Data Mining

머신 러닝 (Machine Learning)

학습 목표

- 머신 러닝에 관련된 공통적인 내용을 알아본다.

주요 내용

1. 머신 러닝
2. 모델의 적합
3. 편향과 분산
4. 데이터셋 분할
5. 혼동 행렬
6. 특징 선택

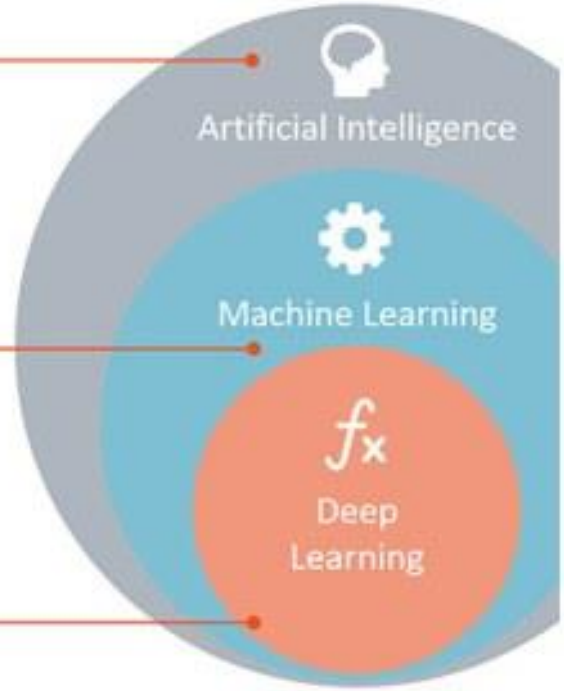
1. 머신 러닝

머신 러닝 (Machine Learning)

Artificial Intelligence
Any technique which enables computers to mimic human behavior.

Machine Learning
Subset of AI techniques which use statistical methods to enable machines to improve with experiences.

Deep Learning
Subset of ML which make the computation of multi-layer neural networks feasible.

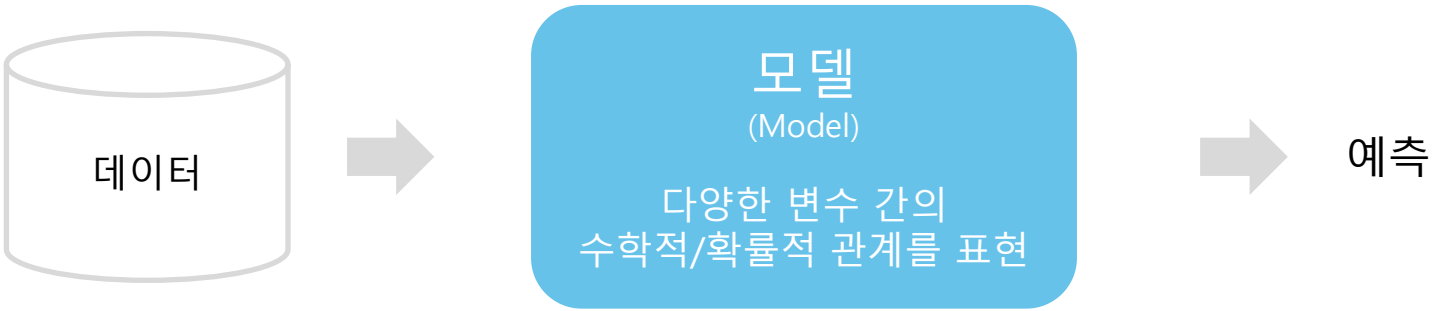


https://www.kdnuggets.com/2017/07/rapidminer-ai-machine-learning-deep-learning.html

머신 러닝 (Machine Learning)

머신 러닝은 데이터를 이용해서 모델을 학습하고

학습된 모델을 이용해서 예측하는 것



- 이메일이 스팸인지 아닌지 예측
- 신용카드 사기 예측
- 쇼핑 고객의 클릭할 확률이 높은 광고 예측
- 슈퍼볼에서 우승할 미식 축구팀 예측

머신 러닝 방법

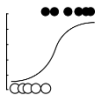
머신 러닝
(Machine Learning)

지도 학습
(Supervised Learning)

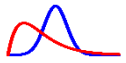
분류
(Classification)

k-최근접 이웃
(k-Nearest Neighbor)

로지스틱 회귀
(Logistic Regression)



나이브 베이즈
(Naïve Bayes)



서포트 벡터 머신
(Support Vector Machine)

결정 트리
(Decision Trees)

회귀
(Regression)

선형 회귀
(Linear Regression)

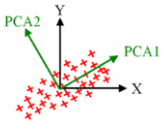
딥러닝
(Deep Learning)

비지도 학습
(Unsupervised Learning)

