Chapter. 15

지도학습 모델 및 파라미터 선택

그리드 서치

FAST CAMPUS
ONLINE
데이터 탐색과 전처리 I

강사. 안길승

I 모델 및 파라미터 선정 문제

- 어떠한 데이터에 대해서도 우수한 모델과 그 하이퍼 파라미터는 절대 존재하지 않음

- 또한, 분석적인 방법으로 좋은 모델과 하이퍼 파라미터를 선정하는 것도 불가능함

I 그리드 서치 개요

- 하이퍼 파라미터 그리드는 한 모델의 하이퍼 파라미터 조합을 나타내며, 그리드 서치란 하이퍼 파라미터 그리드에 속한 모든 파라미터 조합을 비교 평가하는 방법을 의미

- 예시: k-최근접 이웃의 파라미터 그리드

		n_neighbors		
		3	5	7
metric	Manhattan	(Manhattan, 3)	(Manhattan, 5)	(Manhattan, 7)
	Euclidean	(Euclidean, 3)	(Euclidean, 5)	(Euclidean, 7)

➡ 총 여섯 개의 하이퍼 파라미터 조합에 대한 성능을 평가하여, 그 중 가장 우수한 하이퍼 파라미터를 선택

I 그리드 서치 코드 구현

- sklearn을 활용하여 그리드 서치를 구현하려면 사전 형태로 하이퍼 파라미터 그리드를 정의해야 함
  - ➢ Key: 하이퍼 파라미터명 (str)
  - ➢ Value: 해당 파라미터의 범위 (list)

		n_neighbors		
		3	5	7
metric	Manhattan	(Manhattan, 3)	(Manhattan, 5)	(Manhattan, 7)
	Euclidean	(Euclidean, 3)	(Euclidean, 5)	(Euclidean, 7)

```
parameter_grid = {"n_neighbors": [3, 5, 7],
                  "metric": ["Manhattan", "Euclidean"]}
```

I 그리드 서치 코드 구현: GridSearchCV

- sklearn.model_selection.GridSearchCV
  - ➢ 주요 입력
    - estimator: 모델 (sklearn 인스턴스)
    - param_grid: 파라미터 그리드 (사전)
    - cv: k겹 교차 검증에서의 k (2 이상의 자연수)
    - scoring_func: 평가 함수 (sklearn 평가 함수)
  - ➢ GridSearchCV 인스턴스(GSCV)의 주요 method 및 attribute
    - GSCV = GridSearchCV(estimator, param_grid, cv, scoring_func): 인스턴스화
    - GSCV.fit(X, Y): 특징 벡터 X와 라벨 Y에 대해 param_grid에 속한 파라미터를 갖는 모델을 k-겹 교차 검증 방식으로 평가하여, 그 중 가장 우수한 파라미터를 찾음
    - GSCV.get_params(): 가장 우수한 파라미터를 반환
  - ➢ 사용이 편하다는 장점이 있지만, k-겹 교차 검증 방식을 사용하기에 느리고, **성능 향상을 위한 전처리 기법을 적용할 수 없다는 단점**이 있음

I 그리드 서치 코드 구현: ParameterGrid

- sklearn.model_selection.ParameterGrid
  - ➢ param_grid (사전 형태의 하이퍼 파라미터 그리드)를 입력 받아, 가능한 모든 파라미터 조합 (사전)을 요소로 하는 generator를 반환하는 함수

