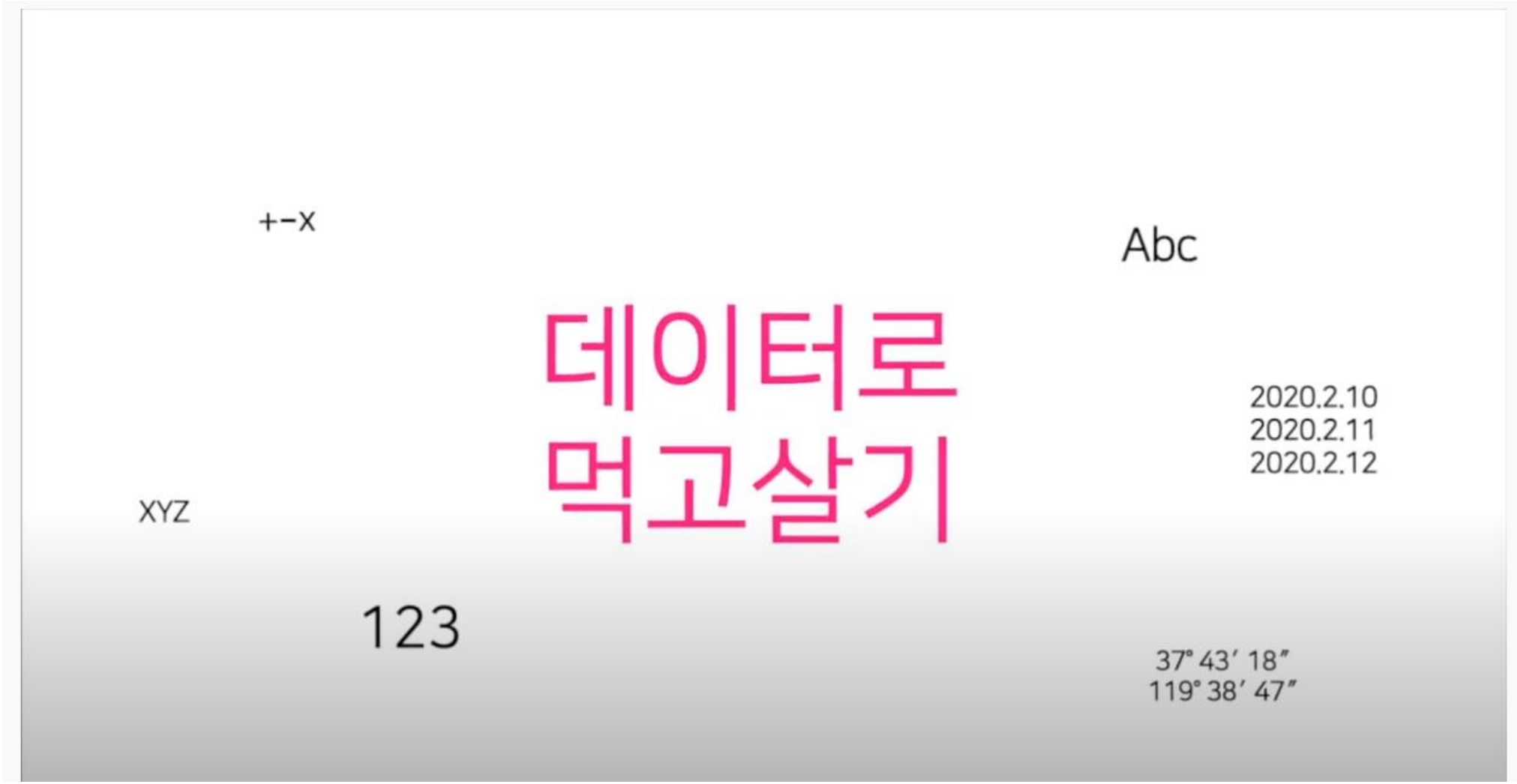
프로젝트 기반 데이터 과학자 양성과정(Data Science)
Machine Learning 및 분석실습

3주차
학습모델과 로지스틱 회귀분석
KNN 최근접 알고리즘

강사 : 최영진

목차

Ice Breaking

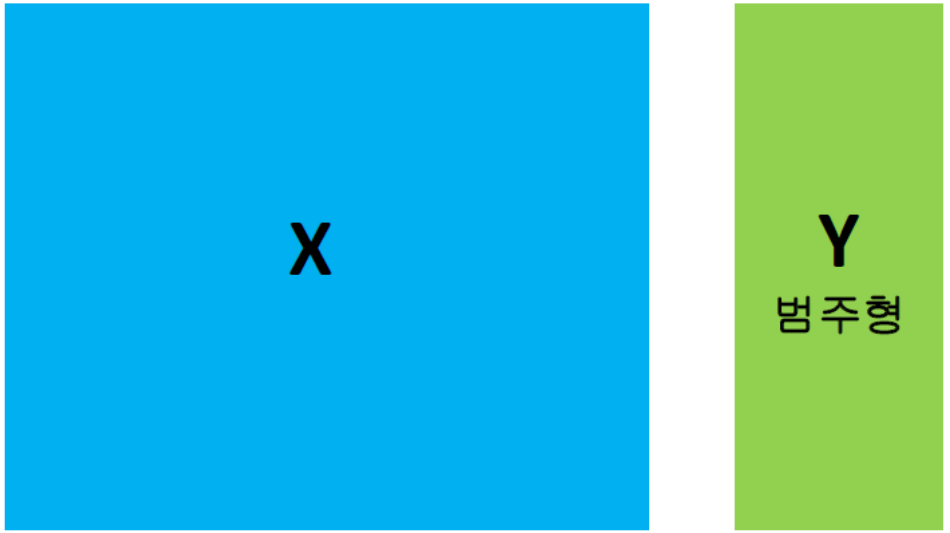


https://www.youtube.com/watch?v=WdF4lVTYRsY

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- 독립변수의의 선형 결합을 이용하여 사건의 발생 가능성을 예측할 때 사용되는 통계기법
- 범주형 반응변수
- 이진변수(반응 변수값 0 or 1) - 회귀의 y값은 0~1 사이의 확률값
- 멀티변수(반응 변수값 1 or 2 or 3 이상)



1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

Men

Vs.

Women

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- **일반 회귀모형을 반응변수가 범주일때로 확장**
- **새로운 관측치가 왔을 때 이를 기존의 범주 중 하나로 분류하는 것이 목적**
  - 제품이 불량인지 양품인지 분류
  - 고객이 이탈 고객인지 잔류 고객인지 분류
  - 카드거래가 정상인지 사기인지 분류
- **로지스틱 회귀분석 과정**
  - 1단계: 각 집단에 속하는 확률의 추정치를 예측. 이진분류의 경우 집단 1에 속하는 확률 P(Y=1)의 추정치로 얻음.
  - 2단계: 추정확률 → 분류 기준 값(cut-off) 적용 → 특정 범주로 분류

P(Y=1) ≥ 0.5 → 집단 1로 분류

P(Y=1) < 0.5 → 집단 0으로 분류

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

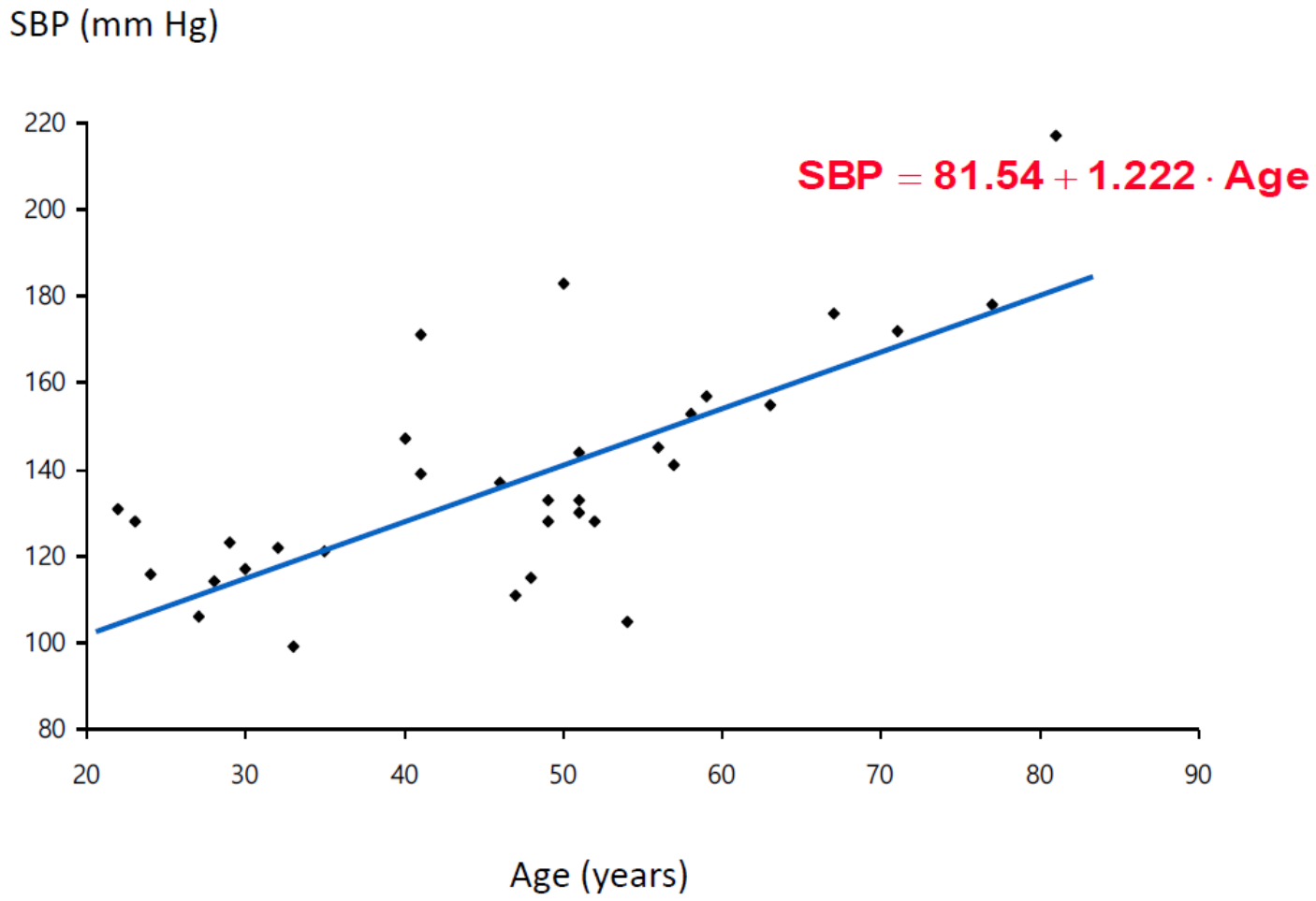
- **33명의 나이와 혈압**

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

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- **33명의 나이와 혈압**



1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

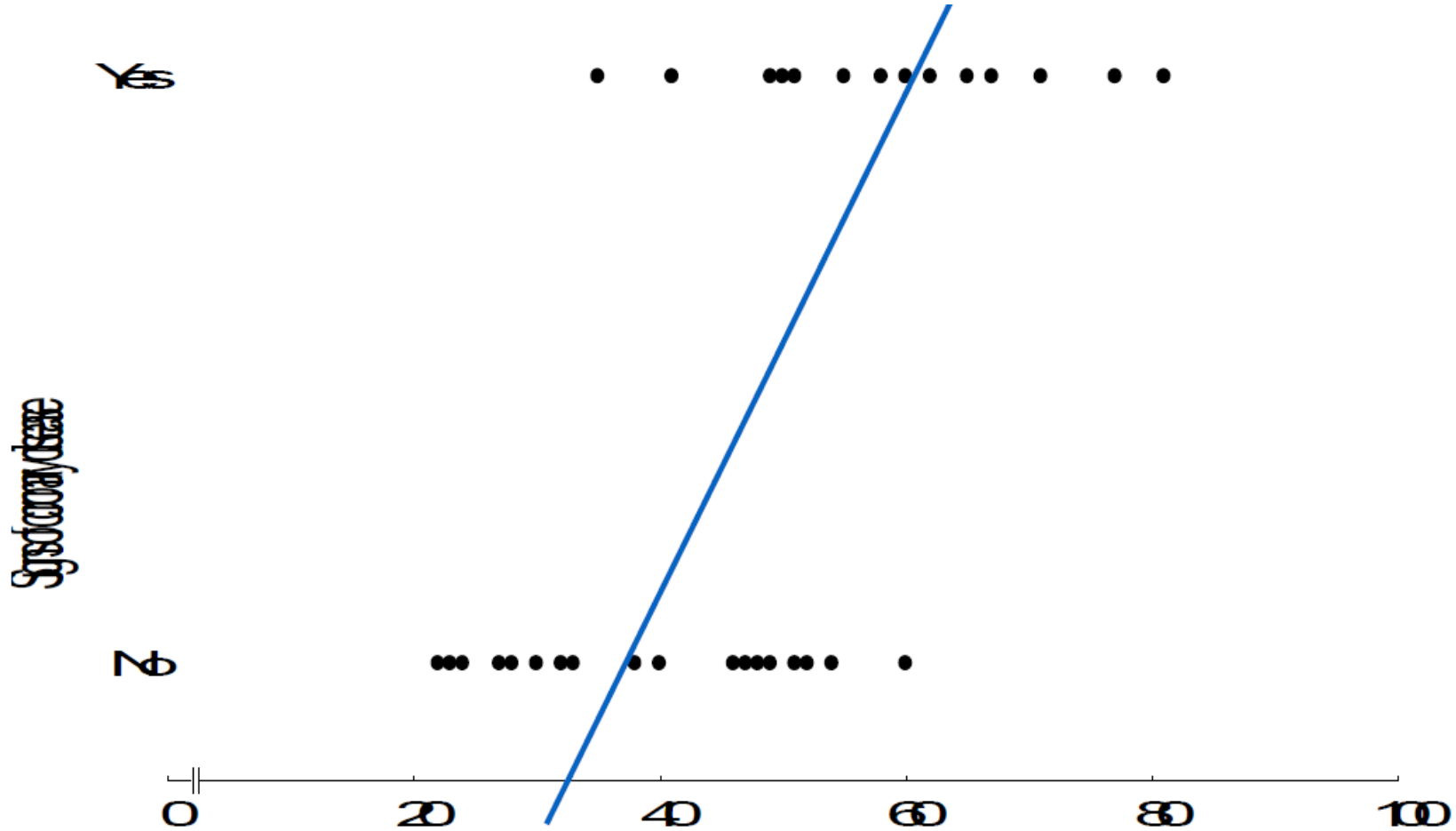
- 연속형 변수가 아닌 이진형(Binary) 변수인 Cancer Diagnosis (CD)를 사용

