프로젝트 기반 데이터 과학자 양성과정(Data Science)
Machine Learning 및 분석실습

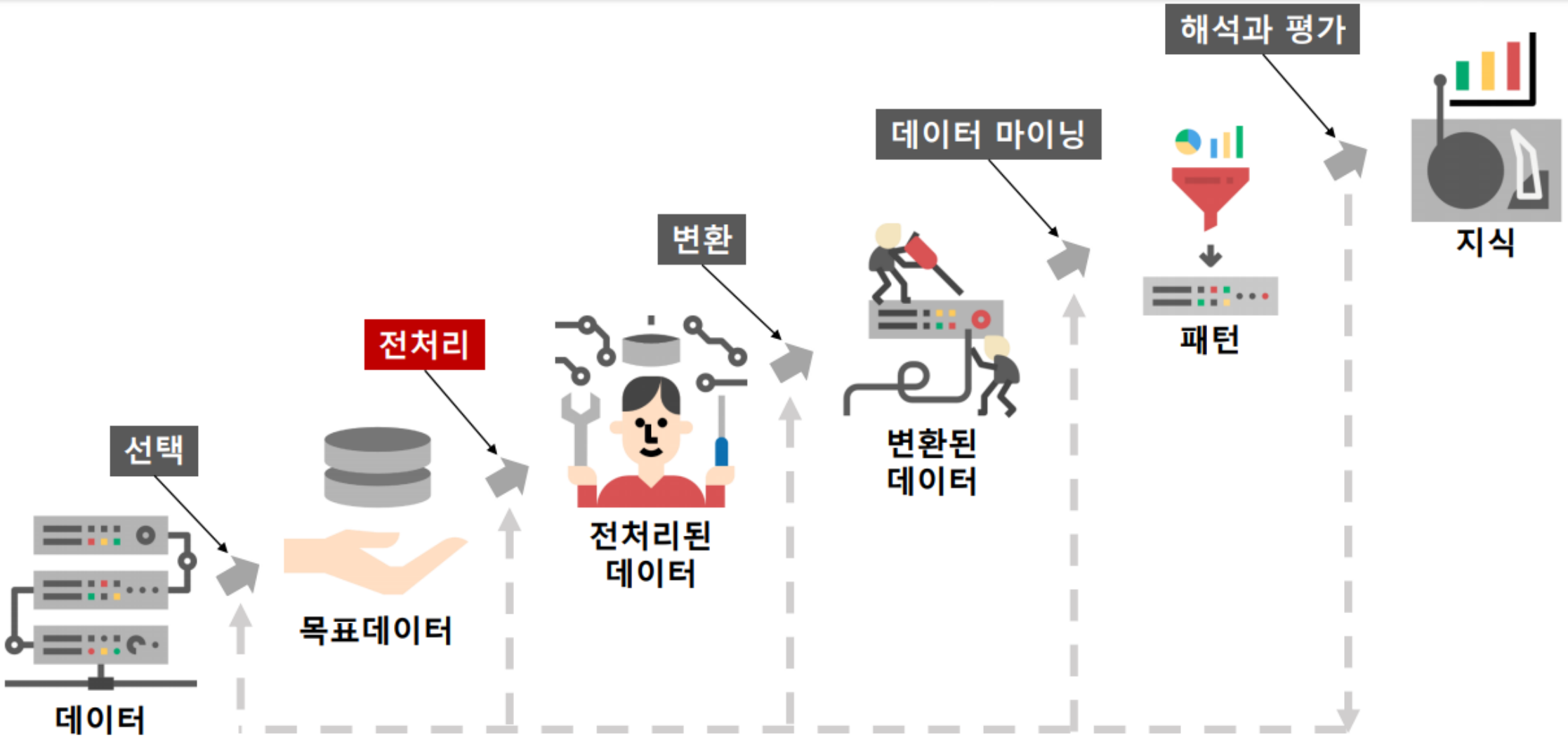
2주차
학습모델과 회귀분석

강사 : 최영진

목차

1. 데이터 전처리

❖ 데이터 분석 단계



1. 데이터 전처리

❖ 데이터 전처리

- 데이터를 분석 및 처리에 적합한 형태로 만드는 과정을 총칭하는 개념
- 데이터 전처리는 데이터 분석 및 처리 과정에서 중요한 단계
- 데이터 분석 , 데이터 마이닝 , 머신 러닝 프로젝트에 적용
- 일반적으로 데이터는 비어있는 부분이 많거나 정합성이 맞지 않는 경우가 많음
- 아무리 좋은 도구나 분석 기법도 품질이 낮은 데이터로는 좋은 결과를 얻을 수 없음

1. 데이터 전처리

❖ 데이터 전처리

- **불완전(incomplete) : 데이터가 비어 있는 경우가 있을 수 있다. 속성에 값이 Null 인 경우**
- **잡음(Noisy) : 에러 또는 잡음이 포함된 경우 예) 나이 : -10**
- **모순된(Inconsistent) : 생년월일과 나이가 맞지 않는 경우, 복사된 레코드의 불일치**

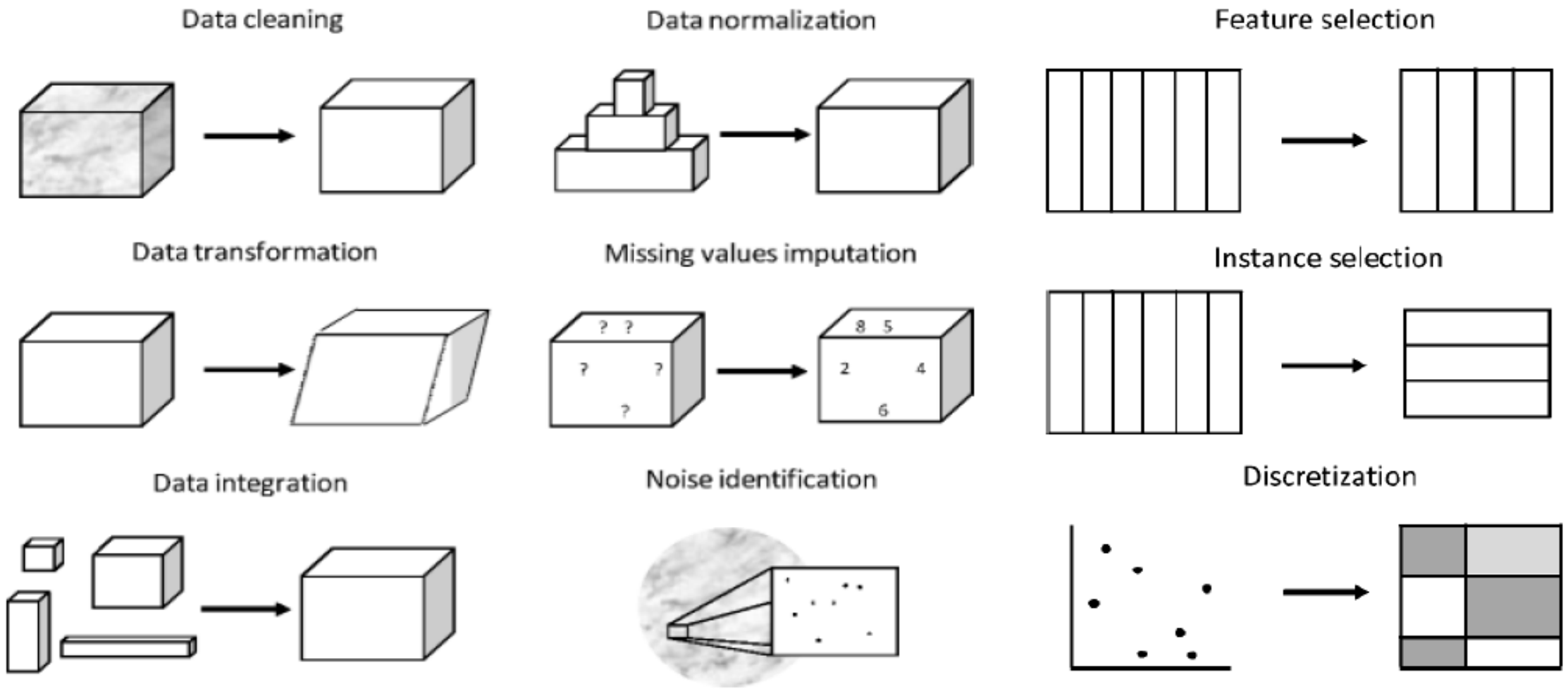
1. 데이터 전처리

❖ 데이터품질 Data Quality

- 완벽한 데이터를 얻는다는 것은 실제에서는 불가능한 일
- 데이터 품질을 저해하는 주요 요인으로는 크게 측정 오류와 수집 과정에서 발생하는 오류로 나눌 수 있음
- 측정 오류 사람의 실수로 잘못된 단위로 기록을 하거나 측정 장비 자체의 한계 등 측정과정에서 발생하는 오류
- 수집 과정 오류 : 데이터의 손실 , 중복 등의 문제로 발생하는 오류