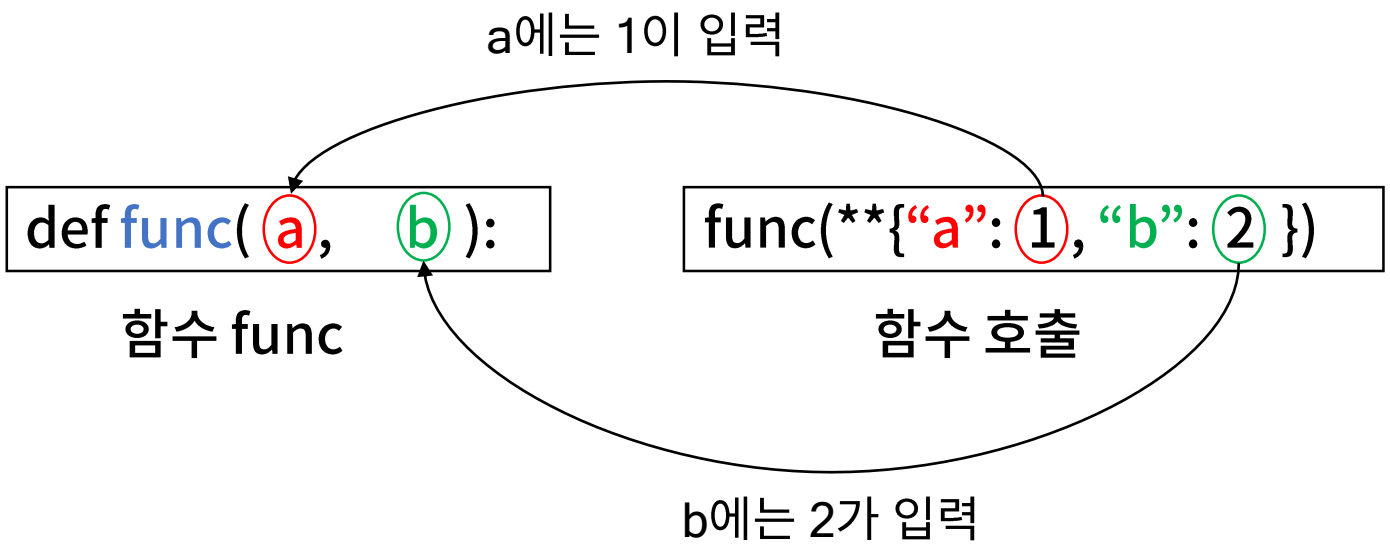
```
1  from sklearn.model_selection import ParameterGrid
2  parameter_grid = {"n_neighbors": [3, 5, 7],
3                    "metric": ["Manhattan", "Euclidean"]}
4  list(ParameterGrid(parameter_grid))
```

```
[{'metric': 'Manhattan', 'n_neighbors': 3},
 {'metric': 'Manhattan', 'n_neighbors': 5},
 {'metric': 'Manhattan', 'n_neighbors': 7},
 {'metric': 'Euclidean', 'n_neighbors': 3},
 {'metric': 'Euclidean', 'n_neighbors': 5},
 {'metric': 'Euclidean', 'n_neighbors': 7}]
```

  - ➢ GridSearchCV에 비해 사용이 어렵다는 단점이 있지만, **성능 향상을 위한 전처리 기법을 적용하는데 문제가 없어서 실무에서 훨씬 자주 사용됨**

ParameterGrid 사용을 위해 알아야 하는 문법 (1/2)

