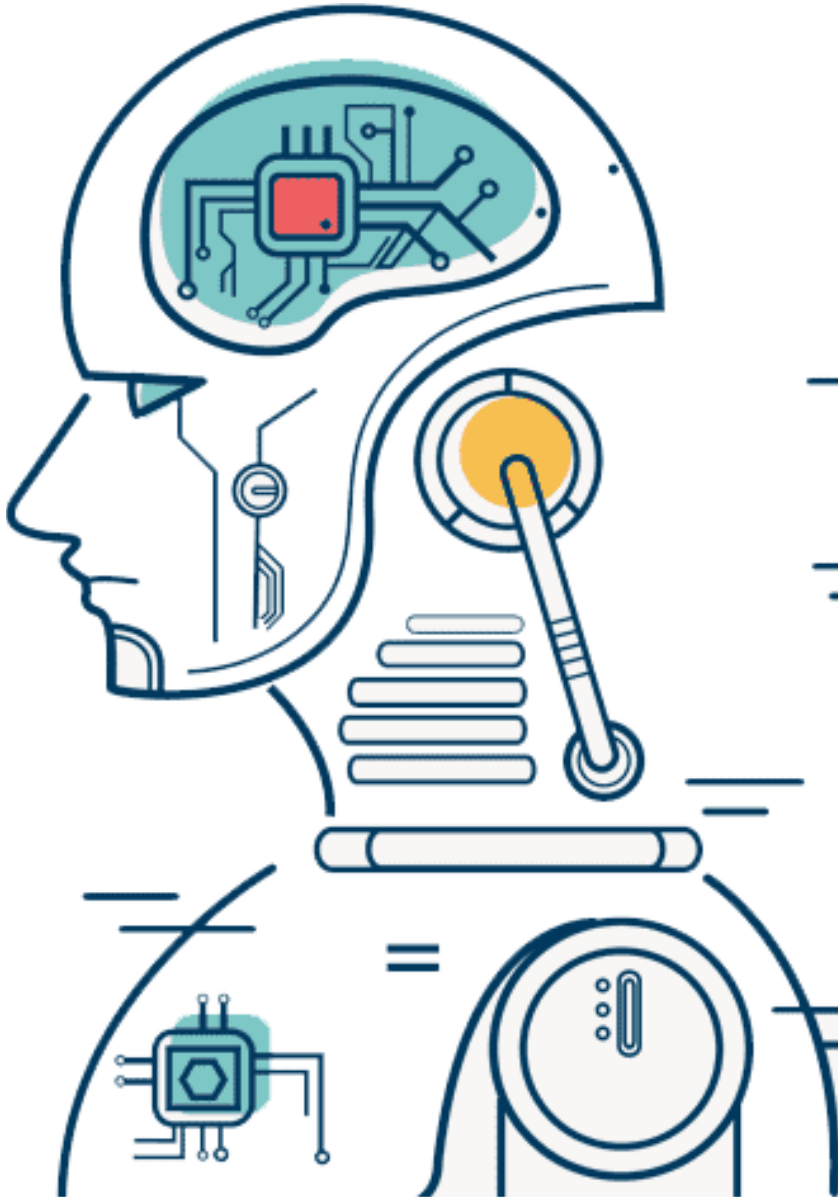
스마트인재개발원
Smart Human Resources Development

Machine Learning

Chapter03 Decision Tree, Label Encoding, One-hot Encoding, Cross validation



강사 손지영

학습목표

스마트인재개발원
Smart Human Resources Development

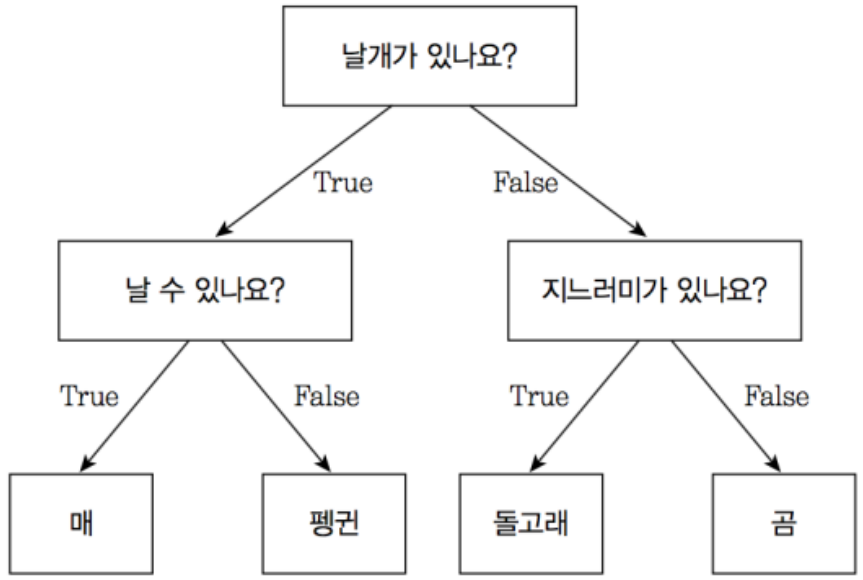
- Decision Tree 알고리즘을 이해 할 수 있다.
- 결측치가 있는지 확인할 수 있다.
- Label 인코딩과 One-hot 인코딩을 이해 할 수 있다.
- 정답(레이블)과 연관 관계가 높은 특성을 선택할 수 있다.
- 교차 검증 기법을 이해 할 수 있다.

Decision Tree

Decision Tree란?
의사결정 나무

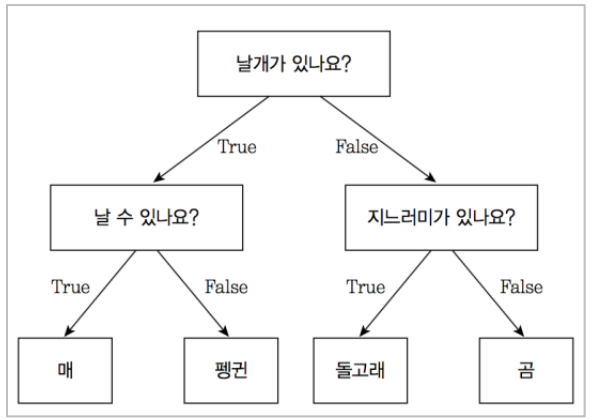
- 스무고개 하듯이 예/아니오 질문을 반복하며 학습
- 특정 기준(질문)에 따라 데이터를 구분하는 모델
- 분류와 회귀에 모두 사용 가능

