

1 데이터분석 단계(Data Analysis Cycle)

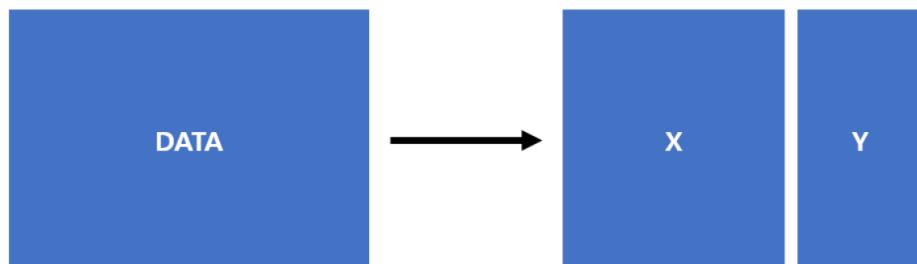
[Open in Colab](#)

✓ 데이터 전처리: (0) 쓸모 없을 뻔한 Raw를 쓸모 있는 Data로 변환

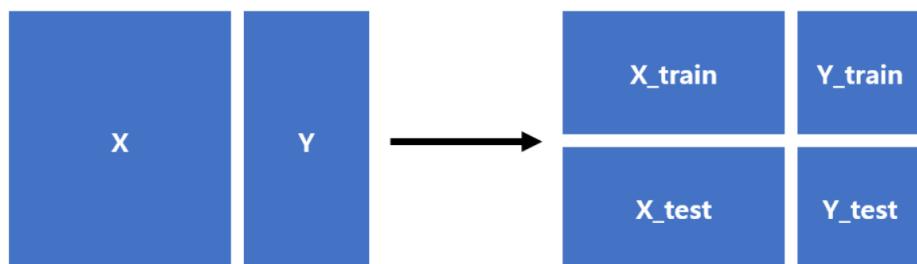
100	T50	횟수	111	TPU	...
few	Gds	Hvi	Rew	Fa	...
Fre	CT	QTP	D	합	...
'1'	1	23	22	NaN	43
76	NaN	43	32	1	8
'Hi'	NaN	NaN	NaN	NaN	87
23	98	NaN	64	46	NaN
c	90	'WW'	24	'KK'	4
t	NaN	2	NaN	NaN	6
64	NaN	90	'IU'	4	76

번호	시간	총량	기간	누적	...
1	1	23	22	21	43
76	33.3	43	32	1	8
5	33.3	52	35	21	87
23	98	52	64	46	61
90	33.3	2	24	33	4
55	2	52	35	21	6
64	33.3	90	11	4	76

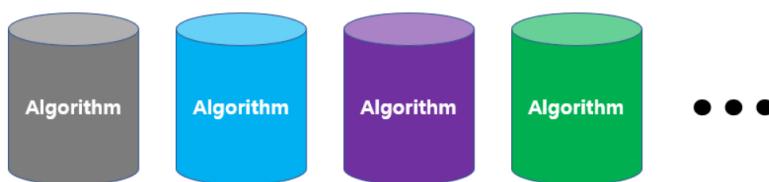
✓ 데이터 분할: (1) 목표/종속변수 Y와 설명/독립변수 X설정



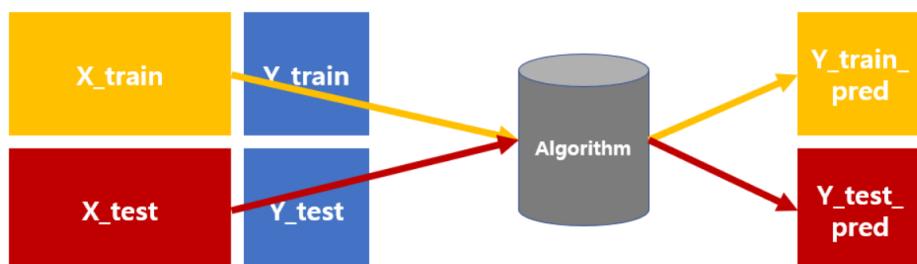
✓ 데이터 분할: (2) 학습데이터 Train과 예측 데이터 Test로 분할



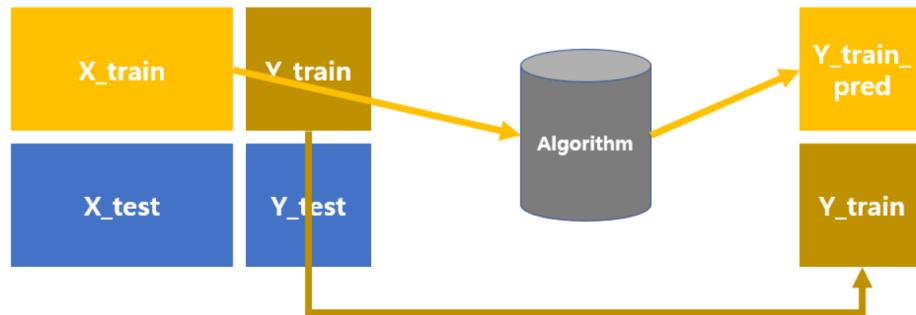
✓ 모델링: (3) 분석 목적에 맞는 알고리즘(Base & Advanced) 후보들 준비



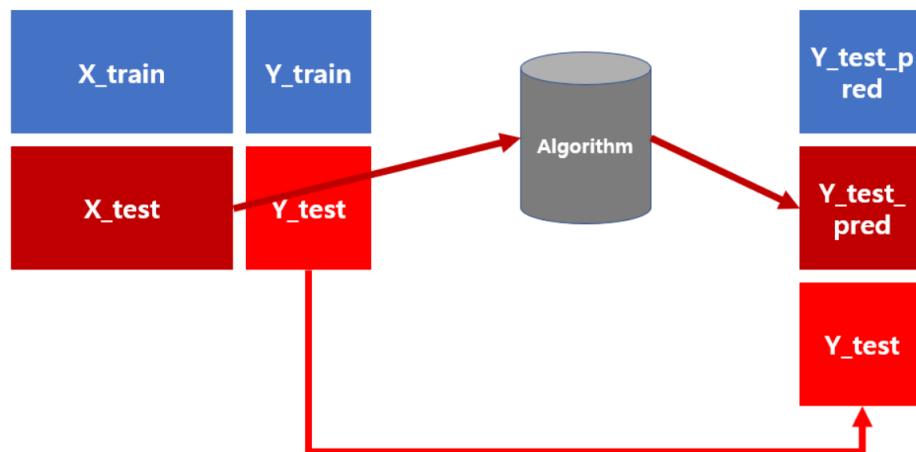
✓ 모델링 & 학습: (4) 알고리즘 평가를 위해 Train/Test의 예측값 추정