- 파이썬 함수의 입력으로 사전 자료형을 사용하는 경우에는 **를 사전 앞에 붙여야 함

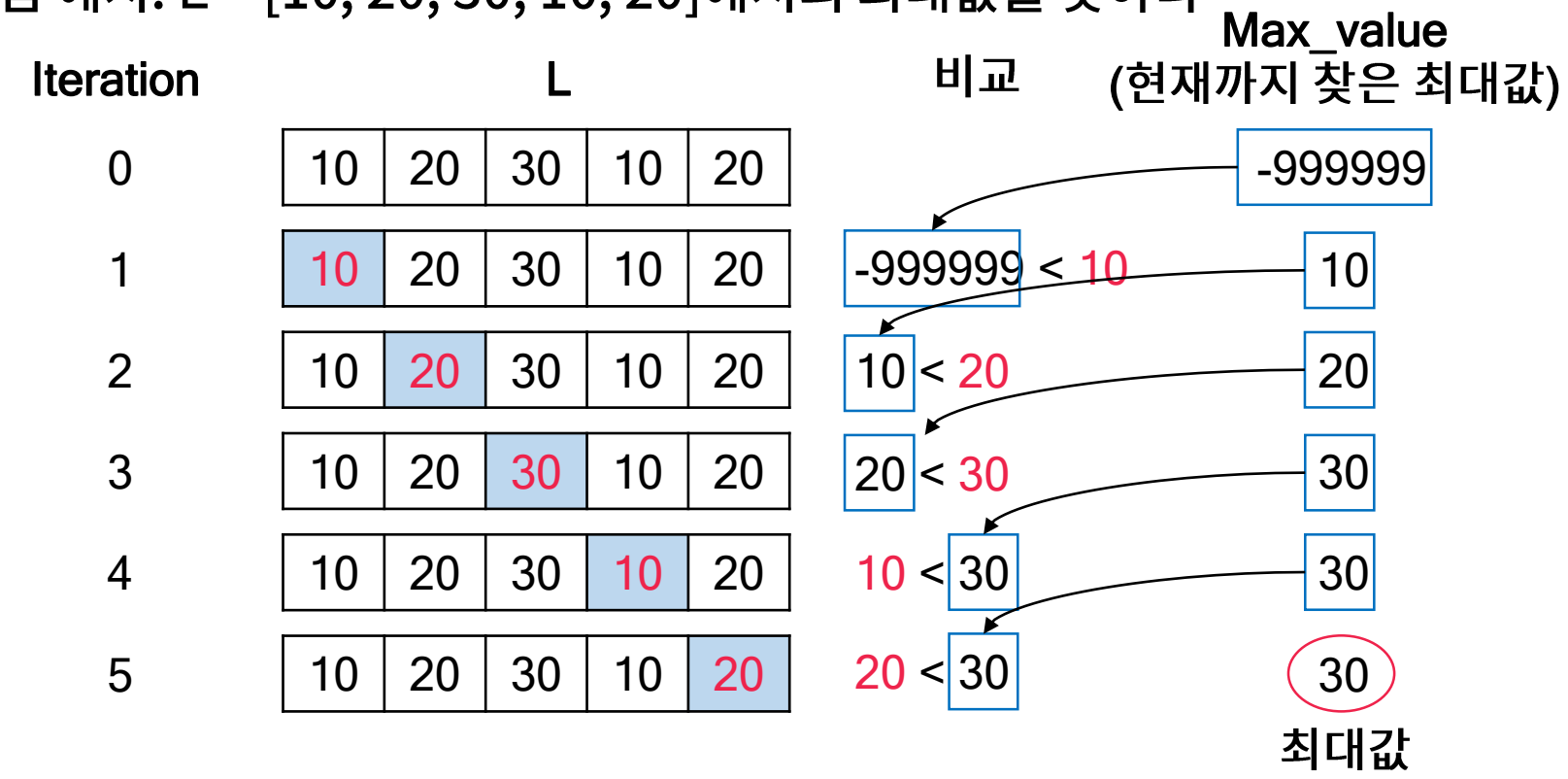


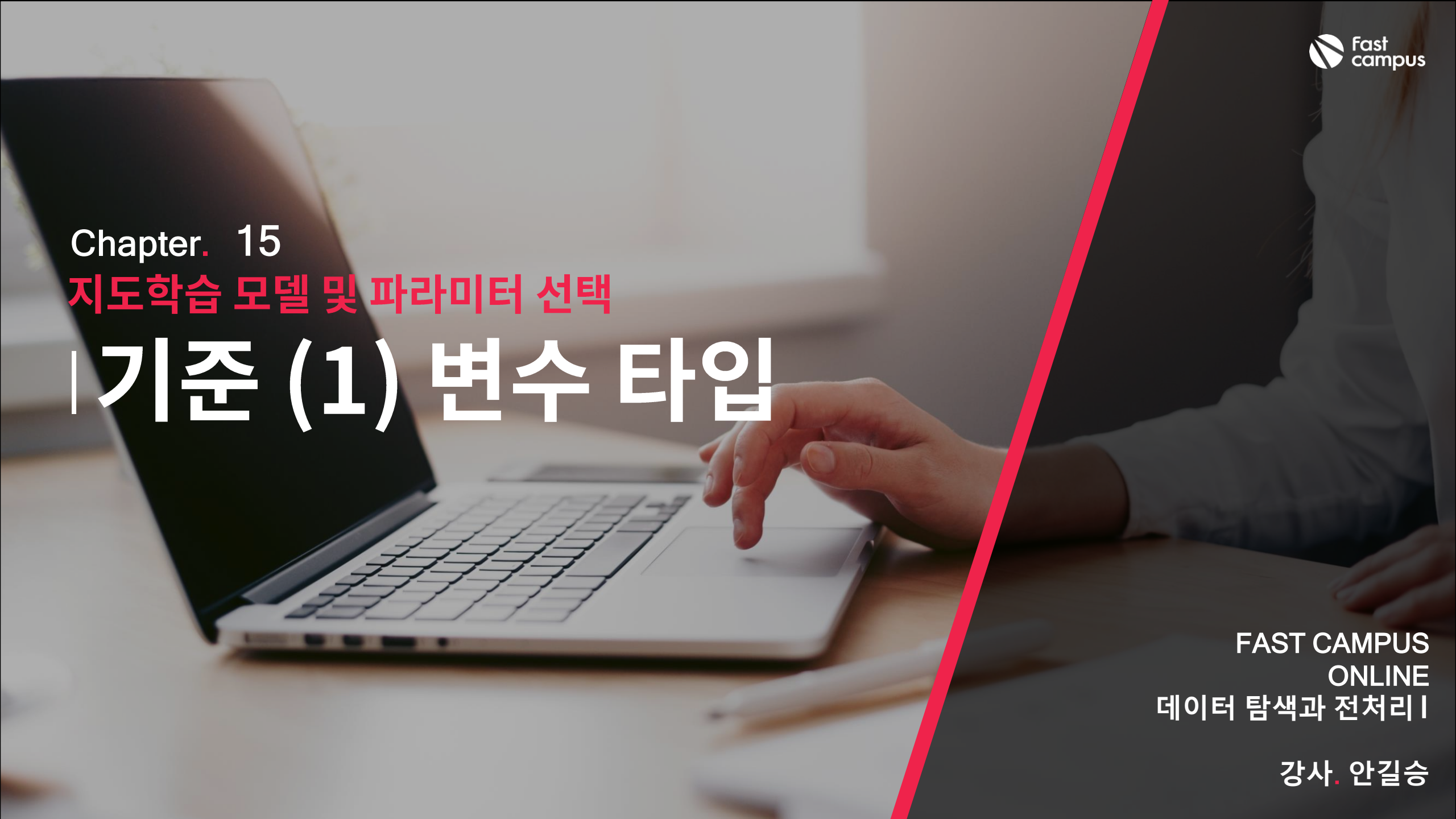
- 이를 활용하면, ParameterGrid 인스턴스를 순회하는 사전 자료형인 변수(파라미터)를 모델의 입력으로 넣을 수 있음