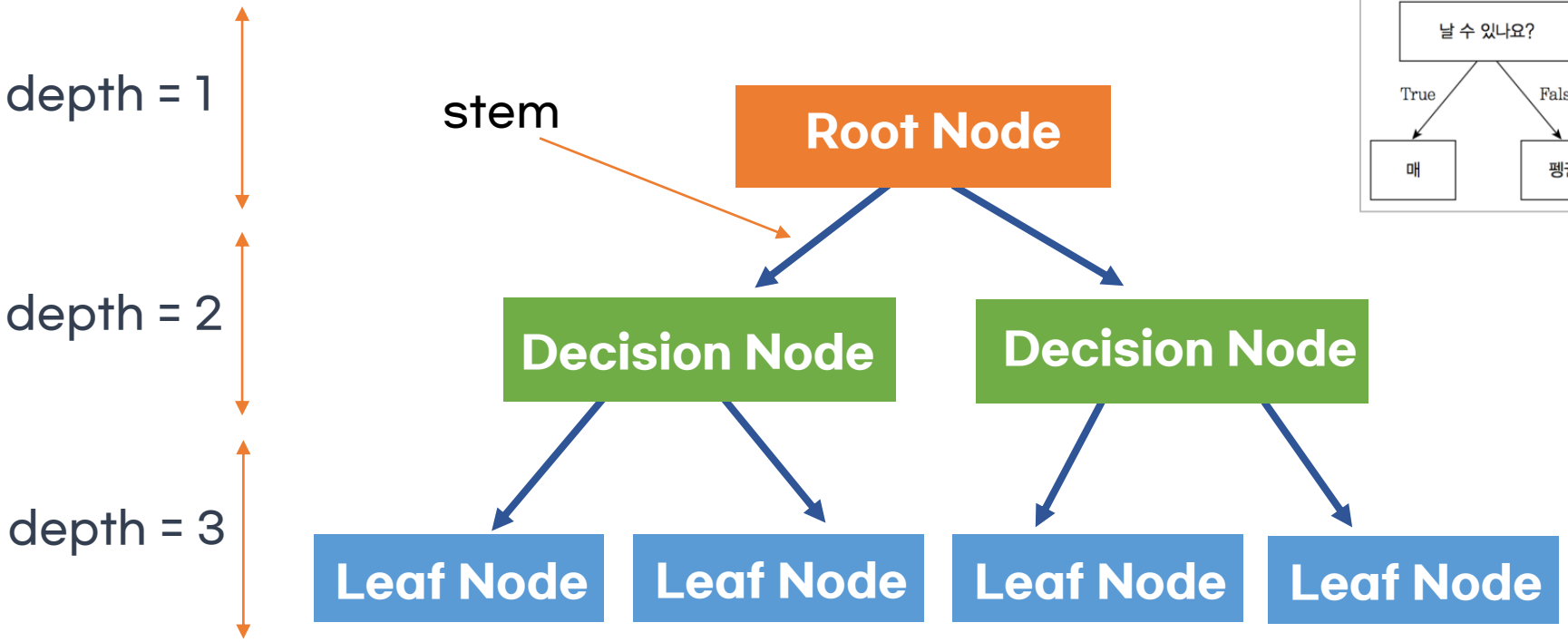
예) 매, 펭귄, 돌고래, 곰을 구분



Decision Tree 용어 정리

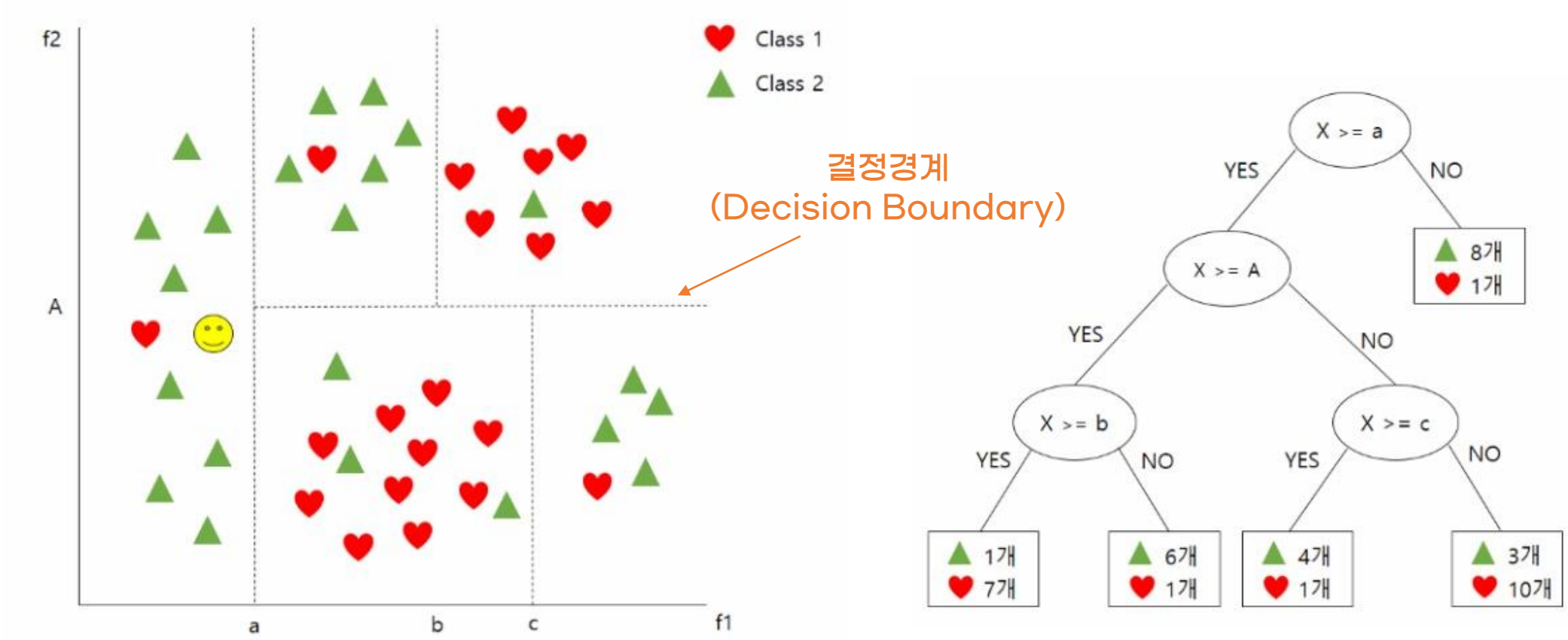
의사결정 나무





Decision Tree 최적의 분리를 찾는 방법: 불순도가 낮아지는 방향

의사결정 나무



Decision Tree 최적의 분리를 찾는 방법: 불순도가 낮아지는 방향

의사결정 나무

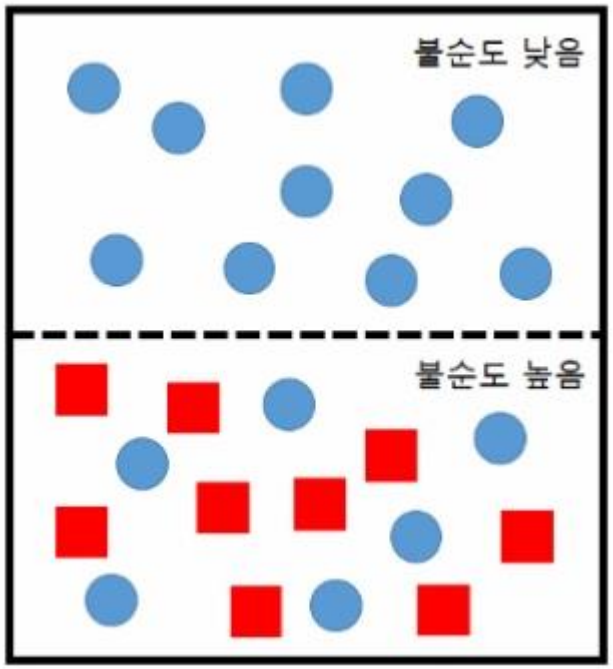
분리 후 결과가 하나의 클래스에만 속하는 것

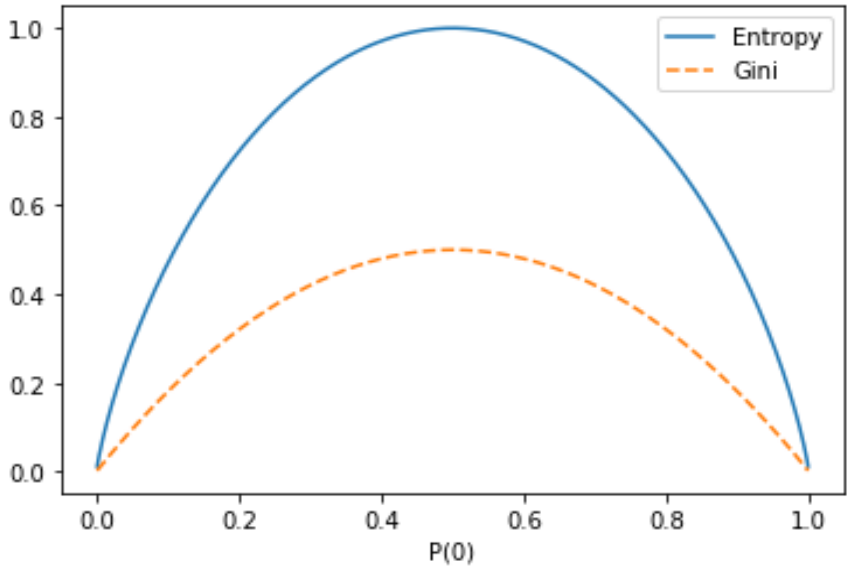
즉, 어떤 기준을 통해 나타난 결과의 불순도(impurity)가 낮을수록 더 좋은 의사결정 나무

- 최적의 의사결정 나무는 어떻게 찾는가?
- 무엇을 기준으로 평가할 수 있는가?

Decision Tree 최적의 분리를 찾는 방법
의사결정 나무

- 불순도 측정: Gini Index (지니계수) $G(S) = 1 - \sum_{i=1}^{c} p_i^2$ 1-((5/10)**2+(5/10)**2)=0.5





Decision Tree 사용법
의사결정 나무

```
# 필요한 라이브러리 임폴트
from sklearn.tree import DecisionTreeClassifier


# 결정 트리 분류 모델 생성 함수 호출, 결정 트리 분류 모델 객체 생성
clf = DecisionTreeClassifier(하이퍼 파라미터, random_state)
```

Decision Tree 주요 매개변수(hyperparameter)

의사결정 나무

```
DecisionTreeClassifier(
criterion,
max_depth,
min_samples_split,
min_samples_leaf,
max_leaf_nodes)
```

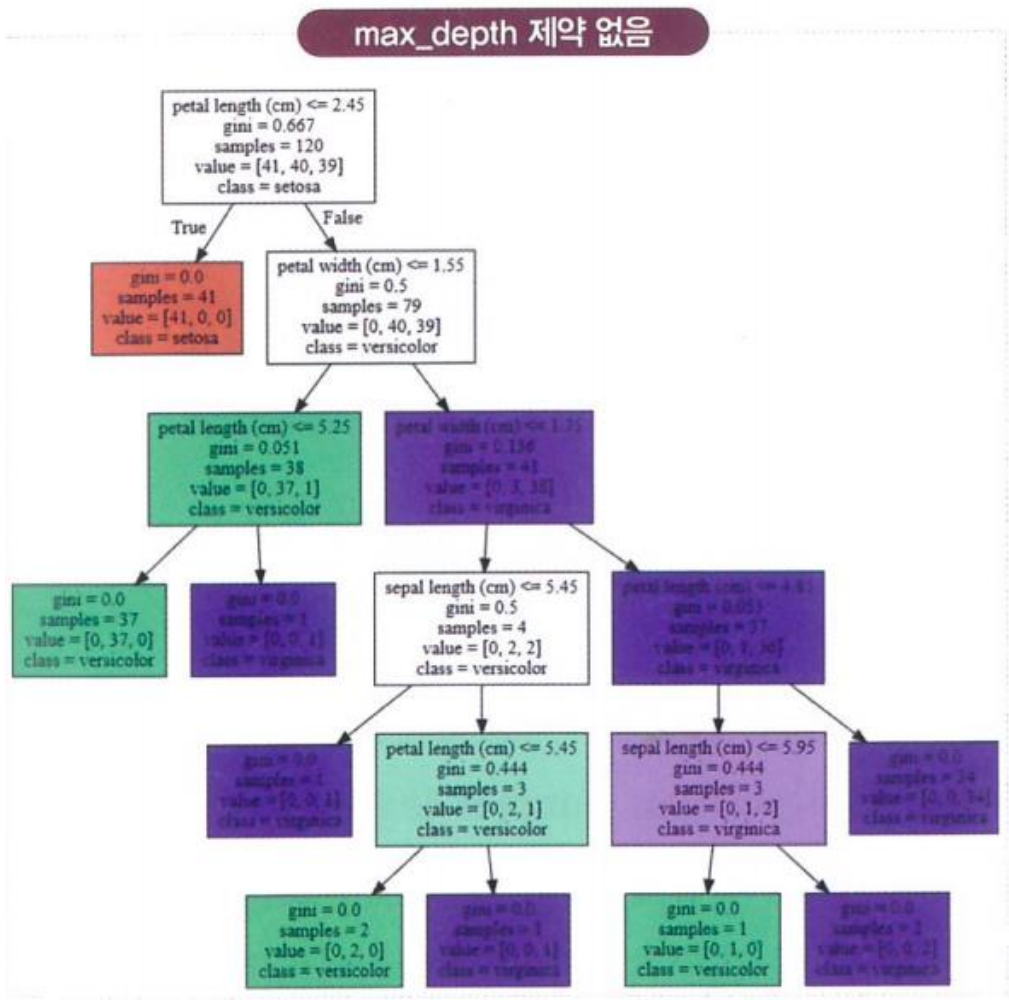
- criterion → 불순도 측정 방법 (gini, entropy)
- max_depth → 트리의 최대 깊이
- min_samples_split → 노드를 분할하기 위한 최소 샘플 수
- min_samples_leaf → 리프 노드가 가져야할 최소 샘플 수
- max_leaf_nodes → 리프 노드의 최대 개수

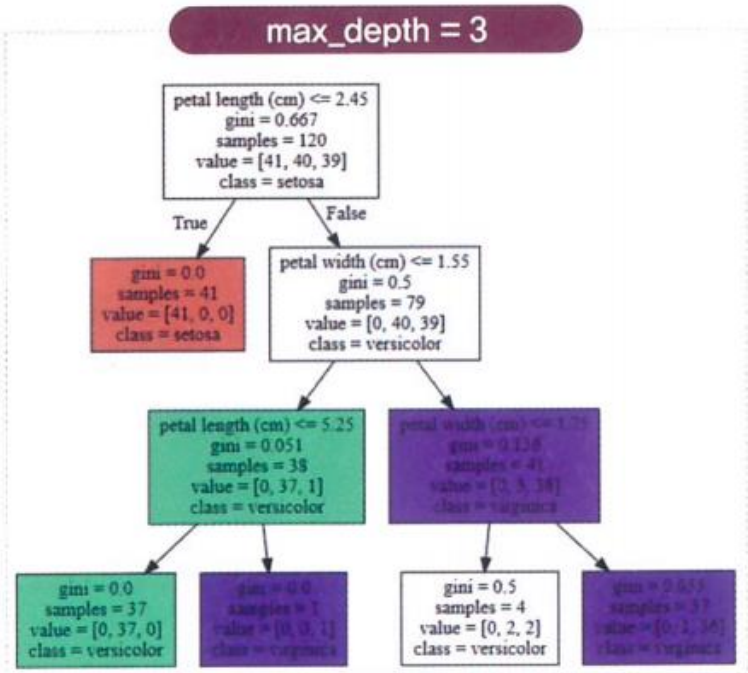
Decision Tree 과대적합 제어
의사결정 나무

하이퍼 파라미터	설명
max_depth (트리의 최대 깊이)	값이 클수록 모델의 복잡도 상승
max_leaf_nodes (리프 노드의 최대 개수)	크게 설정될수록 분할되는 노드 증가
min_samples_split (노드를 분할하기 위한 최소 샘플 수)	작게 설정될수록 분할되는 노드 증가, 복잡도 상승
min_samples_leaf (리프 노드가 가져야할 최소 샘플 수)	작게 설정될수록 분할되는 노드 증가, 복잡도 상승