1. 데이터 전처리

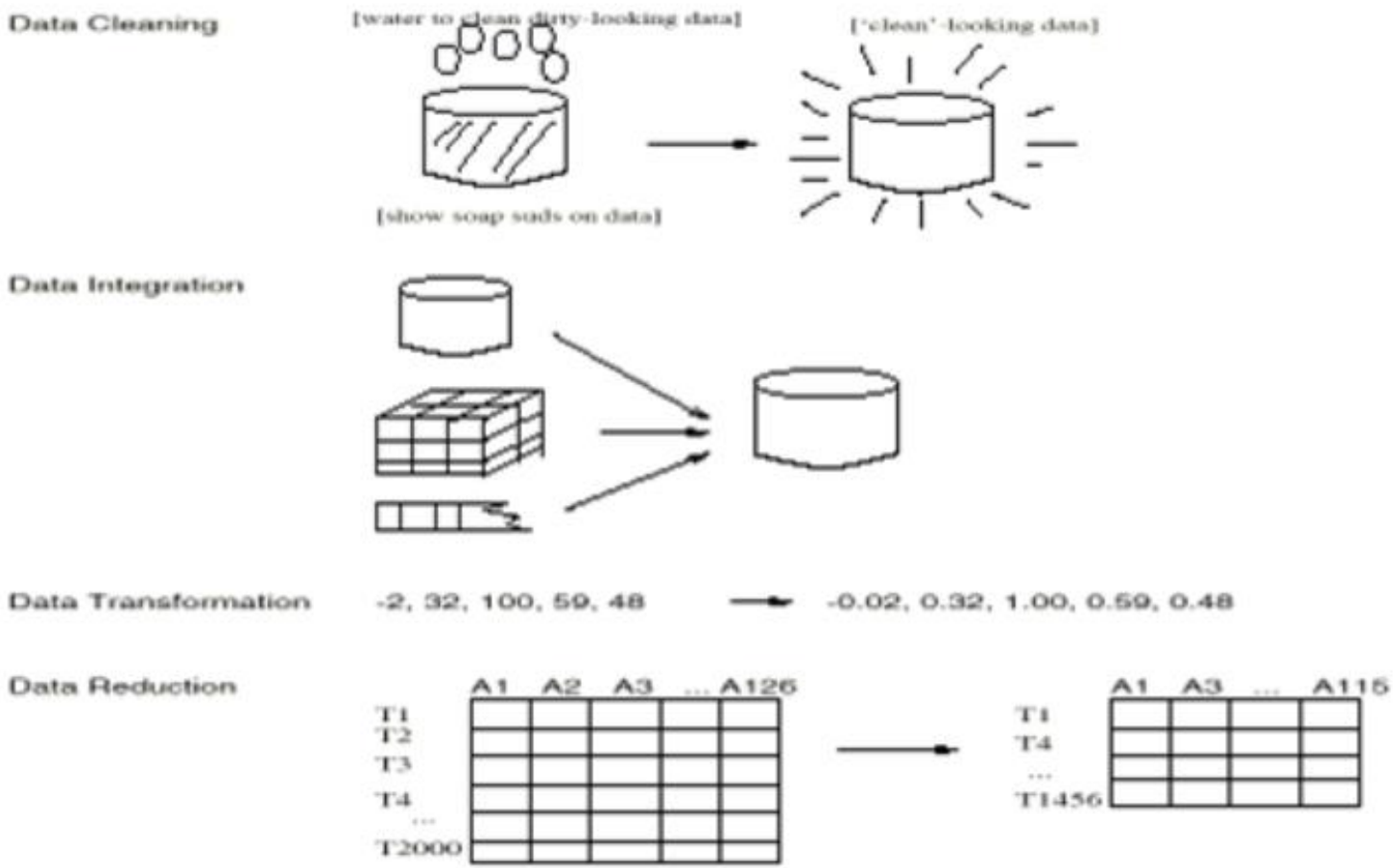
❖ 데이터 전처리



https://bdataanalytics.biomedcentral.com/articles/10.1186/s41044-016-0014-0

1. 데이터 전처리

❖ 데이터 전처리



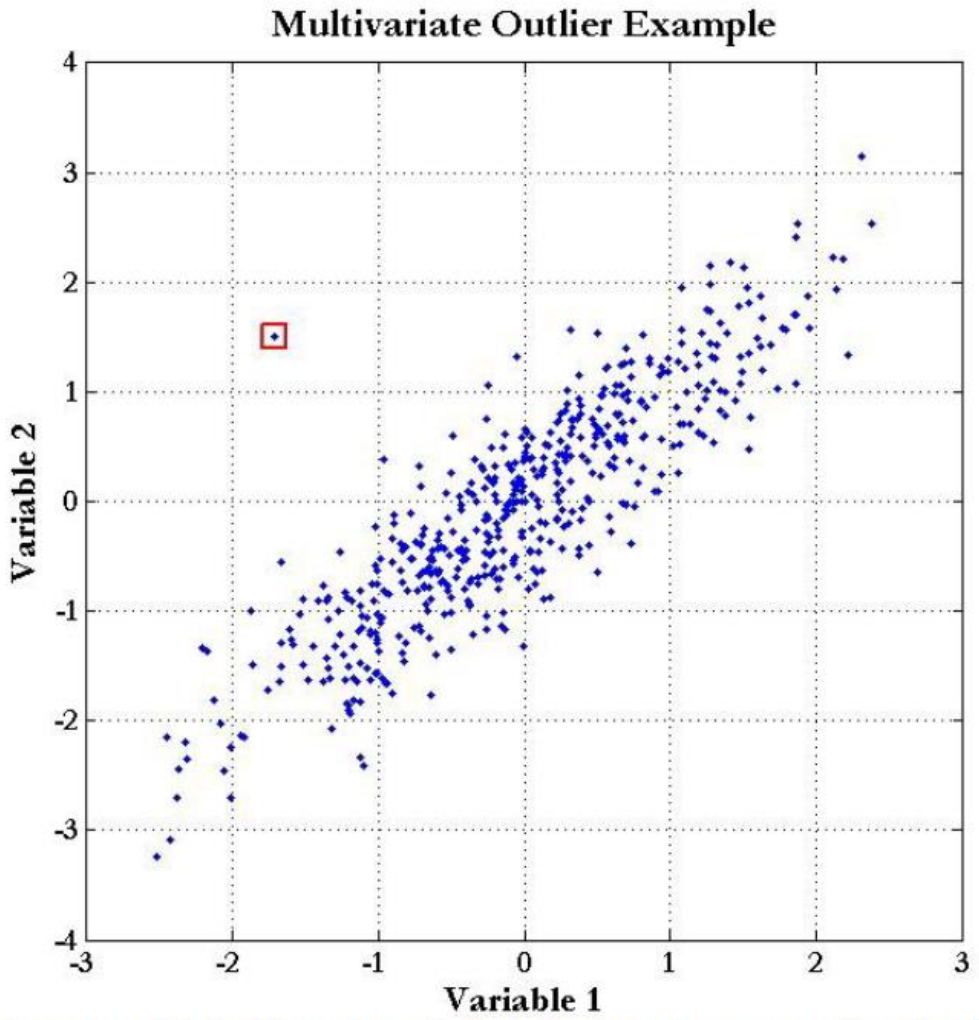
1. 데이터 전처리

❖ 이상치 Outlier

- 대부분의 데이터와 다른 특성을 보이거나 특정 속성의 값이 다른 개체들과 달리 유별난 값을 가지는 데이터를 의미
- 이상치의 중요한 점은 잡음과는 다름
- 잡음이 임의로 발생하는 예측하기 어려운 요인임에 반해 이상치는 적법한 하나의 데이터로서 그 자체가 중요한 분석의 목적이 될 수도 있음
- 예를 들어 네트워크의 침입자 감시와 같은 응용에 있어서는 대다수의 일반 접속 중 예외적으로 발생하는 불법적인 접속시도와 같은 이상치를 찾는 것이 주된 목표

1. 데이터 전처리

❖ 이상치 Outlier



https://madhureshkumar.wordpress.com/2015/06/18/trend-and-outlier/

1. 데이터 전처리

❖ 이상치 Outlier

1. 데이터 전처리

❖ 결측치 Missing values

- 데이터의 결측은 일반적인 경우는 아니지만 드물게 발생하는 문제
- 설문조사의 경우 몇몇 사람들은 자신의 나이나 몸무게와 같은 사적인 정보를 공개하는 것을 꺼리는 경우가 발생하며 이러한 값들은 조사에 있어 결측 값으로 남게 됨

1. 데이터 전처리

❖ 중복 Duplicate data

- 데이터의 중복은 언제든지 발생 가능
- 문제는 중복된 데이터 사이에 속성의 차이나 값의 불일치가 발생할 수 있다는 것
- 기본적으로 모든 속성 및 값이 동일하다면 하나의 데이터는 삭제할 수 있지만 ,
- 그렇지 않은 경우에는 두 개체를 합쳐서 하나의 개체를 만들거나 ,
- 응용에 적합한 속성을 가진 데이터를 선택하는 등의 추가 적인 작업을 필요로 하게 됨

First Name	Last Name	City	Age
John	Barrow	London	32
Aaron	Gomez	Manchester	28
Ralph	Johnson	Newcastle	25
Monica	Robert	Plymouth	37
Craig	Miller	Leicester	30
Paul	Martin	Manchester	32
Claire	Rudolf	London	22
Eric	Plymouth	Birmingham	39
Monica	Robert	Plymouth	37

https://www.opentechguides.com/how-to/article/excel-2016/127/remove-duplicate-data.html

1. 데이터 전처리

❖ 데이터 전처리 단계

데이터 수집 Data Collection
분석이나 학습에 필요한 데이터를 부분 혹은 전체를 수집하는 작업

데이터 정제 Data Cleansing
비어있는 데이터나 잡음, 모순된 데이터 등을 정합성이 맞도록 교정하는 작업

데이터 통합 Data Integration
여러 개의 데이터베이스, 데이터집합 또는 파일을 통합하는 작업

데이터 축소 Data Reduction
샘플링, 차원축소, 특징 선택 및 추출을 통해 데이터 크기를 줄이는 작업

데이터 변환 Data Transformation
데이터를 정규화, 이산화 또는 집계를 통해 변환하는 작업

1. 데이터 전처리

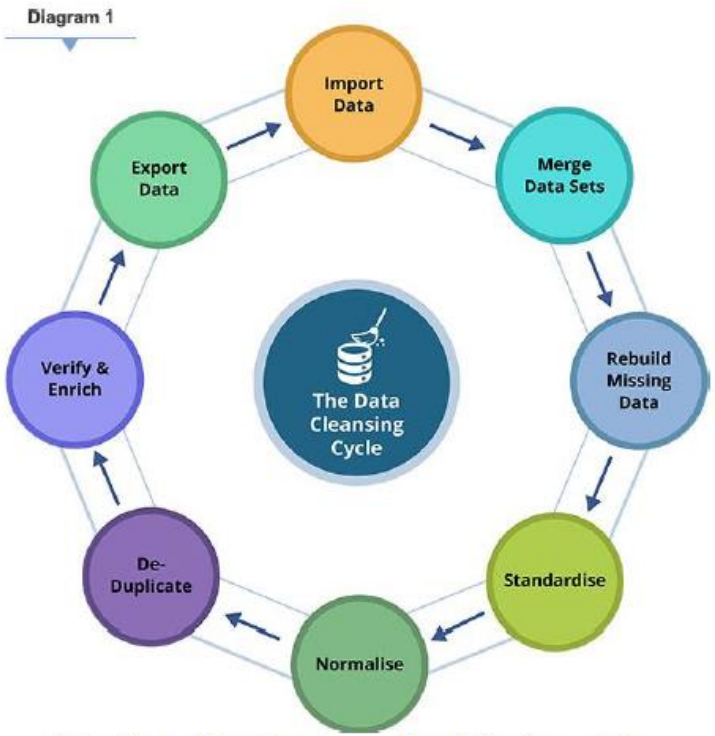
❖ 데이터 수집 Data Collection

- 데이터 수집이 데이터 처리 분석 및 모델 생성의 첫 과정
- 목적과 목표가 되는 정보를 수집하고 측정하기 위해 정의가 필요
- 문제의 정의와 문제해결을 위한 데이터 분석 기획과 시나리오가 중요
- 문제를 식별하고 탐색함으로써 정보 수집시기 및 방법을 결정
- 데이터 종류에 따라서 내부 또는 외부 , 질적 또는 양적 데이터 수집

1. 데이터 전처리

❖ 데이터 정제 Data Cleansing

- 데이터를 활용할 수 있도록 만드는 과정
- 데이터의 누락값 , 불일치 오류의 수정
- 컴퓨터가 읽을 수 없는 요소의 제거
- 숫자나 날짜 등의 형식에 대해 일관성 유지
- 적합한 파일 포맷으로 변환



https://www.dataentryoutsourced.com/blog/cxos-guide-to-marketing-and-sales-data-cleansing-and-enrichment/

1. 데이터 전처리

❖ 데이터 통합 Data Integration

- 서로 다른 출처의 여러 데이터를 결합
- 서로 다른 데이터 세트가 호환이 가능하도록 통합
- 같은 객체 , 같은 단위나 좌표로 데이터를 통합
- 링크 데이터의 핵심 목표 중 하나는 데이터 통합을 완전히 또는 거의 완전히 자동화하는 것

1. 데이터 전처리

❖ 데이터 축소 Data Reduction

- 일반적으로 데이터는 매우 크기 때문에 대용량 데이터에 대한 복잡한 데이터 분석은 실행하기 어렵거나 불가능한 경우가 많음
- 데이터 축소는 원래 용량 기준보다 작은 양의 데이터 표현결과를 얻게 되더라도 원 데이터의 완결성을 유지하기 위해 사용
- 데이터를 축소하면 데이터 분석 시 좀 더 효과적이고 원래 데이터와 거의 동일한 분석 결과를 얻어낼 수 있는 장점

https://www.cohesity.com/blog/cohesity-data-reduction-lock-stock-barrel/

1. 데이터 전처리

❖ 데이터 변환 Data Transformation

- 데이터를 한 형식이나 구조에서 다른 형식이나 구조로 변환
- 원본 데이터와 대상 데이터간에 필요한 데이터 변경 내용을 기반으로 데이터 변환이 간단하거나 복잡 할 수 있음
- 데이터 변환은 일반적으로 수동 및 자동 단계가 혼합되어 수행
- 데이터 변환에 사용되는 도구 및 기술은 변환되는 데이터의 형식 , 구조 , 복잡성 및
- 볼륨에 따라 크게 다를 수 있음

1. 데이터 전처리

❖ 데이터 셋 확인

▪ 변수 확인

- 독립/종속 변수의 정의
- 각 변수의 유형(범주형인지 연속형인지),
- 변수의 데이터 타입(Date인지, Character인지, Numeric 인지 등)

▪ RAW 데이터 확인

- **단변수 분석**

  변수 하나에 대해 기술 통계 확인을 하는 단계

  Histogram이나 Boxplot을 사용해서 평균, 최빈값, 중간값 등과 함께 각 변수들의 분포를 확인

  범주형 변수의 경우 Boxplot을 사용해서 빈도 수 분포를 체크

- **이변수 분석**

  변수 2개 간의 관계를 분석하는 단계

1. 데이터 전처리

❖ 결측값 처리 (Missing value treatment)

- **결측값이 있는 상태로 모델을 만들게 될 경우 변수간의 관계가 왜곡될수 있기 때문에 모델의 정확성이 떨어지게 됨**
- **삭제**
  - 결측값이 발생한 모든 관측치를 삭제하거나 (전체 삭제, **Listwise Deletion**),
  - 데이터 중 모델에 포함시킬 변수들 중 관측값이 발생한 모든 관측치를 삭제
- **다른 값으로 대체 (평균, 최빈값, 중간값)**

1. 데이터 전처리

❖ 결측값 처리 (Missing value treatment)

```
import numpy as np
import pandas as pd
data = {'x1' : [13,np.nan,17,20,22,21,11,56,999,64],
        'x2' : [9,555,17,11,np.nan,10,17,777,22,34],
        'y' : [20,np.nan,30,27,np.nan,32,17,20,22,899]}




df=pd.DataFrame(data)

# df_0 = df.fillna(0)
print(df)
```

```
      x1     x2     y
0   13.0    9.0   20.0
1    NaN  555.0    NaN
2   17.0   17.0   30.0
3   20.0   11.0   27.0
4   22.0    NaN    NaN
5   21.0   10.0   32.0
6   11.0   17.0   17.0
7   56.0  777.0   20.0
8  999.0   22.0   22.0
9   64.0   34.0  899.0
```

1. 데이터 전처리

❖ 결측값 처리 (Missing value treatment)

```
df_0 = df.fillna(0)
print(df_0)
```

```
     x1     x2     y
0  13.0    9.0  20.0
1   0.0  555.0   0.0
2  17.0   17.0  30.0
3  20.0   11.0  27.0
4  22.0    0.0   0.0
5  21.0   10.0  32.0
6  11.0   17.0  17.0
7  56.0  777.0  20.0
8 999.0   22.0  22.0
9  64.0   34.0 899.0
```

1. 데이터 전처리

❖ 결측값 처리 (Missing value treatment)

```
1 df1=df.fillna(method='bfill')
2 df2=df.fillna(method='backfill')
3 print(df)
4 print(df1)
```

```
     x1     x2     y
0  13.0    9.0  20.0
1   NaN  555.0   NaN
2  17.0   17.0  30.0
3  20.0   11.0  27.0
4  22.0    NaN   NaN
5  21.0   10.0  32.0
6  11.0   17.0  17.0
7  56.0  777.0  20.0
8 999.0   22.0  22.0
9  64.0   34.0 899.0
     x1     x2     y
0  13.0    9.0  20.0
1  17.0  555.0  30.0
2  17.0   17.0  30.0
3  20.0   11.0  27.0
4  22.0   10.0  32.0
5  21.0   10.0  32.0
6  11.0   17.0  17.0
7  56.0  777.0  20.0
8 999.0   22.0  22.0
9  64.0   34.0 899.0
```

1. 데이터 전처리

❖ 결측값 처리 (Missing value treatment)

```
# mean= df.mean()
a=df.fillna(df.mean())
b=df.where(pd.notnull(df), df.mean(), axis='columns')

print(a)
print(b.round(2))
```

```
           x1          x2       y
0   13.000000    9.000000   20.000
1  135.888889  555.000000  133.375
2   17.000000   17.000000   30.000
3   20.000000   11.000000   27.000
4   22.000000  161.333333  133.375
5   21.000000   10.000000   32.000
6   11.000000   17.000000   17.000
7   56.000000  777.000000   20.000
8  999.000000   22.000000   22.000
9   64.000000   34.000000  899.000
      x1      x2       y
0  13.00    9.00   20.00
1  135.89  555.00  133.38
2  17.00   17.00   30.00
3  20.00   11.00   27.00
4  22.00  161.33  133.38
5  21.00   10.00   32.00
6  11.00   17.00   17.00
7  56.00  777.00   20.00
8  999.00   22.00   22.00
9  64.00   34.00  899.00
```

