6일차. 캐글 데이터 분석 실습 2

심선영 교수, 이주민 교수

강의 목표

- ❖ 로지스틱 회귀분석을 이해하고 활용한다.
- ❖ 머신러닝 프로세스에 따라 Titanic 생존자를 예측할 수 있다.
- ❖ 다양한 머신러닝 기법을 활용하여 예측력을 높일 수 있다.

강의 스케쥴

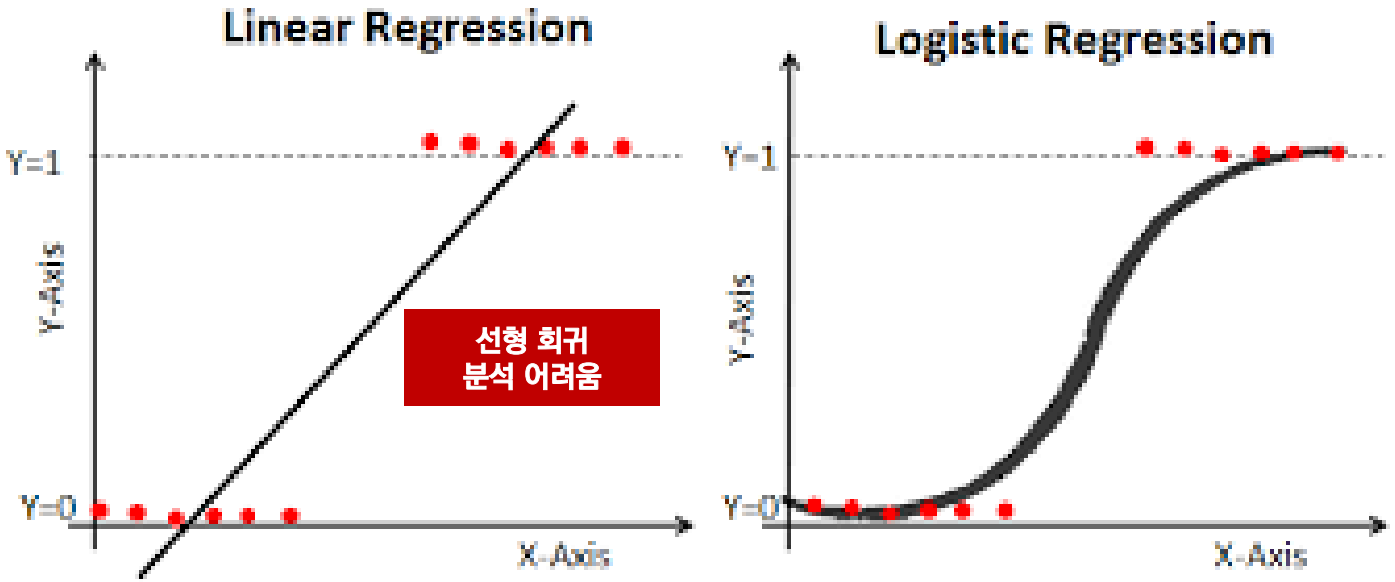
목차	활동
Review & Overview	Day6 개요 & 추가 학습 - Day6_0.lambda함수_학생용.html
로지스틱 회귀분석 실습	유방암 예측 실습 - Day6_1.LogReg_학생용.html
캐글 데이터 분석 실습	- 데이터 분석 기본 실습 - Day6_2_1.titanic(1).html - 예측력 높이기 - Day6_2_2.titanic(2).html
Wrap-up	