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

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

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



1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

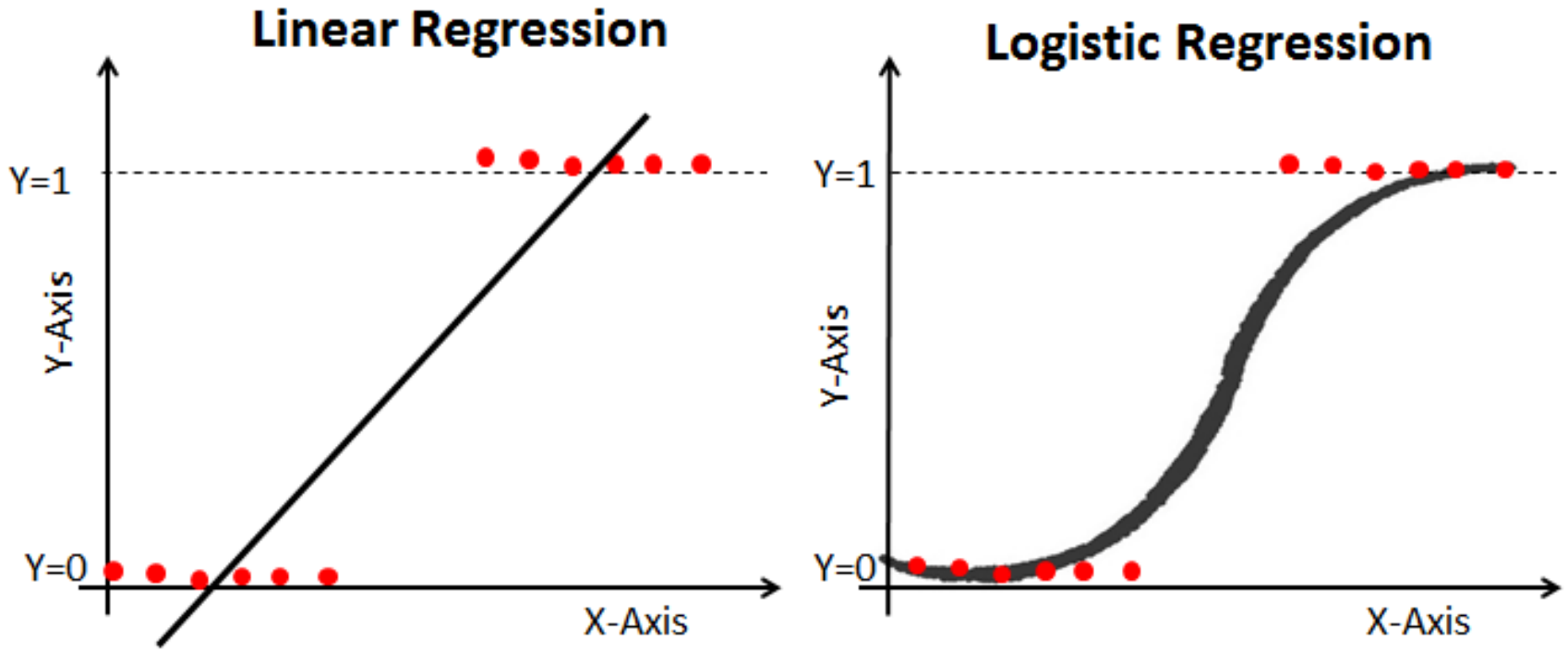
- 연속형 변수가 아닌 이진형(Binary) 변수인 Cancer Diagnosis (CD)를 사용

		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

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- 회귀분석과의 차이점



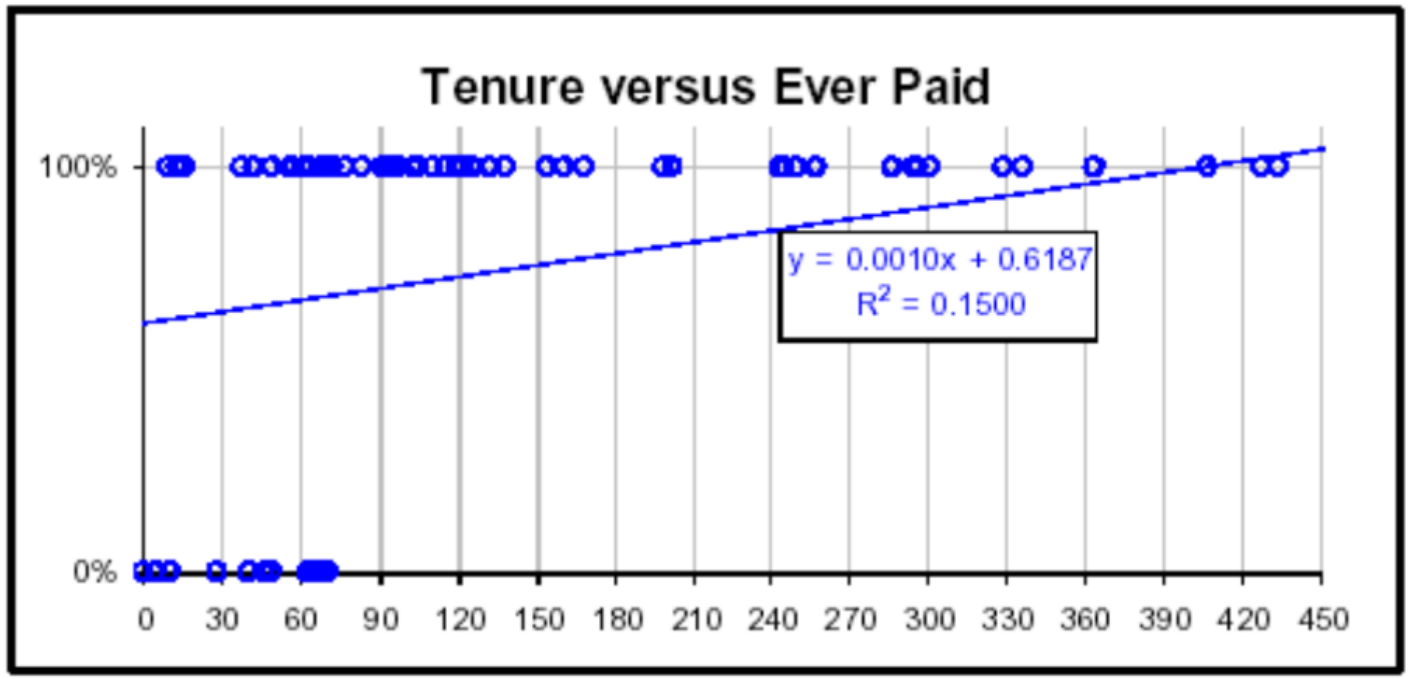
1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- 회귀분석

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \cdots + \hat{\beta}_d x_d$$

✓ May have a probability that is greater than 1 or less than 0



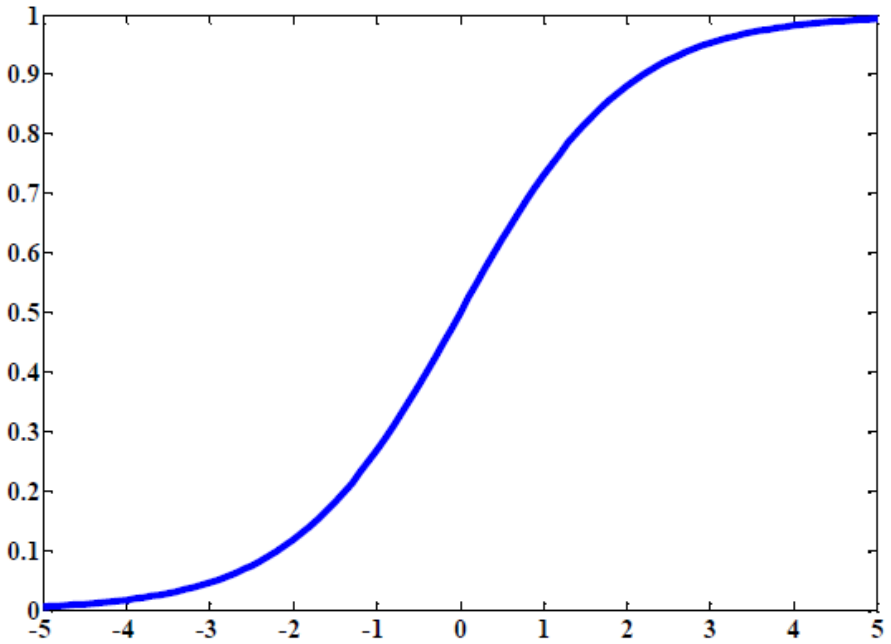
1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- 로지스틱 회귀분석 - Probability that an observation x belongs to class 1

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

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



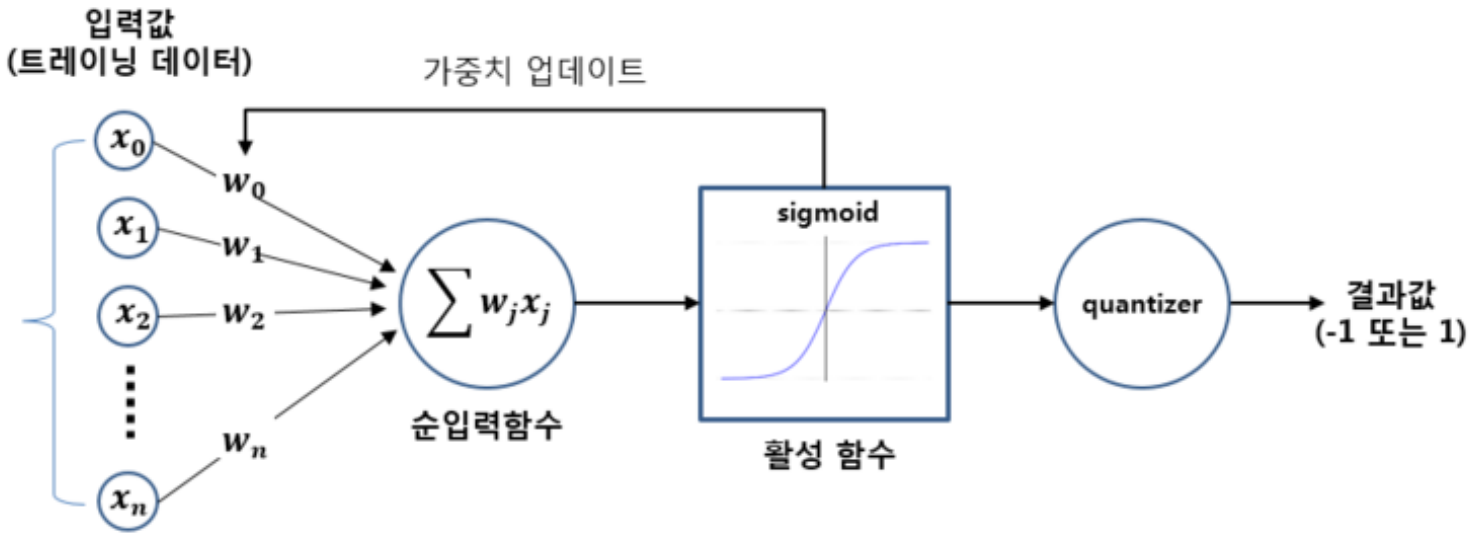
1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- 로지스틱 회귀분석 순입력 함수의 값을 Sigmoid 함수에 대입
- Sigmoid 함수의 값은 특정 클래스에 속할 확률로 , 입력되는 데이터의 특정 클래스에 포함될 예측 확률값 계산

  P(Y=1) ≥ 0.5 → 집단 1로 분류

  P(Y=1) < 0.5 → 집단 0으로 분류



1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 – odds

- **로지스틱 모형 식은 독립 변수가 [-∞,∞]의 어느 숫자이든 상관 없이 종속 변수 또는 결과 값이 항상 범위 [0,1] 사이**

- 오즈 (odds) & 로짓 변환

  성공 확률이 실패 확률에 비해 몇 배 더 높은가를 나타내며 그 식은 아래와 같다.

$$\text{odds} = \frac{p(y=1|x)}{1-p(y=1|x)}$$

- 로짓 변환

  오즈에 로그를 취한 함수로서 입력 값의 범위가 [0,1] 일때 출력 값의 범위를 $(-\infty, +\infty)$로 조정한다.

$$\text{logit}(p) = \log \frac{p}{1-p}$$

- 로지스틱 함수 (logistic function):

  로지스틱 함수의 그래프는 Figure 1과 같고 이는 독립 변수 x가 주어졌을 때 종속 변수가 1의 범주에 속할 확률을 의미한다. 즉, $p(y=1|x)$를 의미한다.