I ParameterGrid 사용을 위해 알아야 하는 문법 (2/2)

- ParameterGrid 인스턴스를 순회하면서 성능이 가장 우수한 값을 찾으려면 최대값(최소값)을 찾는 알고리즘을 알아야 함
  - ➢ 내장 함수인 max 함수나 min 함수를 사용해도 되지만, 평가해야 하는 하이퍼 파라미터 개수가 많으면 불필요한 메모리 낭비로 이어질 수 있으며, 더욱이 모델도 같이 추가되야 하므로 **메모리 에러**로 이어지기 쉬움

- 알고리즘 예시: L = [10, 20, 30, 10, 20]에서의 최대값을 찾아라





Chapter. 15

지도학습 모델 및 파라미터 선택

기준 (1) 변수 타입

FAST CAMPUS
ONLINE
데이터 탐색과 전처리 I

강사. 안길승

I 변수 타입 확인 방법

- DataFrame.dtypes
  - ➢ DataFrame에 포함된 컬럼들의 데이터 타입 (object, int64, float64, bool 등)을 반환

- DataFrame.infer_objects().dtypes
  - ➢ DataFrame에 포함된 컬럼들의 데이터 타입을 추론한 결과를 반환
  - ➢ (예) [‘1’, ‘2’]라는 값을 가진 컬럼은 비록 object 타입이나, int 타입이라고 추론할 수 있음

- 주의: string type이라고 해서 반드시 범주형이 아니며, int 혹은 float type이라고 해서 반드시 연속형은 아님. 반드시 상태 공간의 크기와 도메인 지식 등을 고려해야 함

I 변수 타입에 따른 적절한 모델

- 주의: 모델 성능에는 변수 타입만 영향을 주는 것이 아니므로, 다른 요소도 반드시 고려해야 함

모델 / 변수타입	이진형 only	정수형 only	연속형 only	이진형 + 정수형	정수형 + 연속형	이진형 + 연속형	이진형+정수형+ 연속형
회귀모델	X	△	O	X	O	X	X
의사결정나무 및 앙상블 모델	O	◎	◎	◎	O	△	△
k - 최근접 이웃	O	O	◎	X	△	X	X
베르누이 나이브베이즈	◎	X	X	X	X	X	X
다항 나이브베이즈	△	◎	X	O	X	X	X
가우시안 나이브베이즈	△	△	O	X	△	X	X
신경망	△	O	◎	O	◎	O	O
SVM	◎	O	◎	O	O	O	O

◎: 매우 적합, O: 적합, △: 보통 (가능하면 미 사용 추천), X: 사용 자제

I 혼합형 변수에 적절하지 않은 모델 (1) 회귀 모델

- 혼합형 변수인 경우에는 당연히 **변수의 스케일 차이가 존재**하는 경우가 흔함
- 변수의 스케일에 따라 계수 값이 크게 달라지므로, **예측 안정성이 크게 떨어짐**
  - ➢ 모든 특징이 라벨에 독립적으로 영향을 준다면, 이진형 특징의 계수 절대값이 스케일이 큰 연속형 특징의 계수 절대값보다 크게 설정됨
  - ➢ 이진형 특징 값에 따라 예측 값이 크게 변동함
- 스케일링을 하더라도 이진형 특징의 분포가 변하지 않으므로, 이진형 특징의 값에 따른 영향력이 크게 줄지 않음