군집화
(Clustering)

연관 규칙
(Association)

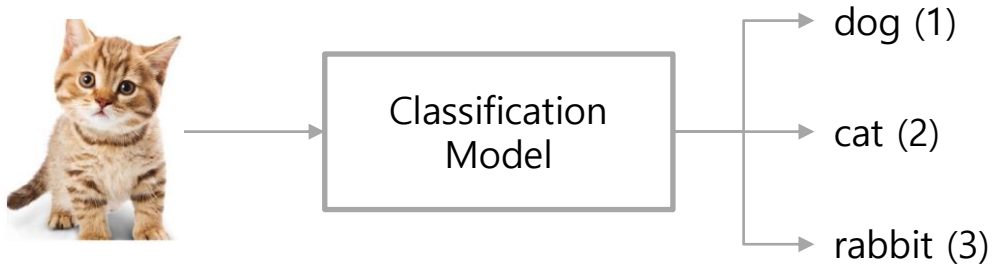
차원 축소
(Dimension Reduction)



추천 시스템 – 협업 필터링
(Collaboration Filtering)

분류 vs. 회귀

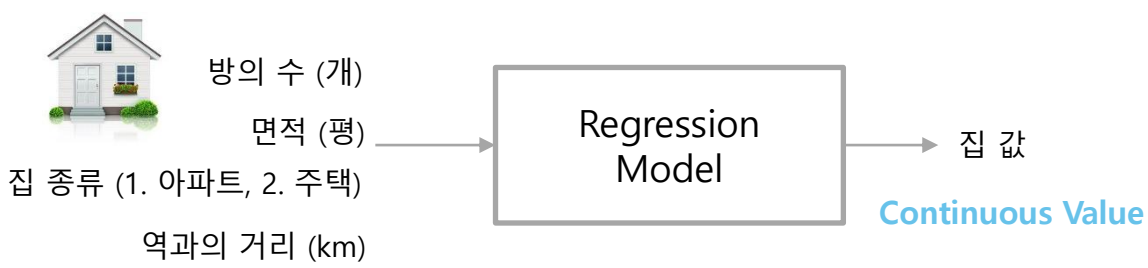
분류 문제 (Classification)



Discrete Value

- **입력 데이터에 대한 클래스를 또는 카테고리를 예측하는 문제**
- 출력은 입력 데이터가 속할 클래스
- **확률 모델** : 입력 데이터가 각 Class에 속할 확률 분포를 예측
  ex) Bernoulli, Categorical Distribution

회귀 문제 (Regression)

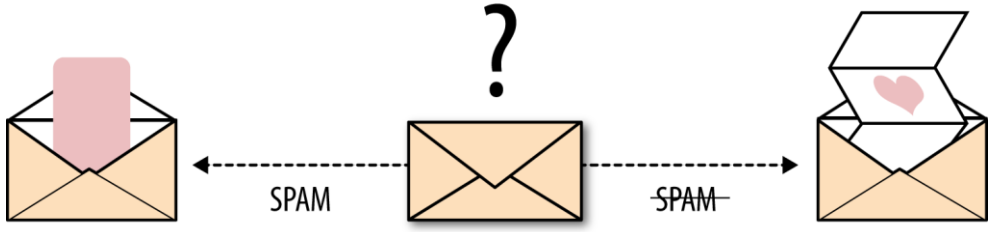


Continuous Value

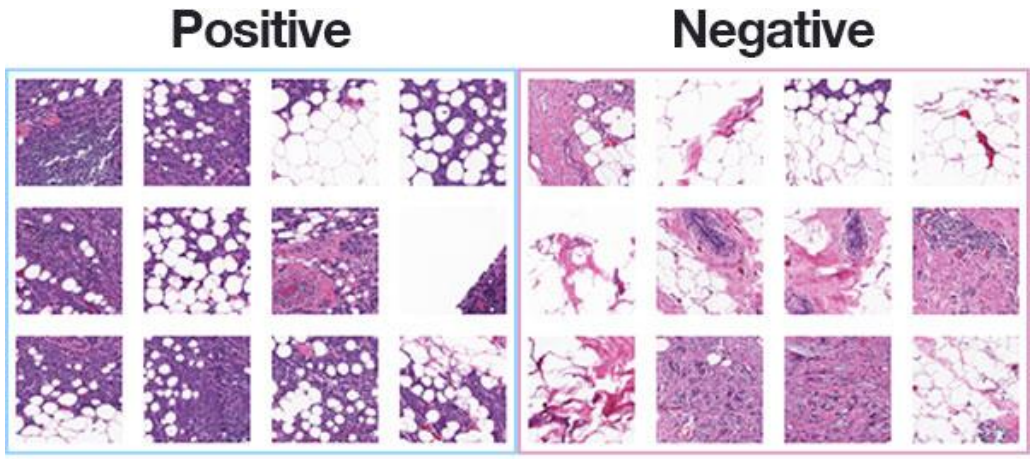
- **여러 독립 변수와 종속 변수의 관계를 함수 형태로 분석하는 문제**
- 출력은 입력 데이터에 대한 함수 값
- **확률 모델** : 관측 값에 대한 확률 분포를 예측
  ex) Gaussian Distribution

이진 분류 (Binary Classification)

메일이 스팸일 확률은?