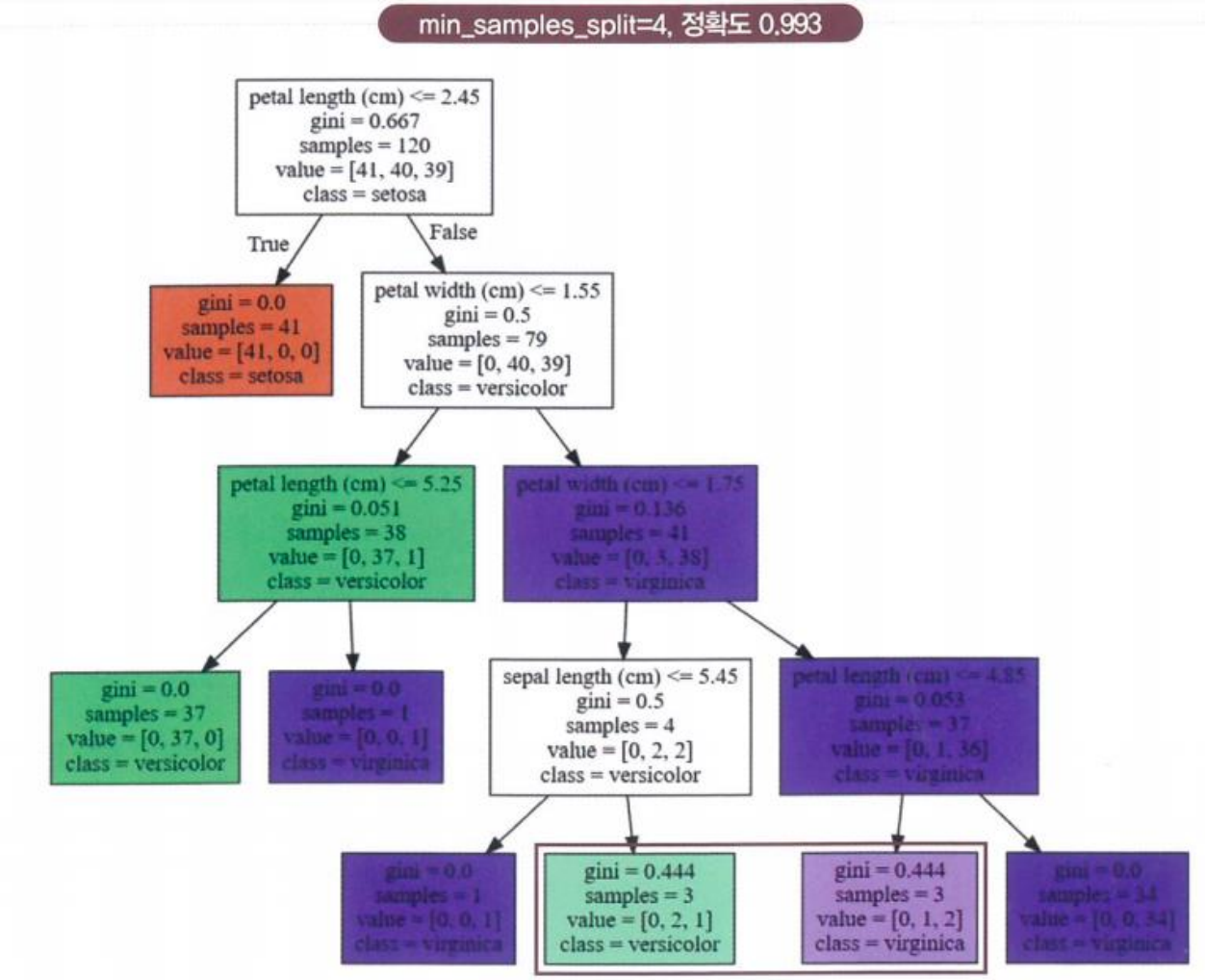
Decision Tree 사전 가지치기

의사결정 나무

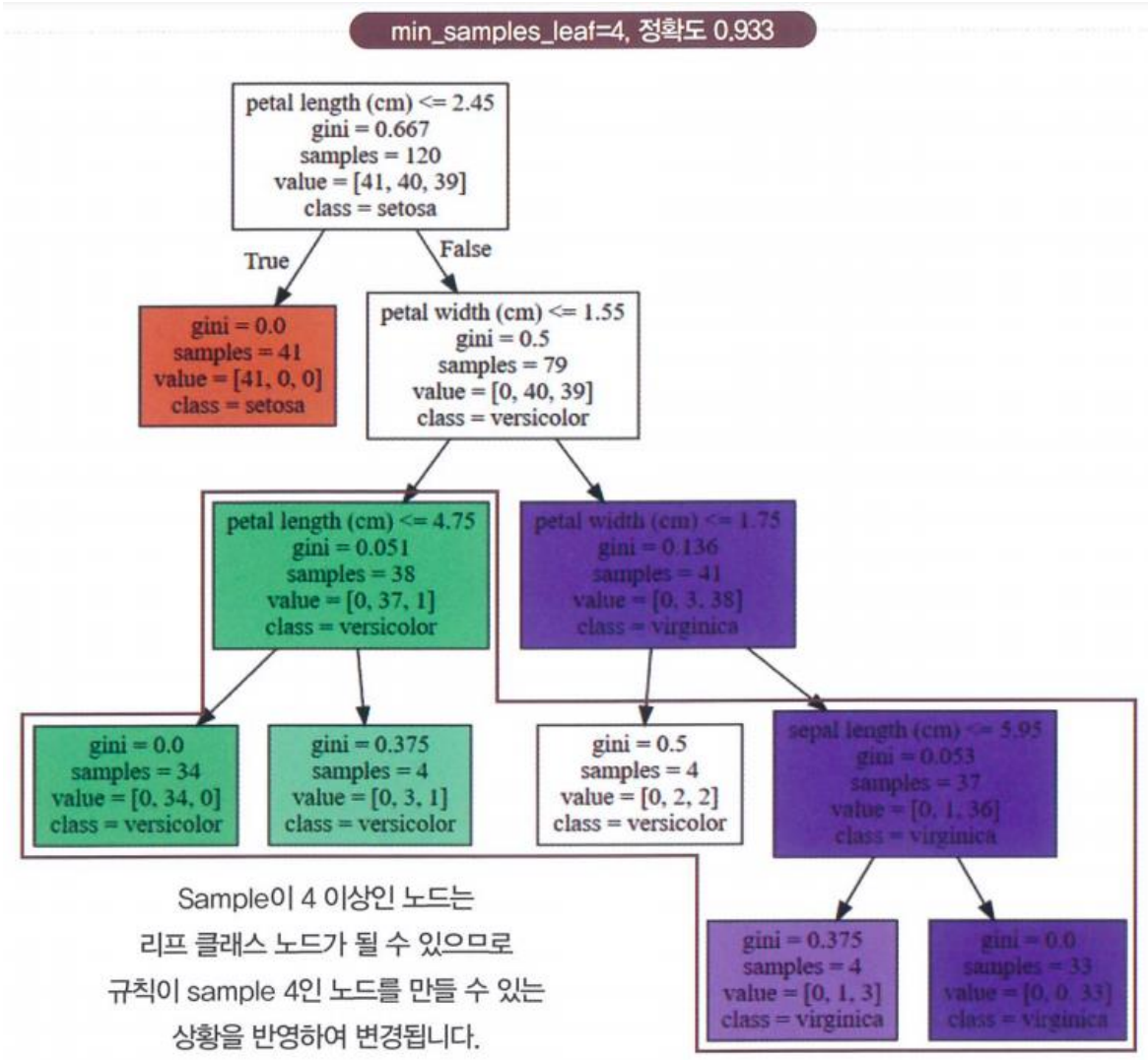




Decision Tree 사전 가지치기
의사결정 나무



Decision Tree 사전 가지치기
의사결정 나무



Decision Tree 장단점

의사결정 나무

- 쉽고, 직관적인 모델
- 해석 가능한 모델(예측에 대한 설명이 가능)
- 과대적합이 발생하기 쉬움
- 과대적합을 극복하기 위해 사전에 하이퍼 파라미터를 제한하는 튜닝이 필요
  (사전 가지치기)

Decision Tree 사용
의사결정 나무

```
# trian_test_split() : 데이터를 학습용, 테스트용으로 분할
X_train,X_test, y_train, y_test = train_test_split(X,y, random_state = 7)

# 결정트리 분류 모델 생성 함수 호출, 하이퍼 파라미터는 전부 기본값 설정,
분류 모델 객체 생성
from sklearn.tree import DecisionTreeClassifier
tree_model = DecisionTreeClassifier()

# 분류 모델 학습 진행
tree_model.fit(X_train, y_train)
```

Decision Tree 사용
의사결정 나무

```
# 테스트 데이터를 이용해서 예측
y_pred = tree_model.predict(X_test)
```

```
# 테스트 데이터에 대한 분류 모델의 성능(평균 정확도) 확인
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

Decision Tree 사용
의사결정 나무

특성 선택 : 정답(레이블)과 연관 관계가 높은 특성을 선택하는 작업