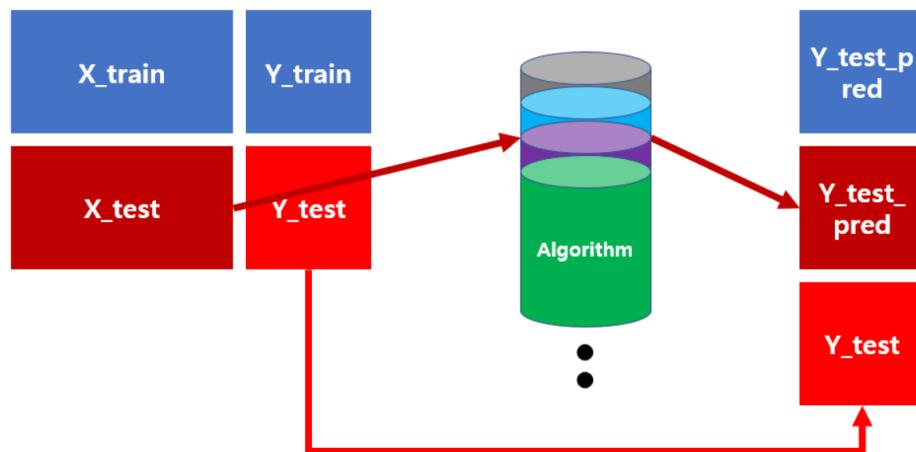
✓ 평가: (5) 학습(Train)이 잘 되었는지 알고리즘 성능검증



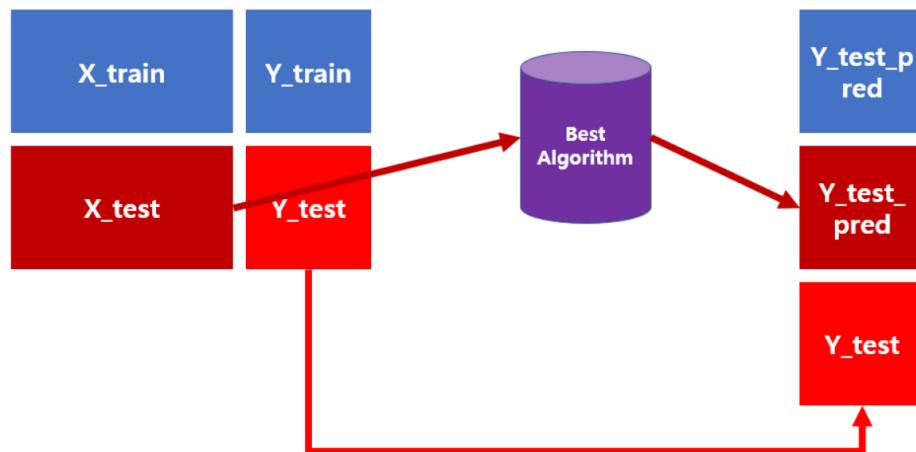
✓ 평가: (6) 예측(Test)이 잘 되었는지 알고리즘 성능검증



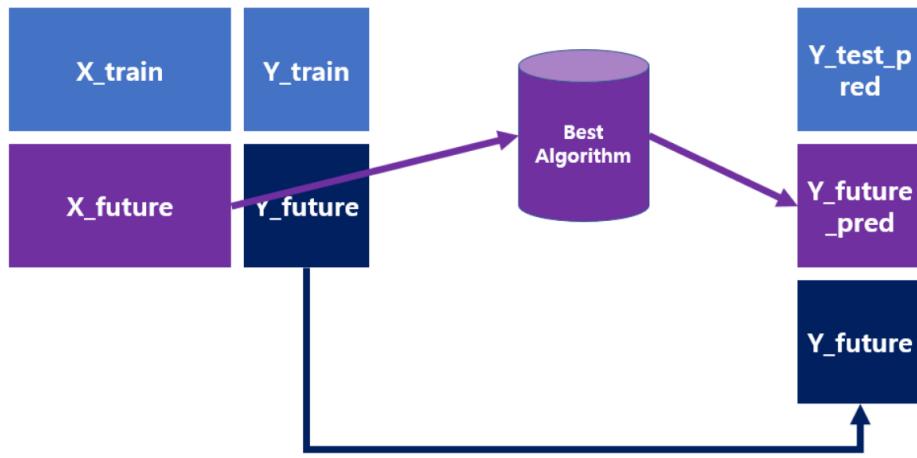
✓ 최적 알고리즘 선택: (7) 알고리즘을 변경하여 위 과정 반복 후



✓ 최적 알고리즘 선택: (7) 최고 성능의 알고리즘 선택

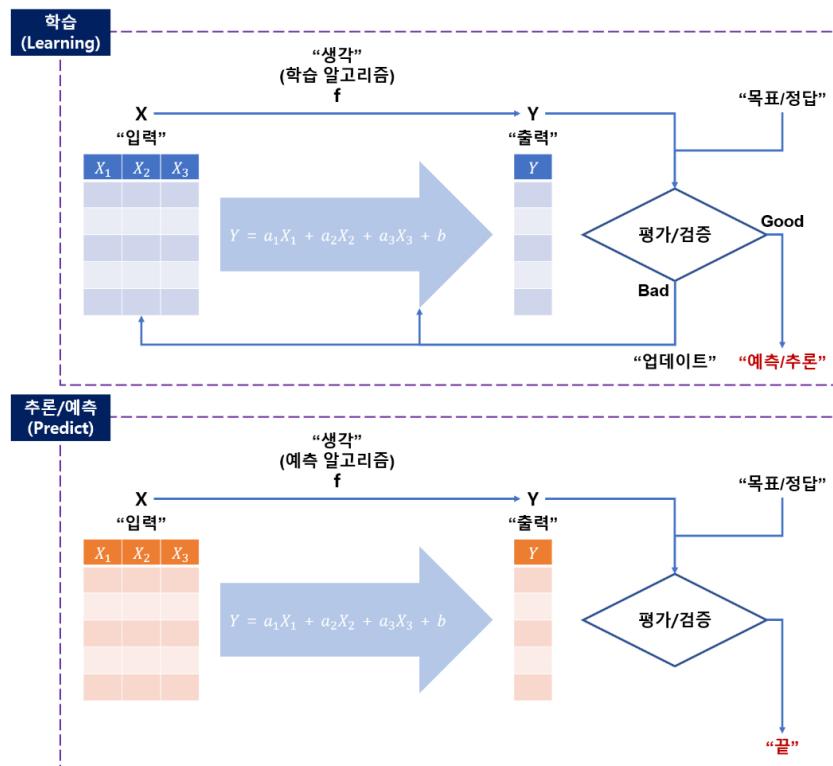


✓ 배포: (8) 실제 비즈니스 서비스 현업 적용 및 매출/수익/개선 정도 평가



2 비지도학습(Unsupervised) 알고리즘: 차원변환

- 데이터분석 과정: 학습 + 추론/예측



"비지도학습(Unsupervised Learning)은 정답 레이블이 없기 때문에, 주로 데이터를 새롭게 표현하여 원래 데이터보다 쉽게 해석하거나 특성을 추가적으로 파악하는데 주로 사용"

- 차원변환:** 비지도학습 알고리즘 중 다차원 특성파악을 위해 사용되는 가장 기본(Baseline) 알고리즘

(비수학적) "일상 속 문제의 다양한 풀이법들의 우선순위를 파악하는 문제"

- 유사한 고객 정보를 가진 사람들의 공통된 쇼핑 취향을 파악하면서 그룹(레이블)으로 추론하는 문제가 군집문제
- 다양한 고객 정보들에서 신규 쇼핑 취향과 같은 추가적인 특성을 파악하기 위해 고객 정보들을 차원변환하여 문제를 다각도로 살펴보는 것

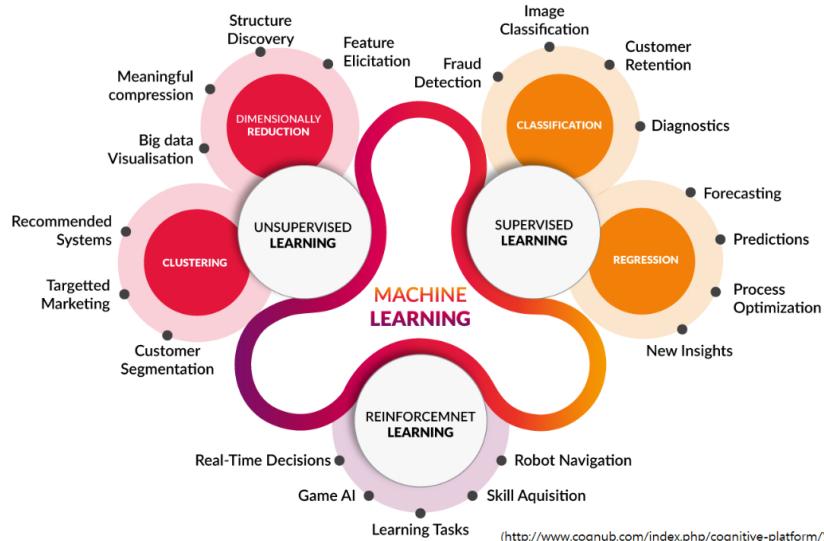
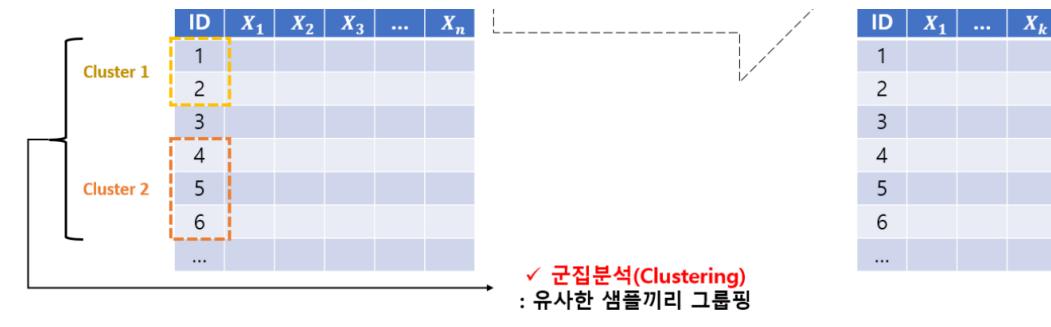
(수학적) "특정 출력(증속변수)/입력(독립변수)의 구분이나 관계 추론도 없고 학습을 위한 목표값도 없이, 주어진 데이터의 추가적인 특성확인을 위해 다른 차원으로 변환(Reduction) 하는 알고리즘"

- 군집문제: 데이터에서 유사한 값들을 가진 군집(Cluster)을 예측 하며 레이블을 활동
- 차원변환: 데이터를 다각도로 살펴보기 위해 다른(차원) 관점으로 추가적인 변수를 예측 하며 특성 확인

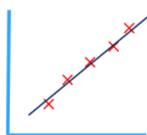
n Dimension
(High-dimensional)

✓ 차원축소(Compression)
: 차원의 수를 줄여 특성추출

k Dimension
(High-dimensional)



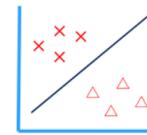
- How much is the stock of Samsung Electronics tomorrow?

 **Regression** – Looking for a statistical relationship across variables that may give us an estimate of a particular outcome. (Supervised)

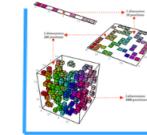
- Are Samsung Electronics and Naver similar business companies?

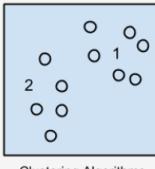
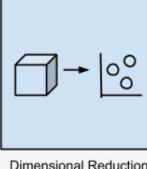
 **Clustering** – Do not have predefined classes but trying to find groups or sets based upon data at hand. (Unsupervised)

- Will Samsung Electronics' stocks rise or fall tomorrow?

 **Classification** – Similar to regression but looking for separations in the data given predefined classes. (Supervised)

- What are the representatives among all stocks in the KOSPI?