  로지스틱 함수는 로짓 변환을 통해 만들어지고, 그 형태는 다음과 같다.

$$\text{logistic function} = \frac{e^{\beta \cdot X_i}}{1+e^{\beta \cdot X_i}}$$

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 – odds

▪ 오즈 (odds) : '실패' (0) 에 대한 '성공' (1) 의 비율

- 집단 1(Y=1)에 속하는 승산은 집단 1에 속하는 확률에서 집단 0에 속하는 확률을 나눈 비율

- 오즈 (odds) & 로짓 변환

  성공 확률이 실패 확률에 비해 몇 배 더 높은가를 나타내며 그 식은 아래와 같다.

$$\text{odds} = \frac{p(y=1|x)}{1-p(y=1|x)}$$

$$Odds = \frac{P(\text{특정 사건이 발생하는 경우})}{P(\text{특정 사건이 발생하지 않는 경우})} = \frac{P(A)}{1-P(A)}$$

1. 로지스틱 회귀분석

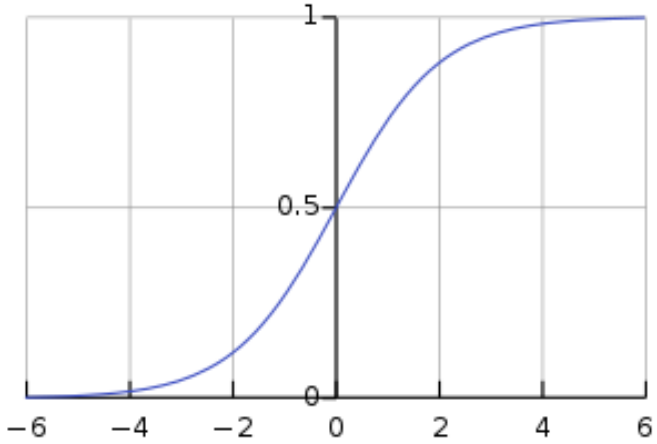
❖ 로지스틱 회귀분석 – odds

- **로짓 (logit) : (0 ~ 1 이 아니라) ±∞의 범위에서 어떤 클래스에 속할 확률을 결정하는 함수**
- Logit 변환, Odds에 log를 씌운 것
  - 일반회귀식으로는 0 또는 1을 예측할 수 없으므로 y값을 Odds 비로 변환하여 예측

$$\text{logit}(p) = \log \frac{p}{1-p}$$

$$Odds\ of\ passing = \frac{0.7}{0.3} = 2.33$$

$$Log\ odds\ of\ passing = log(2.33) = 0.847$$



이항 로지스틱 회귀 : 종속 변수가 (성공, 실패)와 같은 binary 형태
다항 로지스틱 회귀 : 종속 변수가 (삼성, LG, 애플)과 같이 3개 이상의 Multi 형태

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 – odds

- 로그 오즈를 이용한 회귀분석 식은

$$log(Odds) = log\left(\frac{p}{1-p}\right) = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \cdots + \hat{\beta}_d x_d$$

- 양변에 로그를 제하면,

$$\frac{p}{1-p} = e^{\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \cdots + \hat{\beta}_d x_d}$$

- 마지막으로 성공확률에 대한 식으로 표현

$$p = \frac{1}{1 + e^{-(\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \cdots + \hat{\beta}_d x_d)}}$$

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 – odds

- **Predict a new customer whether he/she will accept the bank's personal loan offer**

일련 번호	나이	경력	소득	가족 수	월별 신용카드 평균사용액	교육 수준	담보부 채권	개인 대출	증권 계좌	CD 계좌	온라인 뱅킹	신용 카드
1	25	1	49	4	1.60	UG	0	No	Yes	No	No	No
2	45	19	34	3	1.50	UG	0	No	Yes	No	No	No
3	39	15	11	1	1.00	UG	0	No	No	No	No	No
4	35	9	100	1	2.70	Grad	0	No	No	No	No	No
5	35	8	45	4	1.00	Grad	0	No	No	No	No	Yes
6	37	13	29	4	0.40	Grad	155	No	No	No	Yes	No
7	53	27	72	2	1.50	Grad	0	No	No	No	Yes	No
8	50	24	22	1	0.30	Prof	0	No	No	No	No	Yes
9	35	10	81	3	0.60	Grad	104	No	No	No	Yes	No
10	34	9	180	1	8.90	Prof	0	Yes	No	No	No	No
11	65	39	105	4	2.40	Prof	0	No	No	No	No	No
12	29	5	45	3	0.10	Grad	0	No	No	No	Yes	No
13	48	23	114	2	3.80	Prof	0	No	Yes	No	No	No
14	59	32	40	4	2.50	Grad	0	No	No	No	Yes	No
15	67	41	112	1	2.00	UG	0	No	Yes	No	No	No
16	60	30	22	1	1.50	Prof	0	No	No	No	Yes	Yes
17	38	14	130	4	4.70	Prof	134	Yes	No	No	No	No
18	42	18	81	4	2.40	UG	0	No	No	No	No	No
19	46	21	193	2	8.10	Prof	0	Yes	No	No	No	No
20	55	28	21	1	0.50	Grad	0	No	Yes	No	No	Yes

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 – odds

$$p = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n)}}$$

Input variables	Coefficient	Std. Error	p-value	Odds
Constant term	-13.20165825	2.46772742	0.00000009	*
Age	-0.04453737	0.09096102	0.62439483	0.95643985
Experience	0.05657264	0.09005365	0.5298661	1.05820346
Income	0.0657607	0.00422134	0	1.06797111
Family	0.57155931	0.10119002	0.00000002	1.77102649
CCAvg	0.18724874	0.06153848	0.00234395	1.20592725
Mortgage	0.00175308	0.00080375	0.02917421	1.00175464
Securities Account	-0.85484785	0.41863668	0.04115349	0.42534789
CD Account	3.46900773	0.44893095	0	32.10486984
Online	-0.84355801	0.22832377	0.00022026	0.43017724
CreditCard	-0.96406376	0.28254223	0.00064463	0.38134006
EducGrad	4.58909273	0.38708162	0	98.40509796
EducProf	4.52272701	0.38425466	0	92.08635712

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 – 계수(Coefficient)

- 로지스틱 회귀분석에서 각 변수에 대응하는 베타 값
- 선형회귀분석에서는 해당변수가 1단위 증가할 때 종속변수의 변화량을 의미하나,
- 로지스틱 회귀분석에서는 해당변수가 1단위 증가할 때 로그odds의 변화량
- 양수이면 성공확률과 양의 상관관계, 음수이면 성공확률과 음의 상관관계

Input variables	Coefficient	Std. Error	p-value	Odds
Constant term	-13.20165825	2.46772742	0.00000009	*
Age	-0.04453737	0.09096102	0.62439483	0.95643985
Experience	0.05657264	0.09005365	0.5298661	1.05820346
Income	0.0657607	0.00422134	0	1.06797111
Family	0.57155931	0.10119002	0.00000002	1.77102649
CCAvg	0.18724874	0.06153848	0.00234395	1.20592725
Mortgage	0.00175308	0.00080375	0.02917421	1.00175464
Securities Account	-0.85484785	0.41863668	0.04115349	0.42534789
CD Account	3.46900773	0.44893095	0	32.10486984
Online	-0.84355801	0.22832377	0.00022026	0.43017724
CreditCard	-0.96406376	0.28254223	0.00064463	0.38134006
EducGrad	4.58909273	0.38708162	0	98.40509796
EducProf	4.52272701	0.38425466	0	92.08635712

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 – 유의확률: p-value

- 로지스틱 회귀분석에서 해당변수가 통계적으로 유의미한지 여부를 알려주는 지표
- 0에 가까울수록 모델링에 중요한 변수이며, 1에 가까울수록 유의미하지 않은 변수임
- 특정 유의수준(α)을 설정하여 해당 값 미만의 변수만을 사용하여 다시 로지스틱 회귀분석을 구축 (주로α=0.05(혹은0.01사용)

Input variables	Coefficient	Std. Error	p-value	Odds
Constant term	-13.20165825	2.46772742	0.00000009	*
Age	-0.04453737	0.09096102	0.62439483	0.95643985
Experience	0.05657264	0.09005365	0.5298661	1.05820346
Income	0.0657607	0.00422134	0	1.06797111
Family	0.57155931	0.10119002	0.00000002	1.77102649
CCAvg	0.18724874	0.06153848	0.00234395	1.20592725
Mortgage	0.00175308	0.00080375	0.02917421	1.00175464
Securities Account	-0.85484785	0.41863668	0.04115349	0.42534789
CD Account	3.46900773	0.44893095	0	32.10486984
Online	-0.84355801	0.22832377	0.00022026	0.43017724
CreditCard	-0.96406376	0.28254223	0.00064463	0.38134006
EducGrad	4.58909273	0.38708162	0	98.40509796
EducProf	4.52272701	0.38425466	0	92.08635712

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 – 승산비율(Odd Ratio)

- 나머지 변수는 모두 고정시킨 상태에서 한 변수를 1만큼 증가시켰을 때 변화하는 Odds의 비율

Input variables	Coefficient	Std. Error	p-value	Odds
Constant term	-13.20165825	2.46772742	0.00000009	*
Age	-0.04453737	0.09096102	0.62439483	0.95643985
Experience	0.05657264	0.09005365	0.5298661	1.05820346
Income	0.0657607	0.00422134	0	1.06797111
Family	0.57155931	0.10119002	0.00000002	1.77102649
CCAvg	0.18724874	0.06153848	0.00234395	1.20592725
Mortgage	0.00175308	0.00080375	0.02917421	1.00175464
Securities Account	-0.85484785	0.41863668	0.04115349	0.42534789
CD Account	3.46900773	0.44893095	0	32.10486984
Online	-0.84355801	0.22832377	0.00022026	0.43017724
CreditCard	-0.96406376	0.28254223	0.00064463	0.38134006
EducGrad	4.58909273	0.38708162	0	98.40509796
EducProf	4.52272701	0.38425466	0	92.08635712

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 odd 예시

- 한국의 학교, 소년원, 분류심사원을 방문하여 비행청소년의 소년원, 분류심사원의 재소기록과 학생 청소년의 학교생활기록부를 참조하여 조사
- 서울과 지방의 12개 중학교와 12개 고등학교에 재학 중인 학생 청소년 1,254명과 6개 소년원, 2개 분류심사원에 재원중인 비행청소년 1,121명을 연구대상 집단으로 선정
- 비행청소년과 학생 청소년을 판별해주는 요인(흡연, 음주, 약물사용, 폭력물 시청시간)

Table 5. Parameter estimates for logistic regression model

Variables	df	Parameter estimate	Standard error	Chi-square	p	Odds ratio
Intercept	1	- 7.40	0.483	235.06	0.0001	
Drink(Y/N)	1	2.34	0.299	61.30	0.0001	10.38
Drug(Y/N)	1	1.08	0.195	30.71	0.0001	2.95
Smoke(Y/N)	1	3.48	0.250	193.19	0.0001	32.32
Sexual abuse	1	0.52	0.107	23.54	0.0001	1.68
Murder	1	- 0.02	0.014	1.34	0.2477	0.98
Violence	1	0.58	0.072	63.49	0.0001	1.78
Pornography	1	0.22	0.103	4.49	0.0340	1.25
Family violence	1	0.04	0.026	2.49	0.1144	1.04

Y/N : yes/no

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 odd 예시

- **odds ratio는 흡연 : 32.32 , 음주 : 10.38, 약물사용 : 2.95 ,폭력물 시청시간 : 1.79 으로 흡연이 가장 빈도가 높아 비행청소년 판별 요인임을 확인**
- **흡연의 경우 담배를 피우지 않는 학생보다 피우는 청소년이 비행청소년이 될 확률이 32배 이상 더 큰 것으로 확인**

Table 5. Parameter estimates for logistic regression model

Variables	df	Parameter estimate	Standard error	Chi-square	p	Odds ratio
Intercept	1	- 7.40	0.483	235.06	0.0001	
Drink(Y/N)	1	2.34	0.299	61.30	0.0001	10.38
Drug(Y/N)	1	1.08	0.195	30.71	0.0001	2.95
Smoke(Y/N)	1	3.48	0.250	193.19	0.0001	32.32
Sexual abuse	1	0.52	0.107	23.54	0.0001	1.68
Murder	1	- 0.02	0.014	1.34	0.2477	0.98
Violence	1	0.58	0.072	63.49	0.0001	1.78
Pornography	1	0.22	0.103	4.49	0.0340	1.25
Family violence	1	0.04	0.026	2.49	0.1144	1.04

Y/N : yes/no

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- **from sklearn.linear_model import LogisticRegression**
  - LogisticRegression 모델을 생성하고, 그 안에 속성들(features)과 그 레이블(labels)을 fit
- **model = LinearRegression()**
- **model.fit(features, labels)**
  - fit() 메서드는 모델에 필요한 두 가지 변수를 전달.

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석

- **계수: model.coef_**
- **절편: model.intercept_**
  - predict()를 사용하면 새로운 속성들을 넣었을 때 그 레이블에 속하는지 아닌지 1또는 0으로 구성된 벡터를 반환
- **model.predict(features)**
  - 해당 레이블로 분류될 확률 값을 알고 싶다면 이렇게 .predict_proba()를 확률을 0에서 1사이의 값으로 돌려줌
- **model.predict_proba(features)**

1. 로지스틱 회귀분석

❖ 주피터 단축키

모드	내용	단축키
셀 선택 모드	셀 추가	위에 셀추가: a, 아래에 셀추가: b,
	셀 삭제	dd
	복사/잘라내기	잘라내기: x, 복사하기: c, 붙여넣기: p
	아래 셀과 합치기	shift + m
	셀 타입 변경	마크다운: m, 코드: y
	파일 저장	ctrl + s 또는 s
	코드편집 모드	[enter]
코드 입력 모드	실행	셀 실행: ctrl + enter 실행 후 다음 셀로 이동: shift + enter 실행 취소: ctrl + z, 셀 다시 실행: ctrl +y
	커서에서 셀 나누기	shift + ctrl + -
	셀 선택 모드로 가기	esc 또는 ctrl + m
	주석처리	crtl +/

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.preprocessing import StandardScaler

x_data = np.array([
    [2, 1],
    [3, 2],
    [3, 4],
    [5, 5],
    [7, 5],
    [2, 5],
    [8, 9],
    [9, 10],
    [6, 12],
    [9, 2],
    [6, 10],
    [2, 4]
])
y_data = np.array([0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0])

labels = ['fail', 'pass']

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.2, random_state=2)


```

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
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 0.6666666666666666
```

1. 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.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
```

1. 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
```

1. 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
```

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석 – scaler 변환

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import MaxAbsScaler

from sklearn.metrics import confusion_matrix

scaler = MaxAbsScaler()
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("MaxAbsScaler_accuracy", metrics.accuracy_score(y_test, y_pred))


```

```
MaxAbsScaler_accuracy 0.965034965034965
```

1. 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
```

1. 로지스틱 회귀분석

❖ 지도학습(Supervised learning)

▪ 지도 학습 알고리즘과 분류 및 회귀 문제

번호	알고리즘 이름	분류 문제	회귀 문제
01	선형회귀(linear regression)	×	○
02	정규화(regularization)	×	○
03	로지스틱 회귀(logistic regression)	○	×
04	서포트 벡터 머신(support vector machine)	○	○
05	커널(kernel) 기법을 적용한 서포트 벡터 머신	○	○
06	나이브 베이즈 분류(Naïve Bayes classification)	○	×
07	랜덤 포레스트(random forest)	○	○
08	신경망(neural network)	○	○
09	k-최근접 이웃 알고리즘(k-nearest neighbors algorithm, kNN)	○	○

1. 로지스틱 회귀분석

❖ 모델 성능 평가 척도

- **모델의 예측결과와 성능을 살펴볼 수 있는 척도**

		예측값	
		Class = yes	Class = no
실제값	Class = yes	a (TP : true positive)	b (FN : false negative)
	Class = no	c (FP : false positive)	d (TN : true negative)

1. 로지스틱 회귀분석

❖ 모델 성능 평가 척도

▪ 모델의 예측결과와 성능을 살펴볼 수 있는 척도

- **실제값**: 데이터의 실제 카테고리 / **예측값**: 모델이 분류, 예측한 데이터의 카테고리
- A, TP (True Positive): 실제 **yes** 카테고리의 데이터 중 모델이 **yes** 카테고리로 예측한 데이터의 건 수
- B, FN (False Negative): 실제 **yes** 카테고리의 데이터 중 모델이 **no** 카테고리로 예측한 데이터의 건수
- C, FP (False Positive): 실제 **no** 카테고리의 데이터 중 모델이 **yes** 카테고리로 예측한 데이터의 건수
- D, TN (True Negative): 실제 **no** 카테고리의 데이터 중 모델이 **no** 카테고리로 예측한 데이터의 건수

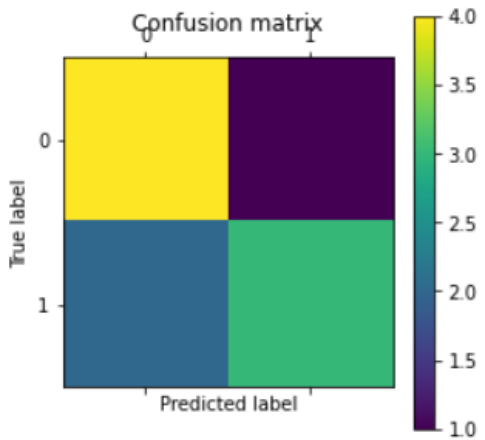
		예측값	
		Class = yes	Class = no
실제값	Class = yes	a (TP : true positive)	b (FN : false negative)
	Class = no	c (FP : false positive)	d (TN : true negative)

1. 로지스틱 회귀분석

❖ 모델 성능 평가 척도