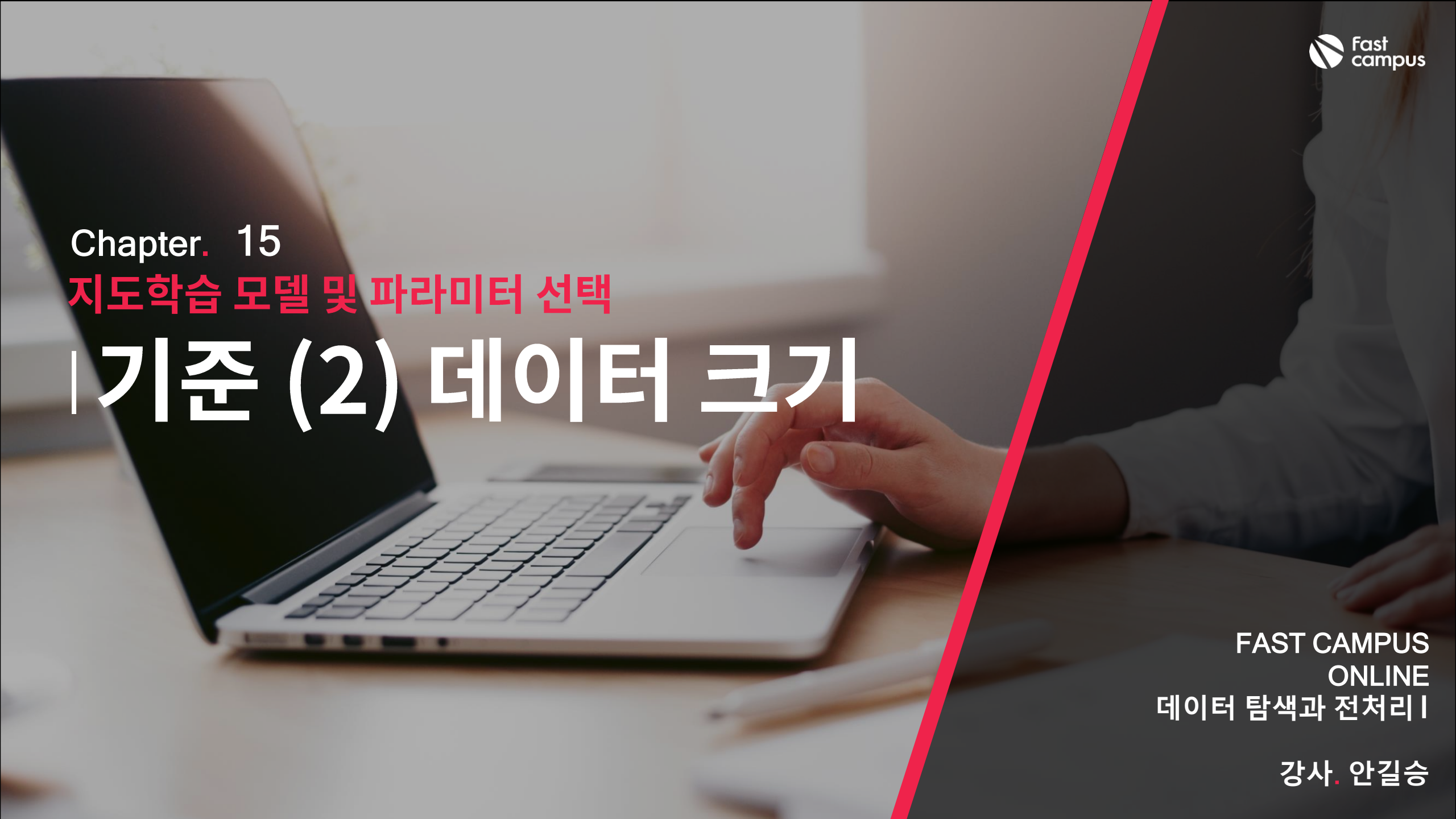
I 혼합형 변수에 적절하지 않은 모델 (2) 나이브 베이즈

- 나이브베이즈는 하나의 확률 분포를 가정하기 때문에, 혼합형 변수를 가지는 데이터에 부적절함
  - ➢ (예시) 베르누이 분포는 연속형 값을 가지는 확률 분포 추정에 매우 부적절

- 따라서 나이브베이즈는 **혼합형 변수인 경우에는 절대로 고려해서는 안 되는** 모델임

l 혼합형 변수에 적절하지 않은 모델 (3) k-최근접 이웃

- 스케일이 큰 변수에 의해 거리가 사실상 결정되므로, k-NN은 혼합형 변수에 적절하지 않음
- 단, 코사인 유사도를 사용하는 경우나, 스케일링을 적용하는 경우에는 큰 무리없이 사용 가능함