 **Dimensionality Reduction** – Transformation of data from high-dimensional into a low-dimensional space so that it retains some meaningful properties of the origin data. (Unsupervised)

Clustering Algorithms	Association Rule Learning Algorithms	Dimensionality Reduction Algorithms	Ensemble Algorithms	Deep Learning Algorithms
	(A,B) → C (D,E) → F (A,E) → G			
k-Means	Apriori algorithm	Principal Component Analysis (PCA)	Boosting	Deep Boltzmann Machine (DBM)
k-Medians	Eclat algorithm	Principal Component Regression (PCR)	Bootstrapped Aggregation (Bagging)	Deep Belief Networks (DBN)
Expectation Maximisation (EM)	-	Partial Least Squares Regression (PLSR)	AdaBoost	Convolutional Neural Network (CNN)
Hierarchical Clustering	-	Sammon Mapping	Stacked Generalization (blending)	Stacked Auto-Encoders
-	-	Multidimensional Scaling (MDS)	Gradient Boosting Machines (GBM)	-
-	-	Projection Pursuit	Gradient Boosted Regression Trees (GBRT)	-
-	-	- Linear Discriminant Analysis (LDA)	Random Forest	-
-	-	- Mixture Discriminant Analysis (MDA)	-	-

- 차원축소: 차원변환 알고리즘은 다양하며, 매우 많은 변수들로 구성된 다차원 데이터를 의미 있는 특성을 유지하면서 차원을 축소하여 저차원 데이터로 재표현(주로 축소) 하는 것

- Why?

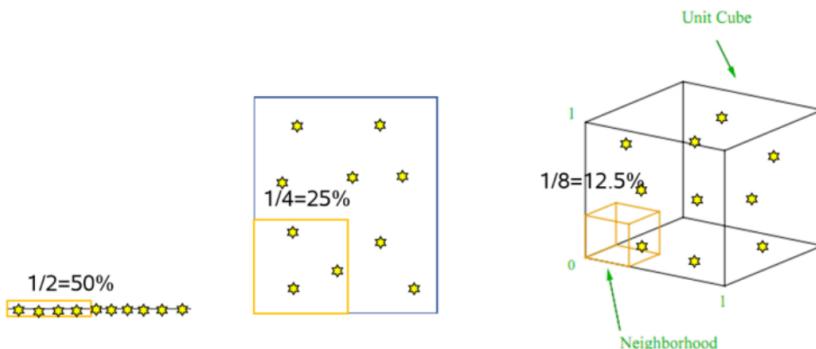
(비수학적)

"현실 데이터는 고차원 공간으로 구성되어 있고 우리는 인위적으로 저차원 공간 데이터를 수집하며, 유사관점에서 보유하고 있는 빅데이터(저차원+고차원)를 사람이 인식가능한 스몰데이터로 표현 가능할 것"

"단순히 데이터를 압축하는 것이 목적이 아니라, 일부 정보 손실이 있지만 이를 최소화하면서 고차원 특성 잘 파악하여 잠재적인 요소를 추출하는 것이 목적"

(수학적)

"차원이 증가(변수의 증가) 할수록 데이터 간의 거리가 기하급수적으로 증가하기 때문에 데이터의 밀도가 희소(Sparse)한 구조를 가지게 되어 이를 사용한 모델링의 알고리즘 성능 하락하는 차원의 저주 발생"



"차원이 증가(변수의 증가)하면 특정 독립변수는 다른 독립변수에 영향을 받는(설명되는) 경우가 발생하며 이를 다중공선성(Multicollinearity)라고 함"

- How?

(1) 변수 선택(Feature/Variable Selection): 특정 변수가 다른 변수들로 생성될 수 있는 경우, 특정 변수의 종속성이 강하다고 하고 간단히 제거를 통해 중요 변수들만 구성하는 차원축소

- 장점: 남은 변수들을 통해 중요도와 해석이 용이
- 단점: 변수들 간의 종속성/상관성을 명확하게 고려하기 어려움
- Ridge, Lasso, VIF(Variance Inflation Factor) 등

(2) 변수 추출(Feature/Variable Extraction): 변수들 간의 상관관계를 고려하여 새로운 중요 변수를 생성하는 차원축소

- 장점: 변수들 간의 상관성을 고려하기 용이하고 변수들의 갯수를 많이 줄일 수 있음
- 단점: 새롭게 추출된 변수들의 의미나 해석이 어려움
- PCA(Principal Component Analysis), FA(Factor Analysis) 등

- 종류: 차원변환 알고리즘은 다양하고, 데이터 특성/구조/목적에 맞는 적절한 선택 필요

- (Linear) Projection: 선형 기준으로 데이터를 근사하여 직관적으로 차원을 축소
- (Non-linear) Manifold Learning: 직관적으로 파악이 어려운 데이터의 비선형적 관계를 반영하여 차원을 축소

| 접근방법 | 알고리즘 | :---: | (Linear) Projection | Eigen Value Decomposition | Singular Value Decomposition | Truncated SVD | Principal Component Analysis | Factor Analysis | Linear Discriminant Analysis | Quadratic Discriminant Analysis | (Non-linear) Manifold Learning | Kernel Principal Component Analysis | Locally Linear Embedding (LLE) | Isomap | Multi Dimensional Scaling | Spectral Embedding | t-distributed Stochastic Neighbor Embedding (t-SNE) | Autoencoders | Self Organizing Map (SOP) |

- Target Algorithm:

- Eigen Value Decomposition
- Singular Value Decomposition
- Truncated SVD
- Principal Component Analysis
- Principal Component Regression
- Factor Analysis
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Mixture Discriminant Analysis
- Non-Negative Matrix Factorization
- Locally Linear Embedding
- t-distributed Stochastic Neighbor Embedding

3 예제 데이터셋(Dataset)

3.1 statsmodels 모듈 사용 데이터셋

```
# 라이브러리 불러오기
import statsmodels.api as sm

• 대기중 CO2농도 데이터:
data = sm.datasets.get_rdataset("CO2", package="datasets")

• 황체형성 호르몬(Luteinizing Hormone)의 수치 데이터:
data = sm.datasets.get_rdataset("lh")

• 1974~1979년 사이의 영국의 호흡기 질환 사망자 수 데이터:
data = sm.datasets.get_rdataset("deaths", "MASS")

• 1949~1960년 사이의 국제 항공 운송인원 데이터:
data = sm.datasets.get_rdataset("AirPassengers")

• 미국의 강수량 데이터:
data = sm.datasets.get_rdataset("precip")

• 타이타닉호의 탑승자들에 대한 데이터:
data = sm.datasets.get_rdataset("Titanic", package="datasets")
```

- `data`가 포함하는 정보:

- `package`: 데이터를 제공하는 R 패키지 이름
- `title`: 데이터 이름
- `data`: 데이터를 담고 있는 데이터프레임
- `__doc__`: 데이터에 대한 설명 문자열(R 패키지의 내용 기준)

3.2 sklearn 모듈 사용 데이터셋

1) 패키지에 포함된 데이터(`load` 명령어)

```
# 라이브러리 불러오기
from sklearn.datasets import load_boston

• load_boston: 회귀용 보스턴 집값
raw = load_boston()
print(raw.DESCR)
print(raw.keys())
print(raw.data.shape, raw.target.shape)

• load_diabetes: 회귀용 당뇨병 자료
• load_linnerud: 회귀용 linnerud 자료
• load_iris: 분류용 붓꽃(iris) 자료
• load_digits: 분류용 숫자(digit) 필기 이미지 자료
• load_wine: 분류용 포도주(wine) 등급 자료
• load_breast_cancer: 분류용 유방암(breast cancer) 진단 자료
```

2) 인터넷에서 다운로드할 수 있는 데이터(`fetch` 명령어)

```
# 라이브러리 불러오기
from sklearn.datasets import fetch_california_housing

• fetch_california_housing: 회귀용 캘리포니아 집값
raw = fetch_california_housing()
print(raw.DESCR)
print(raw.keys())
print(raw.data.shape, raw.target.shape)

• fetch_covtype: 회귀용 토지 조사 자료
• fetch_20newsgroups: 뉴스 그룹 텍스트 자료
• fetch_olivetti_faces: 얼굴 이미지 자료
• fetch_lfw_people: 유명인 얼굴 이미지 자료
• fetch_lfw_pairs: 유명인 얼굴 이미지 자료
• fetch_rcv1: 로이터 뉴스 말뭉치
• fetch_kddcup99: Kddcup 99 Tcp dump 자료
```

3) 확률분포를 사용한 가상 데이터(`make` 명령어)

```
# 라이브러리 불러오기
from sklearn.datasets import make_regression
```

- make_regression: 회귀용 가상 데이터
 $X, y, c = \text{make_regression}(\text{n_samples}=100, \text{n_features}=10, \text{n_targets}=1, \text{bias}=0, \text{noise}=0, \text{coef=True}, \text{random_state}=0)$
- make_classification: 분류용 가상 데이터 생성
- make_blobs: 클러스터링용 가상 데이터 생성