Review & Overview

로지스틱 회귀분석

로지스틱 회귀

❖ Target 데이터가 이산적 형태를 보일 때 사용하는 회귀 모델

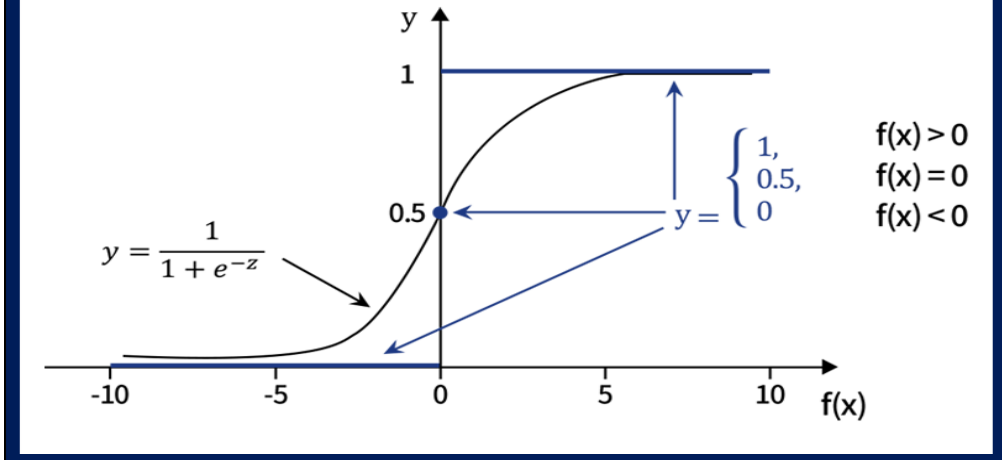


로지스틱 회귀

❖ 로지스틱 함수

■ 입력 값을 0이나 1에 근사한 출력 값으로 전환

로지스틱 함수



$$y = \frac{1}{1+e^{-f(x)}}$$

선형 회귀 분석 대입

$$= \frac{1}{1+e^{-(w^T x+b)}}$$

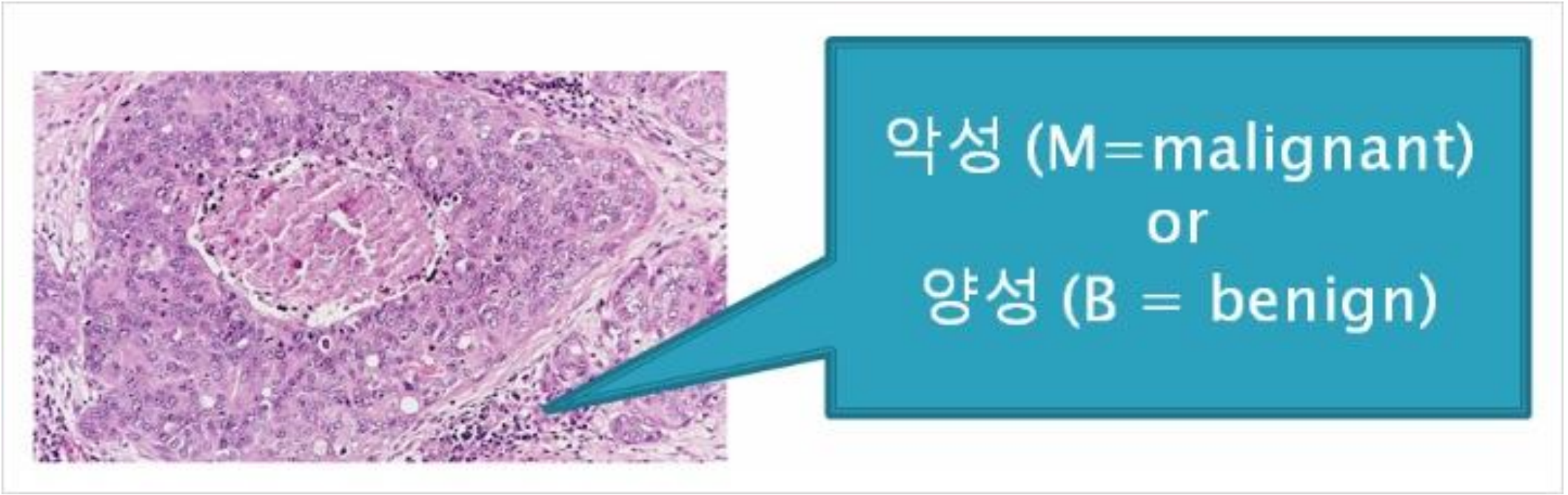
$$ln\frac{y}{1-y} = w^T x + b$$

y	f(x)가 양의 값일 가능성
$1-y$	f(x)가 음의 값일 가능성
$\frac{y}{1-y}$	x가 양의 값일 상대적 가능성

오즈비 (Odds Ratio)