```
# 의사결정나무 모델 특성 중요도 확인
importance = tree_model.feature_importances_

# 보기 쉽게 DataFrame으로 변경
df = pd.DataFrame(importance,
                  index=X_one_hot.columns)

# 특성 중요도 높은 순부터 출력되게 내림차순 정렬
df.sort_values(by='name', ascending=False)
```

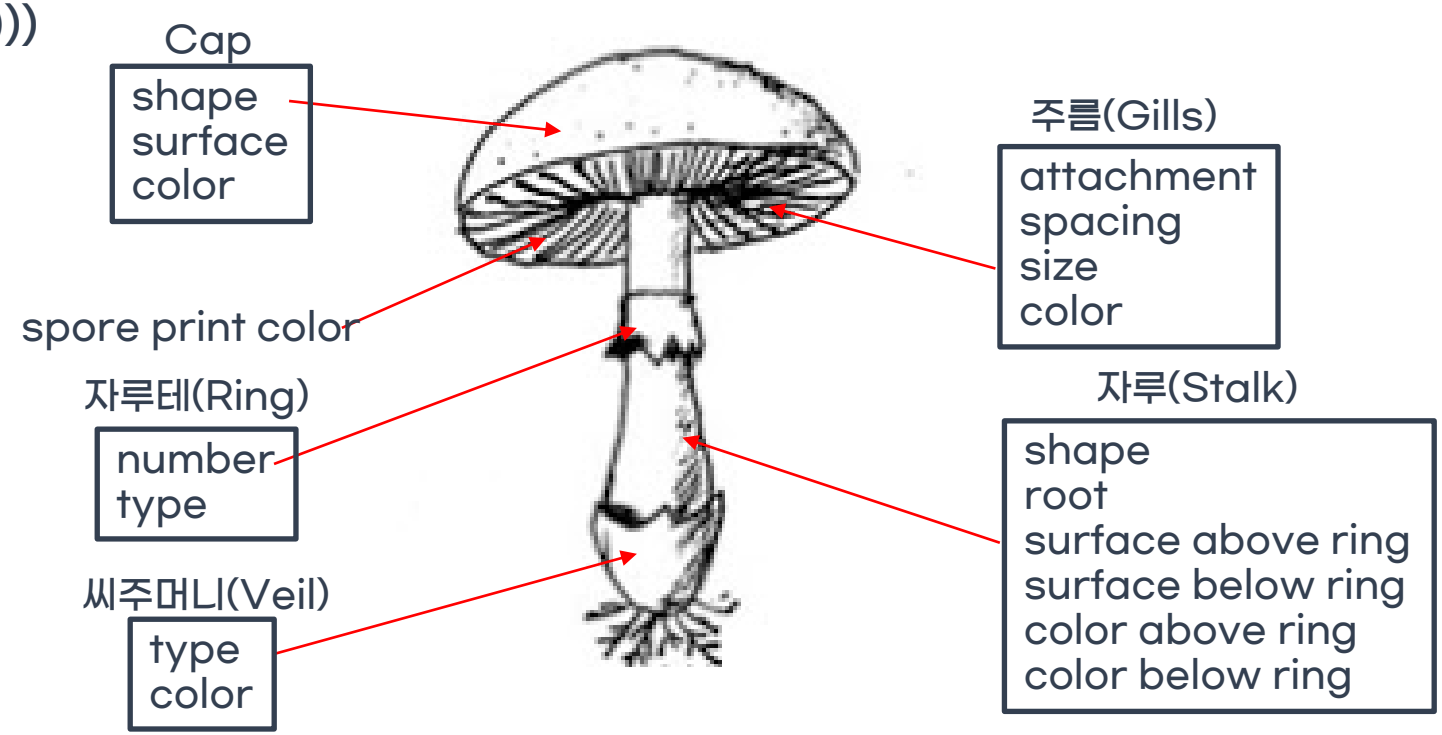
```
odor_n                        0.611853
stalk-root_c                  0.177321
stalk-root_r                  0.089846
spore-print-color_r           0.033995
odor_a                        0.023077
odor_l                        0.022260
stalk-surface-below-ring_y    0.017375
stalk-surface-above-ring_k    0.013956
ring-number_o                 0.005403
cap-surface_g                 0.002102
```

Decision Tree 실습
의사결정 나무

Mushroom 데이터 이용
Decision Tree 독성/식용 분류 실습

Mushroom Dataset

- 8124개의 버섯 종류 데이터
- 22개의 특징 (18개의 버섯 특징, 4개의 다른 특징 (Habitat (서식지), Population(분포 형태), Bruises(타박상), Odor(냄새)))
- 라벨 : 독성 (p), 식용(e)



Mushroom Dataset - 컬럼 및 데이터 의미

poisonous : 독버섯(poisonous), 식용버섯(edible)
cap-shape : 갓 모양(b,c,x,f,k,s) : 원뿔/평면/볼록 등
cap-surface : 갓 표면(f,g,y,s) : 섬유질/비늘모양/부드러움 등
cap-color : 갓 색(n,b,c,g,r,p,u,e,w,y) : 계피/회색/노란색 등
bruises : 타박상(t,f) : 예/아니오
odor : 냄새(a,l,c,y,f,m,n,p,s) : 아몬드,생선,매운 등
gill-attachment(자실층 위치), gill-spacing(자실층 간격), gill-size(자실층 크기), gill-color(자실층 색), stalk-shape(자루 모양), stalk-root(자루 뿌리), stalk-surface-above-ring(자루 표면 위 자루테), stalk-surface-below-ring(자루 표면 아래 자루테), stalk-color-above-ring(자루 색 위 자루테), stalk-color-below-ring(자루 색 아래 자루테), veil-type(베일 유형), veil-color(베일 색), ring-number(링 번호), ring-type(링 타입), spore-pring-color(포자 색), population(인구), habitat(서식지)

Mushroom Dataset - 정보 확인

함수1. info() 함수 - 결측치 확인

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
poisonous                 8124 non-nul  object
cap-shape                 8124 non-nul  object
cap-surface               8124 non-nul  object
cap-color                 8124 non-nul  object
bruises                   8124 non-nul  object
odor                      8124 non-nul  object
gill-attachment           8124 non-nul  object
gill-spacing              8124 non-nul  object
gill-size                 8124 non-nul  object
gill-color                8124 non-nul  object
stalk-shape               8124 non-nul  object
stalk-root                8124 non-nul  object
stalk-surface-above-ring  8124 non-nul  object
stalk-surface-below-ring  8124 non-nul  object
stalk-color-above-ring    8124 non-nul  object
stalk-color-below-ring    8124 non-nul  object
veil-type                 8124 non-nul  object
veil-color                8124 non-nul  object
ring-number               8124 non-nul  object
ring-type                 8124 non-nul  object
spore-print-color         8124 non-nul  object
population                8124 non-nul  object
habitat                   8124 non-nul  object
dtypes: object(23)
memory usage: 1.4+ MB
```

Mushroom Dataset - 데이터 종류 개수 확인

함수2. value_counts() 함수 - 데이터의 종류와 개수 확인

X['cap-shape'].value_counts()

```
x      3656
f      3152
k       828
b       452
s        32
c         4
Name: cap-shape, dtype: int64
```

Mushroom Dataset - 데이터 종류 확인

함수3. unique() 함수 - 데이터의 종류 확인

X['cap-shape'].unique()

```
['x' 'b' 's' 'f' 'k' 'c']
```

Mushroom Dataset - 범주형(이산형) 데이터 수치화

Mushroom 데이터셋 : 범주형(이산형) 데이터, 인코딩 필요

	class	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	gill-spacing	gill-size	gill-color	...	stalk-surface-below-ring	stalk-color-above-ring	stalk-color-below-ring	veil-type	veil-color
0	p	x	s	n	t	p	f	c	n	k	...	s	w	w	p	w
1	e	x	s	y	t	a	f	c	b	k	...	s	w	w	p	w
2	e	b	s	w	t	l	f	c	b	n	...	s	w	w	p	w
3	p	x	y	w	t	p	f	c	n	n	...	s	w	w	p	w
4	e	x	s	g	f	n	f	w	b	k	...	s	w	w	p	w

인코딩 방식을 이용해 수치화

Mushroom Dataset - 데이터 전처리

데이터 전처리 : 인코딩(Encoding)

데이터

숫자형(연속형)
숫자로 이루어진
순서가 있는 데이터
예) 키, 몸무게

Binning

Encoding

범주형(이산형)
문자 형태이거나 숫자라
하더라도 순서보다는
종류(범주)를 표시하는 데이터
예) 생존 여부, 혈액형

Label Encoding, One-hot Encoding

데이터 전처리 Label Encoding : 레이블을 숫자로 mapping

```
X1["capshape"] = X1["capshape"].map({"x":0, "f":1, "k":2, "b":3, "s":4, "c":5})
```

cap-shape
x
b
x
k
f
s
b
s
c

cap-shape
0
3
0
2
1
4
3
4
5

데이터 전처리 One hot Encoding : 분류하고자 하는 범주(종류) 만큼의 자릿수를 만들고 단 한 개의 1과 나머지 0으로 채워서 숫자화 하는 방식

X_one_hot = pd.get_dummies(X2)

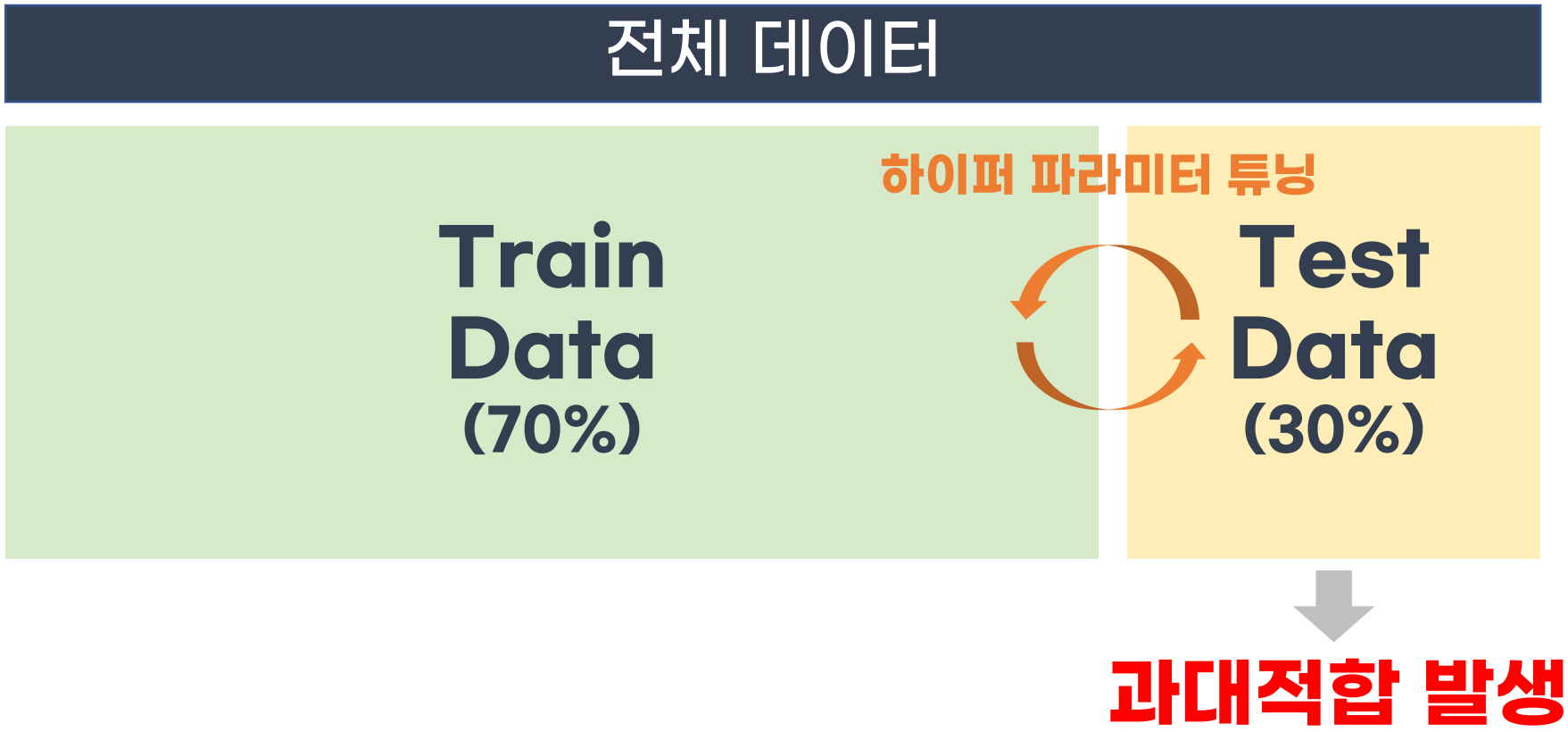
cap-shape
x
b
x
k
f
s
b
s
c

cap-shape_b	cap-shape_c	cap-shape_f	cap-shape_k	cap-shape_s	cap-shape_x
0	0	0	0	0	1
1	0	0	0	0	0
0	0	0	0	0	1
0	0	0	1	0	0
0	0	1	0	0	0
0	0	0	0	1	0
1	0	0	0	0	0
0	0	0	0	1	0
0	1	0	0	0	0