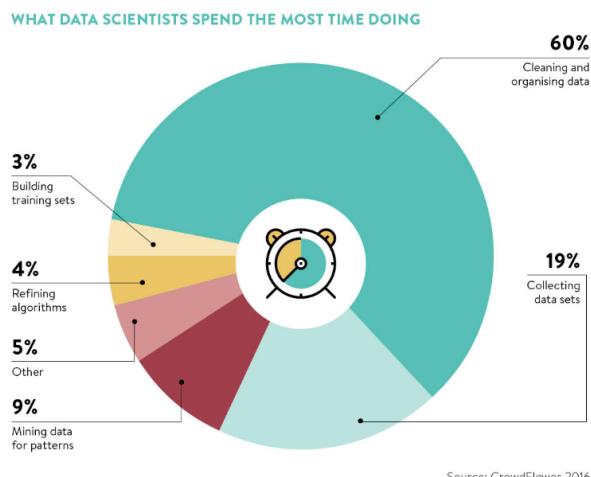
4) load/fetch 명령어 데이터에서 raw가 포함하는 정보: Bunch라는 클래스 객체 형식으로 생성

- data: (필수) 독립 변수 ndarray 배열
- target: (필수) 종속 변수 ndarray 배열
- feature_names: (옵션) 독립 변수 이름 리스트
- target_names: (옵션) 종속 변수 이름 리스트
- DESCR: (옵션) 자료에 대한 설명

4 전처리 방향(Preprocessing)

• 목표:

- 대량으로 수집된 데이터는 그대로 활용 어려움
- 잘못 수집/처리 된 데이터는 엉뚱한 결과를 발생
- 알고리즘이 학습이 가능한 형태로 데이터를 정리



일반적인 전처리 필요항목:

- 데이터 결합
- 결측값 처리
- 이상치 처리
- 자료형 변환
- 데이터 분리
- 데이터 변환
- 스케일 조정 : 단위가 큰 변수들에 의해 차원변환 결과가 왜곡 될 수 있음!

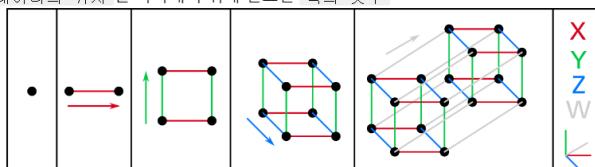
5 함수세팅 및 추정 방향(Modeling): Principal Component Analysis

- Eigen Value Decomposition
- Singular Value Decomposition
- Truncated SVD
- Principal Component Analysis

5.1 차원변환 필수 개념

"차원변환 알고리즘들은 대부분 차원을 다루는 선형대수를 활용하여 벡터기반으로 개발 되고 만들어진 알고리즘"

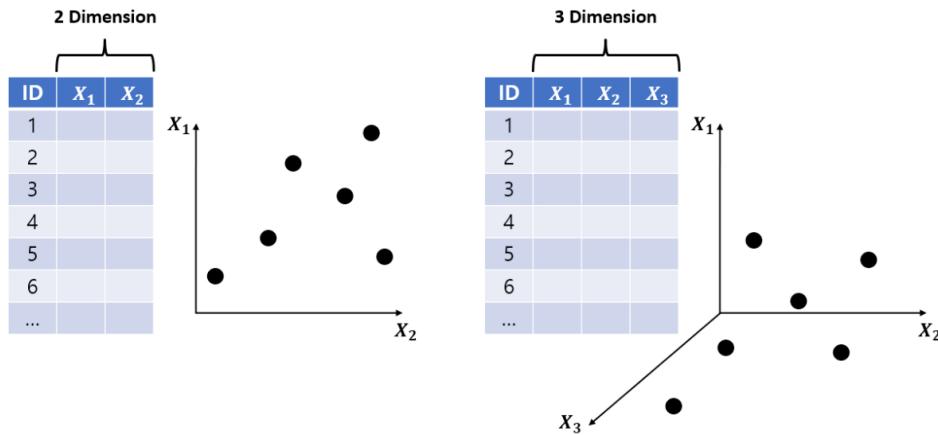
0) 차원(Dimension): 공간 내에서 데이터의 위치를 나타내기 위해 필요한 축의 갯수



0	1	2	3	4	#Dim
(https://commons.wikimedia.org/wiki/File:Dimension_levels.svg)					

- 변수 = 차원 : 데이터가 n 개의 변수를 가지면, n 차원의 좌표축 상에 모든 데이터 위치 표현 가능

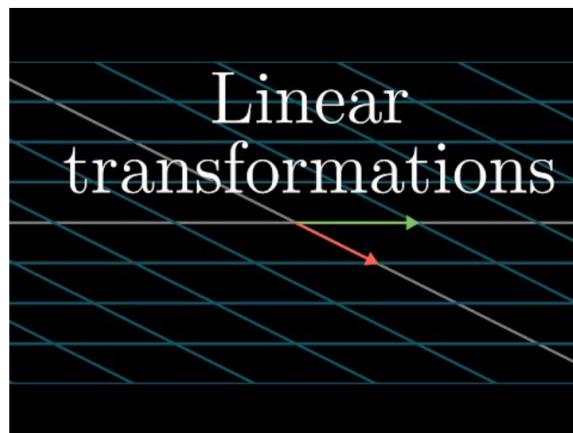
- 각 변수의 값을 사용해 공간상에 하나의 방향을 가진 직선(벡터)로 표현하기 때문에 각 변수를 하나의 공간 표현 차원으로 인식



- 우리가 2차원 데이터, 3차원 데이터 등으로 부르는 경우, 변수의 수가 아니라 인덱스, 변수, 시간 등 사람의 눈으로 해석을 위해 필요한 축의 갯수로 의미 다름

ID	X ₁	X ₂	X ₃
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
...

1) 기저(Basis): 데이터의 독립 변수 갯수를 파악하기 위해 선형대수의 벡터 기반의 용어



데이터1		데이터2			
	변수1	변수2			
샘플1	2	0	샘플1	1	2
샘플2	0	1	샘플2	0	0

변수2 공간 (0,1)	(0,1)로 (2,0) 표현/복제 가능? $(2,0) = ? * (0,1)$	(1,0)로 (2,0) 표현/복제 가능? $(2,0) = 2 * (1,0)$
	(2,0)로 (0,1) 표현/복제 가능? $(0,1) = ? * (2,0)$	(2,0)로 (1,0) 표현/복제 가능? $(1,0) = 0.5 * (2,0)$
	변수1 & 변수2 '독립' ⇒ Basis 2개 ⇒ 독립차원 2개 ⇒ 독립변수 2개	변수1 & 변수2 '종속' ⇒ Basis 1개 ⇒ 독립차원 1개 ⇒ 독립변수 1개

변수1 공간
(2,0)

변수1 공간
(1,0)

변수2 공간
(2,0)

- 데이터의 독립 차원/변수 갯수 파악하기 위해 선형대수에선 Basis 갯수 추정
- Basis의 갯수만큼의 데이터 표현 공간 확인 가능
 - 데이터1은 최대 2차원 공간의 데이터 표현 가능: (5,3) 가능
 - 데이터2는 최대 1차원 공간의 데이터 표현 가능: (5,3) 불가능

- 컴퓨터가 인식하는 최대 독립변수의 수 만큼:

- 데이터1은 변수1 & 변수2가 다른 데이터로 인식
- 데이터2는 변수1 & 변수2를 같은 데이터로 인식
- 회귀문제/분류문제에서 독립변수의 수가 중요했던 이유

$$W = (X^T X)^{-1} X^T Y$$

모든 X 가 독립은 아님

= (기저의 갯수 ≠ 입력변수의 갯수)

= 역행렬이 미존재

= X 가 Full Rank가 아님

= $X^T X$ 가 양의 정부호(Positive Definite)가 아님

=> Full Rank / Positive Definite 계산을 통해 쉽게 Basis 갯수 & 독립변수 갯수 & 역행렬 존재 확인 가능

Full Rank

= (기저의 갯수 = 입력변수의 갯수)

= 역행렬 존재

= 모든 X 가 독립

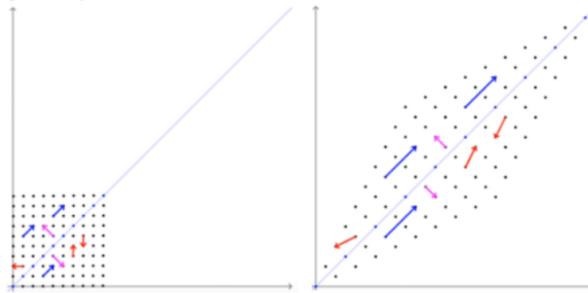
- 예시: 주소를 말하시오!