Skill up 실습

위스콘신 유방암 예측



❖ 참고

- 악성(malignant) - 암o
- 양성(benign) - 암x

실습 파일: Day6_1.LogReg_학생용.html

위스콘신 유방암 예측

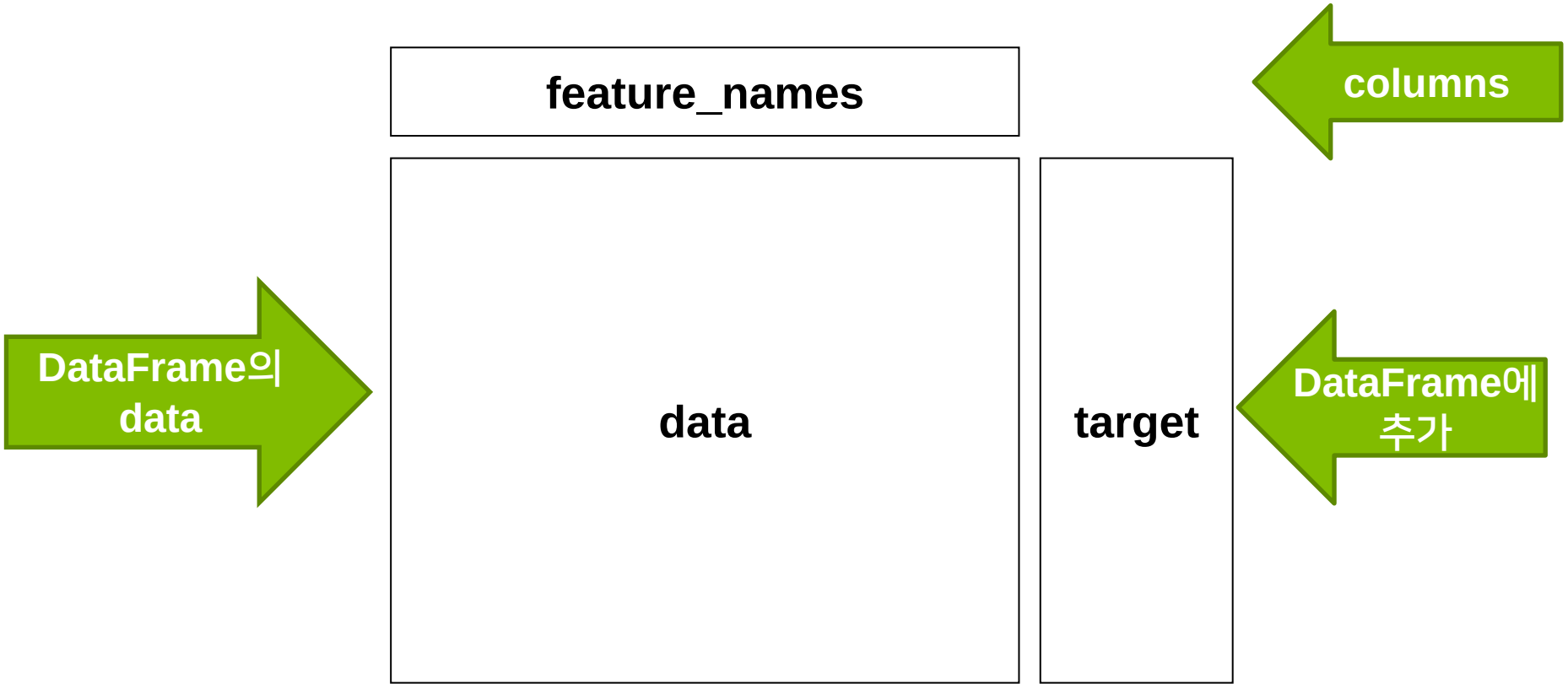
❖ sklearn 의 datasets는 sklearn.utils.Bunch라는 key-value 형식으로 구성되는 딕셔너리 형 타입과 유사한 구조를 가지고 있습니다.

❖ 공통 키 key

- data: 샘플 데이터, Numpy 배열로 이루어져 있습니다.
- target: Label 데이터, Numpy 배열로 이루어져 있습니다.
- feature_names: Feature 데이터의 이름
- target_names: Label 데이터의 이름
  - 'target_names': array(['malignant', 'benign'])
- DESCR: 데이터 셋의 설명
- filename: 데이터 셋의 파일 저장 위치