모델 일반화 성능 평가

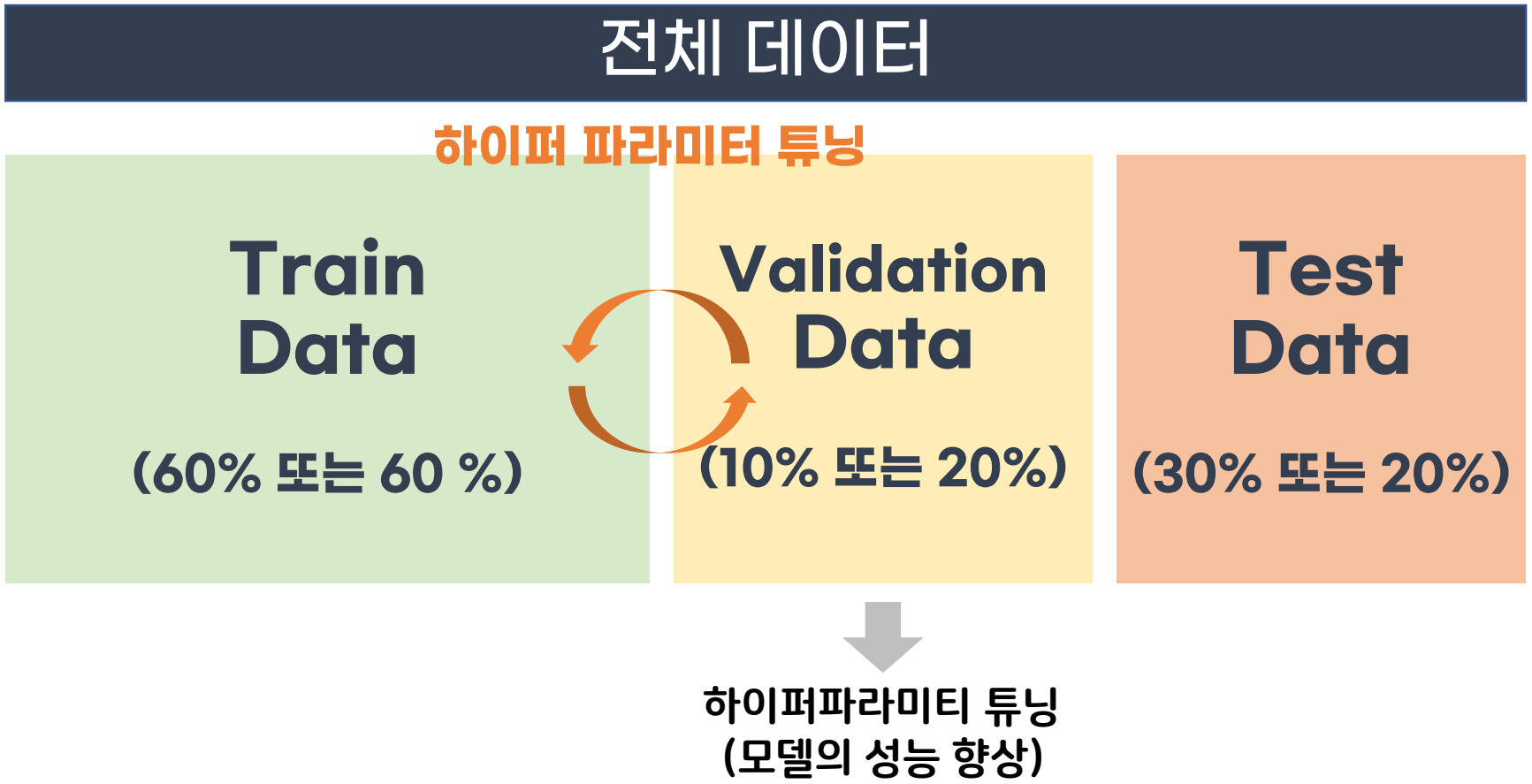
모델 성능 평가

▶ 일반적인 경우



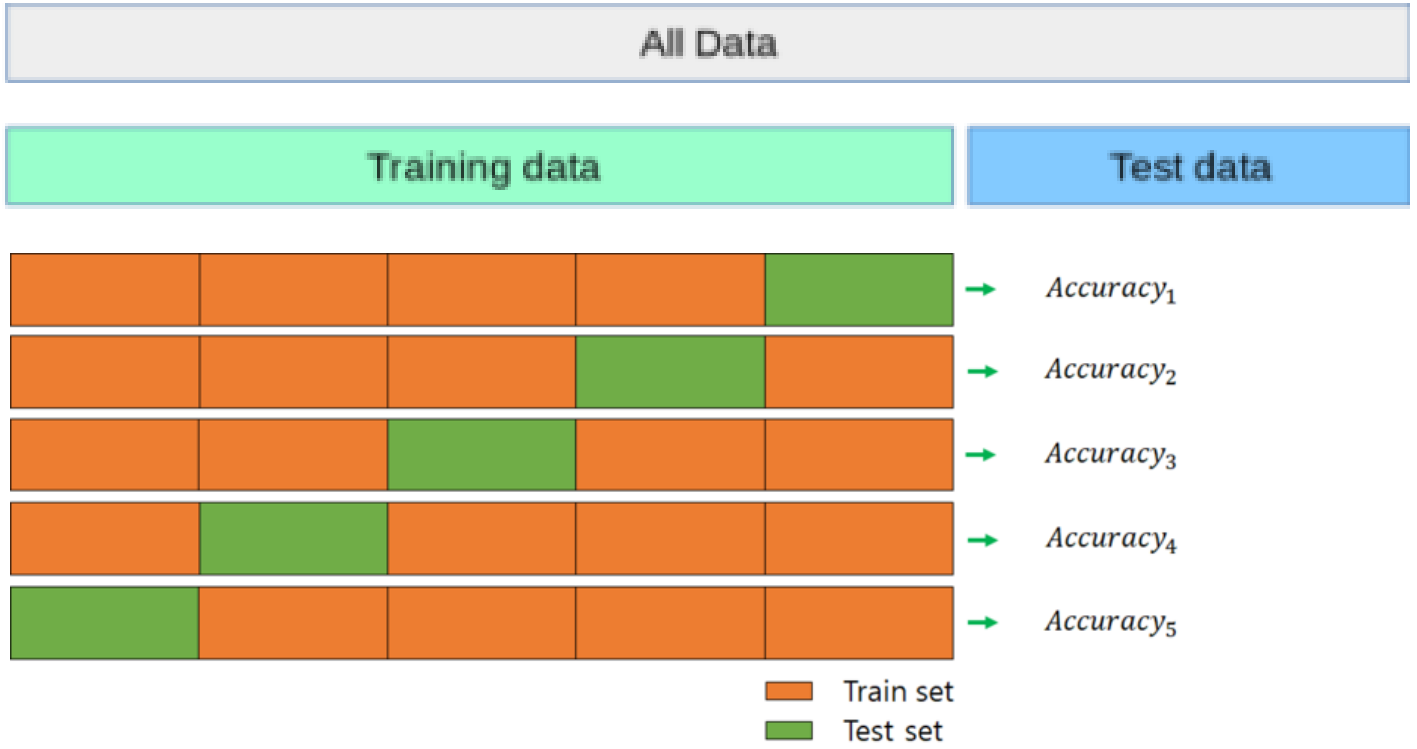
모델 성능 평가

▶ 데이터의 분할과 용도



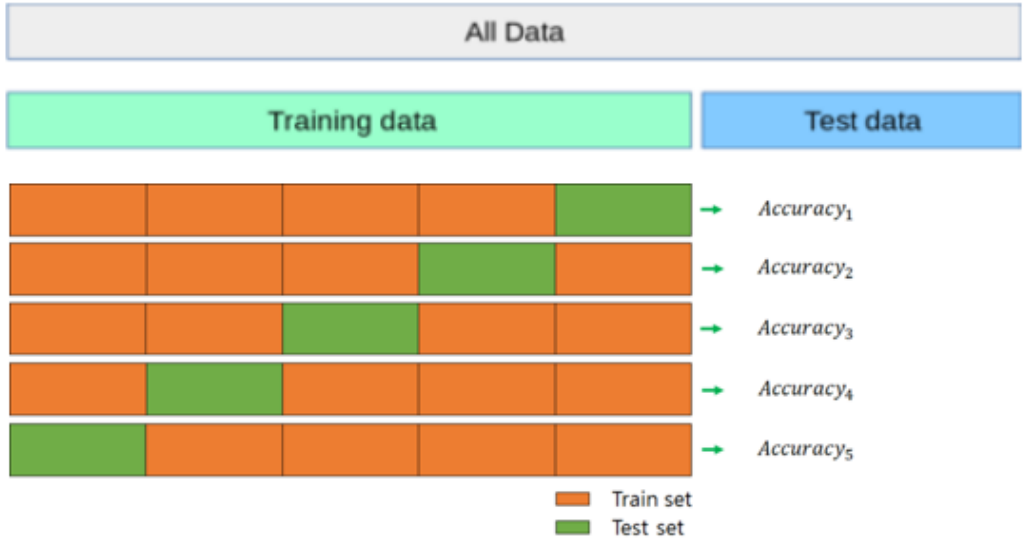
모델 성능 평가

▶ 해결책 : 검증 Data를 하나로 고정하지 않고, Test 데이터의 모든 부분을 사용



모델 성능 평가

▶k-fold cross-validation



1. Test 데이터를 k개의 그룹으로 나누기
2. K-1개의 그룹을 학습에 사용
3. 나머지 1개의 그룹을 이용해서 평가 수행
4. 2번, 3번 과정을 k번 반복
5. 모든 결과의 평균을 구하기

모델 성능 평가

▶k-fold cross-validation

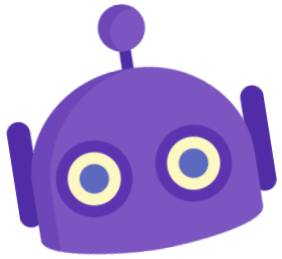
장점	단점
평가에 사용되는 데이터 편중을 막고, 특정 평가 데이터 셋에 과대적합 되는 것 방지	여러 번 학습하고 평가하는 과정을 거치기 때문에 모델 훈련/ 평가 시간 오래 걸림
데이터 세트 크기가 충분하지 않은 경우에도 유용	