예시1

예시2

주소	서울시 용산구 무순동	서울시 청주시 김해시
이해도	어디에 사는지 바로 이해가능	어디에 사는지 이해불가능
입력변수 수	3개(서울시/용산구/무순동)	3개(서울시/청주시/김해시)
독립변수 수	3개(시/구/동)	1개(시)
Basis 수	3개(시/구/동)	1개(시)
역행렬 존재	존재 => 3차원에 사는데 3차원으로 말하니 이해가능	미존재 => 3차원에 사는데 1차원으로 말하면 이해못함

2) 고유벡터(Eigenvector) & 고유값(Eigenvalue):



- 원데이터에 어떠한 차원변환을 하더라도
- 차원 변환된 데이터가 서로 독립이고
- 차원 변환된 데이터의 각 변수의 값이 모두 0이 아니고
- 원데이터의 변수와 차원 변환된 데이터 변수가 크기는 달라도 방향만 같으면

(5-1) Eigenvector: 차원 변환된 데이터 변수

- 차원 변환시, 변화되는 방향

(5-2) Eigenvalue: 원데이터의 변수 대비 차원 변환된 데이터 변수의 크기 변화

- 차원 변환시, 변화되는 크기

$$\text{원데이터} \quad \text{차원 변환된 데이터}$$

$$A \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & \\ \text{샘플2} & \end{matrix} = \lambda \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & \\ \text{샘플2} & \end{matrix}$$

• 예시:

		데이터1		데이터2	
		변수1	변수2	변수1	변수2
샘플1	2	0	샘플1	1	2
샘플2	0	1	샘플2	0	0

```
In [1]: # 고유값 및 고유벡터 추정
import numpy as np
from numpy.linalg import eig

data1 = [[2,0],[0,1]]
data2 = [[1,2],[0,0]]
display('Data1: ', np.matrix(data1))
display('Data2: ', np.matrix(data2))
display('Eigenvalue & Eigenvector of Data1: ', eig(data1)[0], eig(data1)[1])
display('Eigenvalue & Eigenvector of Data2: ', eig(data2)[0], eig(data2)[1])
executed in 167ms, finished 15:27:47 2022-04-25
```

'Data1: '

```
matrix([[2, 0],
       [0, 1]])
```

'Data2: '

```
matrix([[1, 2],
       [0, 0]])
```

'Eigenvalue & Eigenvector of Data1: '

```
array([2., 1.])
array([[1., 0.],
       [0., 1.]])
```

'Eigenvalue & Eigenvector of Data2: '

```
array([1., 0.])
array([[ 1.        , -0.89442719],
       [ 0.        ,  0.4472136 ]])
```

• 데이터1 풀이: Full Rank 인 데이터1은 고유값과 고유벡터가 아래와 같고 총 2개

$$A \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & 2 \\ \text{샘플2} & 0 \end{matrix} = \lambda \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & 1 \\ \text{샘플2} & 0 \end{matrix}$$

1) 데이터에 어떠한 차원변환을 하더라도

$$= \begin{matrix} \text{상수값} & \text{변수1} & \text{변수2} \\ 2 & & 1 \end{matrix} \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & 1 \\ \text{샘플2} & 0 \end{matrix}$$

2) 차원 변환된 서로 독립인 데이터

$$= \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & 1 \\ \text{샘플2} & 0 \end{matrix}$$

3) 각 변수의 값은 모두 0이 아님

4) 서로 크기는 달라도 방향이 같음

$$\begin{matrix} \text{고유} & \text{고유} \\ \text{값1} & \text{값2} \end{matrix} \times \begin{matrix} \text{고유} & \text{고유} \\ \text{벡터1} & \text{벡터2} \end{matrix}$$

• 데이터2 풀이: Full Rank 가 아닌 데이터2는 고유값과 고유벡터가 아래와 같고 총 1개

$$A \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & 1 \\ \text{샘플2} & 0 \end{matrix} = \lambda \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & 1 \\ \text{샘플2} & 0 \end{matrix}$$

1) 데이터에 어떠한 차원변환을 하더라도

$$= \begin{matrix} \text{상수값} & \text{변수1} & \text{변수2} \\ 1 & & 0 \end{matrix} \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & 1 \\ \text{샘플2} & 0 \end{matrix}$$

2) 차원 변환된 서로 독립인 데이터

$$= \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & 1 \\ \text{샘플2} & 0 \end{matrix}$$

3) 각 변수의 값은 모두 0이 아님

4) 서로 크기는 달라도 방향이 같음

고유
값1
고유
벡터1

상수값	1	0	\times	샘플1	1	-0.89
				샘플2	0	0.45

5.2 차원변환 해결을 위한 세팅 및 추정

- Eigen Value Decomposition
- Singular Value Decomposition
- Truncated SVD
- Principal Component Analysis

1) 고유값 분해(EigenValue Decomposition, EVD):

$$\text{원데이터} \quad \text{차원 변환된 데이터}$$

$$A \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & \\ \text{샘플2} & \end{matrix} = \lambda \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & \\ \text{샘플2} & \end{matrix}$$

차원변환 유도행렬 $A = V\Sigma V^{-1}$.

"데이터를 차원변환 시키는 행렬이 정방행렬(정사각형 행렬)이고 모든 차원변수가 독립(Full Rank)인 경우, 3개 행렬로 분해 가능"

$$A = \begin{bmatrix} 1.2 & -0.5 \\ -1.5 & 1.7 \end{bmatrix},$$

Eigenvalues: $\lambda_1 = 0.5486$ and $\lambda_2 = 2.3514$,

$$\text{Eigenvectors: } \vec{v}_1 = \begin{bmatrix} 0.6089 \\ 0.7933 \end{bmatrix}, \vec{v}_2 = \begin{bmatrix} -0.3983 \\ 0.9172 \end{bmatrix}.$$

$$V = [\vec{v}_1, \vec{v}_2] = \begin{bmatrix} 0.6089 & -0.3983 \\ 0.7933 & 0.9172 \end{bmatrix},$$

$$\Sigma = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \begin{bmatrix} 0.5486 & 0 \\ 0 & 2.3514 \end{bmatrix},$$

$$V^{-1} = \text{Inverse of } V = \begin{bmatrix} 1.0489 & 0.4555 \\ -0.9072 & 0.6963 \end{bmatrix}.$$

$$\text{A} = V\Sigma V^{-1}$$

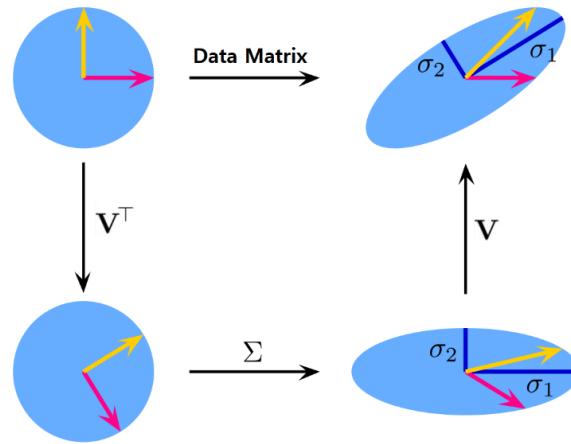
$$\Rightarrow \begin{bmatrix} 1.2 & -0.5 \\ -1.5 & 1.7 \end{bmatrix} = \begin{bmatrix} 0.6089 & -0.3983 \\ 0.7933 & 0.9172 \end{bmatrix} \begin{bmatrix} 0.5486 & 0 \\ 0 & 2.3514 \end{bmatrix} \begin{bmatrix} 1.0489 & 0.4555 \\ -0.9072 & 0.6963 \end{bmatrix}$$

$$\text{원데이터} \quad \text{차원 변환된 데이터}$$

$$A \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & \\ \text{샘플2} & \end{matrix} = \lambda \times \begin{matrix} \text{변수1} & \text{변수2} \\ \text{샘플1} & \\ \text{샘플2} & \end{matrix}$$

- Eigenvector : 차원변환시, 변화되는 방향
- Eigenvalue : 차원변환시, 변화되는 크기

⇒ "원데이터를 특정방향으로 바꾸고 크기를 확대/축소 한 후 원래방향으로 복귀"



2) 특이값 분해(Singular Value Decomposition, SVD):

"데이터를 차원변환 시키는 행렬이 또는 Raw 데이터가, 독립이 아니더라도(Not Full Rank) 그리고 어떠한 형태(정사각형, 직사각형 포함)라도 3개의 행렬로 분해 가능"

$A = U\Sigma V^T$
 where A is $m \times n$ matrix ($m \times n$),
 U is $m \times m$ eigenvector matrix of $A \times A^T$,
 V is $n \times n$ eigenvector matrix of $A^T \times A$,
 and Σ is $m \times n$ diagonal matrix with eigenvalues $\sqrt{\lambda_i}$.

$$A_{m \times n} = U_{m \times m} \times \begin{pmatrix} & & & \\ & & \text{red} & \\ & & \text{pink} & \\ & & \text{white} & \end{pmatrix}_{m \times n} \times V_{n \times n}^T$$

$(m < n)$

$$A_{m \times n} = U_{m \times m} \times \begin{pmatrix} & & & \\ & & \text{red} & \\ & & \text{pink} & \\ & & \text{white} & \end{pmatrix}_{m \times n} \times V_{n \times n}^T$$

$(m > n)$

(<https://www.pikpng.com/transpng/hxRRmbR/>)

- EVD 와 해석 의미는 비슷
- SVD 는 직각각형을 포함한 모든 행렬에서 적용 가능하다는 차이
- EVD 에서 A 의 고유벡터 행렬 이었던 V 와 V^{-1} 가 SVD에서는 각각 AA^T 와 A^TA 의 고유벡터 행렬
- 제곱으로 계산된 고유벡터 대응 고유값은 길이가 2번 곱해진 것이기 때문에, 루트를 사용하여 원래길이로 복원

- 4가지 종류: Full SVD 사용 경우는 드물고, Reduced SVD 사용 일반적

(1) Full SVD:

$$A = U \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_s \\ & & 0 \end{pmatrix} V^T$$

(2) Thin SVD: Σ 에서 비대각파트에 0으로 구성된 부분을 없애고 U 에서는 이에 대응되는 열벡터 제거 한 형태 (A 변경 없음)

$$A = U_s \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_s \end{pmatrix} V^T$$

(3) Compact SVD: Σ 에서 비대각파트 뿐만 아니라 대각파트에도 0으로 구성된 고유값 대응 부분도 제거한 형태 (A 변경 없음)

$$A = U_r \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{pmatrix} V_r^T$$

(4) Truncated SVD: Σ 에서 비대각파트, 0인 고유값 대각파트 뿐만 아니라 0이 아닌 고유값 대각파트도 제거한 형태 (A 변경되어 원래의 A 복원 불가!)