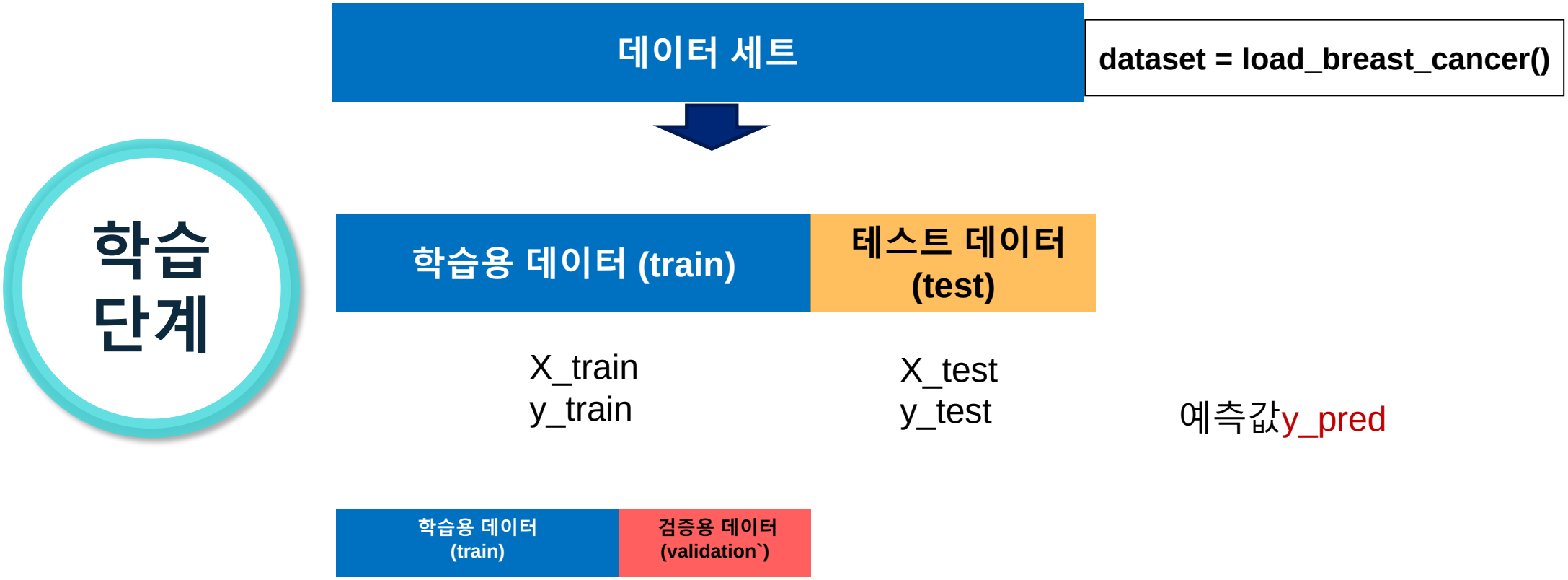
위스콘신 유방암 예측

❖ DataFrame 형태로 변형



위스콘신 유방암 예측

❖ 지도학습



Titanic 생존자 예측 실습
- 1차 도전 -

Titanic 분석 실습(1)

❖ 실습 요약

- 단순하게 분석해 보기

❖ 목표

- 데이터분석 프로세스 따라가보기
- 로지스틱 회귀분석 실시
- 결과값을 캐글에 올려보자!

실습 파일: Day6_2_1titanic(1).html

데이터 모델링에 대한 이해

❖ 각 모델 알고리즘을 불러서 **fit ( ) 함수**를 통해 fitting 진행

- dt = DecisionTreeClassifier().fit(X_train, y_train)
- lr = LogisticRegression().fit(X_train, y_train)
- rf = RandomForestClassifier().fit(X_train, y_train)

❖ **predict ( ) 함수**를 통해 X_test 값에 대한 예측 수행

❖ **score ( ) 함수**를 통해 예측된 값과 정답 값을 비교

Titanic 생존자 예측 실습
- 2차 도전 -

실습 단계

① 라이브러리 불러오기 (NumPy, Pandas, Matplotlib, Seabon 등)

② 데이터 불러오기

③ 데이터 전처리 및 EDA를 통한 변수(feature) 설정

④ 학습데이터 셋(train data set)을 훈련 및 검증 데이터로 분리

⑤ 적합한 머신러닝 알고리즘의 선택하고 학습 (train)

⑥ 학습된 알고리즘에 검증 데이터에 적용하여 예측 (predict)