```python
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

y_test = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]
y_pred = [0, 1, 0, 0, 0, 0, 0, 1, 1, 1]
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)

plt.matshow(confusion_matrix)
plt.title('Confusion matrix')
plt.colorbar()

plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()

```

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



		예측값	
		Class = yes	Class = no
실제값	Class = yes	a (TP : true positive)	b (FN : false negative)
	Class = no	c (FP : false positive)	d (TN : true negative)

1. 로지스틱 회귀분석

❖ 모델 성능 평가 척도

- **Accuracy (정확도)**
  - 모델이 정확하게 분류 또는 예측하는 데이터의 비율
  - TP: 참 긍정개수 TN: 참 부정개수 FP: 거짓 긍정개수 FN : 거짓 부정개수

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

- **Accuracy의 한계**
  - 2 class 문제에서 class yes에 해당하는 데이터는 9900건, class no에 해당하는 데이터는 100건이 존재할 경우
  - 모델이 모든 데이터를 class yes로 예측할 경우에도
    즉, class no를 예측하지 못했음에도 불구하고 accuracy는 9900/10000=99%가 됨
  - 이러한 경우가 있어서 accuracy 모델은 성능을 측정하는 척도로 적합하지 않음

1. 로지스틱 회귀분석

❖ 모델 성능 평가 척도

		예측값	
		Class = yes	Class = no
실제값	Class = yes	a (TP : true positive)	b (FN : false negative)
	Class = no	c (FP : false positive)	d (TN : true negative)

- **Precision (정밀도)**
  - 모델이 검출한 데이터 중 올바르게 검출된 데이터의 비율
  - 예시) 악성종양을 악성으로 예측한 결과
  - TP: 참 긍정개수 TN: 참 부정개수 FP: 거짓 긍정개수 FN : 거짓 부정개수

$$\text{Precision (p)} = \frac{a}{a+c}$$

- **Recall (재현율)**
  - 실제 해당 데이터 중 모델이 올바르게 검출한 데이터의 비율
  - 예시) 전체 악성종양을 실제 예측한 결과

$$\text{Recall (r)} = \frac{a}{a+b}$$

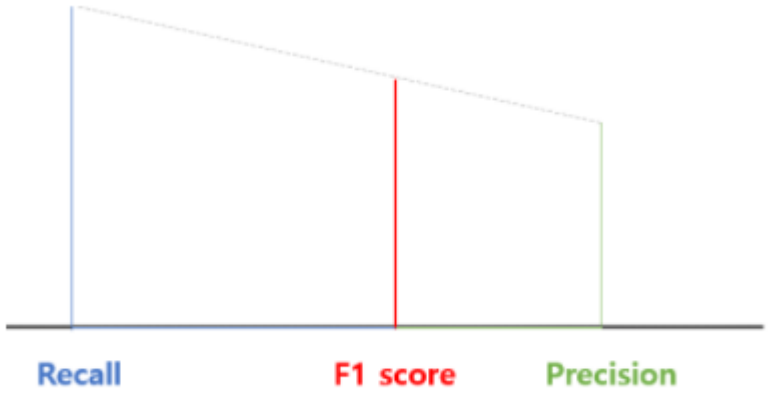
1. 로지스틱 회귀분석

❖ 모델 성능 평가 척도

		예측값	
		Class = yes	Class = no
실제값	Class = yes	a (TP : true positive)	b (FN : false negative)
	Class = no	c (FP : false positive)	d (TN : true negative)

- **F1-Measure**
  - Precision과 recall은 모델의 성능을 객관적으로 판단하기에 부족
  - F1 score는 Precision과 Recall의 조화평균

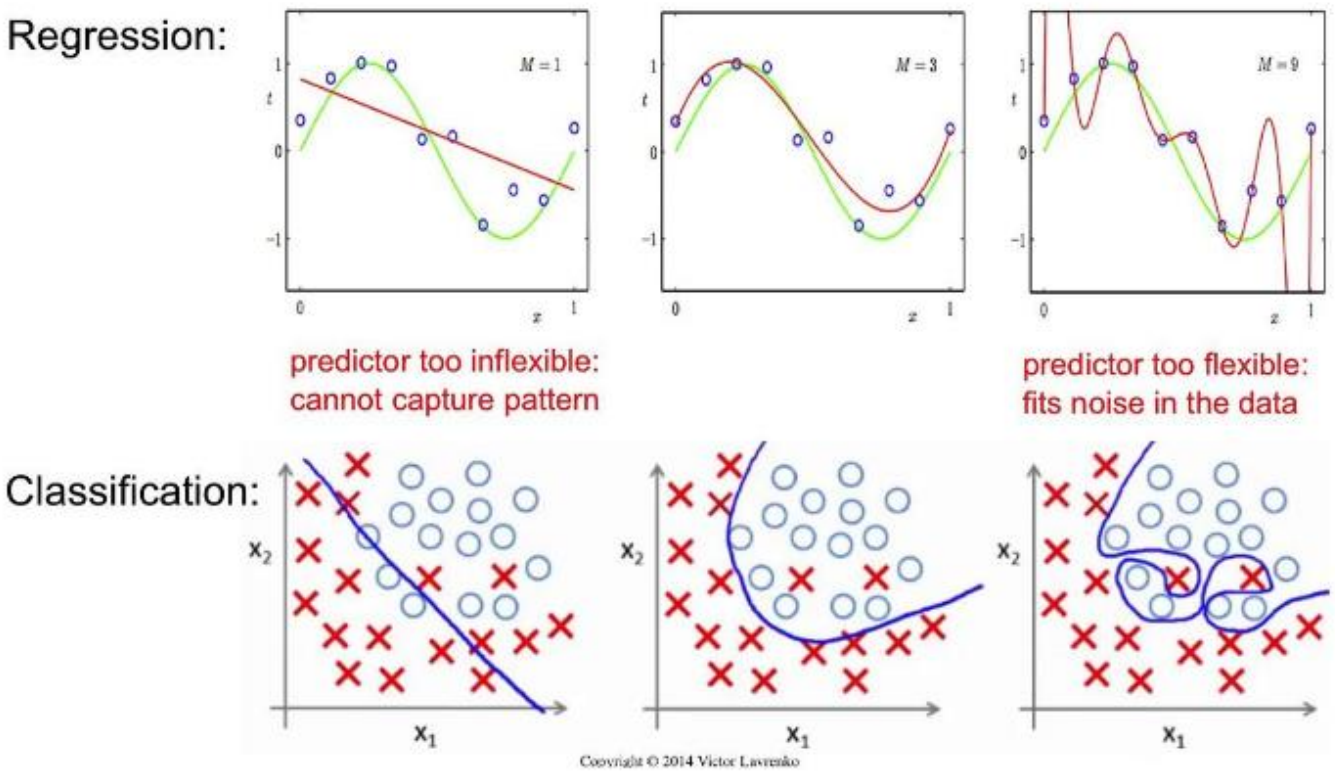
$$(F1\text{-}score) = 2\times\frac{1}{\frac{1}{Precision}+\frac{1}{Recall}} = 2\times\frac{Precision \times Recall}{Precision + Recall}$$



1. 로지스틱 회귀분석

❖ Overfitting 과 Underfitting

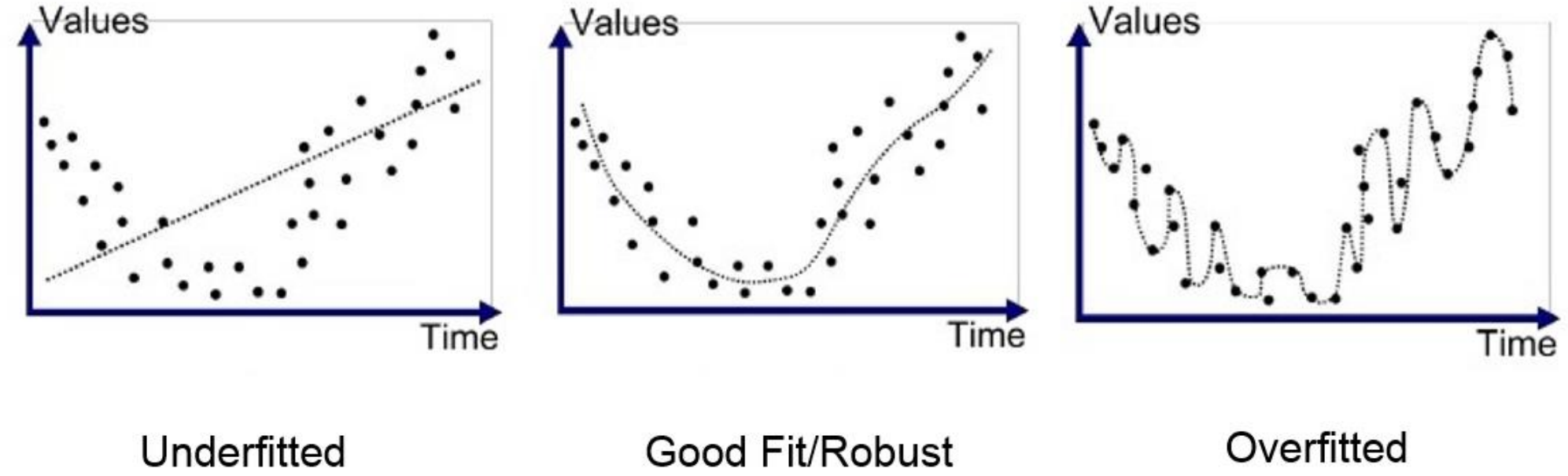
- 언더피팅(underfitting) : 모델이 너무 간단하기 때문에 학습 오류가 줄어들지 않는 것
- 오버피팅(overfitting) : 학습 오류가 테스트 데이터셋에 대한 오류보다 아주 작은 경우



1. 로지스틱 회귀분석

❖ Overfitting 과 Underfitting

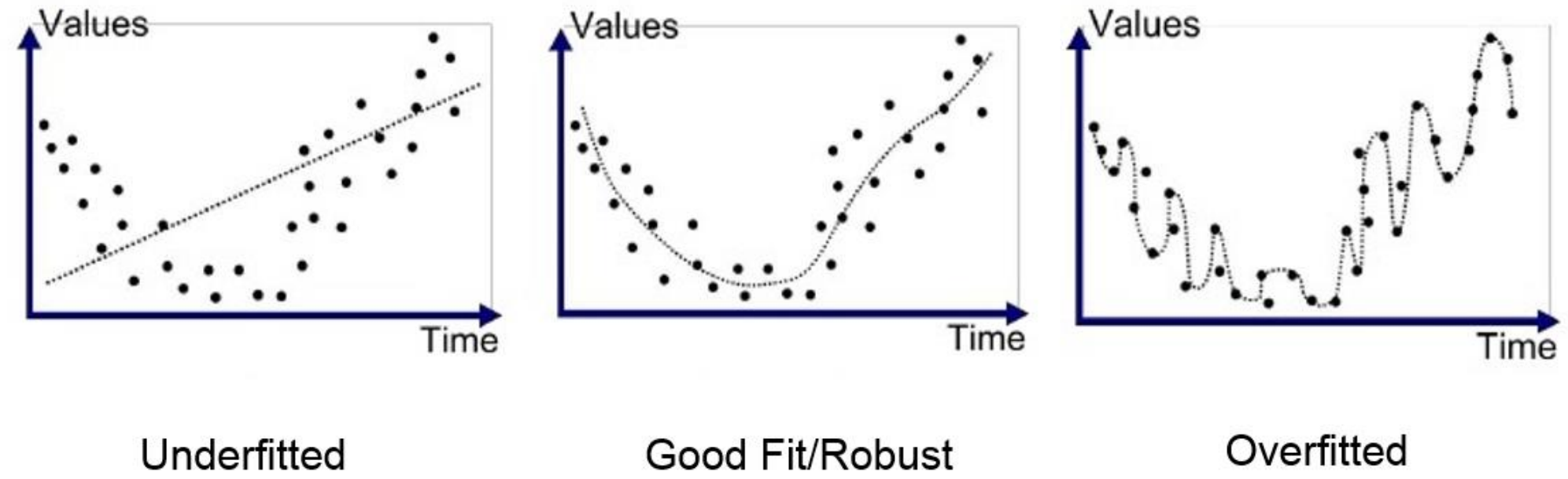
- 오버피팅(overfitting) : 이는 학습 오류가 테스트 데이터셋에 대한 오류보다 아주 작은 경우
- 너무 세밀하게 학습 데이터 하나하나를 다 설명하려고 하다보니 정작 중요한 패턴을 설명할 수 없게 되는 현상



1. 로지스틱 회귀분석

❖ Overfitting 과 Underfitting

- 언더피팅(underfitting) : 모델이 너무 간단하기 때문에 학습 오류가 줄어들지 않는 것
- 학습 데이터가 모자라거나 학습이 제대로 되지 않아서, 트레이닝 데이터에 가깝게 가지 못한 경우



1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.preprocessing import StandardScaler

x_data = np.array([
    [2, 1],
    [3, 2],
    [3, 4],
    [5, 5],
    [7, 5],
    [2, 5],
    [8, 9],
    [9, 10],
    [6, 12],
    [9, 2],
    [6, 10],
    [2, 4]
])
y_data = np.array([0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0])

labels = ['fail', 'pass']

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.2, random_state=2)