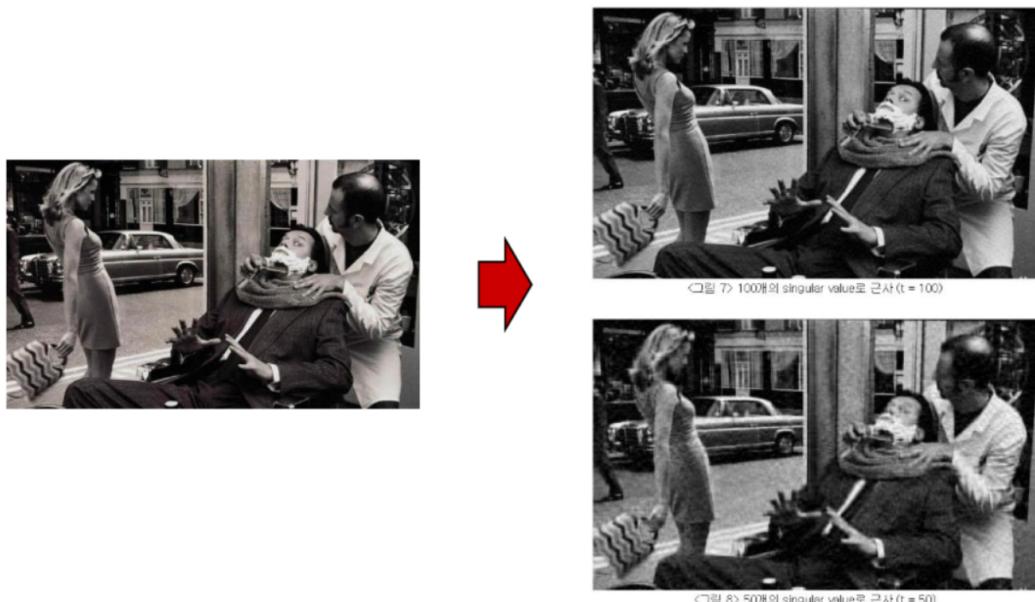
$$A' = U_t \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_t \end{pmatrix} V_t^T$$

(<https://darkpgmr.tistory.com/106>)

"연산이 많을 수 밖에 없는 벡데이터의 차원변환시 Reduced SVD를 통해 연산량 절감 가능"

"Compact SVD 까지는 연산량을 줄여도 차원변환 정도가 그대로지만, Truncated SVD에서는 차원변환 정도가 줄어들기 시작"

"그럼에도 Truncated SVD를 사용하는 큰 이유는, 연산량이 상당히 감소함에도 불구하고 차원변환 정도는 큰 차이가 없기 때문"

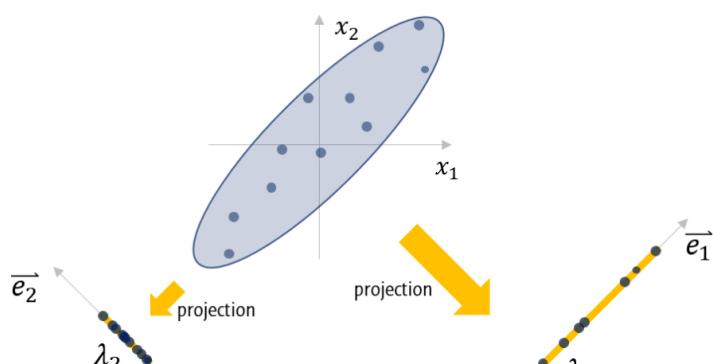
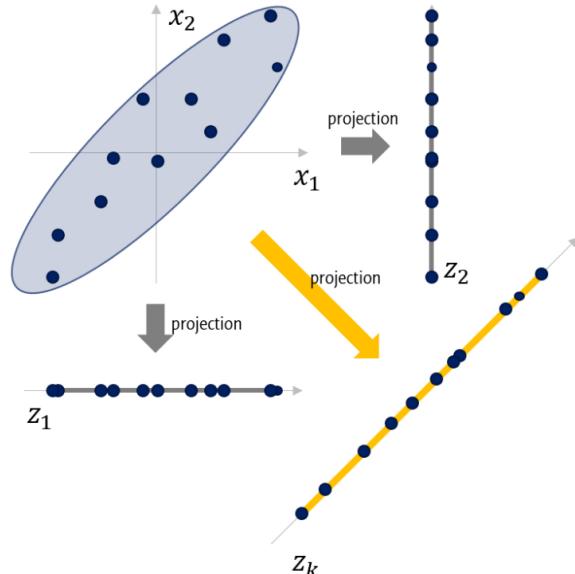


3) 주성분 분석(Principal Component Analysis, PCA): 최적의 차원/변수 조합을 찾아 변수선택 및 차원축소 방법론 중 하나

- **원리:** 모든 차원의 특성을 최대로 유지하면서 원데이터의 손실을 최소화하기 위해 높은 분산을 갖는 기저(Basis)를 찾아 차원축소

"비수학적(기하학적)": 데이터의 특성을 잘 표현하는 데이터의 차원(주성분)을 뽑아 데이터의 양을 줄이는 것

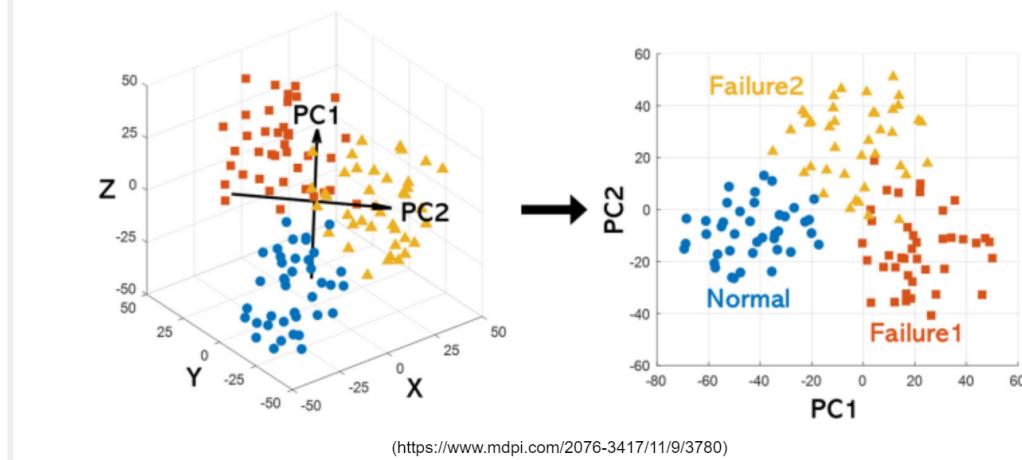
- (1) 데이터의 특징을 잘 표현(분산이 큰)하는 주축(Principle)을 찾음
- (2) 주축으로 데이터를 차원변환 한 상태에서 분산이 큰 성분축(Component)을 찾음
- (3) 성분축을 기준으로 (2)를 지속 반복하여 변수의 갯수 만큼의 모든 주축+성분축 찾음
- (4) 변수의 갯수 보다 적은 주축+성분축 만 골라 최종 차원변환 행렬 생성
- (5) 최종 차원변환 행렬에 데이터를 곱하여 새로운 저차원 데이터 생성



(<https://tyami.github.io/machine%20learning/PCA/>)

z_k

λ_1



(<https://www.mdpi.com/2076-3417/11/9/3780>)

"수학적": 변수들 간에 존재하는 분산(상관관계)을 이용하여 주성분을 추출하면서 차원을 축소하는 기법

(1) 데이터의 표준화/정규화 하여 크기를 조정하고 평균으로 원점을 옮김

$$X_s = \frac{X - \mu}{\sigma}$$

(2) 데이터의 공분산 행렬 계산

- **공분산(Covariance):** 2개 이상의 변수들의 상관관계
- 공분산은 변수의 수에 따라 증가하기 때문에, 변수의 수로 나눔

$$Cov(X_s) = \frac{1}{n-1} X_s^T X_s$$

(3) 공분산 행렬에서 SVD를 사용하여 고유벡터 및 고유값 추정

(4) 고유값이 큰 순서대로 정렬

- 고유값이 가장 큰 경우의 고유벡터는 데이터에서 가장 큰 분산을 가진 기저(Basis)
- 고유값 = 고유벡터의 크기 = 데이터의 분산
- 고유값의 크기만큼의 변수의 분산 크기
- 고유값의 수 = 고유벡터의 수 = 차원/변수의 수
- 변수의 수만큼의 고유값/고유벡터

$$V = [\vec{v}_1, \vec{v}_2, \vec{v}_3, \dots]$$

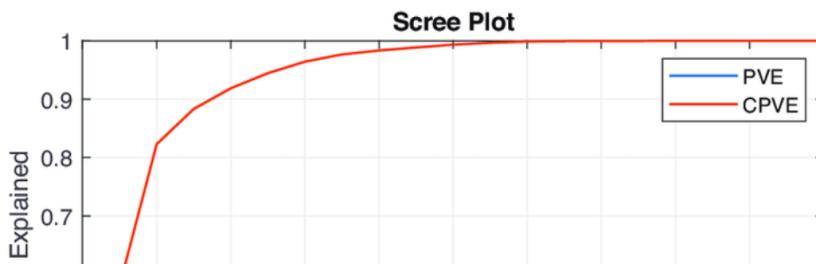
- \vec{v}_1 이 가장 분산이 큰 고유벡터, \vec{v}_2 는 다음으로 분산이 큰 고유벡터
- $\vec{v}_1, \vec{v}_2, \vec{v}_3, \dots$ 는 서로 독립이며 공간상에서는 직각으로 표현됨