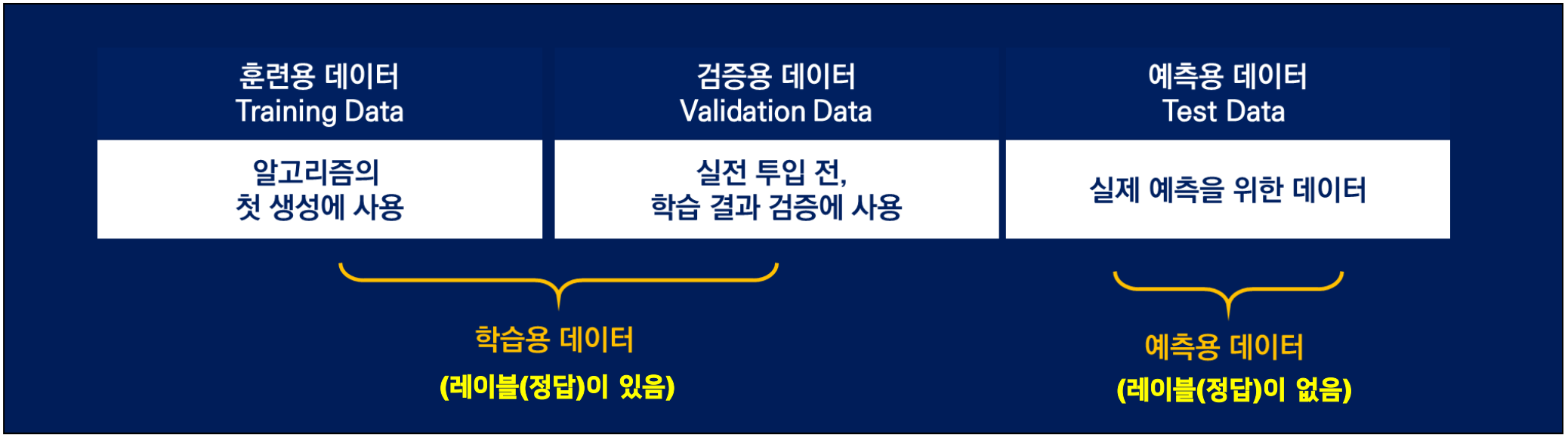
⑦ 예측값과 실제값을 비교해 오차를 측정하여 알고리즘의 성능 평가 (evaluation)

⑧ 3-8단계를 반복하며 알고리즘의 성능 고도화

⑨ 실제 예측을 원하는 데이터 셋(test data set)을 적용하여 최종 예측

머신러닝 데이터 셋

❖ 훈련용(train set), 검증용(validation set), 예측용 (test set)