1. 데이터 전처리

❖ 결측값 처리 (Missing value treatment)

```
a=df.fillna(df.median())
b=df.where(pd.notnull(df), df.median(), axis='columns')

print(a)
print(b)
```

```
     x1     x2     y
0  13.0    9.0  20.0
1  21.0  555.0  24.5
2  17.0   17.0  30.0
3  20.0   11.0  27.0
4  22.0   17.0  24.5
5  21.0   10.0  32.0
6  11.0   17.0  17.0
7  56.0  777.0  20.0
8 999.0   22.0  22.0
9  64.0   34.0 899.0
     x1     x2     y
0  13.0    9.0  20.0
1  21.0  555.0  24.5
2  17.0   17.0  30.0
3  20.0   11.0  27.0
4  22.0   17.0  24.5
5  21.0   10.0  32.0
6  11.0   17.0  17.0
7  56.0  777.0  20.0
8 999.0   22.0  22.0
9  64.0   34.0 899.0
```

1. 데이터 전처리

❖ 이상값 처리 (Outlier treatment)

- **이상값이란 데이터/샘플과 동떨어진 관측치로, 모델을 왜곡할 가능성이 있는 관측치**
- **Boxplot이나 Histogram을, 두개의 변수 간 이상값을 찾기 위해서는 Scatter plot로 확인**

1. 존재할 수 없는 값 제거

- 성별 남, 녀 외

2. 극단적인 값 제거

(1) 논리적 판단

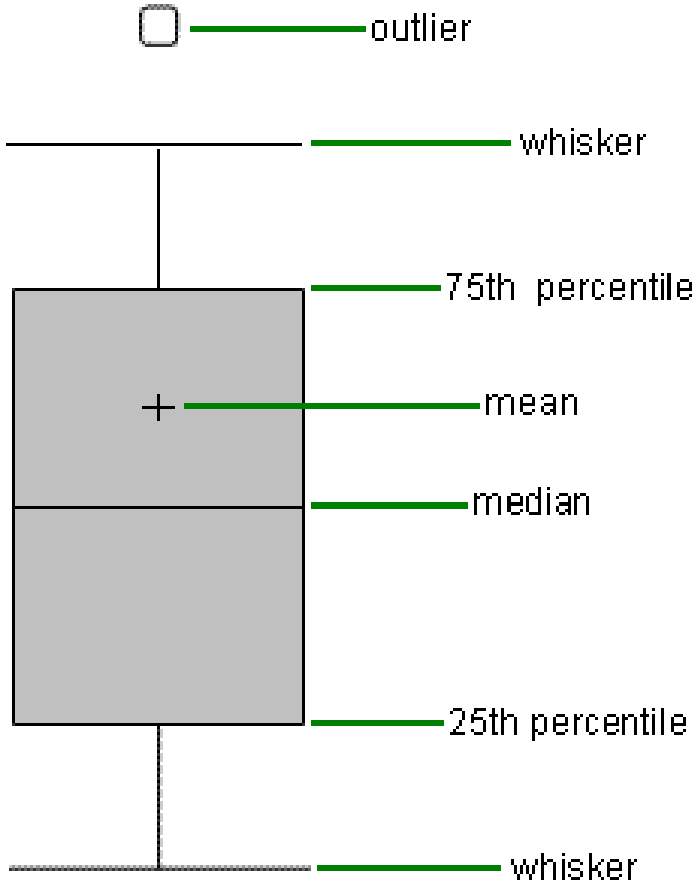
- 야구선수 타율 0.500
- 몸무게 200kg
- <데이터에 대한 도메인 지식>이 가장 중요하다.

(2) 통계적 판단- boxplot IQR 벗어난 값

1. 데이터 전처리

❖ 이상값 처리 (Outlier treatment)

- **중앙값(median): 말그대로 중앙값 50%의 위치**
  - 중앙 값은 짝수일 경우 2개가 될 수도 있고, 그것의 평균이 중앙값이 될 수도 있음
  - 홀수일 경우, 중앙값은 1개가 됨
- **박스(Box): 25%(Q1) ~75%(Q3) 까지 값**
- **수염 (whiskers): 박스의 각 모서리 (Q1, Q3)로 부터 IQR의 1.5배 내에 있는 가장 멀리 떨어진 데이터 점까지 이어져 있는 것이 수염**
- **이상치(Outlier): 수염(whiskers)보다 바깥쪽에 데이터가 존재한다면, 이것은 이상치로 분류**
- **Inter Quartile range (IQR) 이란?**
  Q3 - Q1의 값

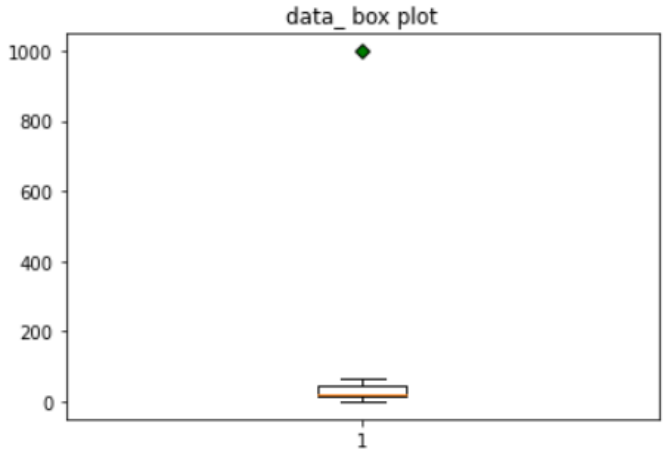


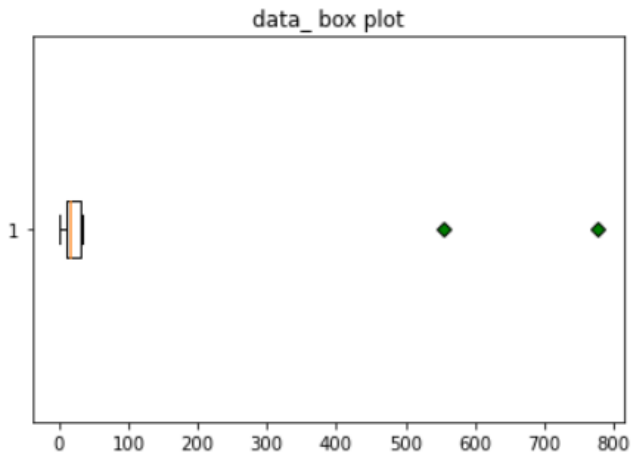
1. 데이터 전처리

❖ 이상값 처리 (Outlier treatment)

```
print(df_0)

green_diamond = dict(markerfacecolor='g', marker='D')
plt.boxplot(df_0['x1'], flierprops=green_diamond)
plt.title("data_ box plot")
plt.show()


green_diamond = dict(markerfacecolor='g', marker='D')
plt.boxplot(df_0['x2'], vert=False, flierprops=green_diamond)
plt.title("data_ box plot")
plt.show()
```

	x1	x2	y
0	13.0	9.0	20.0
1	0.0	555.0	0.0
2	17.0	17.0	30.0
3	20.0	11.0	27.0
4	22.0	0.0	0.0
5	21.0	10.0	32.0
6	11.0	17.0	17.0
7	56.0	777.0	20.0
8	999.0	22.0	22.0
9	64.0	34.0	899.0

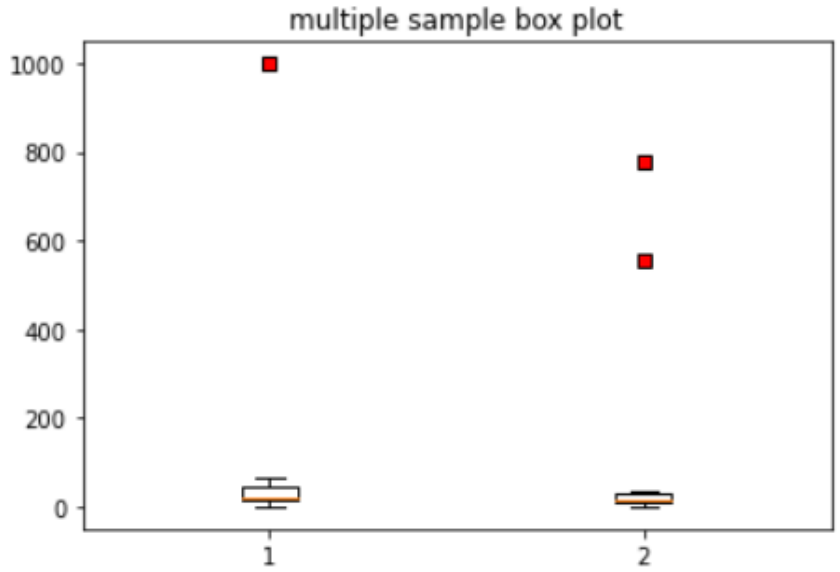




1. 데이터 전처리

❖ 이상값 처리 (Outlier treatment)

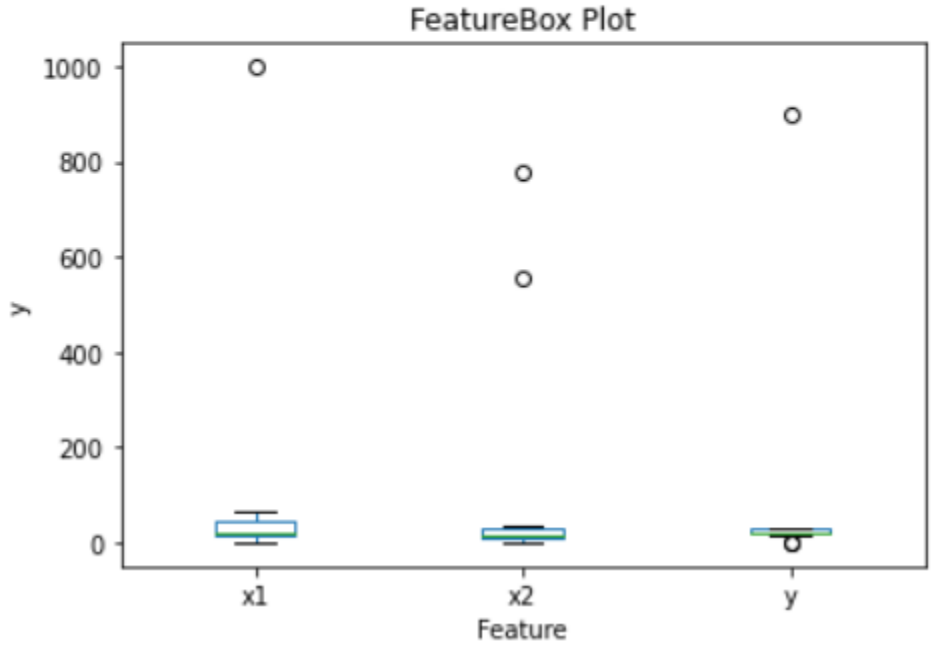
```
data = [df_0['x1'], df_0['x2']]
green_diamond = dict(markerfacecolor='r', marker='s')
plt.boxplot(data, flierprops=green_diamond)
plt.title("multiple sample box plot")
plt.show()
```



1. 데이터 전처리

❖ 이상값 처리 (Outlier treatment)

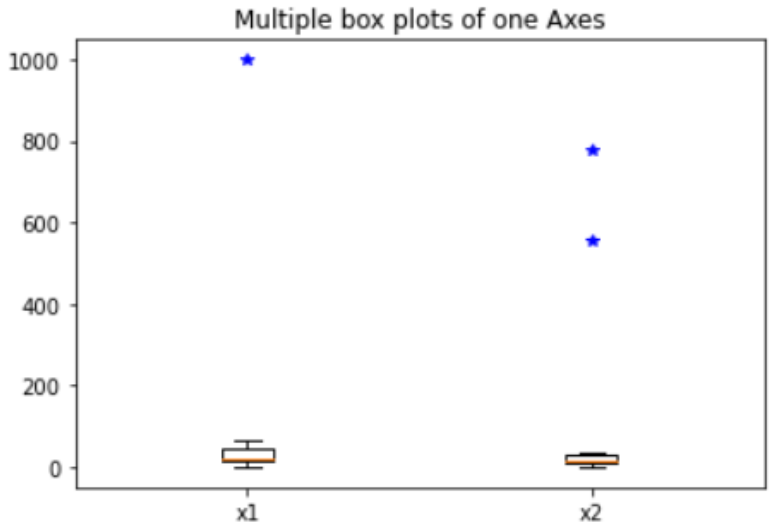
```
df_0.plot.box()
plt.title("FeatureBox Plot")
plt.xlabel("Feature")
plt.ylabel("y")
plt.show()
```



1. 데이터 전처리

❖ 이상값 처리 (Outlier treatment)