```

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
model = LogisticRegression()
model.fit(x_train, y_train)

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


from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred, target_names=['class 0', 'class 1']))

```

```
before_accuracy 0.6666666666666666
              precision    recall  f1-score   support

     class 0       0.00      0.00      0.00         1
     class 1       0.67      1.00      0.80         2

    accuracy                           0.67         3
   macro avg       0.33      0.50      0.40         3
weighted avg       0.44      0.67      0.53         3
```

1. 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
```

1. 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
```

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
1  import pandas as pd
2  import statsmodels.api as sm
3  from sklearn.linear_model import LogisticRegression
4  from sklearn.model_selection import train_test_split
5
6  wine_data = pd.read_csv('D:/big_data/winequality-white.csv',delimiter=';',dtype=float)
7  wine_data.head(10)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6.0
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6.0
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6.0
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6.0
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6.0
5	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6.0
6	6.2	0.32	0.16	7.0	0.045	30.0	136.0	0.9949	3.18	0.47	9.6	6.0
7	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6.0
8	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6.0
9	8.1	0.22	0.43	1.5	0.044	28.0	129.0	0.9938	3.22	0.45	11.0	6.0

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
x_data = wine_data.iloc[:,0:-1]
y_data = wine_data.iloc[:,-1]

# Score 값이 8보다 작으면 0,  8보다 크거나 같으면 1로 값 변경.

y_data = np.array([1 if i>=8 else 0 for i in y_data])

x_data.head(5)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
# 트레인, 테스트 데이터 나누기.

train_x, test_x, train_y, test_y = train_test_split(x_data, y_data, test_size = 0.3,random_state=42)
```

```
from sklearn.metrics import classification_report


log_reg = LogisticRegression()

log_reg.fit(train_x, train_y)


y_true, y_pred = test_y, log_reg.predict(test_x)

print(classification_report(y_true, y_pred))
```

Double click to hide

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1421
           1       0.00      0.00      0.00        49

    accuracy                           0.97      1470
   macro avg       0.48      0.50      0.49      1470
weighted avg       0.93      0.97      0.95      1470
```

1. 로지스틱 회귀분석

❖ 로지스틱 회귀분석 실습

- 로지스틱 회귀분석

```
1  logit = sm.Logit(train_y,train_x).fit()
2
3  logit.summary()
4
5
```

```
Optimization terminated successfully.
         Current function value: 0.143100
         Iterations 9
```

```
1  print (np.exp(logit.params))
```

```
fixed acidity           0.617031
volatile acidity        0.001243
citric acid             0.862660
residual sugar          1.106911
chlorides               0.435785
free sulfur dioxide     1.023655
total sulfur dioxide    1.006192
density                31.667852
pH                      0.379583
sulphates               2.522360
alcohol                 1.765592
dtype: float64
```

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	3428
Model:	Logit	**Df Residuals:**	3417
Method:	MLE	**Df Model:**	10
Date:	Sun, 16 Aug 2020	**Pseudo R-squ.:**	0.1179
Time:	19:06:08	**Log-Likelihood:**	-490.55
converged:	True	**LL-Null:**	-556.12
Covariance Type:	nonrobust	**LLR p-value:**	2.729e-23

	coef	std err	z	P>|z|	[0.025	0.975]
fixed acidity	0.1153	0.138	0.835	0.404	-0.155	0.386
volatile acidity	-2.5585	1.092	-2.343	0.019	-4.699	-0.418
citric acid	-0.8329	0.941	-0.885	0.376	-2.678	1.012
residual sugar	0.0738	0.022	3.362	0.001	0.031	0.117
chlorides	-3.0146	7.500	-0.402	0.688	-17.714	11.684
free sulfur dioxide	0.0235	0.007	3.462	0.001	0.010	0.037
total sulfur dioxide	-0.0014	0.004	-0.411	0.681	-0.008	0.005
density	-16.2508	3.255	-4.992	0.000	-22.631	-9.871
pH	0.6767	0.744	0.910	0.363	-0.781	2.134
sulphates	0.4109	0.713	0.577	0.564	-0.986	1.808
alcohol	0.8842	0.098	8.987	0.000	0.691	1.077

1. 로지스틱 회귀분석

❖ ROC (Receiver Operating Characteristic) Curve

- **ROC 곡선은 Binary Classifier System(이진 분류 시스템)에 대한 성능 평가 기법**

Total Population	모델의 예측 Positive	모델의 예측 Negative
정답이 Positive	True Positive	False Negative
정답이 Negative	False Positive	True Negative

1. 로지스틱 회귀분석

❖ ROC (Receiver Operating Characteristic) Curve

- **ROC 곡선은 Binary Classifier System(이진 분류 시스템)에 대한 성능 평가 기법**

Total Population	모델의 예측 Positive	모델의 예측 Negative
정답이 Positive	True Positive	False Negative
정답이 Negative	False Positive	True Negative

1. 로지스틱 회귀분석

❖ ROC (Receiver Operating Characteristic) Curve

- **ROC 곡선은 Binary Classifier System(이진 분류 시스템)에 대한 성능 평가 기법**

ROC Curve의 X축

$$False\ Positive\ Rate = \frac{\sum False\ Positive\ 수}{\sum 정답의\ Negative\ 수}$$

Total Population	모델의 예측 Positive	모델의 예측 Negative
정답이 Positive	True Positive	False Negative
정답이 Negative	False Positive	True Negative

ROC Curve의 Y축

$$True\ Positive\ Rate = \frac{\sum True\ Positive\ 수}{\sum 정답의\ Positive\ 수}$$

Total Population	모델의 예측 Positive	모델의 예측 Negative
정답이 Positive	True Positive	False Negative
정답이 Negative	False Positive	True Negative

1. 로지스틱 회귀분석

❖ ROC (Receiver Operating Characteristic) Curve

- **Specificity**
  - 특이도, TNR(True Negative Rate)
  - 실제yes 클래스 데이터 중 모델이 예측한 no 클래스 데이터의 비율

$$Specificity = 1 - FPR$$

- **1 -Specificity**
  - FPR(False Positive Rate)
  - 실제no 클래스 데이터 중 모델이yes 클래스로 예측한 데이터의 비율

$$FPR = \frac{FP}{N}$$

1. 로지스틱 회귀분석

❖ ROC (Receiver Operating Characteristic) Curve

- **ROC 곡선은 Binary Classifier System(이진 분류 시스템)에 대한 성능 평가 기법**

교차표		확진 결과	
		질병 有	질병 無
검사	양성	A	B
	음성	C	D
	합계	A+C	B+D

✓ 민감도(Sensitivity) : 어떤 검사를 시행하였을 경우, 이 검사가 질병 有인 사람 중에서 몇 %를 정말 있다고 할 수 있는가에 대한 수치
→ A/(A+C)

✓ 특이도(Specificity) : 반대로 이 검사가 질병 無인 사람 중에서 몇 %를 정말 없다고 할 수 있는가에 대한 수치
→ D/(B+D)

구분			실제 임신 여부	
			예	아니오
임신검사 (간이테스트)	양성	200	① 20	② 180
	음성	1830	③ 10	④ 1820
	합계	2030	30	2000

양성예측도(positive predictive value; PPV)
- 양성결과 중 실제 임신 비율
- PPV=실제 임신/전체 양성
  =TP/(TP+FP)=20/(20+180)=10.0%

음성예측도(negative predictive value; NPV)
- 양성결과 중 실제로는 임신이 아닌 비율
- NPV=실제 비임신/전체 양성
  =TN/(TN+FN)=180/(20+180)=99.5%

민감도(sensitivity)
- 실제 임신 중 양성결과 비율
- Sensitivity=양성/전체 실제 임신
  =TP/(TP+FN)=20/(20+10)=66.7%

특이도(specificity)
- 실제 비임신 중 음성결과 비율
- Specificity=음성/전체 실제 비임신
  =TN/(TN+FP)=1820/(180+1820)=91.0%

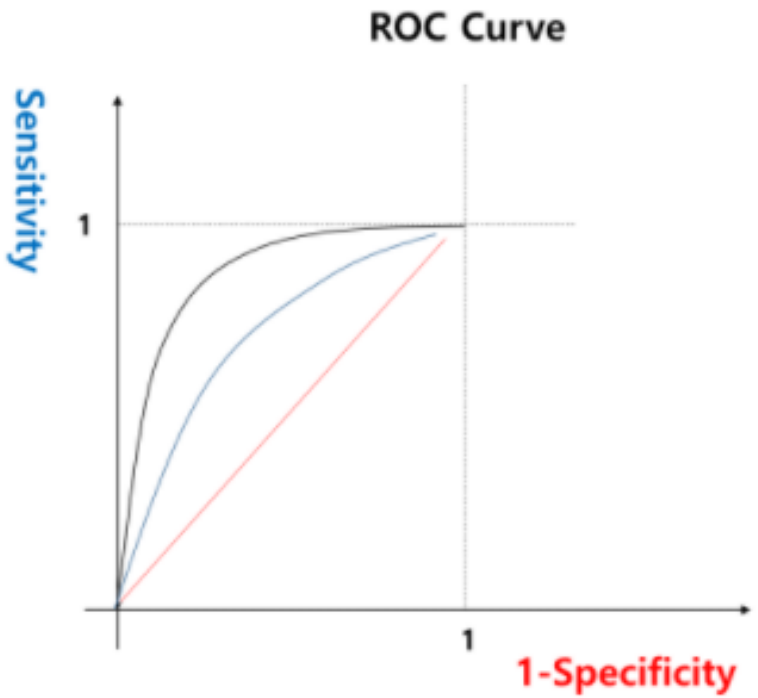
정확도(accuracy)
- 전체 사례 중 검사결과 적중 비율
- Accuracy=적중/전체 사례
  =(20+1820)/(20+180+10+1820)=90.6%

정밀도(precision)
- 양성결과 중 실제 임신 비율
  (양성예측도와 동일)
- 실제 임신/전체 양성=20/(20+180)=10.0%

1. 로지스틱 회귀분석

❖ ROC (Receiver Operating Characteristic) Curve

- 모델이yes 클래스를정확하게예측할수록sensitivity, TPR 값이높아짐
- 빨간색 곡선은 랜덤으로 예측을 한 것이나 마찬가지로 성능이 가장 나쁜 경우
- 곡선이 굽어지면 굽어질수록 AUC가 넓어지므로, 더욱 정확한 모델



Area Under the Curve

ROC curve의 밑면적을 계산한 값

AUC=1.0 ➔ 가장 완벽한 검사 방법

➔ 민감도 및 특이도가 모두 100%

- ✓ 0.90 - 1.00 = Excellent
- ✓ 0.80 - 0.90 = Good
- ✓ 0.70 - 0.80 = Fair
- ✓ 0.60 - 0.70 = Poor
- ✓ 0.50 - 0.60 = Fail

AUC

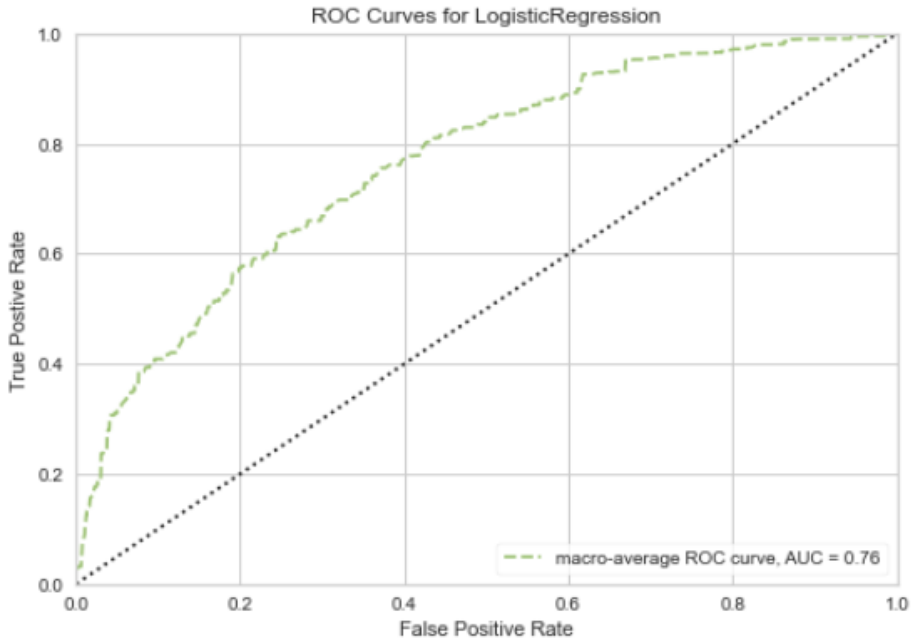
Muller, Matthew P., et al. "Can routine laboratory tests discriminate between severe acute respiratory syndrome and other causes of community-acquired pneumonia?." Clinical infectious diseases 40.8 (2005): 1079-1086.

1. 로지스틱 회귀분석

❖ ROC (Receiver Operating Characteristic) Curve

- **ROC 곡선은 Binary Classifier System(이진 분류 시스템)에 대한 성능 평가 기법**

```
1  from yellowbrick.classifier import ROCAUC
2
3
4  visualizer = ROCAUC(log_reg, classes=[0, 1], micro=False, macro=True, per_class=False)
5  visualizer.fit(train_x, train_y)
6  visualizer.score(train_x, train_y)
7  visualizer.show()
```