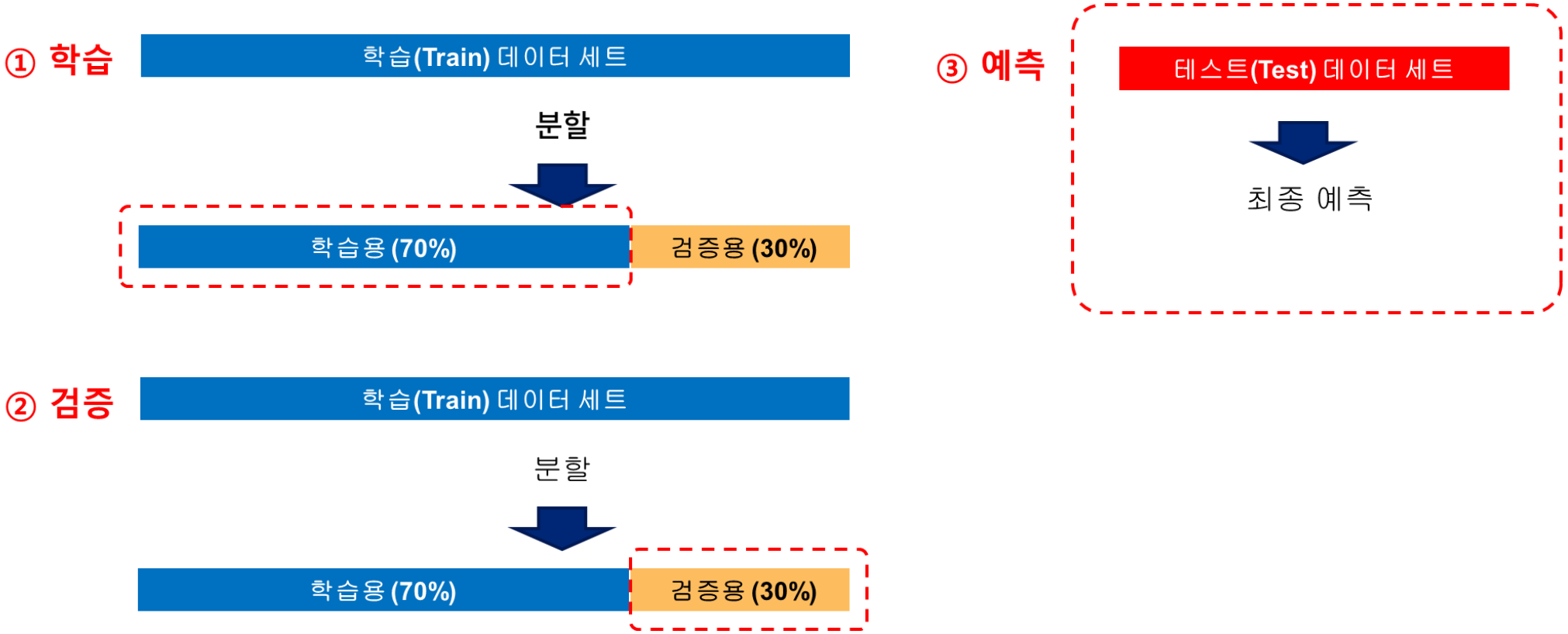


sepal_length	sepal_width	petal_ length	petal_ width	species
5.8	4.0	1.2	0.2	setosa
5.1	2.5	3.0	1.1	versicolor
6.6	3.0	4.4	1.4	versicolor
5.4	3.9	1.3	0.4	setosa
7.9	3.8	6.4	2.0	virginica

sepal_length	sepal_width	petal_ length	petal_ width
5.8	4.0	1.2	0.2
5.1	2.5	3.0	1.1
6.6	3.0	4.4	1.4
5.4	3.9	1.3	0.4
7.9	3.8	6.4	2.0

머신러닝 데이터 셋

❖ 훈련용(train set), 검증용(validation set), 예측용 (test set)



Titanic 생존자 예측력 높이기

- ❖ 데이터 인코딩
  - 원-핫 인코딩
- ❖ 분석 변수 추가
  - SibSp, Parch, Cabin
- ❖ 추가 변수 생성
  - Family, IsAlone 변수 생성
- ❖ 다양한 머신러닝 기법 사용 및 성능 비교
  - Decision Tree, Random Forest 등
- ❖ Decision Tree의 하이퍼 파라미터 최적화
  - 최적의 하이퍼 파라미터 찾기

Titanic 생존자 예측력 높이기

실습 파일: Day6_2_2 titanic(2).html

데이터 인코딩

❖ 레이블 인코딩

- 문자열 값을 숫자형 카테고리 값으로 변환
  - Sex: [남자, 여자] → [0,1]
  - Embarked: [C, S, Q] ➔ [0,1,2]

레이블 인코딩의 문제점?

데이터 인코딩

❖ 원-핫 인코딩

- 새로운 feature 를 추가하고 고유 값에 해당하는 칼럼에만 1을 표시, 나머지는 모두 0을 표시하는 방식
- Pandas의 **get_dummies( )** 또는 사이킷런의 OneHotEncoder() 로 수행

상품 분류
TV
냉장고
전자렌지
컴퓨터
선풍기
선풍기
믹서
믹서

레이블 인코딩

상품분류
0
1
2
4
3
3
5

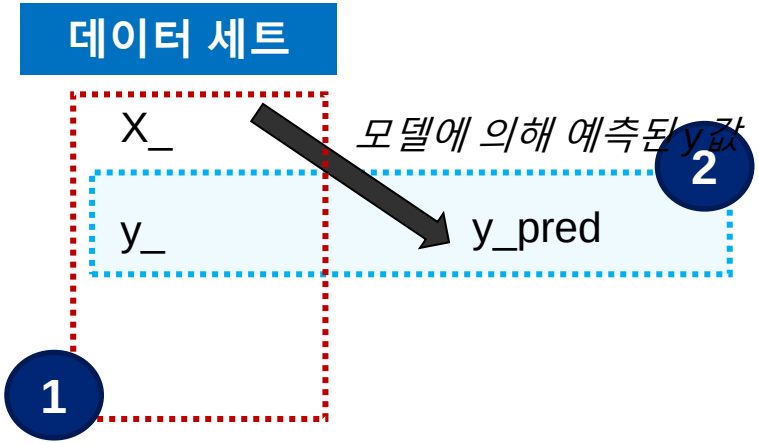
원-핫 인코딩

상품 분류_ TV	상품 분류_ 냉장고	상품 분류_ 전자렌지	상품 분류_ 선풍기	상품 분류_ 컴퓨터	상품 분류_ 믹서
1	0	0	0	0	0
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	0
0	0	0	0	0	1
0	0	0	0	0	1