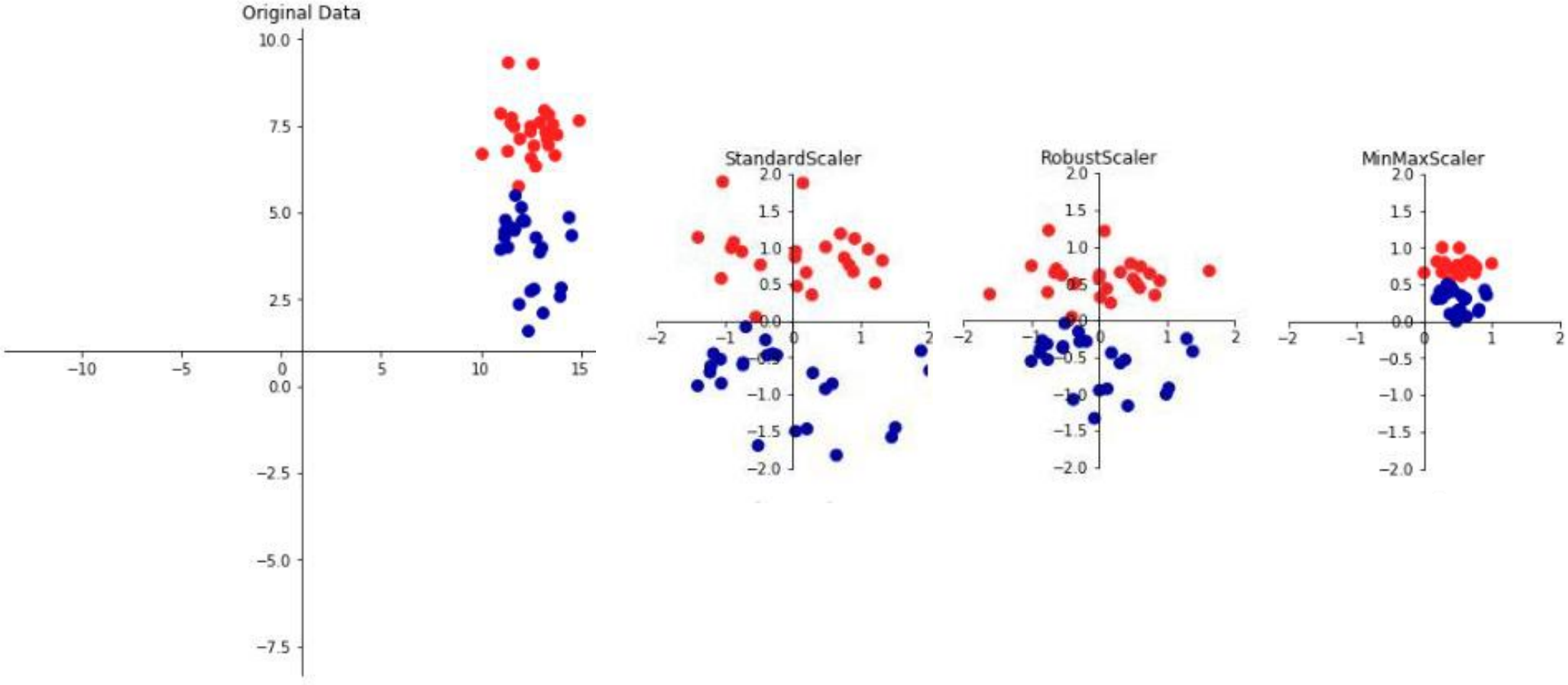
```
ig, ax = plt.subplots()

ax.boxplot([df_O['x1'], df_O['x2']], sym="b*")
plt.title('Multiple box plots of one Axes')
plt.xticks([1, 2],  ['x1', 'x2'])

plt.show()
```



1. 데이터 전처리

❖ 데이터 스케일링 (Data Scaling)

- 데이터 스케일링을 해주는 이유는 데이터의 값이 너무 크거나 혹은 작은 경우에 모델 알고리즘 학습과정에서 0으로 수렴하거나 무한으로 발산



1. 데이터 전처리

❖ 데이터 스케일링 (Data Scaling)

- **StandardScaler**
  - 각 feature의 평균을 0, 분산을 1로 변경
  - 모든 특성들이 같은 스케일
- **MinMaxScaler**
  - 모든 feature가 0과 1사이에 위치
  - 데이터가 2차원 셋일 경우, 모든 데이터는 x축의 0과 1 사이에, y축의 0과 1사이에 위치

1. 데이터 전처리

❖ 데이터 스케일링 (Data Scaling)

- **MaxAbsScaler**
  - 절대값이 0~1사이에 매핑 되도록 즉 -1~1 사이로 재조정
  - 양수 데이터로만 구성된 특징 데이터셋에서는 MinMaxScaler와 유사하게 동작하며, 큰 이상치에 민감.
- **RobustScaler**
  - 모든 특성들이 같은 크기를 갖는다는 점에서 StandardScaler와 비슷하지만,
  - 평균과 분산 대신 median과 quartile을 사용 RobustScaler는 이상치에 영향을 받지 않음

1. 데이터 전처리

❖ 데이터 스케일링 (Data Scaling)

	종류	설명
1	StandardScaler	기본 스케일. 평균과 표준편차 사용
2	MinMaxScaler	최대/최소값이 각각 1, 0이 되도록 스케일링
3	MaxAbsScaler	최대절대값과 0이 각각 1, 0이 되도록 스케일링
4	RobustScaler	중앙값(median)과 IQR(interquartile range) 사용. 아웃라이어의 영향을 최소화

1. 데이터 전처리

❖ 데이터 스케일링 (Data Scaling)

```
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import RobustScaler


scaler = StandardScaler()
scaler.fit(df_0)
X_train_1 = scaler.transform(df_0)


scaler = MinMaxScaler()
scaler.fit(df_0)
X_train_2 = scaler.transform(df_0)



scaler = MaxAbsScaler()
scaler.fit(df_0)
X_train_3 = scaler.transform(df_0)


scaler = RobustScaler()
scaler.fit(df_0)
X_train_4 = scaler.transform(df_0)


print(pd.DataFrame(X_train_1, columns=["x1","x2","y"]))
print(pd.DataFrame(X_train_2,columns=["x1","x2","y"]))
print(pd.DataFrame(X_train_3,columns=["x1","x2","y"]))
print(pd.DataFrame(X_train_4,columns=["x1","x2","y"]))


```

2. Sklearn 소개

❖ Sklearn 소개

```
import numpy as np                                  ## 기초 수학 연산 및 행렬계산
import pandas as pd                                 ## 데이터프레임 사용
from sklearn import datasets                        ## iris와 같은 내장 데이터 사용
from sklearn.model_selection import train_test_split    ## train, test 데이터 분할

from sklearn.linear_model import LinearRegression       ## 선형 회귀분석
from sklearn.linear_model import LogisticRegression     ## 로지스틱 회귀분석
from sklearn.naive_bayes import GaussianNB              ## 나이브 베이즈
from sklearn import svm                                 ## 서포트 벡터 머신
from sklearn import tree                                ## 의사결정나무
from sklearn.ensemble import RandomForestClassifier     ## 랜덤포레스트

import matplotlib.pyplot as plt                         ## plot 그릴때 사용
```

2. Sklearn 소개

❖ 선형회귀분석

- **from sklearn.linear_model import LinearRegression**

```
clf = LinearRegression()
clf.fit(x_train,y_train)  # 모수 추정
clf.coef_                 # 추정 된 모수 확인(상수항 제외)
clf.intercept_            # 추정 된 상수항 확인
clf.predic(x_test)        # 예측
clf.score(x_test, y_test) # 모형 성능 평가
```

2. Sklearn 소개

❖ 로지스틱 회귀분석

- **from sklearn.linear_model import LogisticRegression**

```
clf = LogisticRegression(solver='lbfgs').fit(x_train,y_train)
clf.predict(x_test)
clf.predict_proba(x_test)
clf.score(x_test,y_test)
```

2. Sklearn 소개

❖ 나이브 베이즈

- **from sklearn.naive_bayes import GaussianNB**

```
gnb = GaussianNB()
gnb.fit(x_train, y_train)
gnb.predict(x_test)
gnb.score(x_test, y_test)
```

2. Sklearn 소개

❖ 의사결정나무

- **from sklearn import tree**

```
clf = tree.DecisionTreeClassifier()
clf.fit(x_train, y_train)
clf.predict(x_test)
clf.predict_proba(x_test)
clf.score(x_test, y_test)
```

2. Sklearn 소개

❖ 서포트벡터머신

- from sklearn import svm

```
clf = svm.SVC(kernel='linear')
clf.fit(x_train, y_train)
clf.predict(x_test)
clf.score(x_test, y_test)
```

2. Sklearn 소개

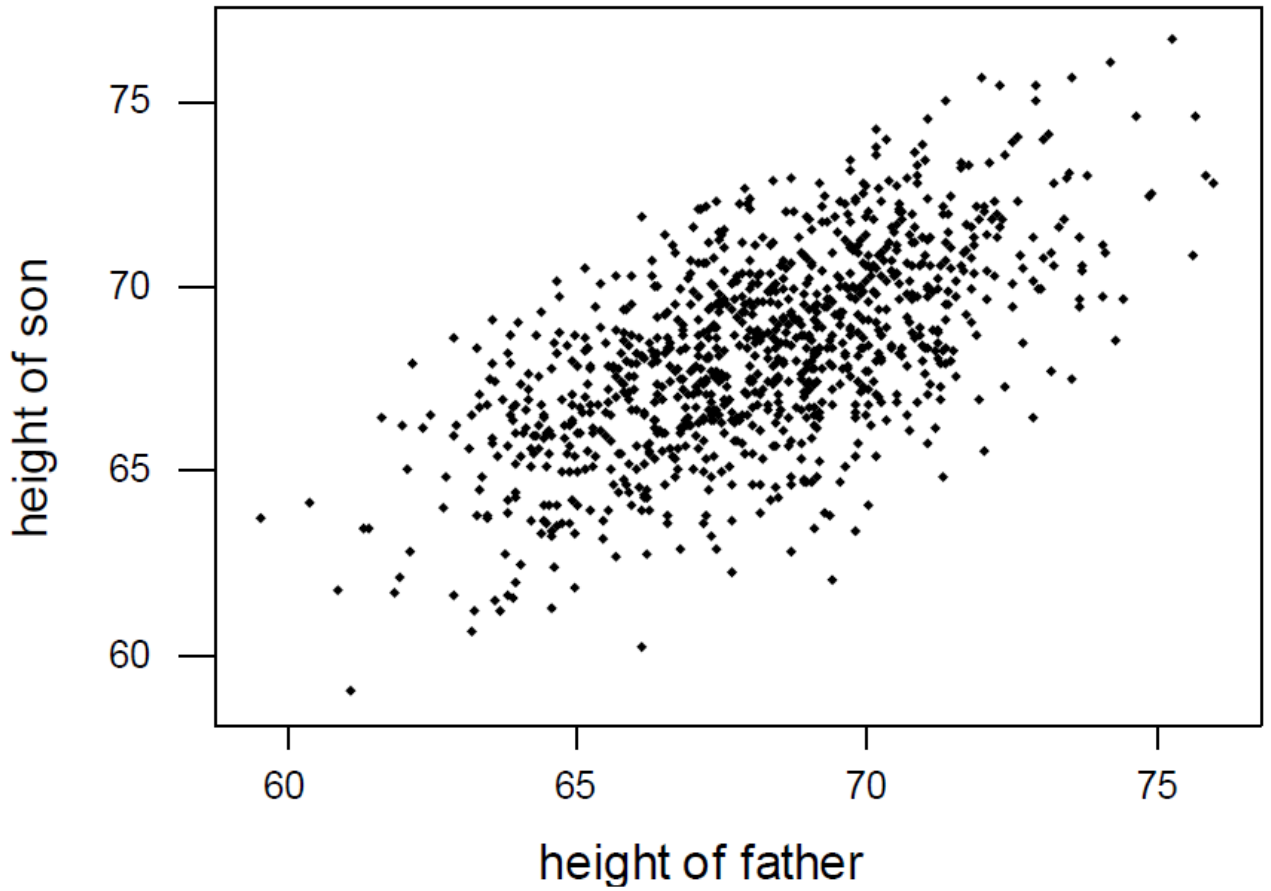
❖ 랜덤포레스트

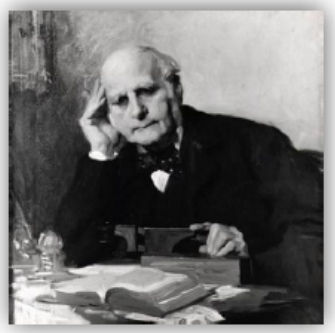
- from sklearn.ensemble import RandomForestClassifier

```
clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(x_train, y_train)
clf.feature_importances_
clf.predict(x_test)
```

3. 회귀분석

❖ 회귀분석 소개





Francis Galton
(1822~1911)

3. 회귀분석

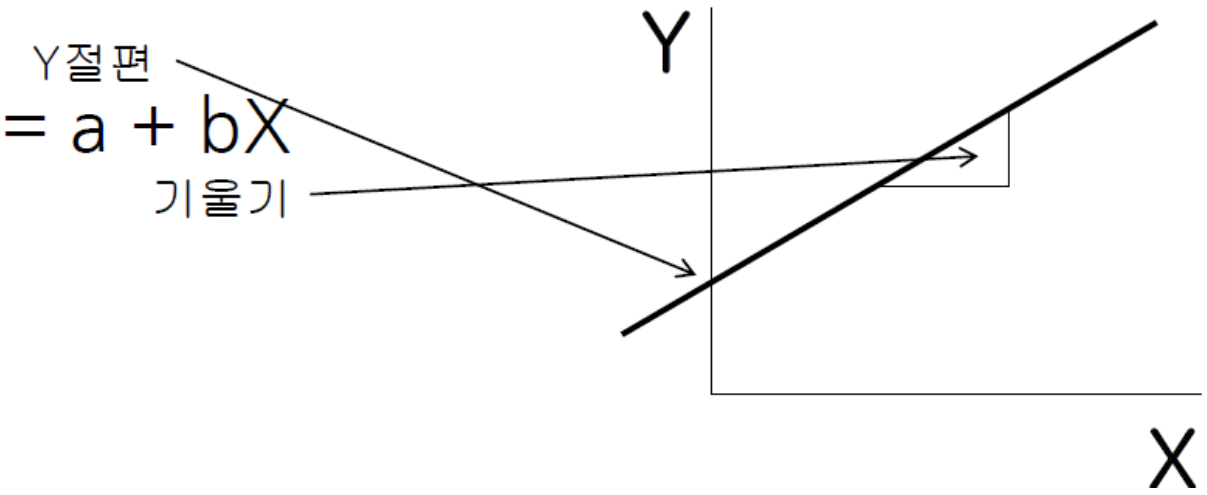
❖ 회귀분석 소개

변수들 사이의 관계를 분석하는 방법

X변수 (예측변수, Predictor), Y변수 (반응변수, Response)