모델 성능 평가

▶k-fold cross-validation 사용법: cross_val_score()

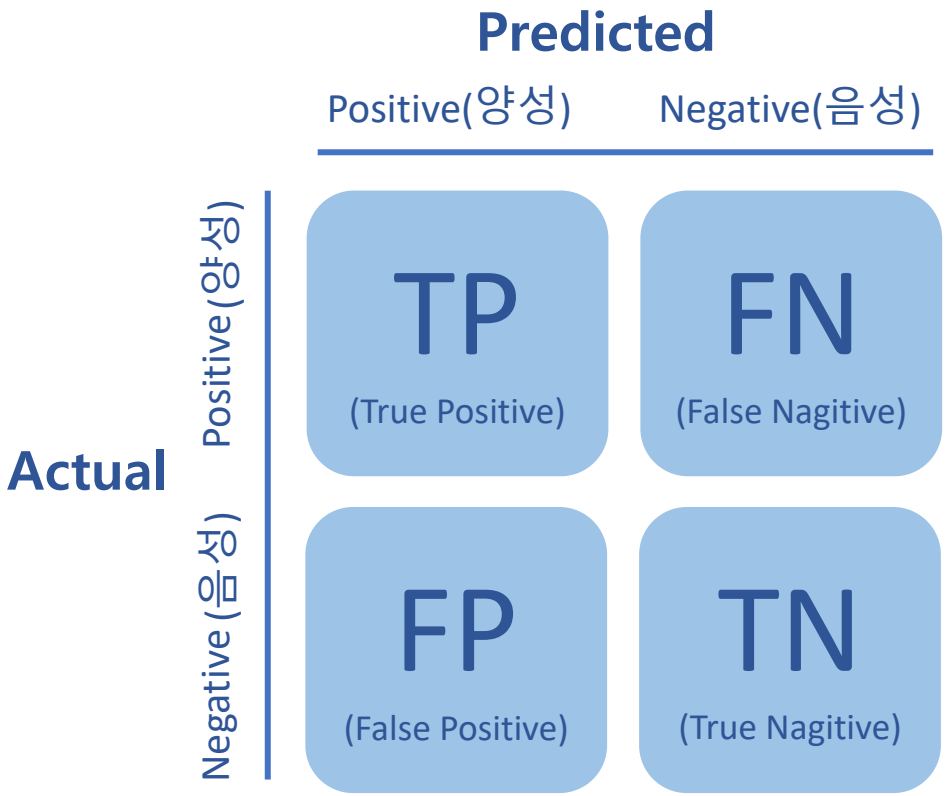
```
# 교차검증 도구 불러오기
from sklearn.model_selection import cross_val_score
score = cross_val_score(모델, X, y, cv=나눌 개수)
score
```

```
[0.86680328 0.87704918 0.875      0.90554415 0.8788501 ]
```



분류 평가 지표

Confusion_matrix



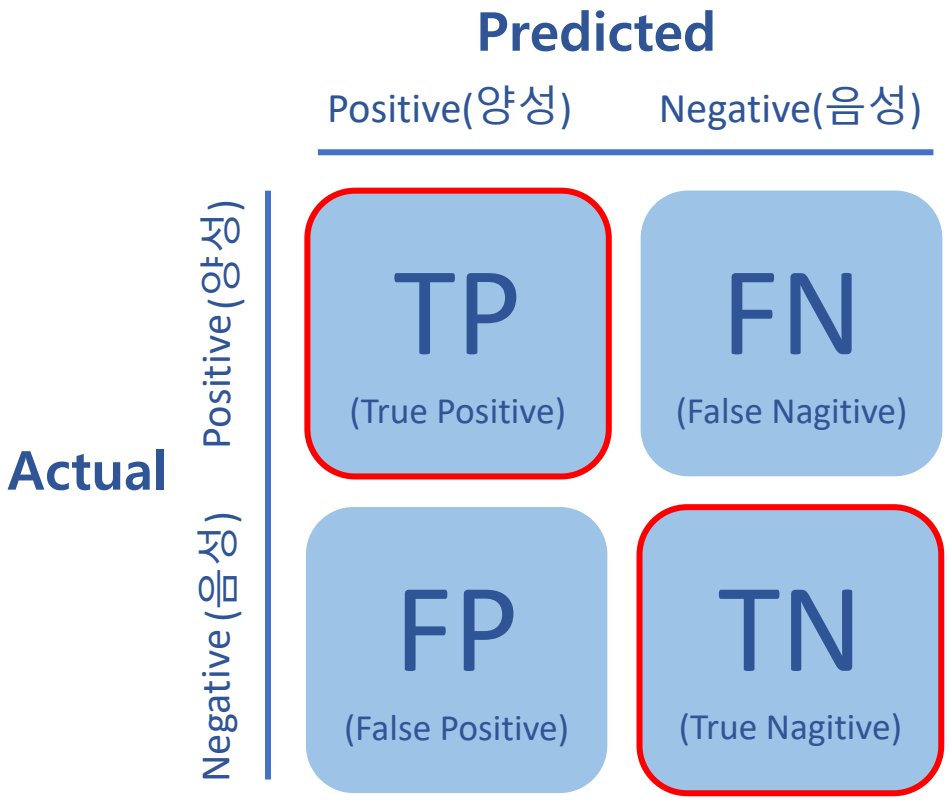
- True Positive(TP)
  실제 True인 정답을 True라고 예측 (정답)

- False Negative(FN)
  실제 True인 정답을 False라고 예측 (오답)

- False Positive(FP)
  실제 False인 정답을 True라고 예측 (오답)

- True Negative(TN)
  실제 False인 정답을 False라고 예측 (정답)

Confusion_matrix



정확도(Accuracy)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- 전체 중 정확하게 맞춘 비율
- 예측값과 실제값이 얼마나 일치하는지 판단

Confusion_matrix

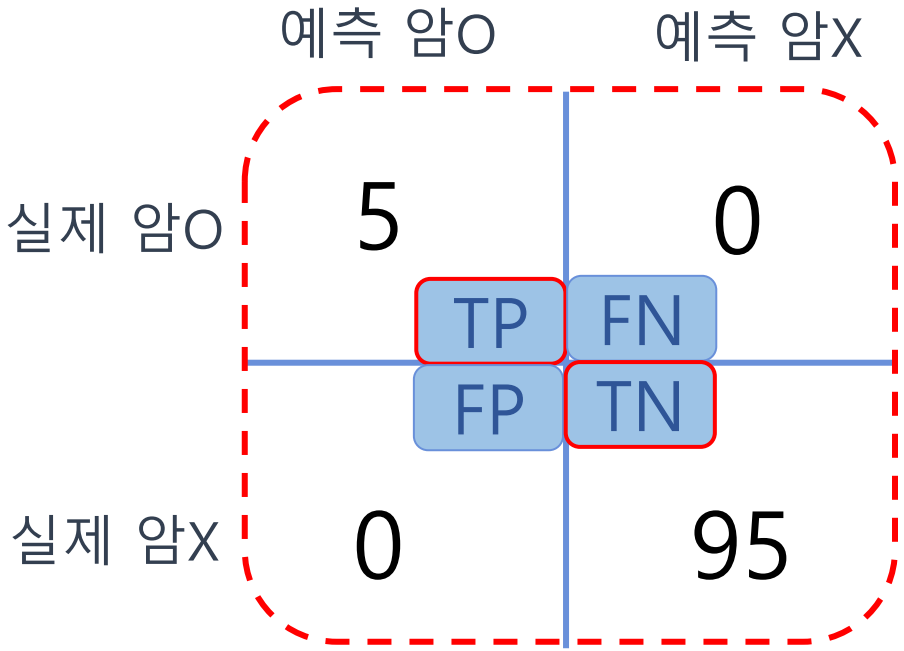
- 100명 중 실제 암 환자는 5명
  95명은 암 x, 5명은 암o 예측