모델 평가 – 정확도 측정 방법

❖ 정확도 보는 방법

- 방법1) 기존 데이터 세트(X와 y)를 인수로 사용할 경우
  - 사이킷런의 score함수 사용
    - ✓ lr_clf.fit() 이후,
    - ✓ lr_ clf. Score(X_train,y_train)
- 방법2) 예측한 값과 기존 데이터(정답) 비교 (accuracy_score 이용)
  - From sklearn.metrics import accuracy_score
    - ✓ Accuracy_score(y_train, y_pred)



하이퍼 파라미터 (Hyper Parameter) 최적화

❖ 머신러닝 모델에서 우리가 직접 설정할 수 있는 값

■ 하이퍼 파라미터 값을 조정하여 머신러닝 알고리즘의 성능 개선 → 하이퍼 파라미터 최적화

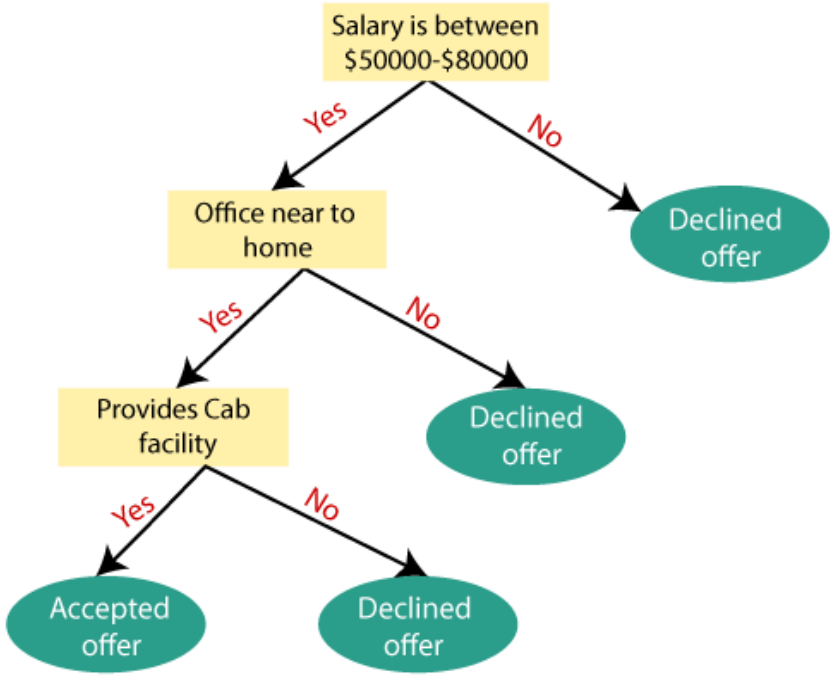
	파라미터 (Parameter)	하이퍼 파라미터 (Hyper Parameter)
의미	• 매개변수 • 모델 내부에서 결정되는 값 • 데이터로부터 학습하여 결정	• 초매개변수 • 모델 학습에 반영되는 값 • 학습 전에 미리 설정
예시	선형회귀, 로지스틱회귀의 계수	비용함수의 종류, 학습률, 결정트리의 최대 깊이(max-depth)
설정 가능 여부	직접 설정 안됨 (X)	직접 설정 가능 (O)

■ 학습률 예시→

하이퍼 파라미터 (Hyper Parameter) 최적화

❖ 의사결정나무 (Decision Tree)

- 결정트리는 매우 쉽고, 스케일링이나 정규화 등 사전 데이터 가공의 영향이 적음
- 예측 성능을 향상시키기 위해 복잡한 규칙구조를 가져야 하고 이로 인한 과적합이 발생, 예측성능 떨어질수도 있음
- 트리의 depth가 깊어질수록 결정 트리의 예측성능은 저하될 수 있음