Chapter. 15

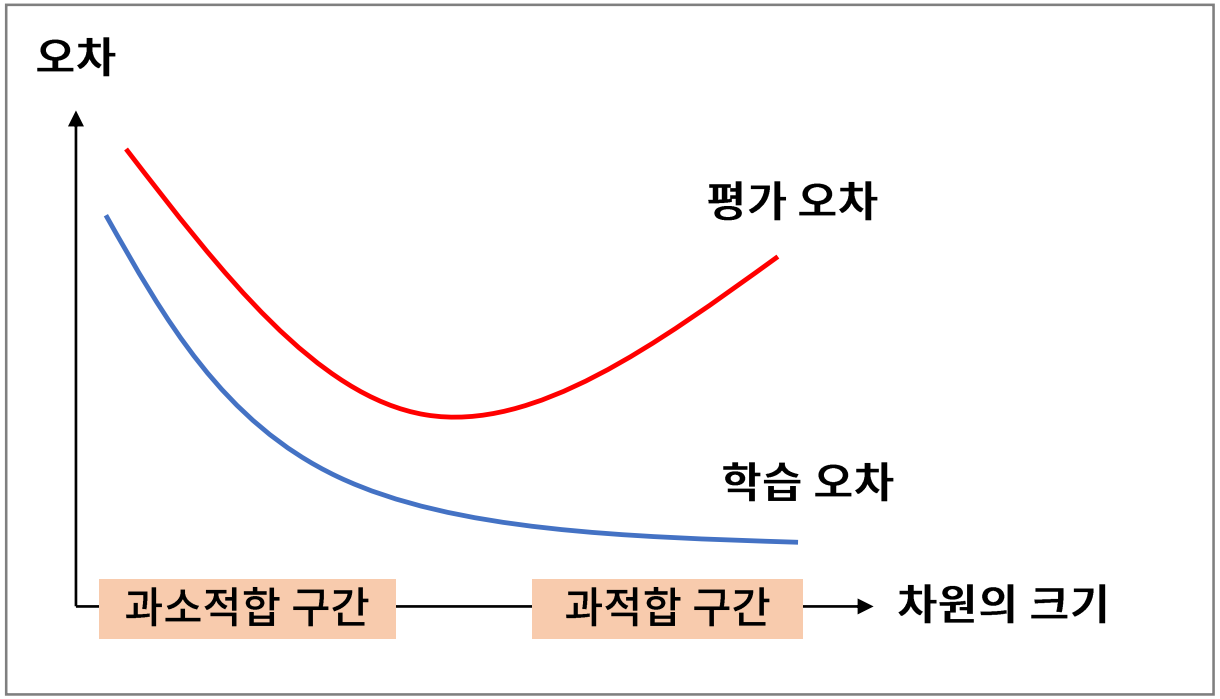
지도학습 모델 및 파라미터 선택

기준 (2) 데이터 크기

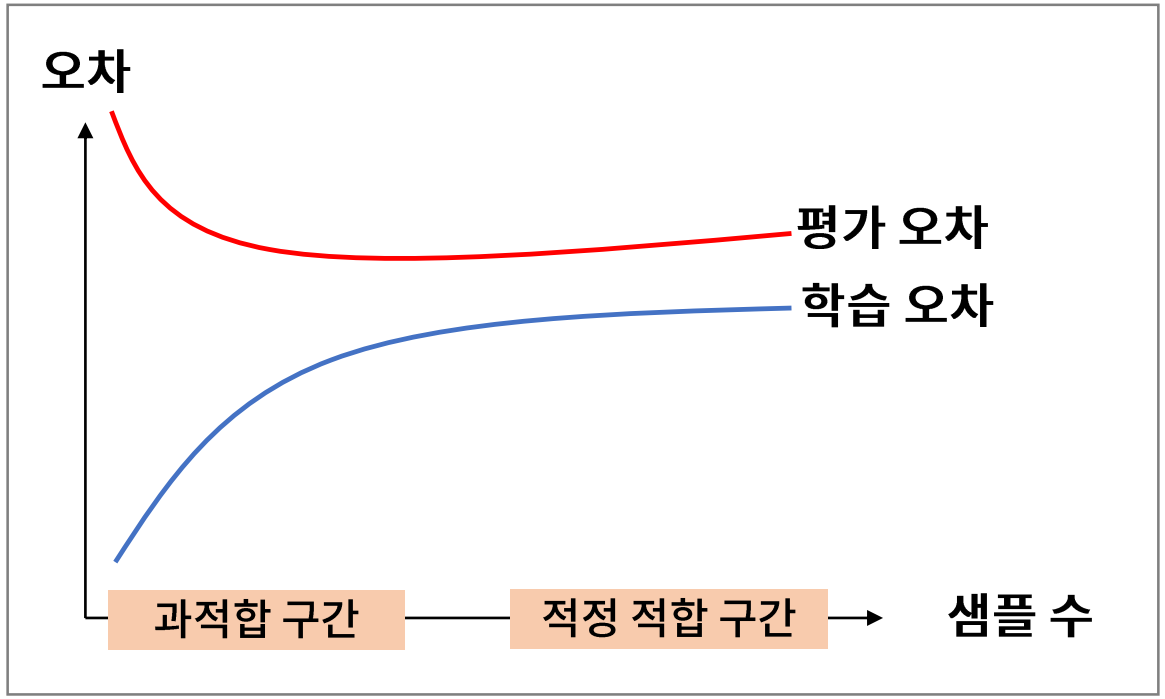
FAST CAMPUS
ONLINE
데이터 탐색과 전처리 I

강사. 안길승

l 샘플 개수와 특징 개수에 따른 과적합 (remind)