정확도

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\frac{100}{100}$$

- True 암o , False 암x



Confusion_matrix

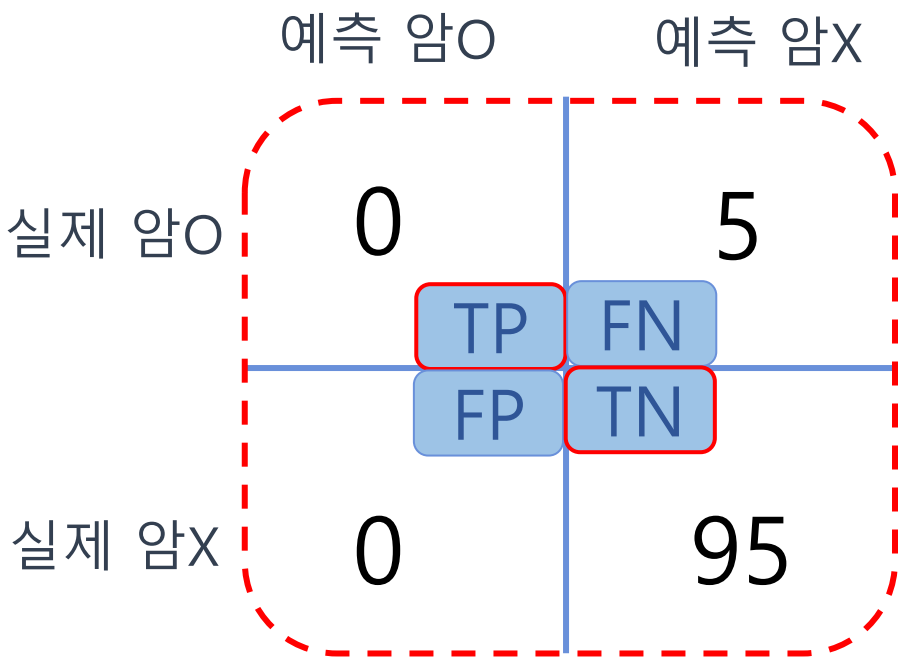
- 100명 중 암 환자는 5명
  100명 모두 암 환자 x 예측

정확도

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

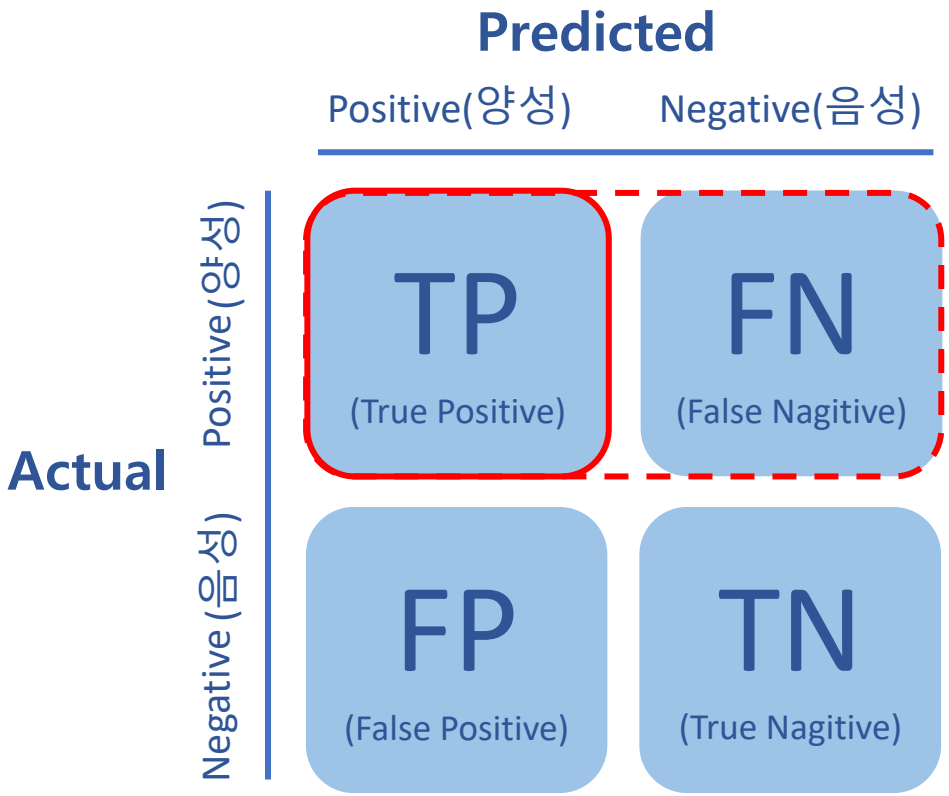
$$\frac{95}{100}$$

- True 암o , False 암x



불균형한(imbalance) 데이터가 들어있을 경우 정확도로 성능을 평가하는 것은 문제가 됨

Confusion_matrix



재현율(Recall)

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- 실제 양성 중에 예측 양성 비율
- 실제 양성 중 얼마나 양성을 정확히 예측했는지 판단

Confusion_matrix

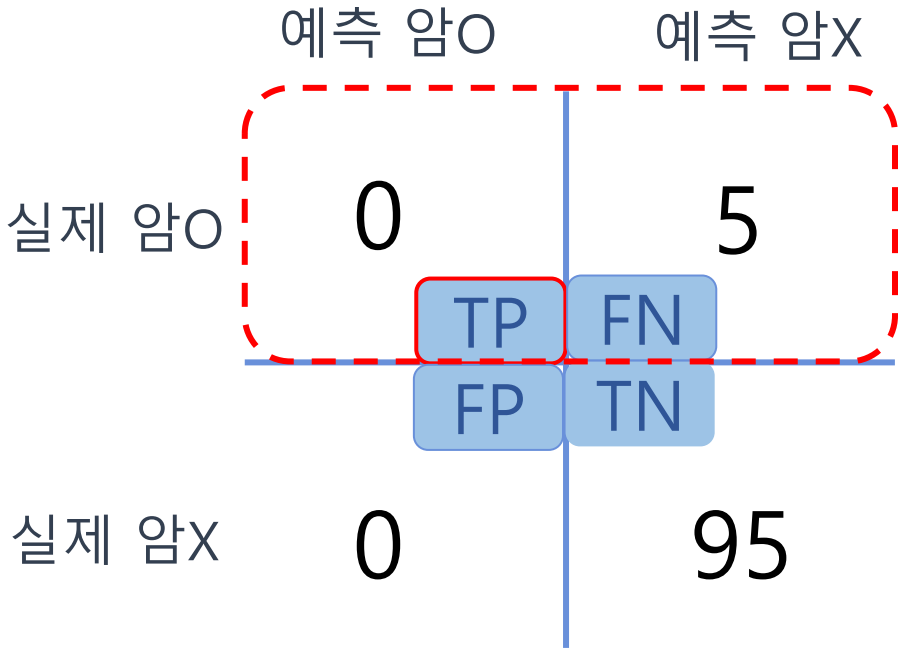
- 100명 중 암 환자는 5명
- 100명 모두 암 환자 x 예측

재현율

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\frac{0}{5}$$

- True 암o , False 암x



Confusion_matrix

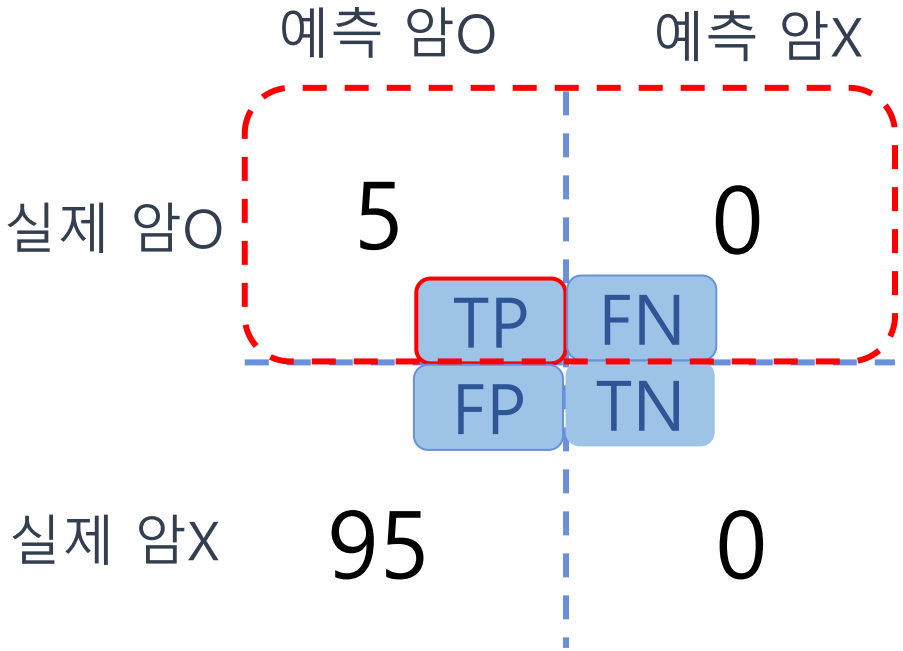
- 100명 중 암 환자는 5명
- 100명 모두 암 환자 o 예측

재현율

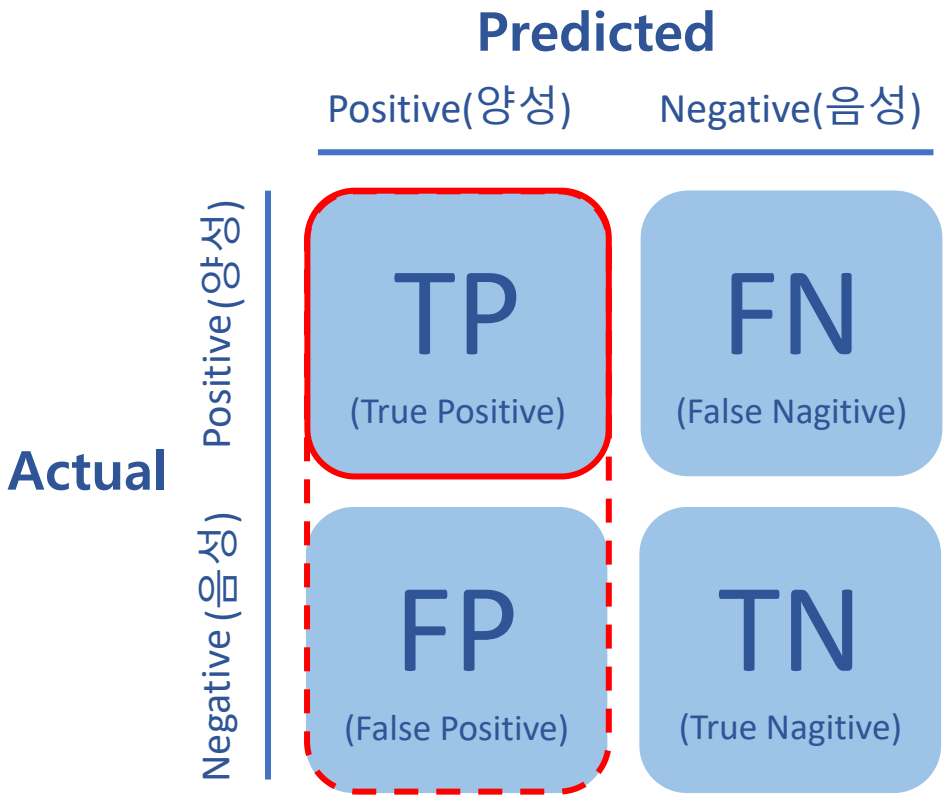
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\frac{5}{5}$$

- True 암o , False 암x



Confusion_matrix



정밀도(Precision)

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- 예측 양성 중에 실제 양성 비율
- 예측된 양성 중 실제 양성이 얼마나 정확한지 판단