의사결정나무에서의 하이퍼 파라미터

❖ 결정 트리의 최대 깊이(max-depth)

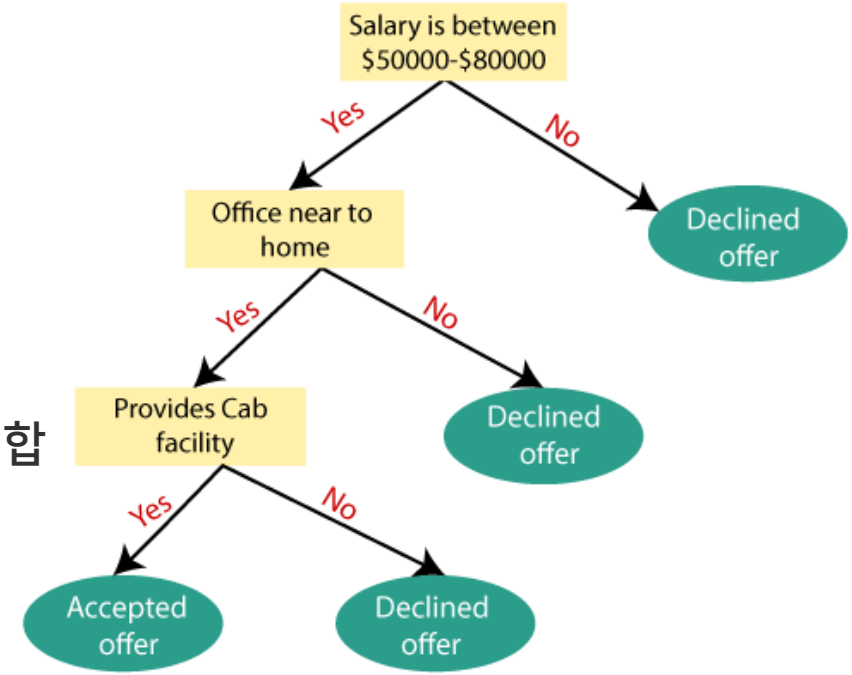
- 깊이가 깊어질수록 모델 복잡 → 과대적합

❖ 노드분할을 위한 최소 샘플 수 (min_samples_split)

- 최소 수가 작아질수록 분할이 많아 모델 복잡 → 과대적합

❖ 마지막 잎들의 최소 샘플수 (min_samples_leaf)

- 최소 수가 작아질수록 분할이 많아 모델 복잡 → 과대적



의사결정나무에서의 하이퍼 파라미터

❖ GridSearchCV

- 최적 하이퍼 파라미터 튜닝을 한번에!
- 분류나 회귀 알고리즘에 적용되는 하이퍼 파라미터를 순차적으로 테스트하여 최적의 파라미터 서치
- Gird(격자) - 촘촘하게 파라미터를 변화시켜 가면서 테스트해본다는 의미

❖ 주요 파라미터

- Estimator: classifier., regressor 등 모델
- param_grid: Key+리스트값을 가지는 딕셔너리. 튜닝을 위해 파라미터명과 사용될 여러 마라미터를 지정함.
- scoring: 예측 성능을 측정할 평가 방법 (우리는 accuracy)
- cv: 교차검증을 위해 분할되는 학습/테스트 세트 개수
- refit: default가 Tue. 가장 최적의 하이퍼 파라미터를 찾은 뒤, 입력된 estimator 객체를 해당 하이퍼파라미터로 재학습 시킴.

```
from sklearn.model_selection import GridSearchCV

parameters = {'max_depth':[2,3,5,10], 'min_samples_split':[2,3,5], 'min_samples_leaf':[1,5,8]}

grid_dclf = GridSearchCV(dt_clf, param_grid = parameters, scoring='accuracy', cv=5)
```

의사결정나무에서의 하이퍼 파라미터

❖ 몇 가지의 경우의 수가 있을까요?

```
from sklearn.model_selection import GridSearchCV

parameters = {'max_depth':[2,3,5,10], 'min_samples_split':[2,3,5], 'min_samples_leaf':[1,5,8]}

grid_dclf = GridSearchCV(dt_clf, param_grid = parameters, scoring='accuracy', cv=5)
```

Wrap-Up

❖ Day6. Wrap-up 설문

- https://forms.gle/a3MoTLLAwovSRAPc6