```
<matplotlib.axes._subplots.AxesSubplot at 0x1f50f39c400>
```

2. KNN은 최근접 이웃 알고리즘

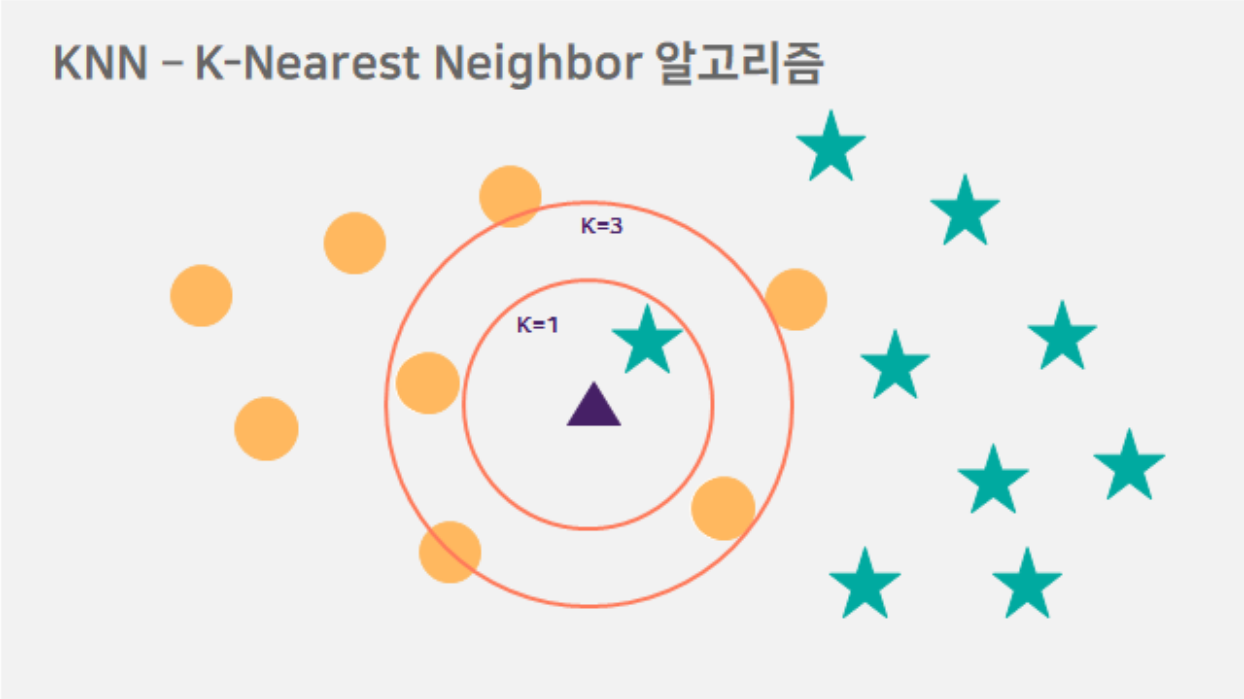
❖ KNN 알고리즘

- 새로운 데이터를 입력 받았을 때 이 데이터와 가장 근접한 데이터들의 종류가 무엇인지 확인하고 많은 데이터의 종류로 분류
- 새로운 데이터와 기존 데이터들간 거리를 측정하고 가까운 데이터들의 종류가 무엇인지 확인하여 새로운 데이터의 종류를 판별

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

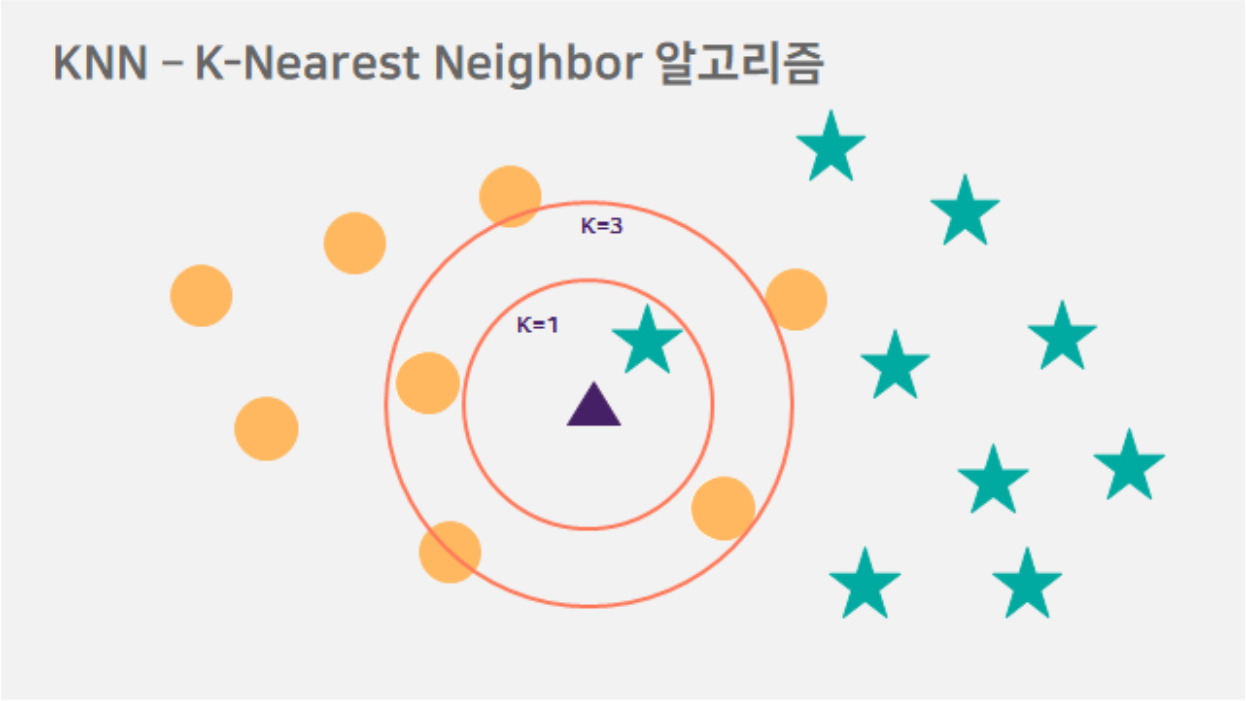
- **K는 인접한 데이터의 개수**
  - K가 1일 때는 가장 가까운 데이터의 종류로 선택 / 세모가 새로운 데이터라면 가장 가까운 별로 분류
  - K가 3일 때는 가장 가까운 데이터의 종류로 선택 / 세모가 새로운 데이터라면 가장 가까운 동그라미2, 별1



2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

- **K는 인접한 데이터의 개수**
  - KNN 알고리즘은 K에 따라 결과가 달라지기 때문에 K를 정해주는 것이 가장 중요한 요소

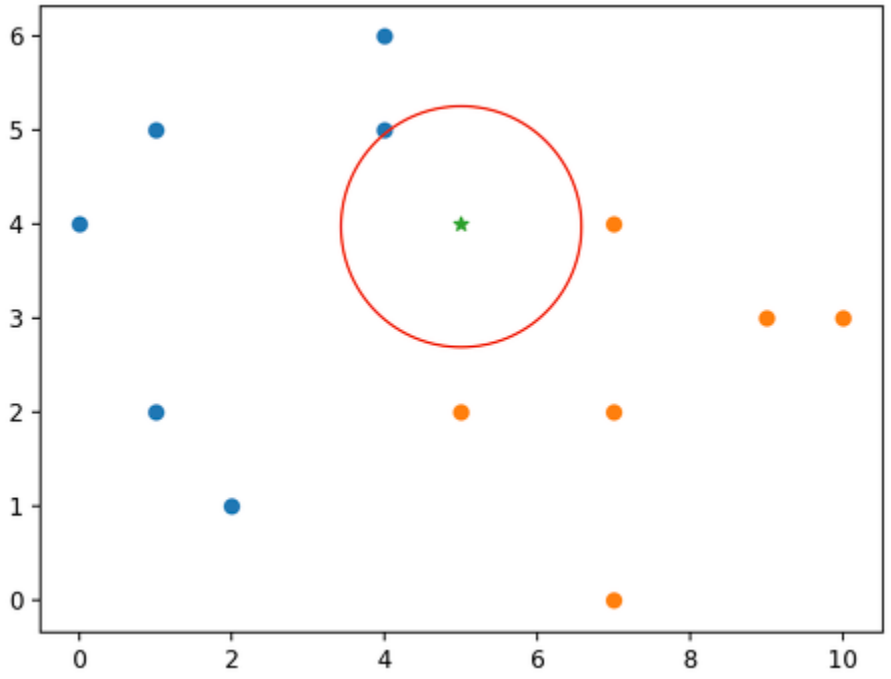


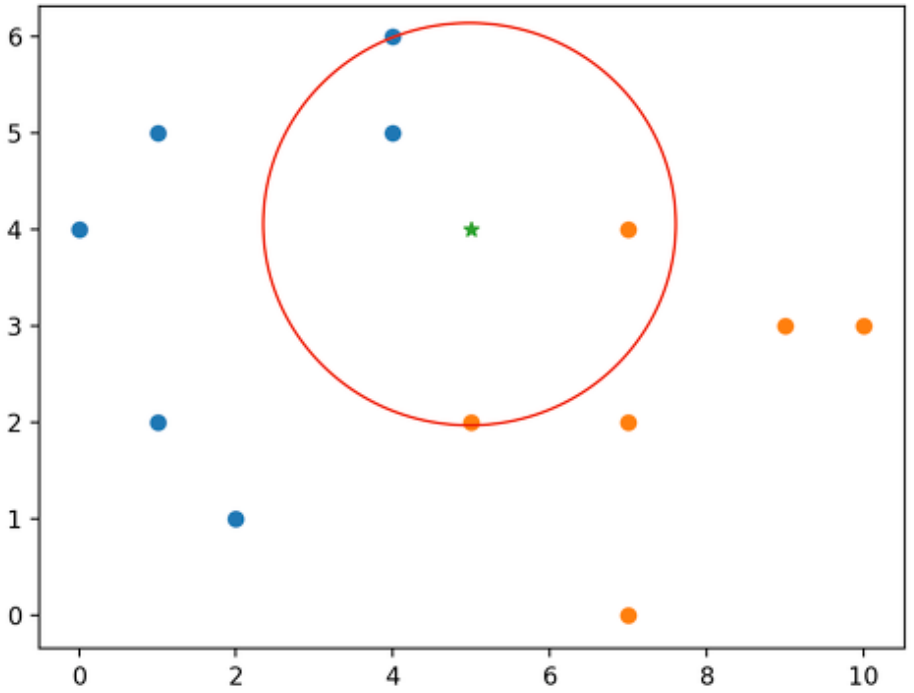
2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

▪ K는 인접한 데이터의 개수

- KNN 알고리즘은 K에 따라 결과가 달라지기 때문에 K를 정해주는 것이 가장 중요한 요소



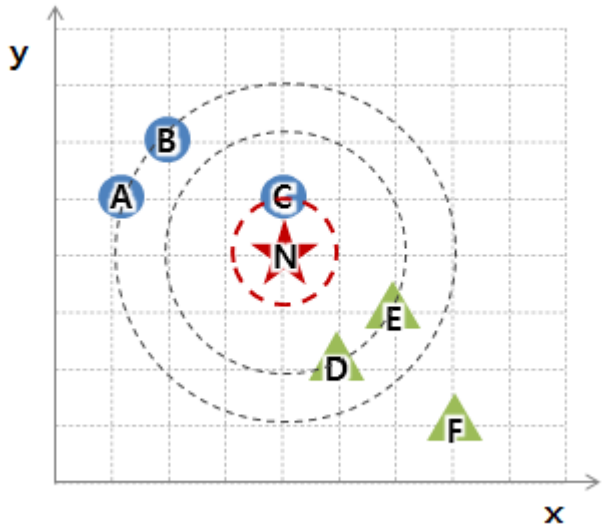


2. KNN은 최근접 이웃 알고리즘

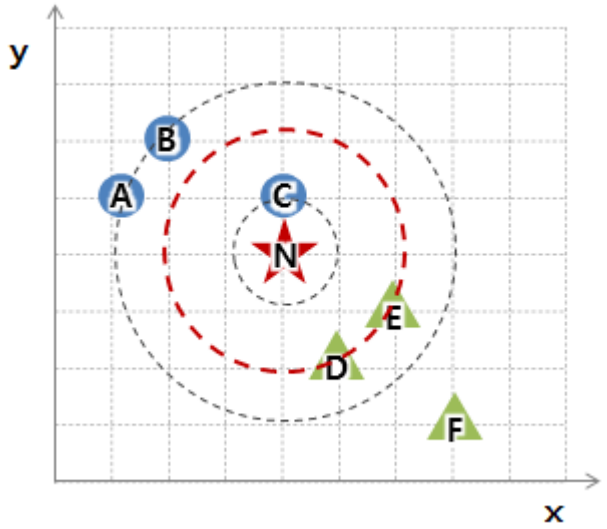
❖ KNN 알고리즘

▪ K는 인접한 데이터의 개수

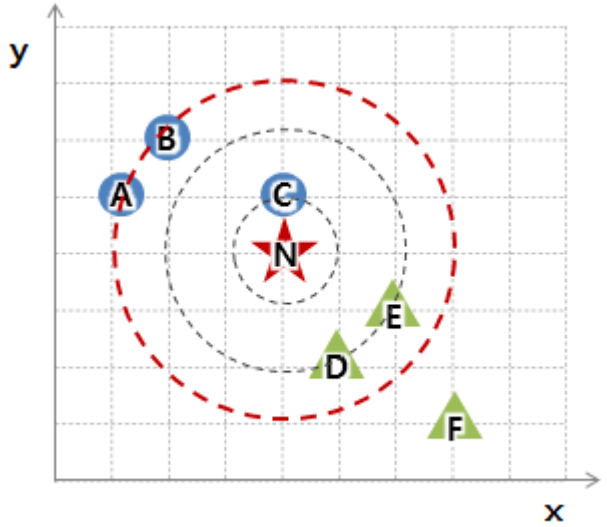
- KNN 알고리즘은 K에 따라 결과가 달라지기 때문에 K를 정해주는 것이 가장 중요한 요소



K=1



K=3



K=5

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

▪ K는 인접한 데이터의 개수

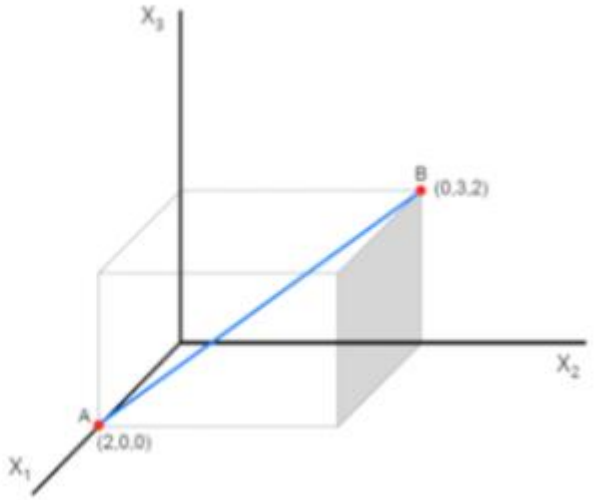
- KNN의 거리를 구하는 공식은 유클리드 거리 공식을 사용

두 점 $A(x_1, y_1)$, $B(x_2, y_2)$ 사이의 거리

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Euclidean Distance

$$X = (x_1, x_2, \ldots, x_n)$$
$$Y = (y_1, y_2, \ldots, y_n)$$
$$d_{(X,Y)} = \sqrt{(x_1 - y_1)^2 + \ldots + (x_n - y_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$



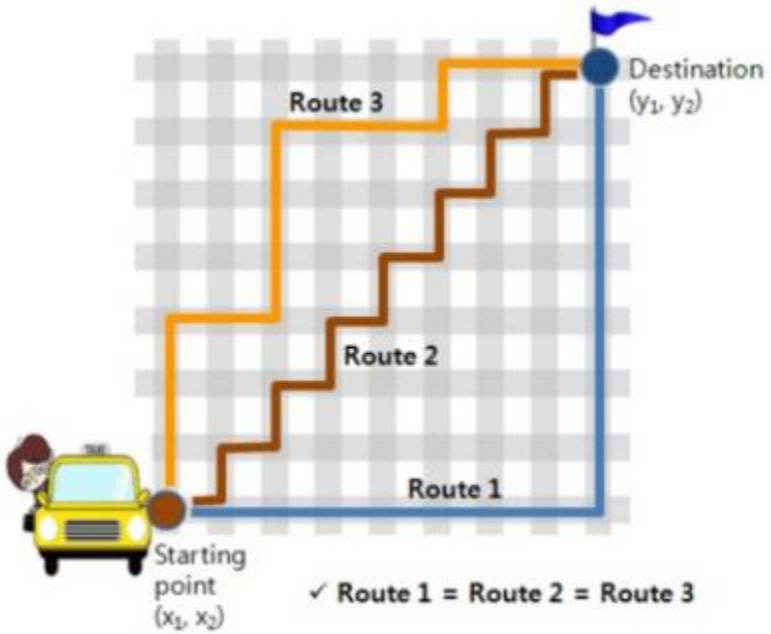
$$d_{(A,B)} = \sqrt{(0-2)^2 + (3-0)^2 + (2-0)^2} = \sqrt{17}$$

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

- **K는 인접한 데이터의 개수**
  - KNN의 거리를 구하는 공식은 **Manhattan Distance** 거리 공식을 사용

$$d_{Manhattan(X,Y)} = \sum_{i=1}^{n} |x_i - y_i|$$

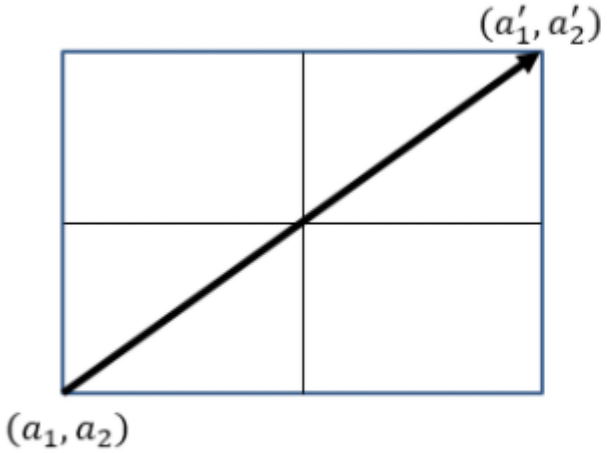


2. KNN은 최근접 이웃 알고리즘

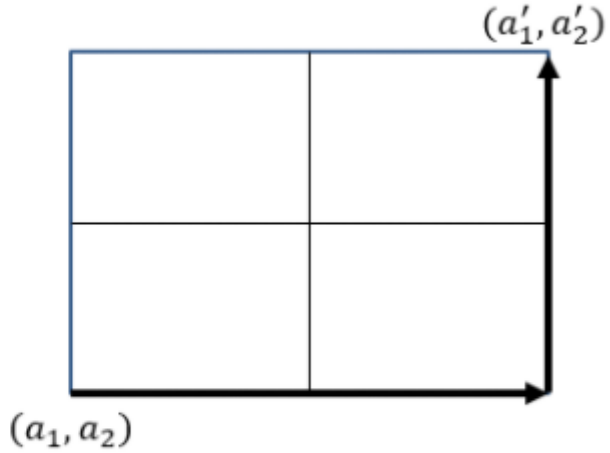
❖ KNN 알고리즘

- **K는 인접한 데이터의 개수**

유클리드 거리



맨하튼 거리



$$d = \sqrt{(a_1' - a_1)^2 + (a_2' - a_2)^2}$$

$$d = (a_1' - a_1) + (a_2' - a_2)$$

$$d = \sqrt[p]{\sum_k (a_k' - a_k)^p}$$

P=1 이면 맨하튼
P=2 이면 유클리드

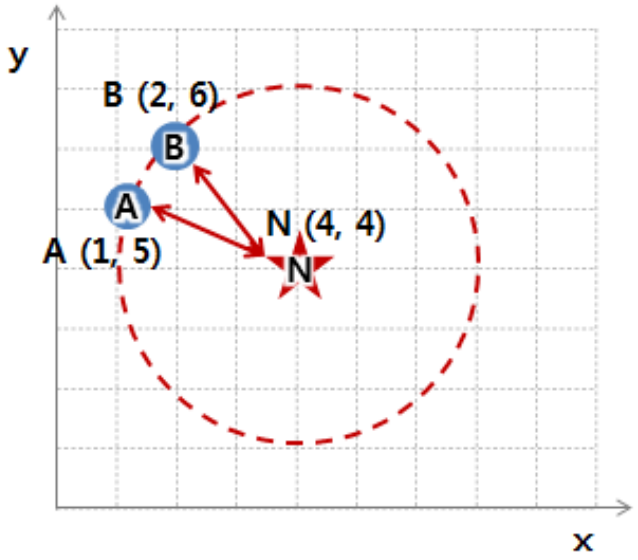
2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

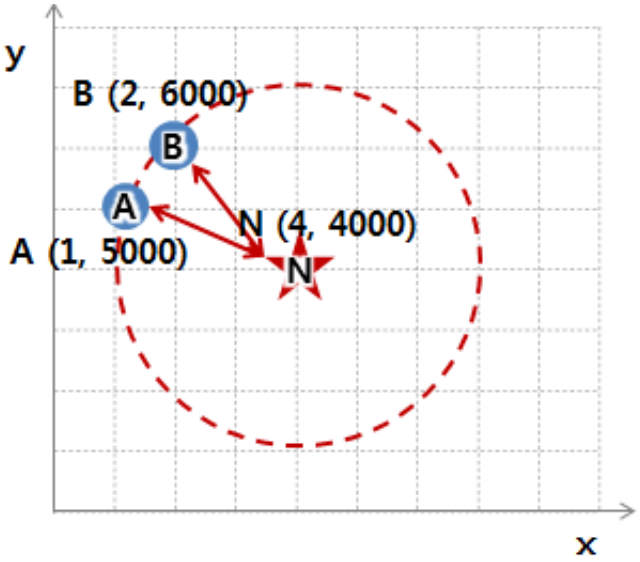
- **정규화(Normalization)**

두 점 $A(x_1, y_1)$, $B(x_2, y_2)$ 사이의 거리

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



A-N 간의 유클리드 거리는 3.162
B-N 간의 유클리드 거리는 2.828



A-N 간의 유클리드 거리는 1000.004
B-N 간의 유클리드 거리는 2000.001

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

▪ 정규화(Normalization)

도시	인구(명)	미세먼지농도(㎍/㎥)
서울	1000만	200
시애틀	67만	40

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

▪ KNN 알고리즘의 장점

- 어떤 분포 든 상관 없음
- 쉽고 이해하기 직관적
- 샘플 수가 많을 때 좋은 분류법

▪ KNN 알고리즘의 단점

- 최적의 k를 선택하기가 어려움
- 데이터가 많을 때 분석속도가 느릴 수 있음
- 특정분포를 가정하지 않기 때문에 샘플수가 많이 있어야 정확도가 좋음

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

- **from** sklearn.neighbors **import** KNeighborsClassifier
- **classifier = KNeighborsClassifier(n_neighbors = 3)**
  - KNeighborsClassifier 모델을 생성해야 하는데, 이 때 n_neighbors로 k를 정해줌
  - metric='minkowski' , p=1,2
- **classifier.fit(X, y)**
  - x 데이터는 여러 개의 차원으로 이루어진 배열(점들의 집합)
    y 데이터는 레이블(각 점들의 분류 결과)/ 0 아니면 1로 분류
- guesses = classifier.predict**()**
- classifier.score**()**
  - 정확도 확인

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
1  from sklearn.neighbors import KNeighborsClassifier
2
3
4  X = [[0], [1], [2], [3]]
5  y = [0, 0, 1, 1]
6  from sklearn.neighbors import KNeighborsClassifier
7  neigh = KNeighborsClassifier(n_neighbors=3)
8  neigh.fit(X, y)
9  print(neigh.predict([[1.1]]))
10 print(neigh.predict_proba([[0.9]]))
11
```

```
[0]
[[0.66666667 0.33333333]]
```