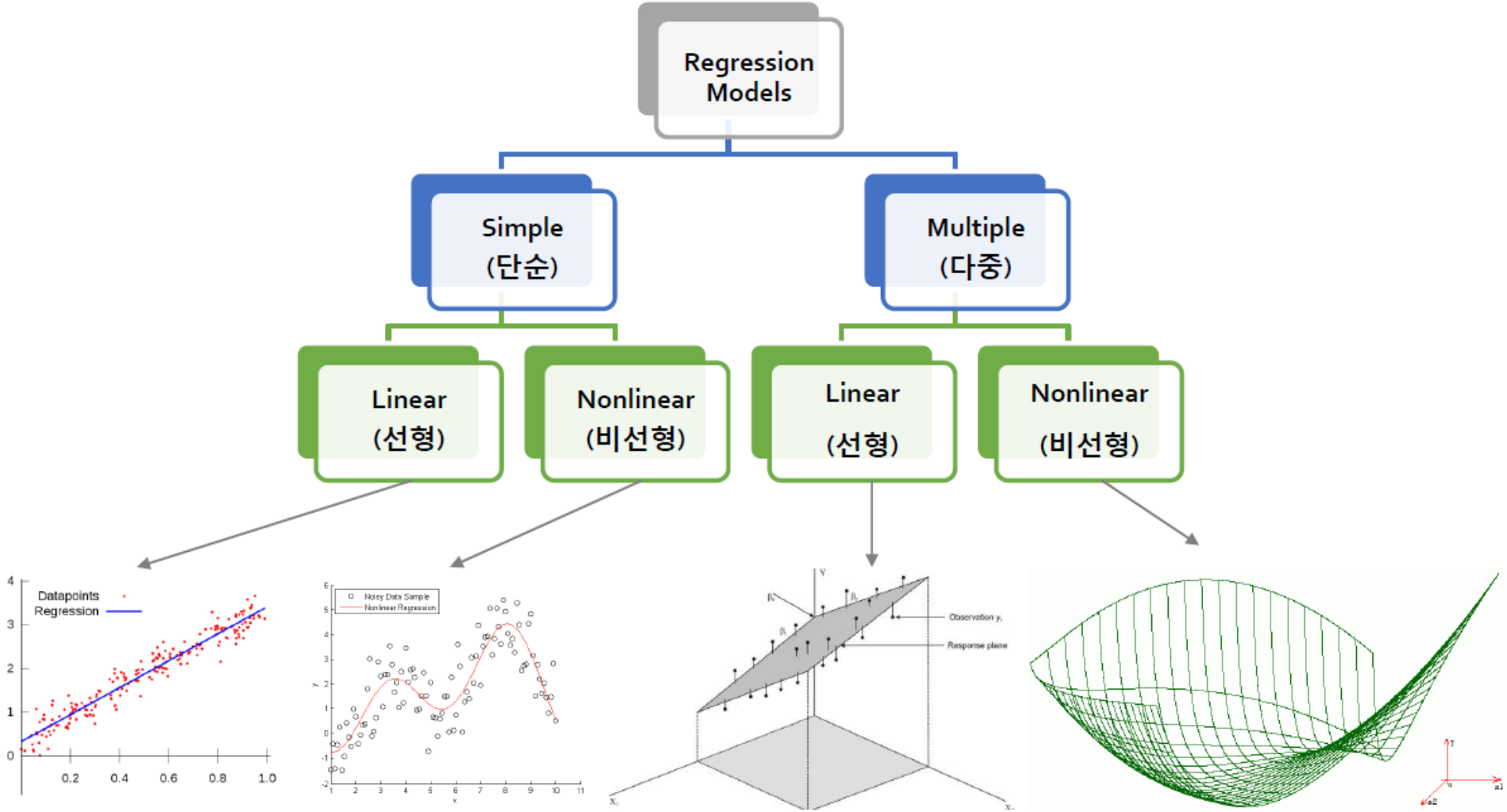
선형회귀모델: 예측변수와 반응변수 사이의 관계를
직선으로 표현

직선 식: $Y = a + bX$



3. 회귀분석

❖ 회귀분석 소개



3. 회귀분석

❖ 회귀분석 소개

- 두변수(종속변수, 독립변수)사이의 함수적 관계를 기술하는 수학적 방정식을 구하는데 사용
- 식은 독립변수의 값이 주어질 때 종속변수의 값을 추정하거나 예측하는데 사용
- 서로 영향을 주고 받는 상관관계를 갖는 두 변수 사이의 관계를 분석
- Python에서는 대표적으로 sklearn(scikit-learn) 패키지에서 Linear regression 회귀분석을 위한 함수를 제공

※종속변수(Dependent Variable)

독립변수의 특정한 값에 따른 그의 값을 예측하고자 하는 변수

※독립변수(Independent Variable)

다른 변수에 영향을 주고 그 변수의 값을 예측하려는 변수

3. 회귀분석

❖ 회귀분석 소개

종속변수	시리얼 수요	금가격	주택가격	중고차 가격
독립변수	제품의 가격	이자율	주택의 크기	차종
	5~12세 아동의 수	물가상승률	침실의 수	배기량
	경쟁회사 제품의 가격	석유가격	도로 접근성	연식
	광고투자	보석용 금에 대한 수요	주택의 위치	관리상태
	금년도 연간 매출액	산업용 금에 대한 수요	주택의 상태	옵션
	과거년도 연간 매출액	상업용 금에 대한 수요		사고여부

3. 회귀분석

❖ 산포도소개

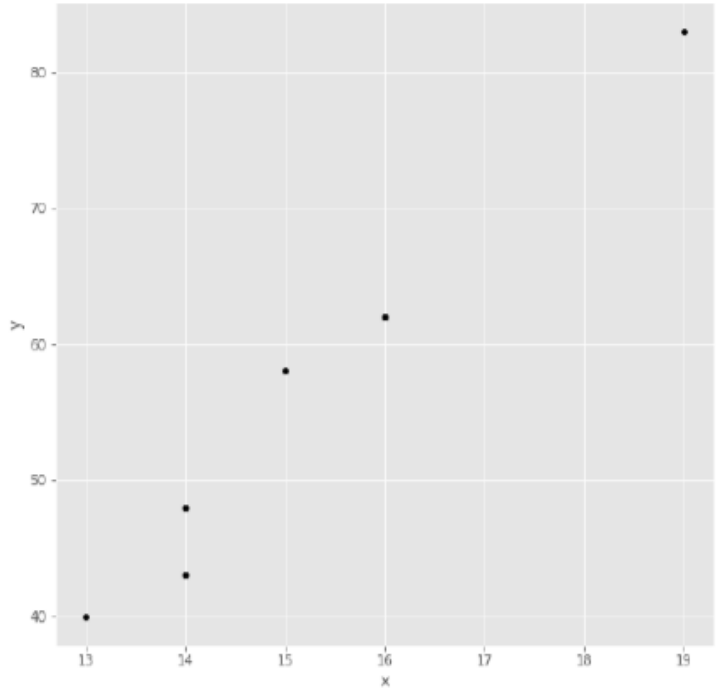
- 회귀분석을 할 때는 먼저 두 변수 사이의 관계를 대략적으로 알아보기 위하여
  산포도(Scatter Diagram)를 그림 (산포도= 산점도)
- 보통 X축에 독립변수, Y축에 종속변수를 설정하고 각 변수의 값을 나타내는 점을 도표에 나타낸 것
- 두 변수 간의 관련성 및 예측을 위한 상관분석이나 회귀분석을 할 만한 자료인지를 미리 알 수 있음

3. 회귀분석

❖ 산포도 예시

차량가격과 판매액 (단위 : 백만 원)

차종	차량 가격(x)	판매액(y)
A	13	40
B	19	83
C	16	62
D	14	48
E	15	58
F	14	43



3. 회귀분석

❖ 단일선형회귀 모델 소개

- **종속변수Y가 독립변수X와 오차항(Error Term)에 어떻게 관련되어 있는가를 나타내는 방정식**
- **오차항(Error Term)**
  - 독립변수X의 값이 주어질 때 종속변수Y의 실제값과 예측값의 차이를
- **단일 선형회귀 모델식**

$$Y_i = \alpha + \beta X_i + \varepsilon_i$$

위 식은 X라는 독립변수가 Y라는 종속변수에 주는 영향력을 식으로 나타낸 것

- X_i: 이미 알려진 독립변수의 i번째 값
- Y_i: i번째 관측치에 대한 종속변수의 값
- α: X값이변해도 Y의 변동에는 영향을 주지 않는 회귀계수
- β: X의 영향력을 크기와 부호로 나타내는 회귀계수, 독립변수 X의 기울기
- ε_i: i번째 관측치에 대한 오차항

3. 회귀분석

❖ 단일선형회귀모델의 가정

- 하나의 종속변수와 하나의 독립변수를 분석
- 독립변수X의 각 값에 대한 Y의 확률분포가 존재함
- Y의 확률분포의 평균은 X값이 변함에 따라 일정한 추세를 따라 움직임
- 종속변수와 독립변수간에는 선형함수 관계가 존재함

3. 회귀분석

❖ 회귀계수 추정

- 수집된 데이터(산포도)에 가장 적절한 회귀직선을 구하는것
- 방법으로는 최소자승법이 사용됨

❖ 최소자승법

- 잔차를 자승한값들의 합이 최소가 되도록 표본회귀식의 a와b를 구하는 방법
- 측정값을 기초로해서 적당한 제곱합을 만들고 그것을 최소로 하는값을 구하여 측정결과를 처리하는 방법

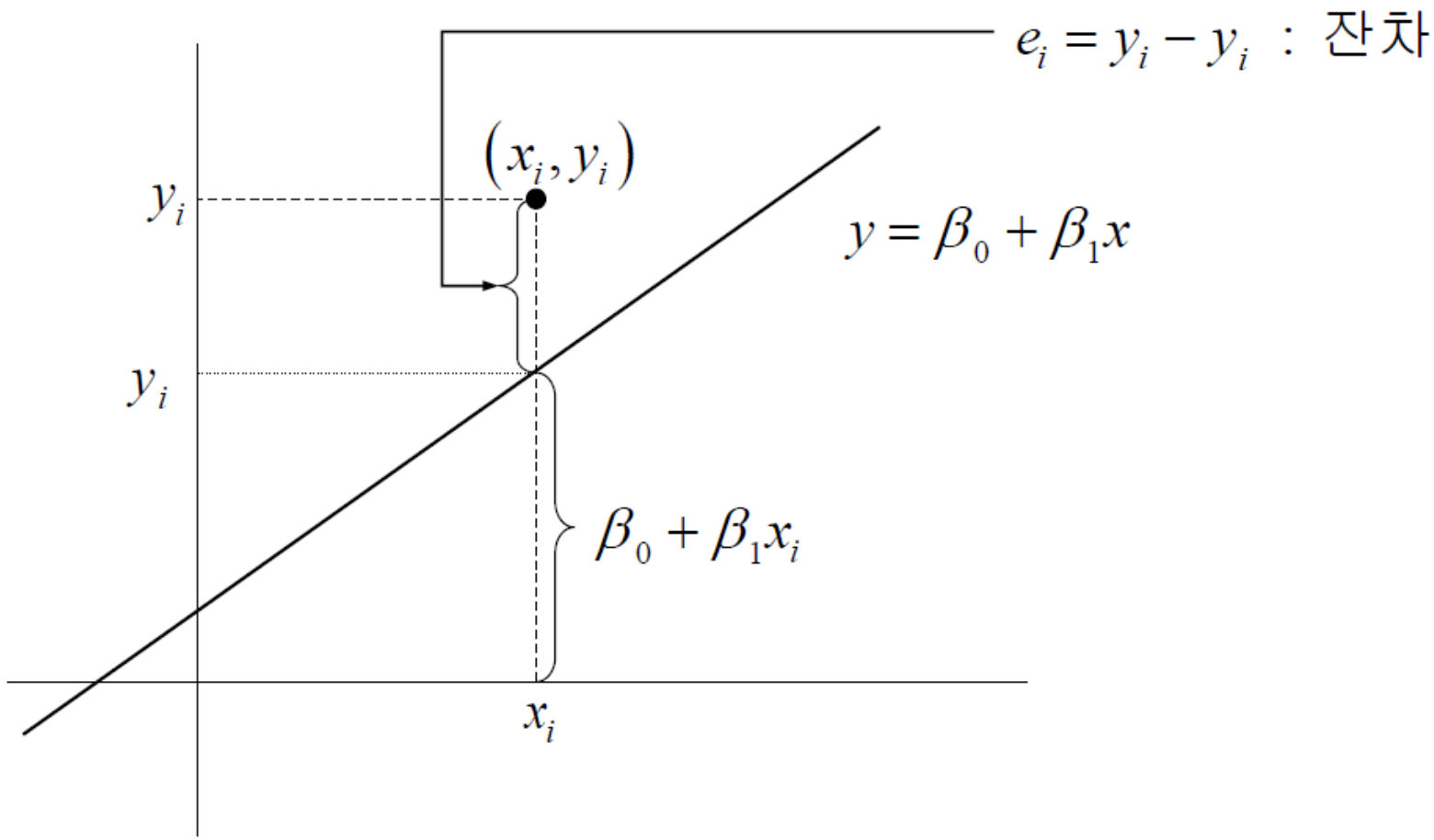
※ 잔차(Residual)