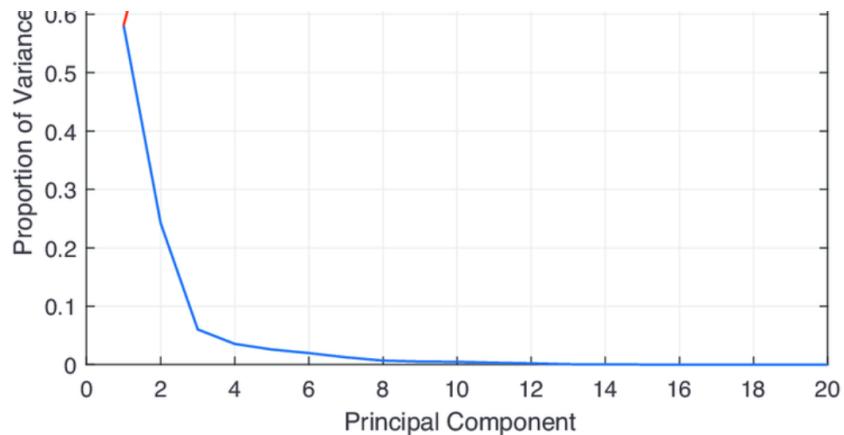
(5) 지정된 최소 분산 크기 이상을 설명하도록, k (변수의 수보다 작은) 번째 고유벡터 까지 선택

$$\begin{aligned} Var(X_s) &\simeq Var(X_{s,k}) \\ &= Var(X_{s,1}) + Var(X_{s,2}) + \dots + Var(X_{s,k}) \\ &= \lambda_1 + \lambda_2 + \dots + \lambda_k \end{aligned}$$

- **PVE(Proportion of Variance Explained):** 주성분들의 분산 비율
- 일반적으로 모든 주성분을 사용하지 않고, 처음 몇 개의 주성분만 사용
- 몇 개의 주성분이 필요한지 Scree Plot 시각화하고 총합은 100%
- 적은 변수 일수록 좋고 많은 분산을 커버할수록 좋음
- 누적 분산의 기울기가 급변하는 지점을 흔히 Elbow(팔꿈치)라고 함

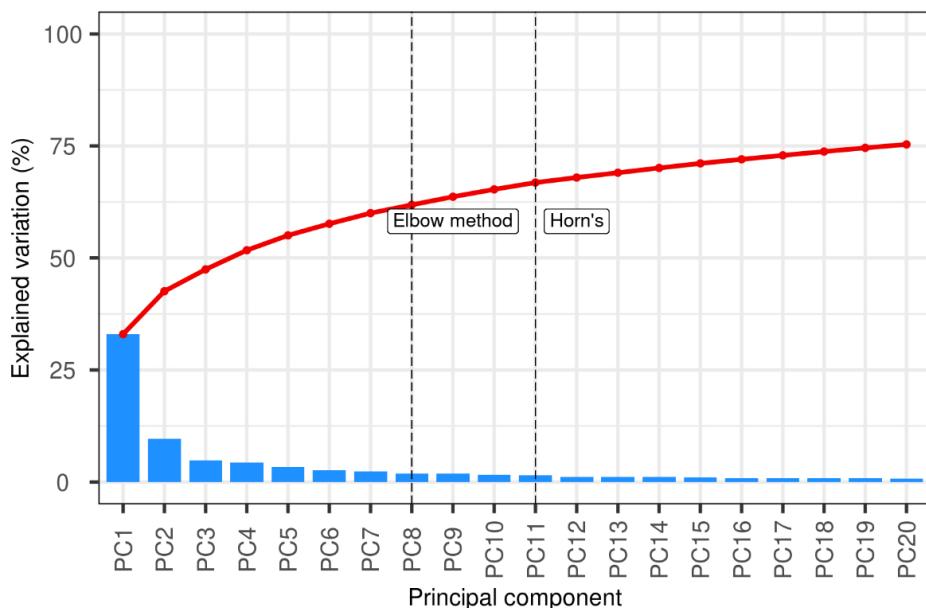
$$\frac{\sum_{j=1}^k \lambda_j}{\sum_{j=1}^n \lambda_j} = 90\%?$$





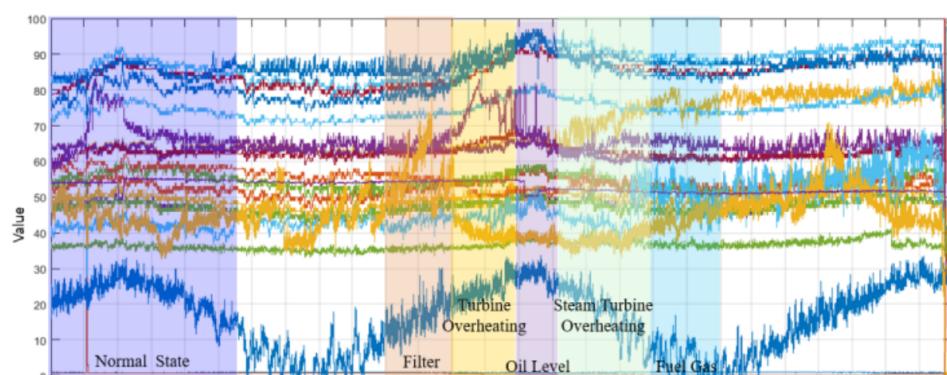
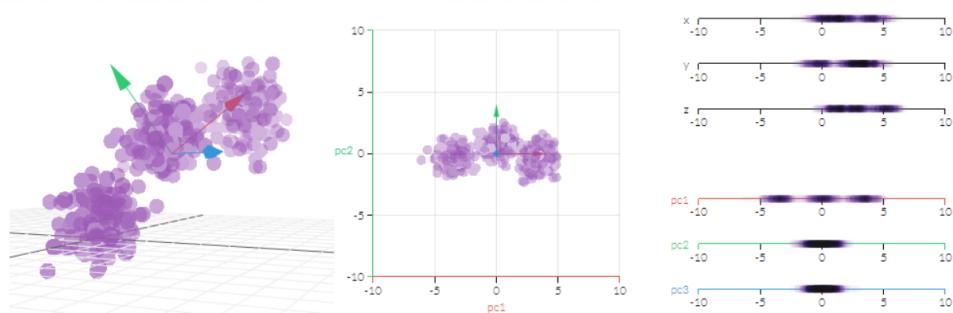
(https://www.researchgate.net/figure/Principal-components-analysis-scree-plot_fig2_350281842)

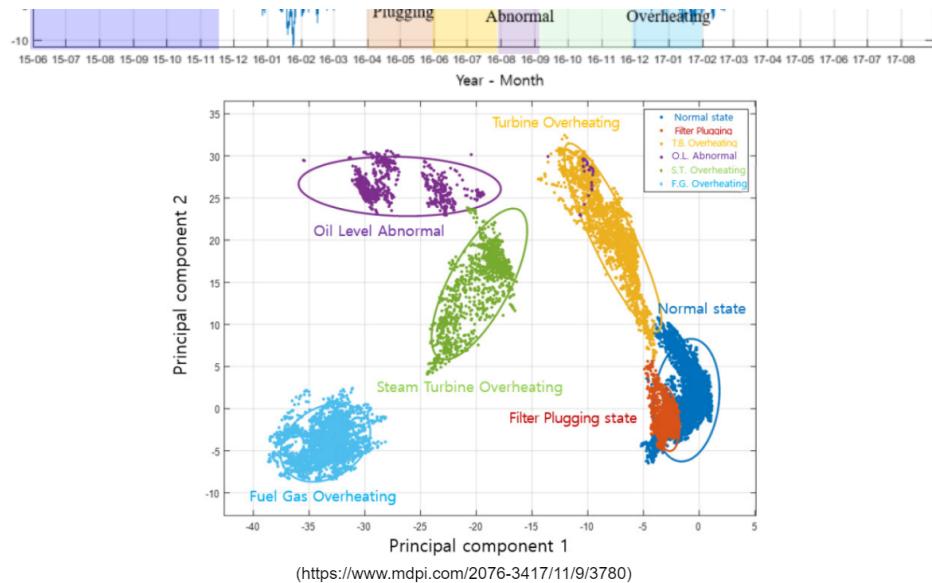
SCREE plot



(<https://bioconductor.org/packages-devel/bioc/vignettes/PCAtools/inst/doc/PCAtools.html>)

(6) 원데이터에 선택된 고유벡터 행렬로 차원변환을 하면 새로운 저차원 데이터





- 정리: 모든 차원의 특성을 최대로 유지하면서 원데이터의 손실을 최소화하기 위해 높은 분산을 갖는 기저(Basis)를 찾아 차원축소

- 데이터의 분포 특성은 일반적으로 평균과 분산 등이 있는데, PCA는 데이터의 편진 정도인 분산(상관관계) 사용
- 차원축소시 벡터와 행렬 연산, 최대화 분산 파악을 위한 고유값, 고유벡터, 공분산, 라그랑주승수 등 여러가지 선형대수 이론 활용
- 원데이터의 분포 특성을 유지한 채, 차원 축소하는 기법으로 특이값 분해(SVD) 방법론 사용

(1) Full SVD:

$$A = U \Sigma V^T$$

A U Σ V^T

Σ is a diagonal matrix with singular values $\sigma_1, \dots, \sigma_s$ and zeros elsewhere.