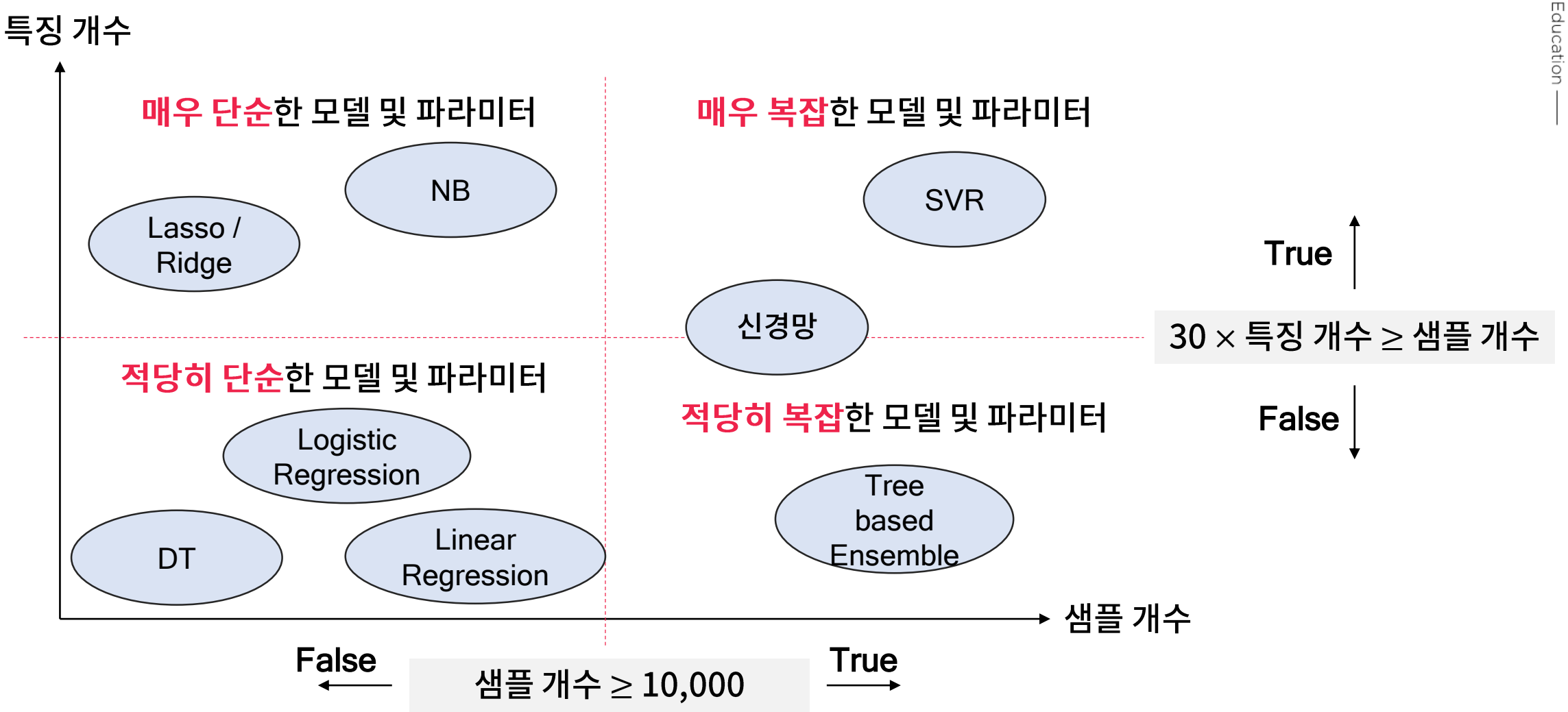


차원의 크기와 과적합 간 관계



샘플 수와 과적합 간 관계

I 샘플 개수와 특징 개수에 따른 적절한 모델



Chapter. 15

지도학습 모델 및 파라미터 선택

복잡도 파라미터 튜닝 방법

FAST CAMPUS
ONLINE
데이터 탐색과 전처리 I

강사. 안길승

I 복잡도 파라미터 튜닝 개요

- **복잡도 파라미터**란 복잡도에 영향을 주는 파라미터로, 이 값에 따라 **과적합 정도가 결정**되므로 매우 신중하게 튜닝해야 함

모델	파라미터	영향
휴리스틱하게 학습되는 모든 모델	max_iter	• 복잡한 모델의 경우 (예: 신경망, SVR), 이 값이 클수록 학습 시간이 오래 소요될 뿐만 아니라, 과적합으로 이어질 위험이 있음 • 따라서 **복잡한 모델을 학습**할 때, **일부러 max_iter를 작게 잡아서** 과적합을 회피하기도 함 (주의: 단순한 모델은 이 값을 작게 잡을 필요가 없음)
정규화 회귀 모델	alpha	• 복잡도와 반비례 관계
의사결정나무	max_depth	• 복잡도와 정비례 관계
	min_samples_leaf	• 복잡도와 반비례 관계
SVM	C, gamma, degree	• 복잡도와 약한 정비례 관계
	kernel	• poly > rbf > linear 순으로 과적합 가능성이 높음

위 표의 파라미터는 sklearn의 인자 이름을 기준으로 작성하였음

I 복잡도 파라미터 튜닝 개요

- **복잡도 파라미터**란 복잡도에 영향을 주는 파라미터로, 이 값에 따라 **과적합 정도가 결정**되므로 매우 신중하게 튜닝해야 함 (계속)