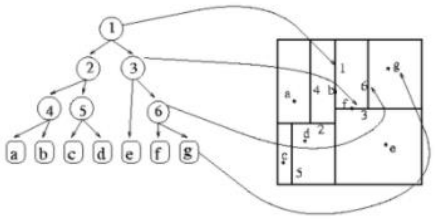
2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

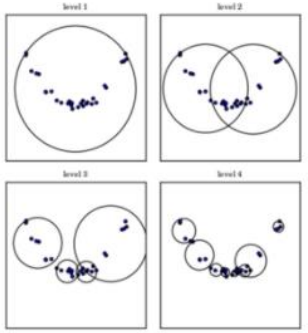
```
1  from sklearn.neighbors import KNeighborsClassifier
2  classifier = KNeighborsClassifier(n_neighbors = 3,weights="distance", metric="euclidean")
3
4
5  training_points = [
6    [0.5, 0.2, 0.1],
7    [0.9, 0.7, 0.3],
8    [0.4, 0.5, 0.7]
9  ]
10 training_labels = [0, 1, 1]
11 classifier.fit(training_points, training_labels)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='distance')
```

- Weight : k개의 이웃 중 거리가 가까운 이웃의 영향을 더 많이 받도록 가중치
- Leaf_size : 트리에서 몇 대 몇으로 나눠서 뻗어 나갈지를 나타내는 값 / 분류 예측 성능
- metric: 거리 측정 방식을 변경하는 매개변수로 default 값은 minkowsi
- metric_params: 메트릭 함수의 추가 키워드로 기본값은 None
- n_jobs: 이웃을 검색하기 위해 실행하는 병렬 작업 수
- n_neighbors: 검색할 이웃의 수로 default 값은 5
- p: minkowski 의 매개변수
- Weights 는 예측에 사용하는 가중치로 uniform 은 각 이웃에 동일한 가중치를 , 'distance'는 가까운 이웃이 멀리 있는 이웃보다 더욱 큰 영향을 줌



kd-tree



ball-tree

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
unknown_points = [
  [0.2, 0.1, 0.7],
  [0.4, 0.7, 0.6],
  [0.5, 0.8, 0.1]
]

import numpy as np

guesses = classifier.predict(unknown_points)
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(training_labels, guesses))


```

```
Accuracy: 0.6666666666666666
```

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
1  from sklearn import datasets
2  import numpy as np
3
4  %matplotlib inline
5
6  # load Iris datasets
7  iris = datasets.load_iris()
8
9  print(iris.feature_names)
10 # here we only select sepal length and width (select first 2 columns)
11 X = iris.data[:, :4]
12 # print(X)
13 y = iris.target
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
```

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(X)
X_scaled = scaler.transform(X)

X=pd.DataFrame(X_scaled,columns =['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'])

# X.columns = iris.feature_names
X.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	0.222222	0.625000	0.067797	0.041667
1	0.166667	0.416667	0.067797	0.041667
2	0.111111	0.500000	0.050847	0.041667
3	0.083333	0.458333	0.084746	0.041667
4	0.194444	0.666667	0.067797	0.041667

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,random_state=30)

clf = KNeighborsClassifier(n_neighbors = 3)
clf.fit(x_train, y_train)

print(clf.score(x_test, y_test))
```

```
0.9333333333333333
```

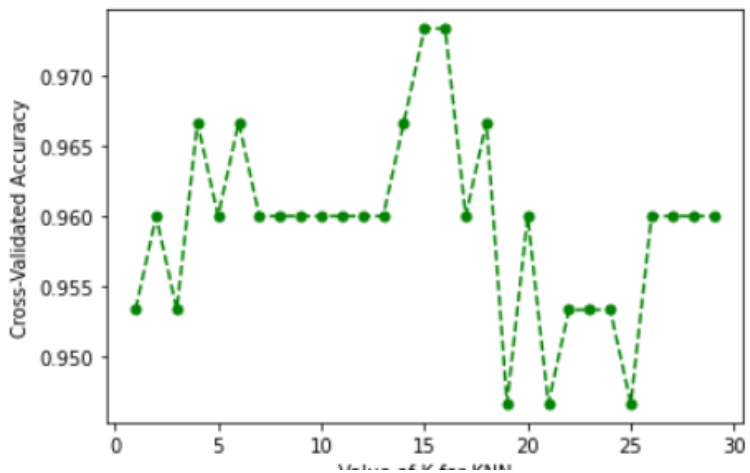
```
from sklearn.model_selection import cross_val_score # K-fold cross-validation module

scores = cross_val_score(classifier,X, y, cv=5, scoring='accuracy')
print(scores)
print(scores.mean())

```

```
[0.96666667 0.96666667 0.93333333 0.9        1.        ]
0.9533333333333334
```

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
from sklearn import model_selection
import matplotlib.pyplot as plt

k_range = range(1, 30)

k_scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = model_selection.cross_val_score(knn, X, y, cv=5, scoring='accuracy')
    k_scores.append(scores.mean())

#Visualizing data
plt.plot(k_range, k_scores, marker='o',color='green',linestyle='dashed',markersize=5)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```



2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
from sklearn.datasets import load_breast_cancer
breast_cancer_data = load_breast_cancer()
```

```
import pandas as pd
X_Data = pd.DataFrame(breast_cancer_data.data)
y = pd.DataFrame(breast_cancer_data.target)
```

```
y.head()
```

	0
0	0
1	0
2	0
3	0
4	0

```
print(breast_cancer_data.target_names)

```

```
['malignant' 'benign']
```

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaler.fit(X_Data)
X_scaled = scaler.transform(X_Data)

X=pd.DataFrame(X_scaled)