독립변수 X의 값이 주어질 때 표본회귀선의 예측값 Y과 실제값 Yi 사이에 표본오차 때문에 발생하는 차이

$$e_i = Y_i - \hat{Y}$$

3. 회귀분석

❖ 회귀계수 추정



3. 회귀분석

❖ 표본회귀계수 구하는 예

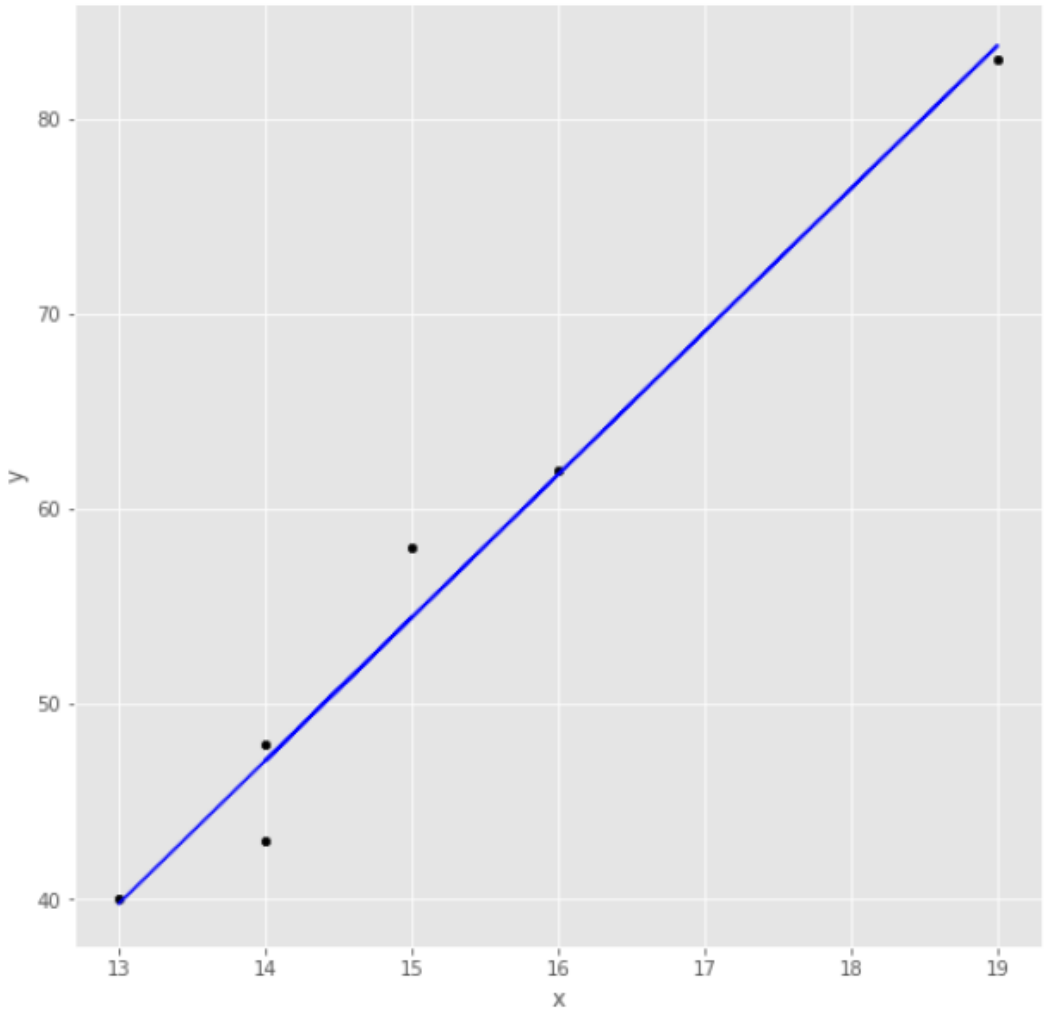
차종	차량 가격(x_i)	판매액(y_i)	x_i^2	$x_i y_i$
A	13	40	169	520
B	19	83	361	1577
C	16	62	256	992
D	14	48	196	672
E	15	58	225	870
F	14	43	196	602
합계	91	334	1403	5233

$$\bar{x} = \frac{91}{6} = 15.17, \quad \bar{y} = \frac{334}{6} = 55.67, \quad b = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2} = \frac{5233 - 6(15.17)(55.67)}{1403 - 6(15.17^2)} = 7.4$$

$$a = \bar{y} - b\bar{x} = 55.67 - 7.46(15.17) = -57.5, \quad \hat{y} = -57.5 + 7.46x$$

3. 회귀분석

❖ 표본회귀계수 구하는 예



3. 회귀분석

❖ 선형회귀식 적용예(아래e는잔차)

차량 가격(x)	판매액(y)	$\hat{y} = -57.5 + 7.46x$	$e = y - \hat{y}$
13	40	39.48	0.52
19	83	84.24	-1.24
16	62	61.86	0.14
14	48	46.94	1.06
15	58	54.4	3.6
14	43	46.94	-3.94

$$\hat{y} = -57.5 + 7.46(20) = 91.7$$

3. 회귀분석

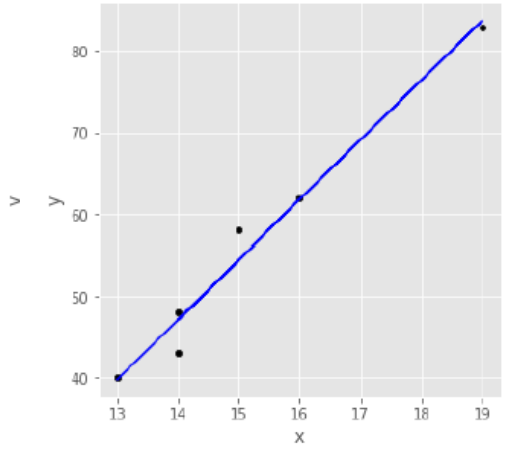
❖ 적합도 검증

- **적합도 검증이란?**
  - 표본자료를 사용하여 구한 표본회귀식이 종속변수의 값을 어느정도 정확하게 예측할 수 있는가의 정도를 검증
  - 두 변수값들이 표본회귀선 주위에 몰려있으면 종속 변수의 실제값과 예측값 차이인 잔차가 줄어들어 예측의 정확성이 높아짐
- **적합도검증방법**
  - 추정의표준오차
  - 결정계수

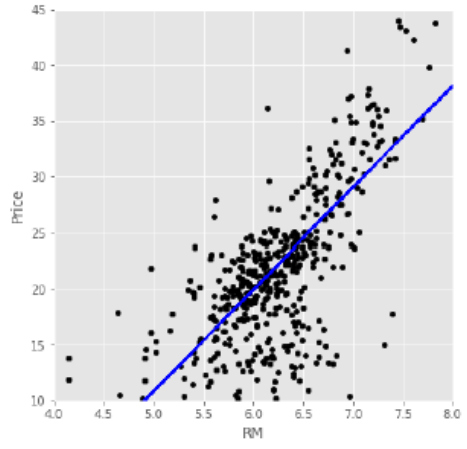
3. 회귀분석

❖ 적합도 검증

- **추정의표준오차(standard error of estimate)**
  - 값이 클수록 실제값들이 표본회귀선 주위로 널리흩어지고
    작을수록 실제값들이 표본회귀선주위로 모여들어 그 표본회귀선을 이용한 종속변수값의 예측에 대한 정확도는 높아짐
- **추정의 표준오차Se 와 표준편차의 차이**
- **추정의 표준오차: 표본들의 실제값들이 표본회귀선 주위로 흩어진 변동을 측정**
- **표준편차: 표본들의 실제값들이 평균주위로 흩어진 변동을 측정**



S_e가 작은 경우



S_e가 큰 경우

3. 회귀분석

❖ 적합도 검증

▪ 추정의 표준 오차식

- 예측값과 실제값의 차이를 잔차라고 하면 추정치의 표준오차Se는 잔차들의 표준편차를 구하기 위한식
- 실제값이 회귀식에서 얼마나 떨어져있는가를 나타내기 위함
- 추정 표준오차 계산기준은 회귀직선, 절편과 기울기의 두 통계량에의해 결정되므로 자유도는2 만큼감소

$$S_e = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{n-2}} = \sqrt{\frac{SSE}{n-2}} = \frac{\sum y^2 - a\sum y - b\sum xy}{n-2}$$

SSE : 오차제곱합

3. 회귀분석

❖ 적합도 검증

▪ 추정의 표준 오차식

x	y	x^2	y^2	xy
13	40	169	1600	520
19	83	361	6889	1577
16	62	256	3844	992
14	48	196	2304	672
15	58	225	3364	870
14	43	196	1849	602
91	334	1403	19850	5233

$$S_e = \frac{\sum y^2 - a\sum y - b\sum xy}{n-2} = \frac{19850 - (-57.5)(334) - 7.46(5233)}{6-2} = 4.205$$

3. 회귀분석

❖ 적합도 검증

▪ 총변동(total variation)

- 총 제곱합(Sum of Squares Total : SST)=
  회귀제곱합(Sum of Squares Regression:SSR)+ 잔차제곱합(Sum of Squares Error:SSE)

$$\sum (y_i - \bar{y})^2 = \sum (\hat{y}_i - \bar{y})^2 + \sum (y_i - \hat{y}_i)^2$$

$$SST = SSR + SSE$$

- SST : 실제값 yi들이 이들의 평균 y로 부터 흩어진정도
- SSR : 예측치와 실제값yi들의 평균 y의 차이의 제곱의 합
- SSE : 예측치와 실제값의 차이의 제곱의 합

3. 회귀분석

❖ 적합도 검증

- **결정계수(Coefficient of Determination)**

$$R^2 = \frac{\text{설명되는 변동}}{\text{총변동}} = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} = \frac{a\sum y_i + b\sum x_i y_i - n\bar{y}^2}{\sum {y_i}^2 - n\bar{y}^2}$$

- **결정계수는 0부터1까지의 값을 가짐**
- **표본 회귀선이 모든 자료에 완전히 적합하면 SSE=0, $R2$=1이 됨**
- **$R2$의 값이1에 가까울수록 표본회귀선으로 종속변수의 실제값yi를 예측하는데 정확성이 더 높음**

3. 회귀분석

❖ 적합도 검증

x	y	$\hat{y}$	$y-\hat{y}$	$(y-\hat{y})^2$	$y-\bar{y}$	$(y-\bar{y})^2$
13	40	39.48	0.52	0.2704	-15.67	245.44
19	83	84.24	-1.24	1.5376	27.33	747.11
16	62	61.86	0.14	0.0196	6.33	40.11
14	48	46.94	1.06	1.1236	-7.67	58.78
15	58	54.4	3.6	12.96	2.33	5.44
14	43	46.94	-3.94	15.5236	-12.67	160.44
91	334	333.86	0.14	31.4348	-0.02	1257.33
		SSE=31.4348		SST=1,257.33		

$$SSR = SST - SSE = 1{,}257.33 - 31.4348 = 1{,}225.895$$

$$R^2 = \frac{SSR}{SST} = \frac{1{,}225.895}{1{,}257.33} = 0.975$$

차량가격이 판매액 변동의 97.5%를 결정하고 다른 요인들이 나머지 2.5%의 영향을 미침

3. 회귀분석

❖ 적합도 검증

- **잔차(Residuals)**
  - 회귀분석 모델의 예측값 y와 실제값yi 사이차이

$$e_i = y_i - \hat{y}$$

- **MSE(Mean Squared Error)**
  - 평균 제곱오차
  - 회귀선과 모델 예측값사이의 오차를 사용
  - 오차를 제곱한값들의 평균

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

- **RMSE(Root Mean Squared Error)**
  - MSE에서 구한값에 루트를 적용한값

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

3. 회귀분석

❖ 다중선형회귀분석이란?

▪ 다중선형회귀분석 소개

- 두개 이상의 독립변수들과 하나의 종속 변수의 관계를 분석하는 방법
- 단순 회귀분석을 확장한 것

▪ 다중선형회귀분석가정

- 오차항 εi는 n개의 독립변수$X1,X2,\ldots,Xn$의 각각에 독립적. 즉, 독립변수와 관련된 측정오차는 존재하지 않음
- 오차항의 기대 값은0이며 일정한분산을 갖는 정규분포를 이룸
- 어떤 두 오차도 서로 상관이 없다. 즉, 그들의 공분산은 0임
- 독립변수들은 서로 선형함수로 완전히 관련되어 있지 않음

3. 회귀분석

❖ 다중선형회귀분석이란?

- **다중선형회귀모델식**

$$y_i = \alpha + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_{k-1} x_{k-1i} + \beta_k x_{ki} + \varepsilon_i$$

- 위식은 x라는 독립변수들이 y라는 종속변수에 주는 영향력을 식으로 나타낸것
- k: 1~k까지 에상수값(여러 개 독립변수중 n번째 독립변수를 뜻함)
- yi: i번째 관측치에 대한 종속변수의값
- xi: 이미 알려진 독립변수의 i번째값
- α: X값이변해도Y의 변동에는 영향을 주지않는 회귀계수
- β: X의 영향력을 크기와 부호로 나타내주는 회귀계수, 독립변수 X의 기울기
- εi: i번째 관측치에 대한 오차항

3. 회귀분석

❖ 다중선형회귀분석이란

- 수집된 데이터(산포도)에 가장 적절한 회귀 직선을 구하는것
- 방법으로는 최소 자승법이 사용됨

❖ 최소자승법

- 잔차를 자승한 값들의 합이 최소가 되도록 표본회귀식의 a와b를 구하는 방법

$$\sum {e_i}^2 = \sum (Y_i - \hat{Y}_i)^2 = \sum (Y_i - a - b_1 X_{1i} - b_2 X_{2i})^2$$

3. 회귀분석

❖ 다중선형회귀분석이란

▪ 표본회귀계수구하는예

차량가격,광고비와 판매액 (단위 : 백만 원)

차종	차량 가격(x_{1i})	광고비(x_{2i})	판매액(y)
A	13	9	20
B	18	7	22
C	17	17	30
D	20	11	27
E	22	8	35
F	21	10	32