모델	파라미터	영향
로지스틱 회귀	C	• 복잡도와 반비례 관계
신경망	hidden_layer_sizes	• 복잡도와 강한 정비례 관계 • 정확히는 hidden_layer_sizes에 따라 가중치 개수가 결정되고, 가중치 개수가 복잡도를 결정함
SVR	epsilon	• 복잡도와 강한 반비례 관계
Tree Ensemble	max_depth	• 복잡도와 정비례 관계이며, 과적합을 피하기 위해 보통 4이하로 설정
	learning_rate (random forest 제외)	• 복잡도와 정비례 관계

위 표의 파라미터는 sklearn의 인자 이름을 기준으로 작성하였음

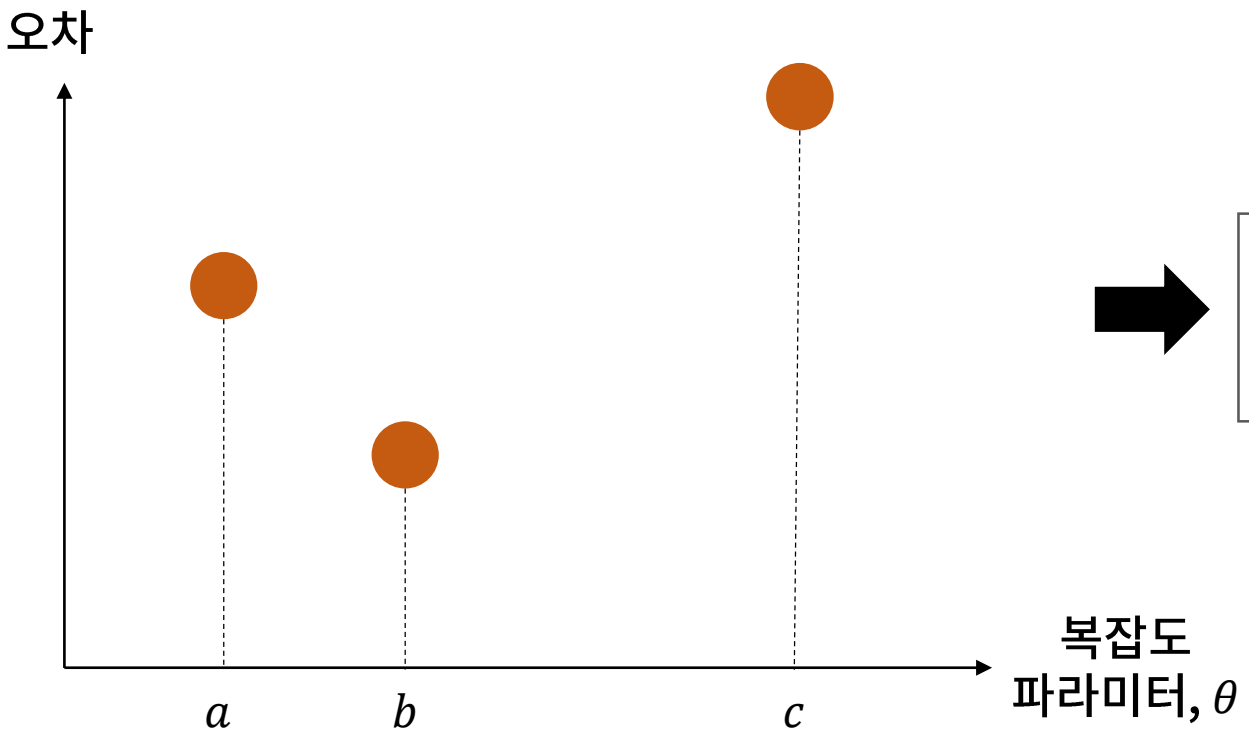
학습시 우연성이 개입되는 모델의 복잡도 파라미터 튜닝

- 경사하강법 등의 방법으로 학습되는 모델 (예: 회귀모델, 신경망 등)은 초기값에 의한 영향이 매우 큼

- 따라서 복잡도 파라미터 변화에 따른 성능 변화의 패턴을 확인하기 어려운 경우가 많으므로, seed를 고정한 뒤 튜닝을 수행해야 함

I 복잡도 파라미터 튜닝

- seed가 고정되어 있거나, 학습 시 우연 요소가 개입되지 않는 모델의 경우에는 복잡도 파라미터에 따른 성능 변화 패턴 확인이 상대적으로 쉬움
- 복잡도 파라미터가 둘 이상인 경우에는 서로 영향을 주기 때문에 반드시 **두 파라미터를 같이 조정**해야 함
- 파라미터 그리드 크기를 줄이기 위해, **몇 가지 파라미터 값을 테스트**한 후 범위를 설정하는 것이 바람직함



적절한 탐색 구간 1: $a \leq \theta \leq b$

적절한 탐색 구간 2: $b \leq \theta \leq c$

Chapter.

지도학습 모델 및 파라미터 선택

감사합니다

FAST CAMPUS
ONLINE
데이터 탐색과 전처리 I

강사. 안길승