X.columns = breast_cancer_data.feature_names
X.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	v
0	1.097064	-2.073335	1.269934	0.984375	1.568466	3.283515	2.652874	2.532475	2.217515	2.255747	...	1.886690	-1.359293	2.303601	2.00
1	1.829821	-0.353632	1.685955	1.908708	-0.826962	-0.487072	-0.023846	0.548144	0.001392	-0.868652	...	1.805927	-0.369203	1.535126	1.89
2	1.579888	0.456187	1.566503	1.558884	0.942210	1.052926	1.363478	2.037231	0.939685	-0.398008	...	1.511870	-0.023974	1.347475	1.45
3	-0.768909	0.253732	-0.592687	-0.764464	3.283553	3.402909	1.915897	1.451707	2.867383	4.910919	...	-0.281464	0.133984	-0.249939	-0.55
4	1.750297	-1.151816	1.776573	1.826229	0.280372	0.539340	1.371011	1.428493	-0.009560	-0.562450	...	1.298575	-1.466770	1.338539	1.22

5 rows × 30 columns

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
1 from sklearn.model_selection import train_test_split
2 X_train, X_test , y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

```
1 print(len(X_train))
2 print(len(X_test))
3 print(len(y_train))
4 print(len(y_test))
```

```
398
171
398
171
```

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
1 from sklearn.neighbors import KNeighborsClassifier
2 classifier = KNeighborsClassifier(n_neighbors = 3)
```

```
1 classifier.fit(X_train, y_train)
```

```
C:₩Users₩choi₩Anaconda3₩envs₩tensor-gpu₩lib₩site-packages₩ipykernel_launcher.py:1: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples, ), for example using ravel().
  """Entry point for launching an IPython kernel.
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

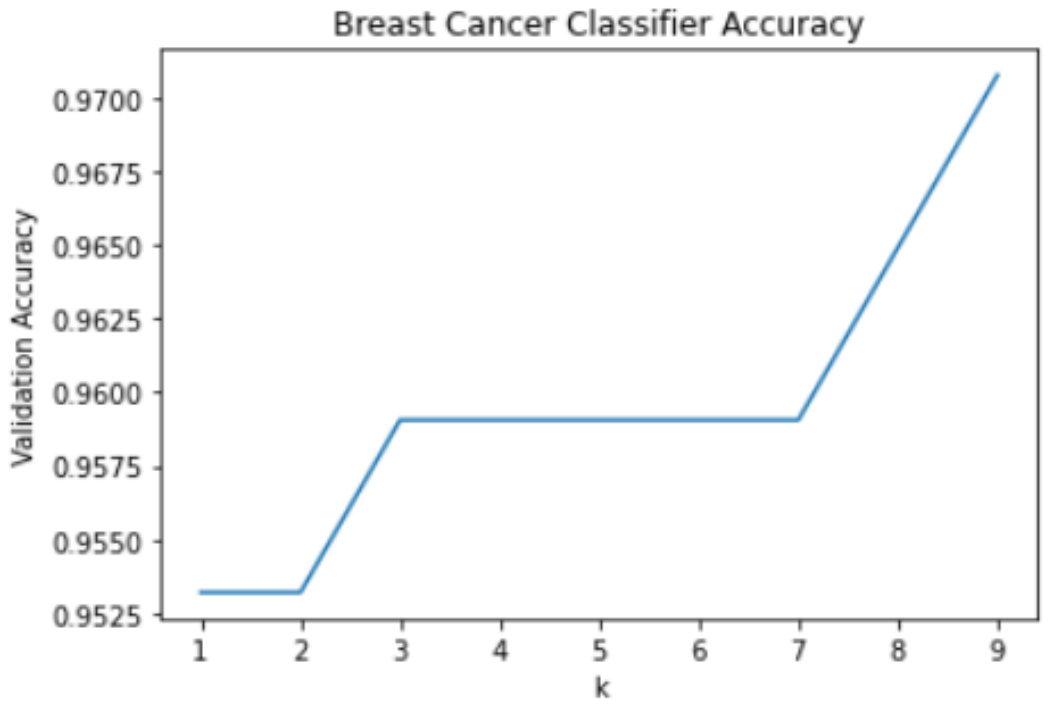
```
1 print(classifier.score(X_test, y_test))
```

```
0.9590643274853801
```

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

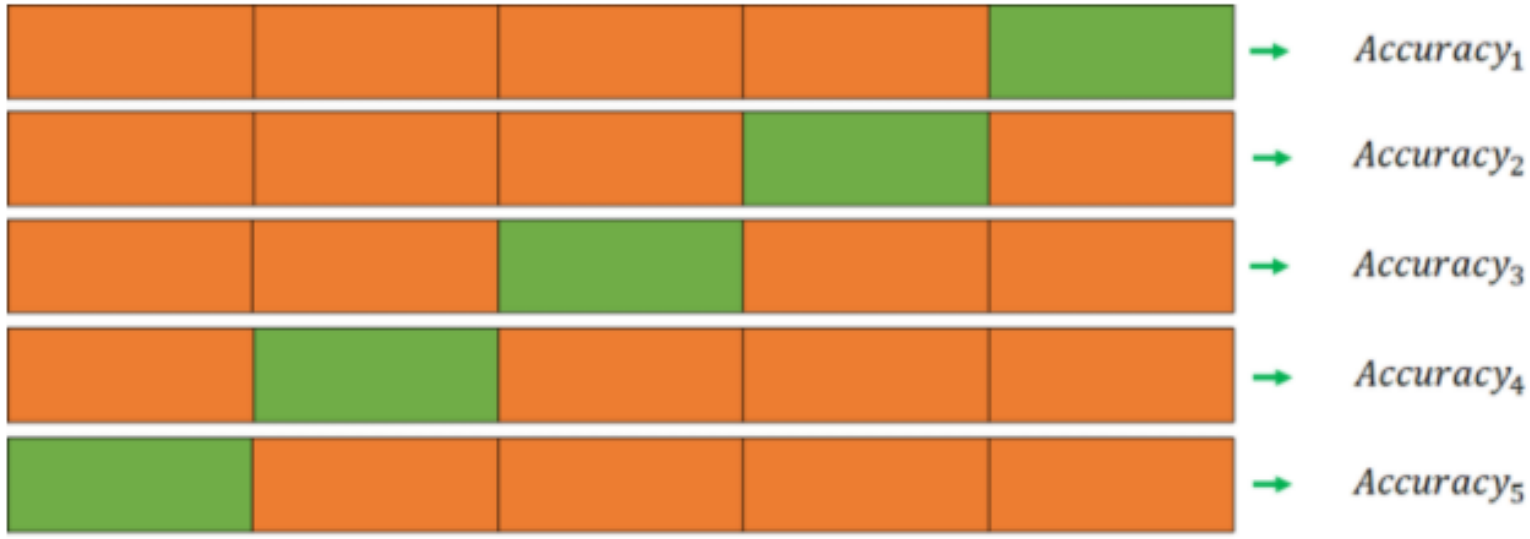
```
import matplotlib.pyplot as plt
k_list = range(1,10)
accuracies = []
for k in k_list:
    classifier = KNeighborsClassifier(n_neighbors = k)
    classifier.fit(X_train, y_train)
    accuracies.append(classifier.score(X_test, y_test))
plt.plot(k_list, accuracies)
plt.xlabel("k")
plt.ylabel("Validation Accuracy")
plt.title("Breast Cancer Classifier Accuracy")
plt.show()
```



2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

- k-겹 교차 검증(k-fold cross validation)



$$Accuracy = Average(Accuracy_1, \cdots, Accuracy_k)$$

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

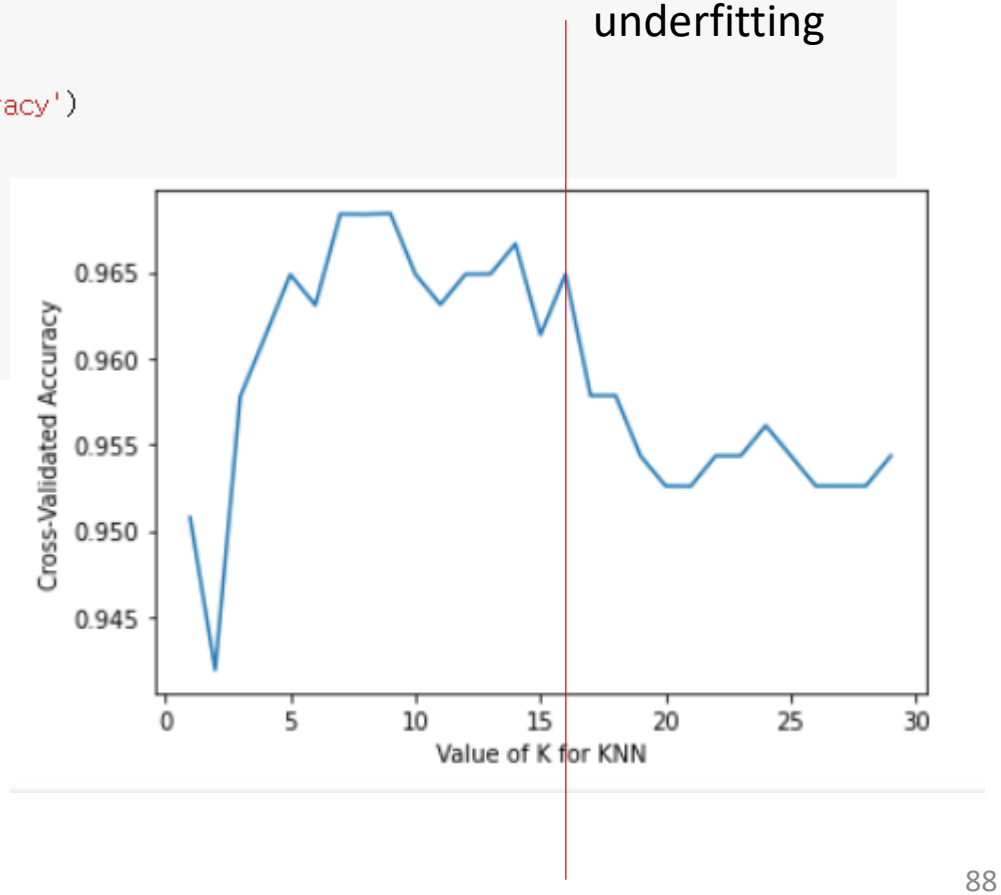
```
1 from sklearn.model_selection import cross_val_score # K-fold cross-validation module
2
3 scores = cross_val_score(classifier,X, y, cv=5, scoring='accuracy')
4 print(scores)
5 print(scores.mean())
6
```

```
[0.97368421 0.95614035 0.98245614 0.94736842 0.92920354]
0.9577705325260053
```

2. KNN은 최근접 이웃 알고리즘

❖ KNN 알고리즘

```
from sklearn import model_selection
import matplotlib.pyplot as plt

k_range = range(1, 30)

k_scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = model_selection.cross_val_score(knn, X, y, cv=5, scoring='accuracy')
    k_scores.append(scores.mean())

#Visualizing data
plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```



2. KNN은 최근접 이웃 알고리즘

❖ KNN 회귀 알고리즘

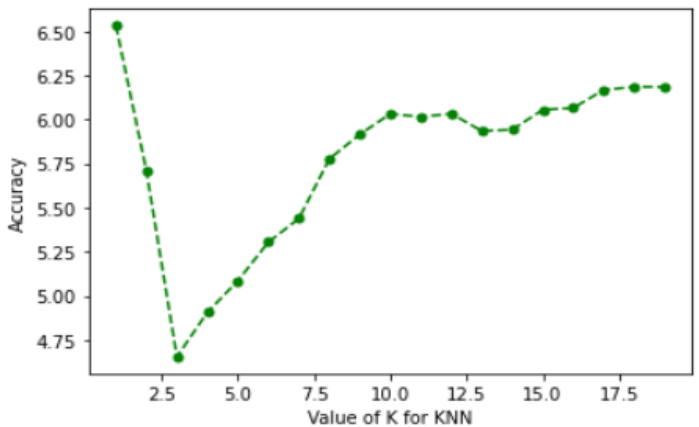
```
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn import neighbors
from sklearn.metrics import mean_squared_error
from math import sqrt
import pandas as pd

boston = load_boston()
X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target ,test_size=0.2, random_state=42)
print(X_train.shape, X_test.shape)

k_range = range(1, 20)

rmse_val = [] #to store rmse values for different k
for K in range(1, 20):

    model = neighbors.KNeighborsRegressor(n_neighbors = K)
    model.fit(X_train, y_train)  #fit the model
    pred=model.predict(X_test) #make prediction on test set
    error = sqrt(mean_squared_error(y_test,pred)) #calculate rmse
    rmse_val.append(error) #store rmse values
    print('RMSE value for k= ' , K , 'is:', error)

plt.plot(k_range, rmse_val, marker='o',color='green',linestyle='dashed',markersize=5)
plt.xlabel('Value of K for KNN')
plt.ylabel('Accuracy')
plt.show()
```

2. KNN은 최근접 이웃 알고리즘

❖ KNN 회귀 알고리즘

```python
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn import neighbors
from sklearn.metrics import mean_squared_error
from math import sqrt
import pandas as pd

boston = load_boston()
X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target ,test_size=0.2,
print(X_train.shape, X_test.shape)

k_range = range(1, 20)

rmse_val = [] #to store rmse values for different k
for K in range(1, 20):

    model = neighbors.KNeighborsRegressor(n_neighbors = K)
    model.fit(X_train, y_train)  #fit the model
    pred=model.predict(X_test) #make prediction on test set
    error = sqrt(mean_squared_error(y_test,pred)) #calculate rmse
    rmse_val.append(error) #store rmse values
    print('RMSE value for k= ' , K , 'is:', error)

plt.plot(k_range, rmse_val, marker='o',color='green',linestyle='dashed',markersize=5)
plt.xlabel('Value of K for KNN')
plt.ylabel('Accuracy')
plt.show()
```

```
(404, 13) (102, 13)
RMSE value for k=  1 is: 6.533458382156615
RMSE value for k=  2 is: 5.709593749498621
RMSE value for k=  3 is: 4.6539825286421435
RMSE value for k=  4 is: 4.906214796301827
RMSE value for k=  5 is: 5.0852851926117255
RMSE value for k=  6 is: 5.3050110934403385
RMSE value for k=  7 is: 5.437776362145272
RMSE value for k=  8 is: 5.7766325625103505
RMSE value for k=  9 is: 5.9162730135904535
RMSE value for k=  10 is: 6.032231158128644
RMSE value for k=  11 is: 6.015260127343712
RMSE value for k=  12 is: 6.032981417805106
RMSE value for k=  13 is: 5.934959335454073
RMSE value for k=  14 is: 5.943177815362587
RMSE value for k=  15 is: 6.055387525401548
RMSE value for k=  16 is: 6.0659727962688335
RMSE value for k=  17 is: 6.1682431059235165
RMSE value for k=  18 is: 6.185559955314015
RMSE value for k=  19 is: 6.184915007192077
```



감사합니다