3. 회귀분석

❖ 다중선형회귀분석이란

▪ 표본회귀계수 구하는 예

x_{1i}	x_{2i}	y_i	x_{1i}^*	x_{2i}^*	y_i^*	x_{1i}^{*2}	x_{2i}^{*2}	$x_{1i}^* x_{2i}^*$	$x_{1i}^* y_i^*$	$x_{2i}^* y_i^*$
13	9	20	-5.5	-1.33	-7.67	30.25	1.78	7.33	42.17	10.22
18	7	22	-0.5	-3.33	-5.67	0.25	11.11	1.67	2.83	18.89
17	17	30	-1.5	6.67	2.33	2.25	44.44	-10	-3.5	15.56
20	11	27	1.5	0.67	-0.67	2.25	0.44	1	-1	-0.44
22	8	35	3.5	-2.33	7.33	12.25	5.44	-8.17	25.67	-17.11
21	10	32	2.5	-0.33	4.33	6.25	0.11	-0.83	10.83	-1.44
111	62	166	0	0	0	53.5	63.33	-9	77	25.67
$\bar{x}_1 = 18.5,$		$\bar{x}_2 = 10.33,$		$\bar{y} = 27.67 x_{1i}^* = x_{1i} - \bar{x}_1,$		$x_{2i}^* = x_{2i} - \bar{x}_2,$		$y_i^* = y_i - \bar{y}$		

$$b_1 = \frac{\sum x_{2i}^{*2} \sum y_i^* x_{1i}^* - \sum x_{1i}^* x_{2i}^* \sum y_i^* x_{2i}^*}{\sum x_{1i}^{*2} \sum x_{2i}^{*2} - (\sum x_{1i}^* x_{2i}^*)^2} = \frac{63.33(77) - (-9)(25.67)}{53.5(63.33) - (-9)^2} = 1.5444$$

$$b_2 = \frac{\sum x_{1i}^{*2} \sum y_i^* x_{2i}^* - \sum x_{1i}^* x_{2i}^* \sum y_i^* x_{1i}^*}{\sum x_{1i}^{*2} \sum x_{2i}^{*2} - (\sum x_{1i}^* x_{2i}^*)^2} = \frac{53.5(25.67) - (-9)(77)}{53.5(63.33) - (-9)^2} = 0.6248$$

$$a = \bar{y} - b_1 \bar{x}_1 - b_2 \bar{x}_2 = 27.67 - 1.5444(18.5) - 0.6248(10.33) = -7.3537$$

$$\hat{y}_i = -7.3537 + 1.5444 x_{1i} + 0.6248 x_{2i}$$

3. 회귀분석

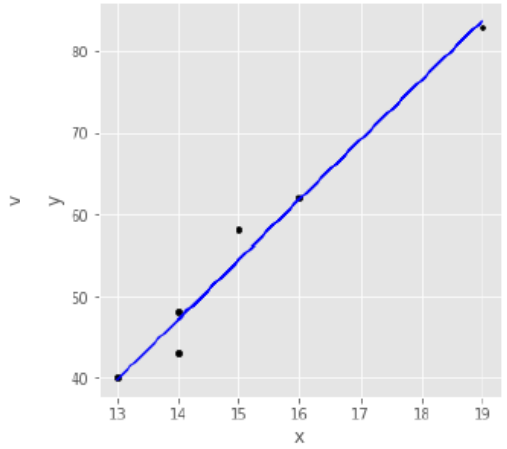
❖ 적합도 검증

- **적합도 검증이란?**
  - 표본자료를 사용하여 구한 표본회귀식이 종속변수의 값을 어느정도 정확하게 예측할 수 있는가의 정도를 검증
  - 두 변수값들이 표본회귀선 주위에 몰려있으면 종속 변수의 실제값과 예측값 차이인 잔차가 줄어들어 예측의 정확성이 높아짐
- **적합도검증방법**
  - 추정의표준오차
  - 결정계수

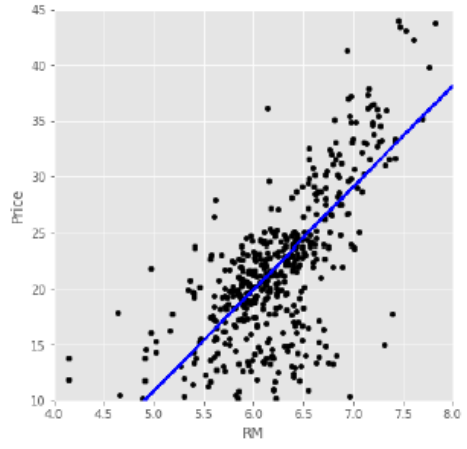
3. 회귀분석

❖ 적합도 검증

- **추정의표준오차(standard error of estimate)**
  - 값이 클수록 실제값들이 표본회귀선 주위로 널리흩어지고
    작을수록 실제값들이 표본회귀선주위로 모여들어 그 표본회귀선을 이용한 종속변수값의 예측에 대한 정확도는 높아짐
- **추정의 표준오차Se 와 표준편차의 차이**
- **추정의 표준오차: 표본들의 실제값들이 표본회귀선 주위로 흩어진 변동을 측정**
- **표준편차: 표본들의 실제값들이 평균주위로 흩어진 변동을 측정**



S_e가 작은 경우



S_e가 큰 경우

3. 회귀분석

❖ 적합도 검증

- **추정의 표준오차 예시**

y_i	$\hat{y}_i$	e_i	e_i^2
20	18.34	1.66	2.76
22	24.81	-2.81	7.9
30	29.51	0.49	0.24
27	30.4	-3.4	11.56
35	31.61	3.39	11.49
32	31.32	0.68	0.46
166	147.65	-1.65	31.65

$$S_e = \sqrt{\frac{\sum e_i^2}{n-(k+1)}} = \sqrt{\frac{31.65}{6-3}} = \sqrt{10.55} = 3.2481$$

3. 회귀분석

❖ 적합도 검증

▪ 총변동(total variation)

- 총 제곱합(Sum of Squares Total : SST)=
  회귀제곱합(Sum of Squares Regression:SSR)+ 잔차제곱합(Sum of Squares Error:SSE)

$$\sum (y_i - \bar{y})^2 = \sum (\hat{y}_i - \bar{y})^2 + \sum (y_i - \hat{y}_i)^2$$

$$SST = SSR + SSE$$

- SST : 실제값 yi들이 이들의 평균 y로 부터 흩어진정도
- SSR : 예측치와 실제값yi들의 평균 y의 차이의 제곱의 합
- SSE : 예측치와 실제값의 차이의 제곱의 합

3. 회귀분석

❖ 적합도 검증

- **결정계수(Coefficient of Determination)**

$$R^2 = \frac{\text{설명되는 변동}}{\text{총변동}} = \frac{SSR}{SST} = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2} = 1 - \frac{SSE}{SST}$$

- **결정계수는 0부터1까지의 값을 가짐**
- **표본 회귀선이 모든 자료에 완전히 적합하면 SSE=0, $R2$=1이 됨**
- **$R2$의 값이1에 가까울수록 표본회귀선으로 종속변수의 실제값yi를 예측하는데 정확성이 더 높음**

3. 회귀분석

❖ 적합도 검증

▪ 조정결정계수

- 독립변수의 수가 증가하면 모형도 복잡해지고, 독립변수의 상관관계가 높아져서 독립 변수들간에 서로영향을 미치는 다중공선성의 문제가 발생하기 때문에 상대적인 조정이 필요.

$$R_a^2 = 1 - (1 - R^2)(\frac{n-1}{n-k-1})$$

▪ 조정결정계수

- 독립변수의 수가 증가하면 모형도 복잡해지고, 독립변수의 상관관계가 높아져서 독립 변수들간에 서로영향을 미치는 다중공선성의 문제가 발생하기 때문에 상대적인 조정이 필요

3. 회귀분석

❖ 적합도 검증

▪ 조정결정계수

y_i	$\hat{y}_i$	$y_i - \bar{y}$	$(y_i - \bar{y})^2$	$\hat{y}_i - \bar{y}$	$(\hat{y}_i - \bar{y})^2$
20	18.34	-7.67	58.8289	-9.33	87.0489
22	24.81	-5.67	32.1489	-2.86	8.1796
30	29.51	2.33	5.4289	1.84	3.3856
27	30.4	-0.67	0.4489	2.73	7.4529
35	31.61	7.33	53.7289	3.94	15.5236
32	31.32	4.33	18.7489	3.65	13.3225
166	147.65	0	169.3334	119.98	134.91

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2} = \frac{134.91}{169.3334} = 0.7967$$

$$R_a^2 = 1 - (1 - R^2)\left(\frac{n-1}{n-k-1}\right) = 1 - (1 - 0.7967)\frac{6-1}{6-2-1} = 0.6612$$

여기서$Ra2$과$R2$은판매액의변동의79.67%와66.12%는각각차량가격과광고비에의하여설명될수있다는것을의미

3. 회귀분석

❖ 다중회귀분석 예시 1

In [1]:
```
from sklearn import linear_model
import numpy as np
import pandas as pd

```

In [2]:
```
data = {'x1' : [13,18,17,20,22,21],
        'x2' : [9,7,17,11,8,10],
        'y' : [20,22,30,27,35,32]}
```

In [4]:
```
import pandas as pd

data =pd.DataFrame(data)
X= data[['x1','x2']]
y=data['y']

data
```

Out[4]:

	x1	x2	y
0	13	9	20
1	18	7	22
2	17	17	30
3	20	11	27
4	22	8	35
5	21	10	32

변수설명

- data : 임의로'x1', 'x2'와'y'이름을 가진 리스트 형식에 데이터를 생성하여 pandas.DataFrame형으로 변경후 저장한 변수

- X : 'data'변수에'x1', x2' 데이터만 저장한 변수 (독립변수)
- y : 'data'변수에'y' 데이터만 저장한변수(종속변수)

함수설명

-pandas.DataFrame() : 2차원의 수정가능한 테이블형태의 데이터구조를 만드는 함수

3. 회귀분석

❖ 다중회귀분석 예시 1

```
linear_regression = linear_model.LinearRegression()
linear_regression.fit(X=pd.DataFrame(X), y=y)
prediction = linear_regression.predict(X=pd.DataFrame(X))
print('Bintercept_: ', linear_regression.intercept_)
print('acoef_: ',linear_regression.coef_)
```

```
Bintercept_:  -7.359201773835938
acoef_:  [1.5443459  0.62472284]
```

```
residuals = y - prediction
residuals.describe()
```

```
5.730691056910575
2.3938861829482567
```

3. 회귀분석

❖ 다중회귀분석 예시 1

```
1 SSE = (residuals**2).sum()
2 SST = ((y-y.mean())**2).sum()
3 R_squared = 1 - (SSE/SST)
4 print('R_squared: ', R_squared)
```

```
R_squared:  0.796944017668523
```

3. 회귀분석

❖ 다중회귀분석 예시 1

```
from sklearn.metrics import mean_squared_error

from math import sqrt

mse= mean_squared_error(y,prediction )
rmse=sqrt(mse)

print('score = ' , linear_regression.score(X=pd.DataFrame(X),y=y))
print('mean_squared_error = ' , mse)
print('RMSE = ' , rmse)

```

```
score =  0.796944017668523
mean_squared_error =  5.730691056910575
RMSE =  2.3938861829482567
```

3. 회귀분석

❖ 다중회귀분석 예시 2

```
from sklearn import datasets
boston_house_prices = datasets.load_boston()

# 로드한 boston 전체 데이터에 key 값을 출력
print(boston_house_prices.keys())
# boston 전체 데이터 중 data에 대한 전체 행, 열 길이를 출력
print(boston_house_prices.data.shape)
# boston 데이터에 컬럼 이름을 출력
print(boston_house_prices.feature_names)
```

```
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename'])
(506, 13)
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

```
print(boston_house_prices.DESCR)
```

3. 회귀분석

❖ 다중회귀분석 예시 2

```
1 import pandas as pd
2 X = pd.DataFrame(boston_house_prices.data)
3 X.tail()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

```
1 X.columns = boston_house_prices.feature_names
2 X.tail()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

3. 회귀분석

❖ 다중회귀분석 예시 2

```
1  X['Price'] =  boston_house_prices.target
2
3  y=X.pop('Price')
4
5  print(y)
6  X.tail()
7
```

```
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: Price, Length: 506, dtype: float64
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

3. 회귀분석

❖ 다중회귀분석 예시 2

```
1  data = {'x1' : [13,18,17,20,22,21],
2          'x2' : [9,7,17,11,8,10],
3          'y' : [20,22,30,27,35,32]}
4
5  y_pop_sample=data.pop('x2')
6  print(y_pop_sample)
```

Series.pop(item)
DataFrame.pop(item)

```
[9, 7, 17, 11, 8, 10]
```

```
1  fruit = ['사과', '배', '체리','바나나']
2  fru= fruit.pop(0)
3
4  print(fruit)
5  print(fru)
```

```
['배', '체리', '바나나']
사과
```

3. 회귀분석

❖ 다중회귀분석 예시 2

```
linear_regression = linear_model.LinearRegression()
linear_regression.fit(X=pd.DataFrame(X), y=y)
prediction = linear_regression.predict(X=pd.DataFrame(X))
print('a value: ', linear_regression.intercept_)
print('b value: ',linear_regression.coef_)
```

```
a value:  36.45948838509001
b value:  [-1.08011358e-01  4.64204584e-02  2.05586264e-02  2.68673382e+00
 -1.77666112e+01  3.80986521e+00  6.92224640e-04 -1.47556685e+00
  3.06049479e-01 -1.23345939e-02 -9.52747232e-01  9.31168327e-03
 -5.24758378e-01]
```

```
residuals = y - prediction
residuals.describe()

```

```
count    5.060000e+02
mean     4.296958e-15
std      4.683822e+00
min     -1.559447e+01
25%     -2.729716e+00
50%     -5.180489e-01
75%      1.777051e+00
max      2.619927e+01
Name: Price, dtype: float64
```

3. 회귀분석

❖ 다중회귀분석 예시 2

```
SSE = (residuals**2).sum()
SST = ((y-y.mean())**2).sum()
R_squared = 1 - (SSE/SST)
print('R_squared: ', R_squared)
```

```
R_squared:  0.7406426641094094
```

```

mse= mean_squared_error(y,prediction )

print(mse)

rmse=sqrt(mse)

print(rmse)

print('score = ' , linear_regression.score(X=pd.DataFrame(X),y=y))
print('mean_squared_error = ' , mse)
print('RMSE = ' , rmse)
```

```
21.894831181729206
4.679191295697282
score =  0.7406426641094094
mean_squared_error =  21.894831181729206
RMSE =  4.679191295697282
```