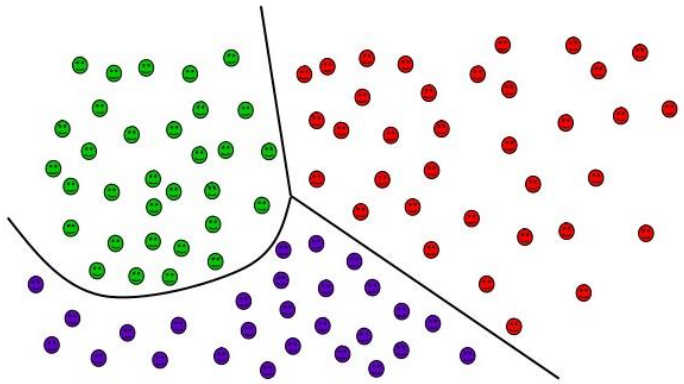
세포 조직이 유방암 양성일 확률은?



얼굴이 가짜 얼굴일 확률은?

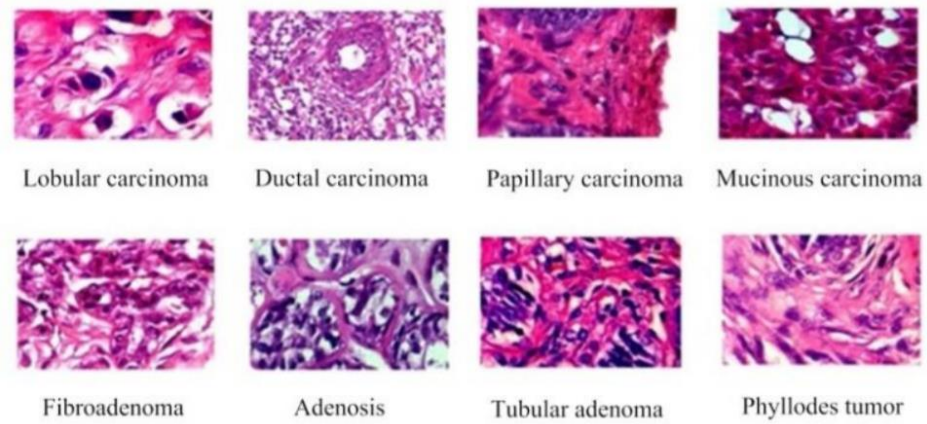
다중 분류 (Multiclass Classification)

데이터가 각 Class에 속할 확률은?