(4) Truncated SVD: Σ 에서 비대각파트, 0인 고유값 대각파트 뿐만 아니라 0이 아닌 고유값 대각파트도 제거한 형태 (A 변경되어 원래의 A 복원 불가!)

$$A' = U_t \Sigma_t V_t^T$$

A' U_t Σ_t V_t^T

Σ_t is a diagonal matrix containing only the first t non-zero singular values of Σ .

(https://darkpgmr.tistory.com/106)

- 장단점:

"장점"

- 데이터의 차원/변수의 수가 크면 데이터를 한눈에 인식하기 어려운데, PCA가 도움
- 데이터의 주요한 특징을 추출하기 때문에 지도학습을 위한 전처리, 시각화의 도구로도 많이 사용
- 많은 양의 정보를 효과적으로 시각화 가능
- 독립변수들이 독립적이지 않아 발생하는 다중공선성(Multicollinearity)를 해결하는데 기여
- 데이터 저작공간 확보 및 전송효율과 연산속도 유리

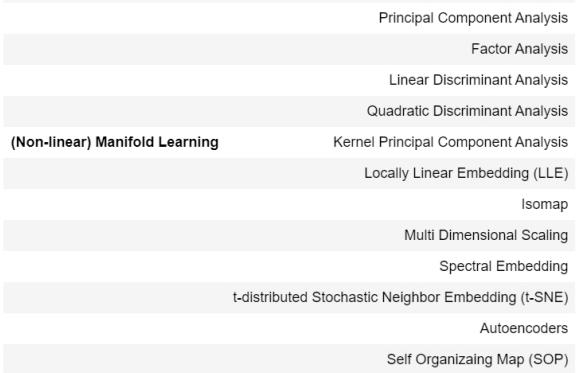
"단점"

- 알고리즘으로서 예측하여 성능 검증이 어려워 이보다 데이터 특성 확인에 많이 사용
- 원데이터 대비 정도의 차이일 뿐 필연적으로 정보 손실이 발생
- 변환된 데이터가 무슨 의미인지? 무슨 값인지? 해석의 어려움
- 공분산이 중요한 기준이기 때문에, 분산이 작은게 좋은 데이터에서는 부적합
- 데이터 분산이 적교하지 않으면 부적합 = 데이터 변수가 독립이 아니면 부적합

5.3 차원변환 알고리즘의 진화

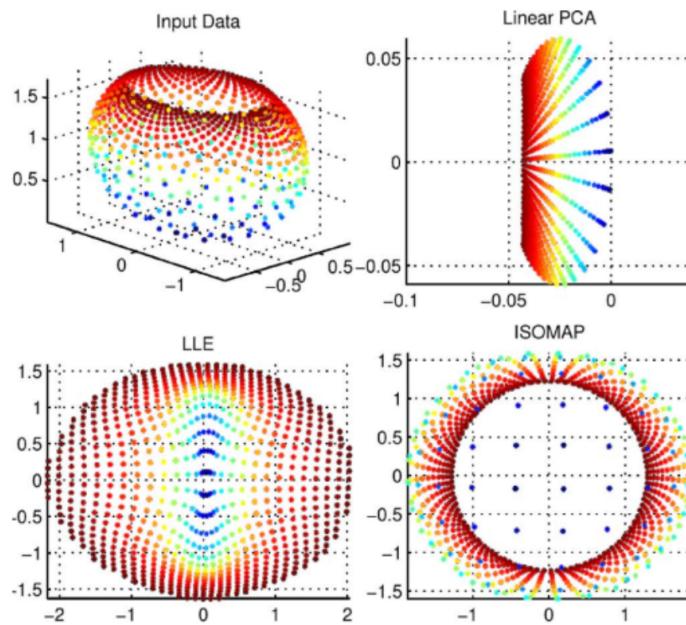
- (Linear) Projection vs (Non-linear) Manifold Learning:

접근방법	알고리즘
(Linear) Projection	Eigen Value Decomposition
	Singular Value Decomposition
	Truncated SVD

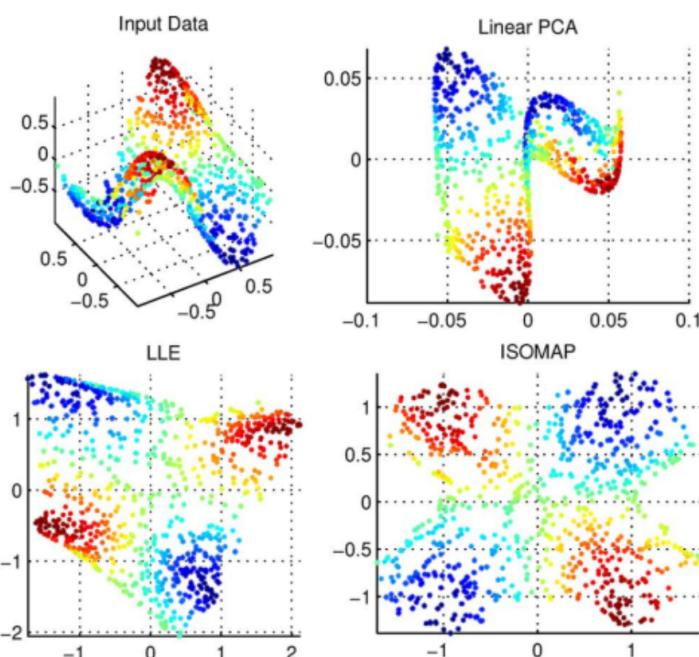


- Comparison:

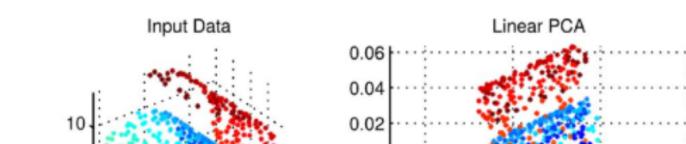
(1)

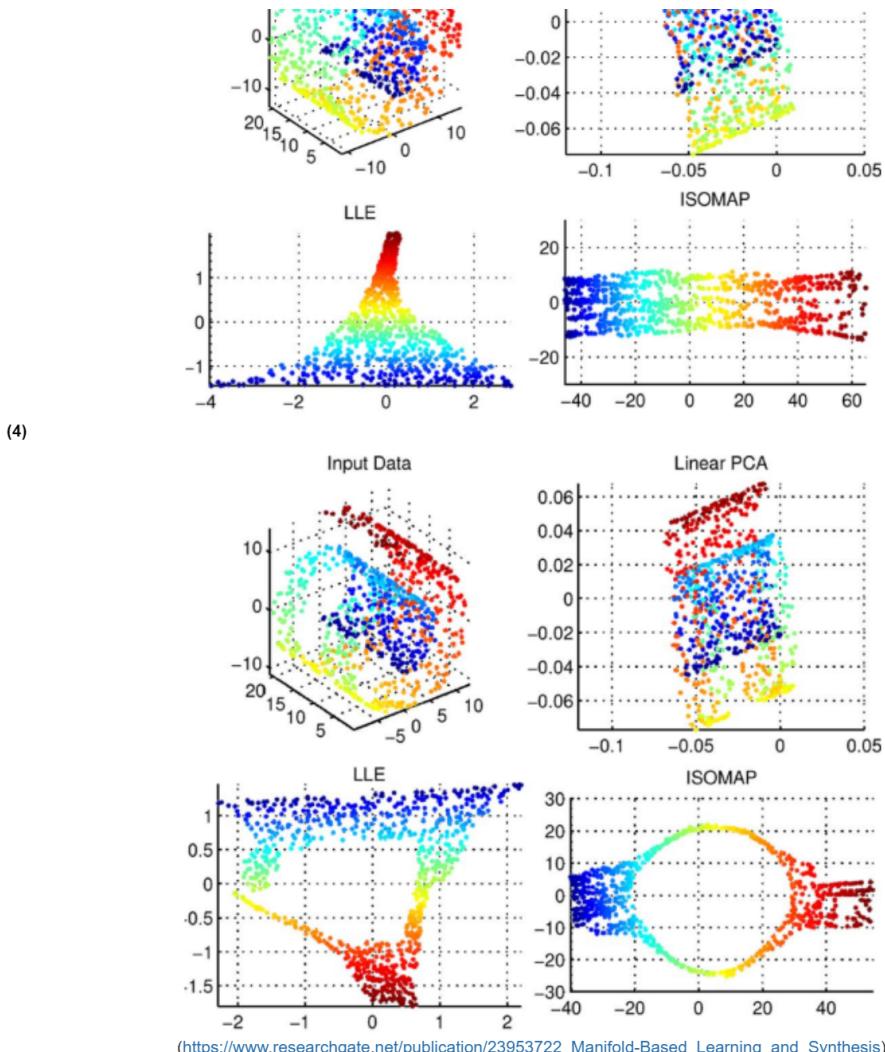


(2)



(3)





[\(5\) Manifold Learning](#)