3. 회귀분석

❖ 다중회귀분석 예시 2

```
1  import numpy as np
2
3  X = pd.DataFrame(np.c_[X['LSTAT'], X['RM']], columns = ['LSTAT','RM'])
4  Y = y
5
6  print(X)
7  print(Y)
```

```
     LSTAT     RM
0     4.98  6.575
1     9.14  6.421
2     4.03  7.185
3     2.94  6.998
4     5.33  7.147
..     ...    ...
501   9.67  6.593
502   9.08  6.120
503   5.64  6.976
504   6.48  6.794
505   7.88  6.030

[506 rows x 2 columns]
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
```

두 개의 1차원 배열을 칼럼으로 세로로 붙여서 2차원 배열 만들기
(Stack 1-D arrays as columns into a 2-D array)

```
np.c_[a, b]
np.column_stack([a, b])
np.concatenate((c.T, d.T), axis = 1) # for 2D~ array
```

3. 회귀분석

❖ 다중회귀분석 예시 2

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.4, random_state=1)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(303, 2)
(203, 2)
(303,)
(203,)
```

3. 회귀분석

❖ 다중회귀분석 예시 2

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

reg_1 = LinearRegression()
reg_1.fit(X_train, Y_train)

# model evaluation for training set
y_train_predict = linear_regression.predict(X_train)
rmse = (np.sqrt(mean_squared_error(Y_train, y_train_predict)))
r2 = r2_score(Y_train, y_train_predict)

print("The model performance for training set")
print("--------------------------------------")
print('RMSE is {}'.format(rmse))
print('R2 score is {}'.format(r2))
print("₩n")
```

```
The model performance for training set
--------------------------------------
RMSE is 5.289697660346464
R2 score is 0.650374492157888
```

3. 회귀분석

❖ 다중회귀분석 예시 2

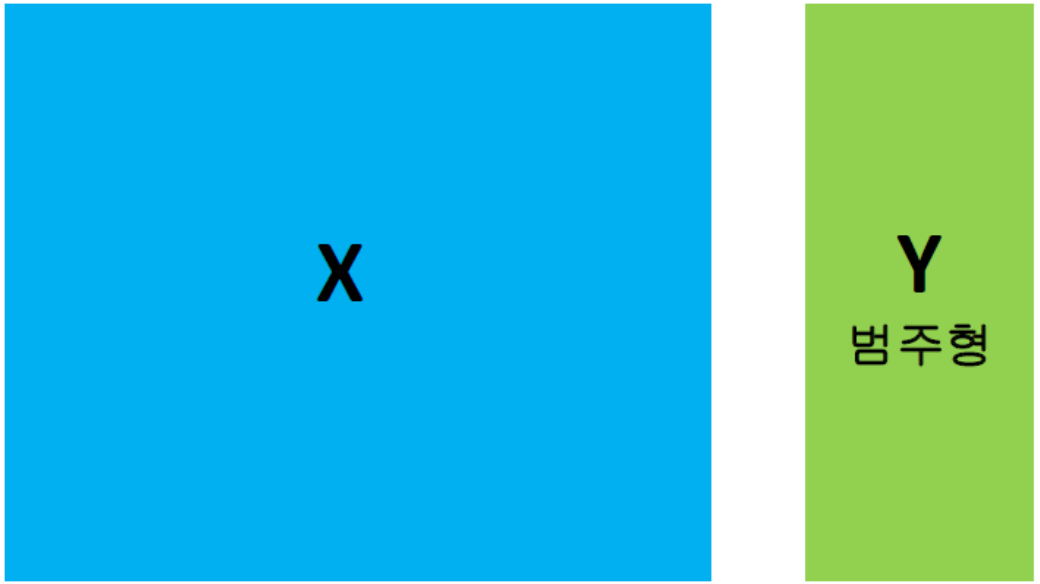
```
19 # model evaluation for testing set
20 y_test_predict = linear_regression.predict(X_test)
21 rmse = (np.sqrt(mean_squared_error(Y_test, y_test_predict)))
22 r2 = r2_score(Y_test, y_test_predict)
23
24 print("The model performance for testing set")
25 print("--------------------------------")
26 print('RMSE is {}'.format(rmse))
27 print('R2 score is {}'.format(r2))
```

```
The model performance for testing set
--------------------------------
RMSE is 5.893914836058081
R2 score is 0.6153562498551477
```

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- 범주형 반응변수
- •이진변수(e.g., 반응 변수값 0 or 1)
- •멀티변수(e.g., 반응 변수값 1 or 2 or 3 이상)
- •일반회귀분석과는 다른방식으로 접근해야될 필요성



3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- **일반 회귀모형을 반응변수가 범주일때로 확장**
- **새로운 관측치가 왔을때 이를 기존의 범주 중 하나로 분류하는것이 목적**

- 제품이 불량인지 양품인지 분류
- 고객이 이탈 고객인지 잔류 고객인지 분류
- 카드거래가 정상인지 사기인지 분류

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- **33명의 나이와 혈압**

Age	SBP	Age	SBP	Age	SBP
22	131	41	139	52	128
23	128	41	171	54	105
24	116	46	137	56	145
27	106	47	111	57	141
28	114	48	115	58	153
29	123	49	133	59	157
30	117	49	128	63	155
32	122	50	183	67	176
33	99	51	130	71	172
35	121	51	133	77	178
40	147	51	144	81	217

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- 연속형 변수가 아닌 이진형(Binary) 변수인 Cancer Diagnosis (CD)를 사용

Age	CD	Age	CD	Age	CD
22	0	40	0	54	0
23	0	41	1	55	1
24	0	46	0	58	1
27	0	47	0	60	1
28	0	48	0	60	0
30	0	49	1	62	1
30	0	49	0	65	1
32	0	50	1	67	1
33	0	51	0	71	1
35	1	51	1	77	1
38	0	52	0	81	1

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

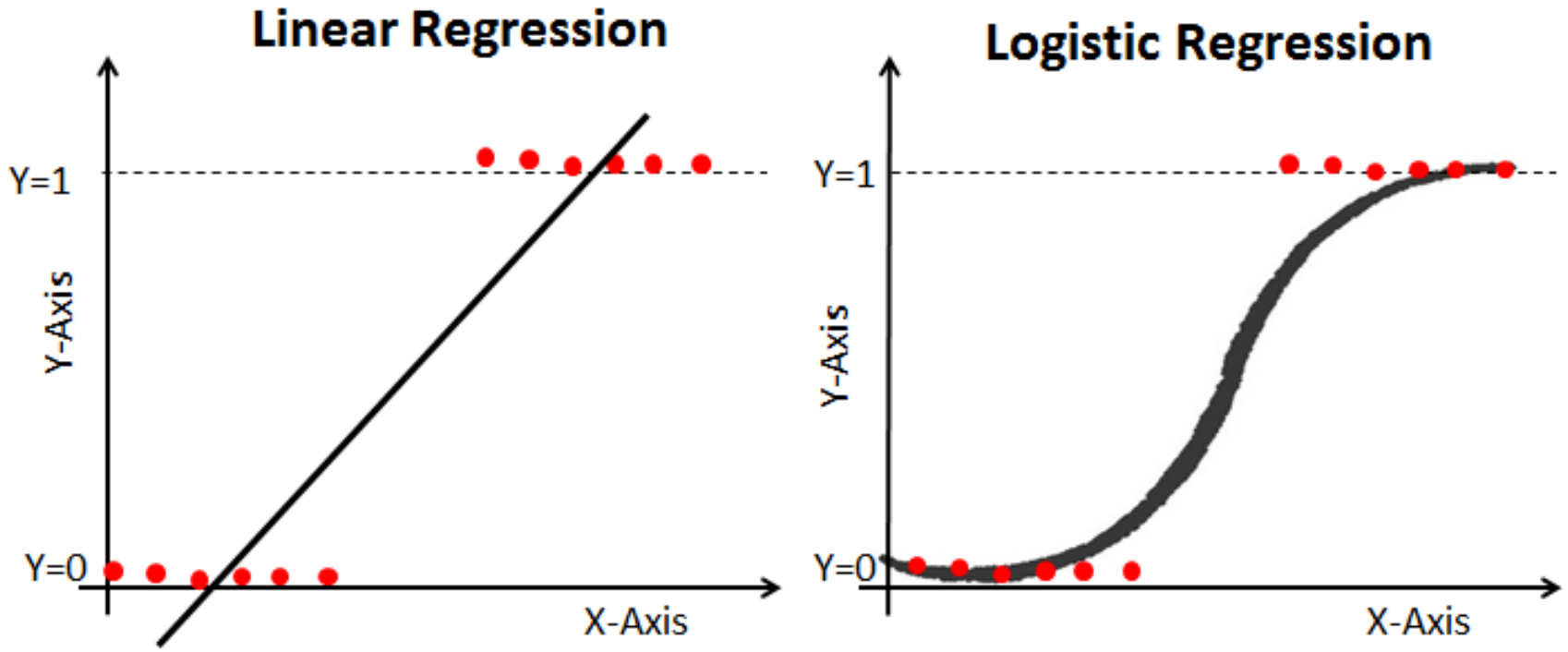
- 연속형 변수가 아닌 이진형(Binary) 변수인 Cancer Diagnosis (CD)를 사용

		CH Disease	
Age group	# in group	#	%
20 - 29	5	0	0
30 - 39	6	1	17
40 - 49	7	2	29
50 - 59	7	4	57
60 - 69	5	4	80
70 - 79	2	2	100
80 - 89	1	1	100

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- 회귀분석과의 차이점



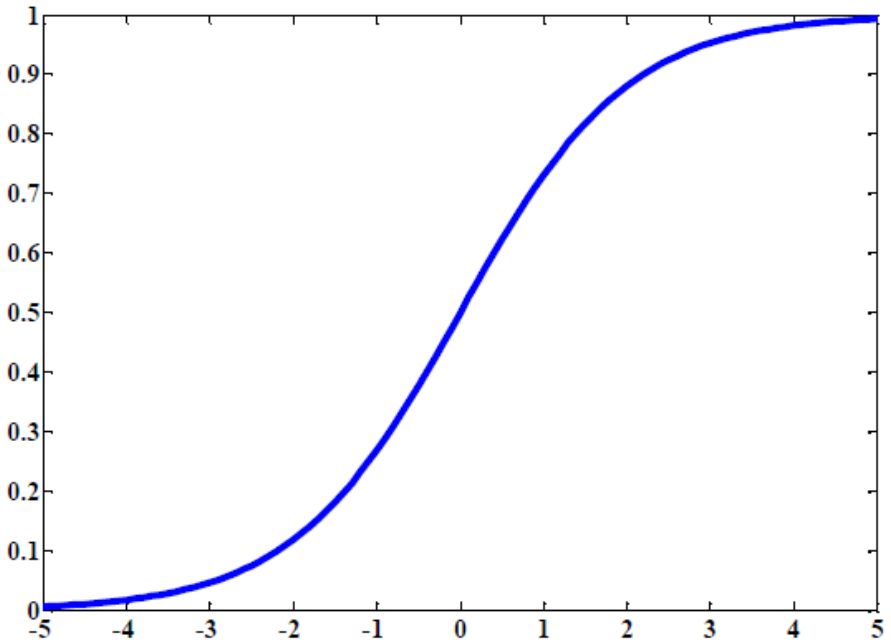
3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- 로지스틱 회귀분석

$$\pi(x) = \frac{1}{1+e^{-\left(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n\right)}} = \frac{1}{1+e^{-\beta X}}$$

$$\pi(x) = \frac{1}{1+e^{-\beta x}}$$



3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn import metrics
cancer = load_breast_cancer()
from sklearn.preprocessing import StandardScaler

model = LogisticRegression()
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
                                                    random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("before_accuracy", metrics.accuracy_score(y_test, y_pred))
```

```
before_accuracy 0.965034965034965
```

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
19 from sklearn.metrics import classification_report
20
21 print(classification_report(y_test, y_pred, target_names=['class 0', 'class 1']))
22 y_test=y_test.tolist()
23 print(y_test.count(1))
24
25
```

```
before_accuracy 0.965034965034965
              precision    recall  f1-score   support

     class 0       0.96      0.94      0.95        54
     class 1       0.97      0.98      0.97        89

    accuracy                           0.97       143
   macro avg       0.96      0.96      0.96       143
weighted avg       0.97      0.97      0.96       143

89
```

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석 – scaler 변환

```
from sklearn import metrics
cancer = load_breast_cancer()
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
data_scaled = scaler.fit_transform(cancer.data)

X_train, X_test, y_train, y_test = train_test_split(data_scaled, cancer.target,
                                                    random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("StandardScaler_accuracy", metrics.accuracy_score(y_test, y_pred))

```

```
StandardScaler_accuracy 0.9790209790209791
```

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석 – scaler 변환

```
from sklearn.preprocessing import MinMaxScaler


scaler = MinMaxScaler()
data_scaled = scaler.fit_transform(cancer.data)

X_train, X_test, y_train, y_test = train_test_split(data_scaled, cancer.target,
                                                    random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("MinMaxScaler_accuracy", metrics.accuracy_score(y_test, y_pred))

```

```
MinMaxScaler_accuracy 0.986013986013986
```

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석 – scaler 변환

```
1  from sklearn.linear_model import LogisticRegression
2  from sklearn.preprocessing import MaxAbsScaler
3
4  from sklearn.metrics import confusion_matrix
5
6  scaler = MaxAbsScaler()
7  data_scaled = scaler.fit_transform(cancer.data)
8
9  X_train, X_test, y_train, y_test = train_test_split(data_scaled, cancer.target,
10                                                     random_state=42)
11
12 model = LogisticRegression()
13 model.fit(X_train, y_train)
14
15 y_pred = model.predict(X_test)
16 print("MaxAbsScaler_accuracy", metrics.accuracy_score(y_test, y_pred))
17
18
```

```
MaxAbsScaler_accuracy 0.965034965034965
```

3. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석 – scaler 변환

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import confusion_matrix

scaler = RobustScaler()
data_scaled = scaler.fit_transform(cancer.data)

X_train, X_test, y_train, y_test = train_test_split(data_scaled, cancer.target,
                                                    random_state=42)

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
print("RobustScaler_accuracy", metrics.accuracy_score(y_test, y_pred))


```

```
RobustScaler_accuracy 0.986013986013986
```

감사합니다