Confusion_matrix

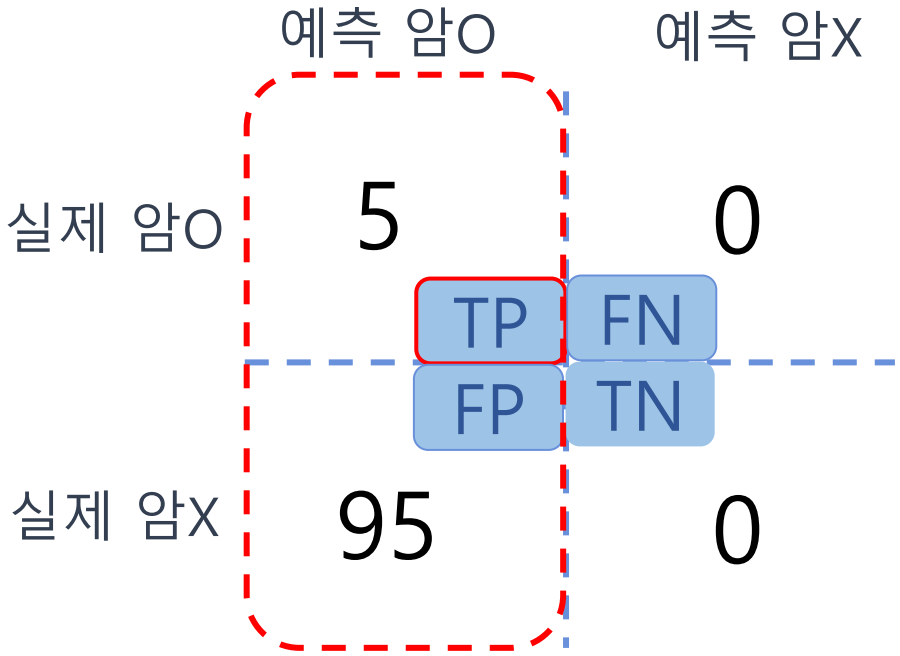
- 100명 중 암 환자는 5명
- 100명 모두 암 환자 o 예측

정밀도

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\frac{5}{100}$$

- True 암o , False 암x



Confusion_matrix

상황에 따른 재현율과 정밀도의 상대적인 중요도

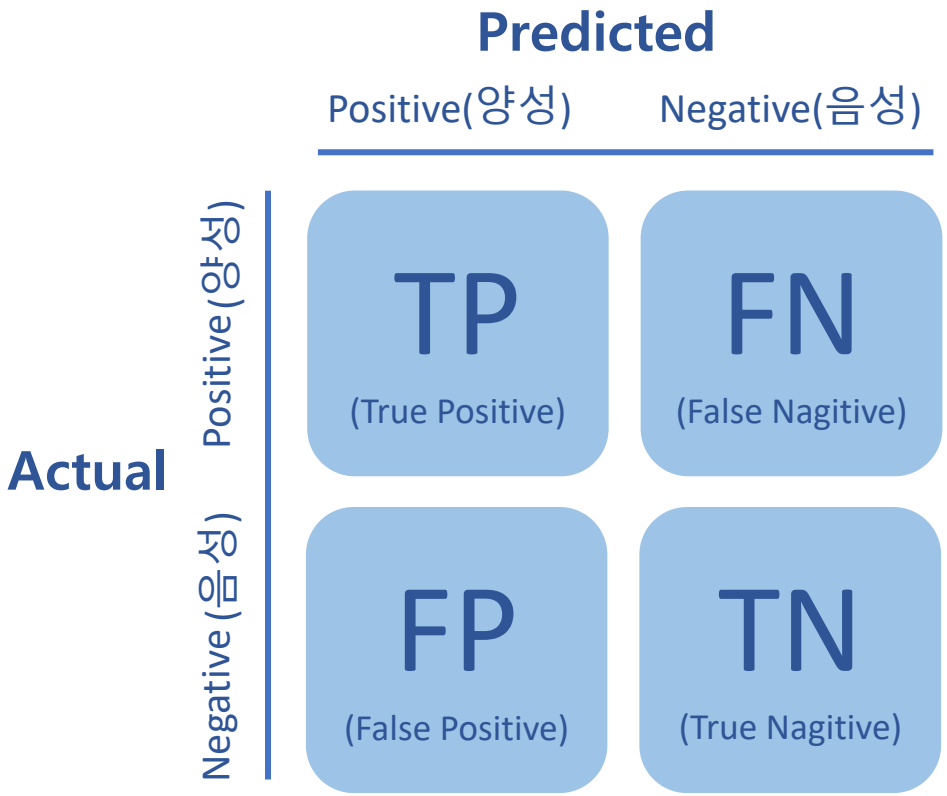
재현율(Recall)를 선호하는 경우

- 실제 positive(양성)인 데이터 예측을 Negative(음성)으로 잘못 판단하게 되면 업무상 큰 영향을 줌
- 암 진단, 금융사기 판별, 도둑 판별

정밀도(Precision)를 선호하는 경우

- 실제 Negative(음성)인 데이터 예측을 Positive(양성)으로 잘못 판단하게 되면 업무상 큰 영향을 줌
- 스펨메일(스팸메일 양성, 정상메일 음성)
- 유아 컨텐츠(안전 영상 양성, 비안전 영상 음성)

Confusion_matrix

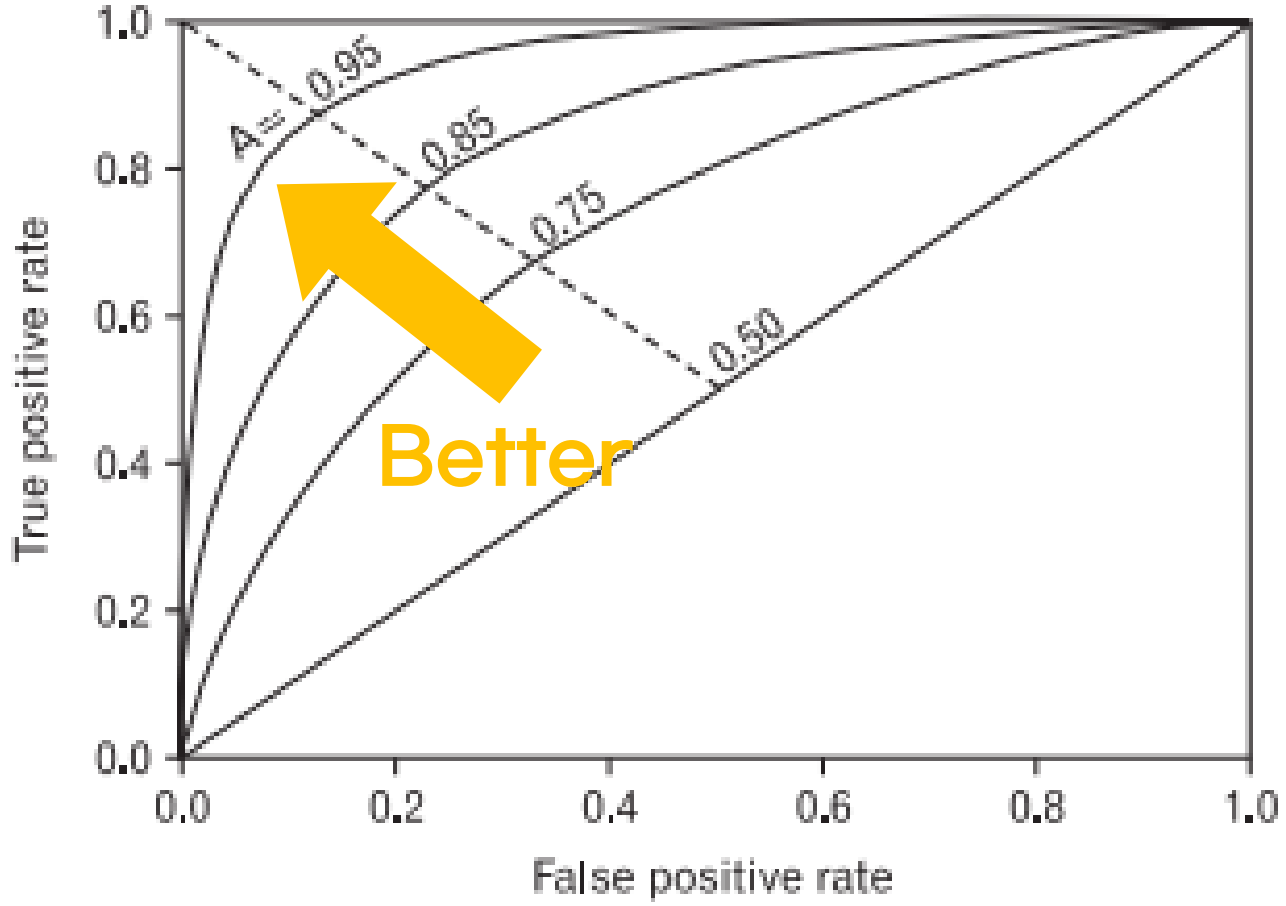


조화평균(F1-Score)

$$\mathrm{F} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

- 정밀도와 재현율의 조화평균

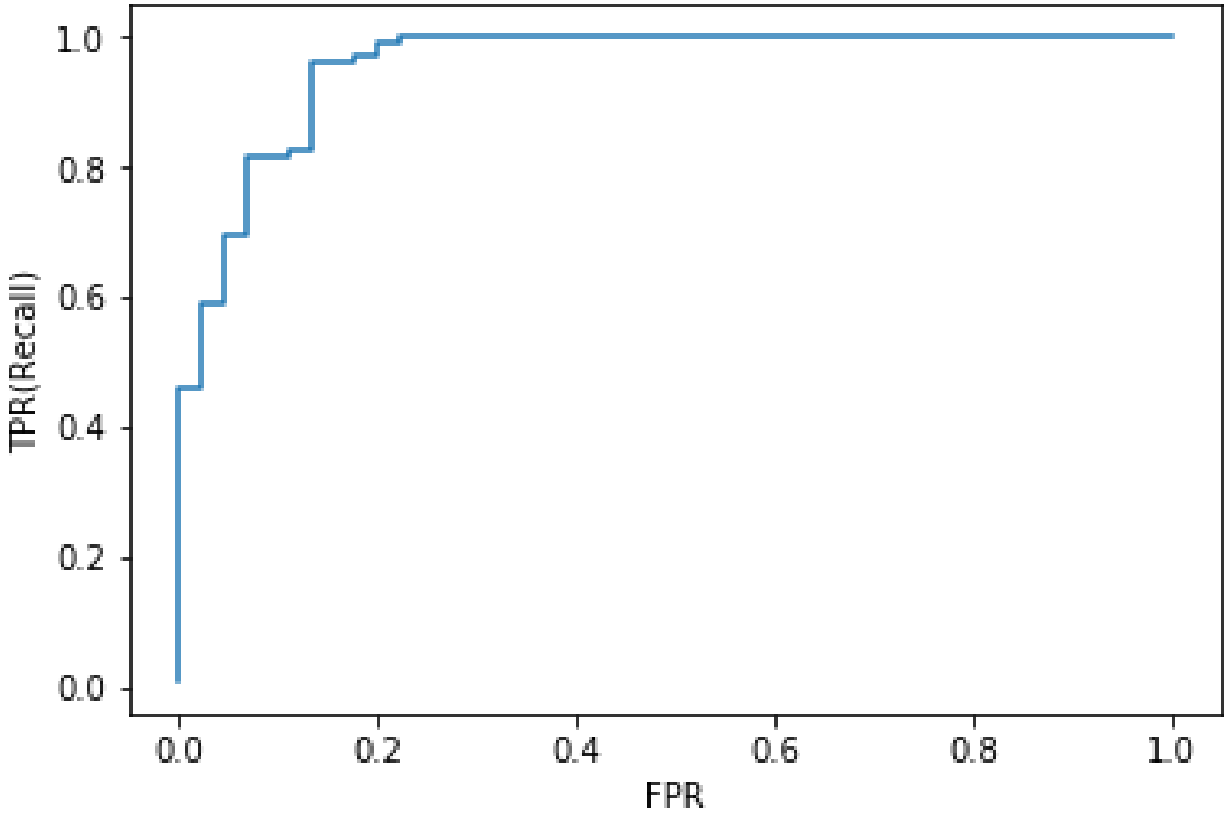
ROC curve



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

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

ROC curve



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

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

다음 시간에 만나요!

앙상블 모델
그리드서치