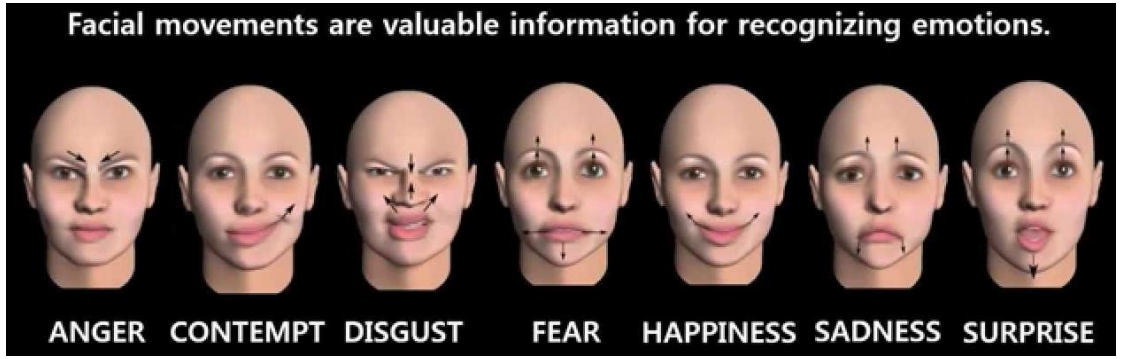


이미지 인식

유방암 종류 인식



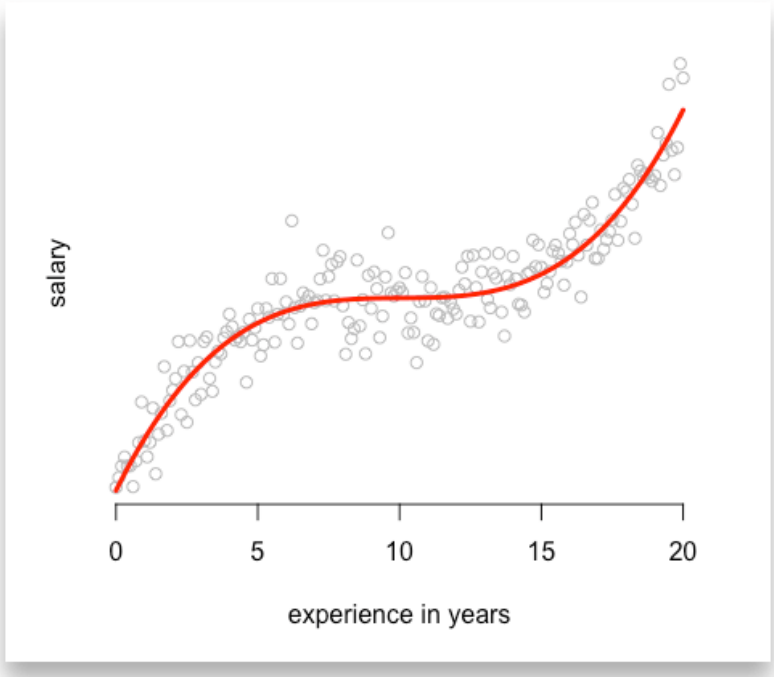
감정 인식



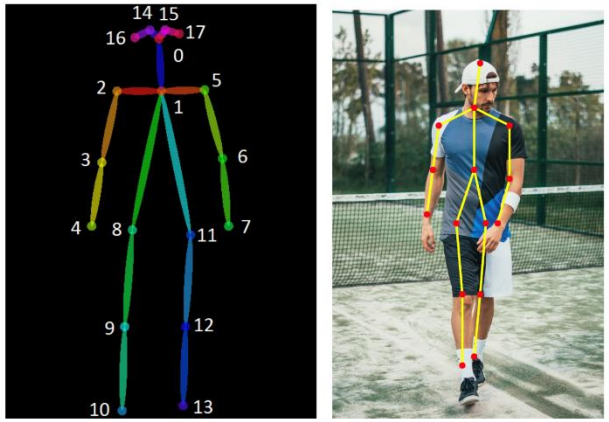
https://www.kaggle.com/ashishpatel26/tutorial-facial-expression-classification-keras

회귀 (Regression)

입력 값에 대한 연속 함수의 함수 값을 예측

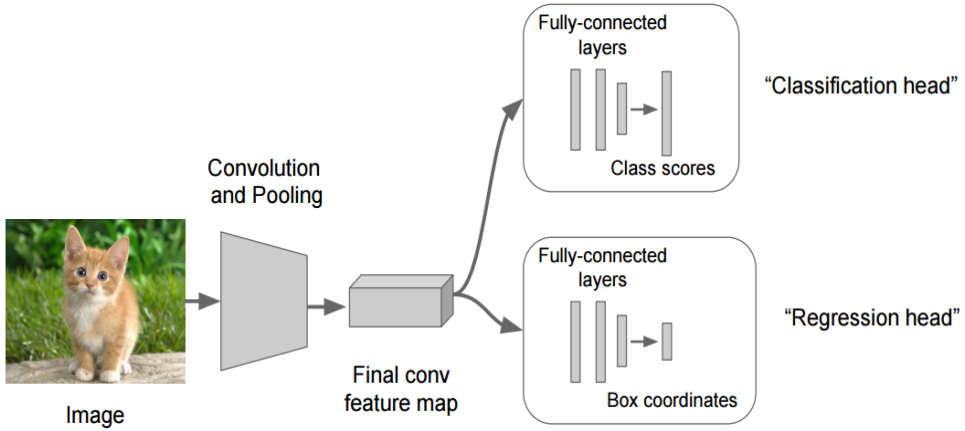


Human Pose Estimation



https://blog.nanonets.com/human-pose-estimation-2d-guide/

Object Detection에서 Bounding Box 예측

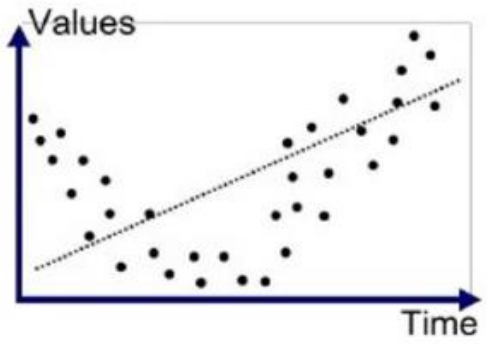


2. 모델의 적합

모델의 적합 (fit)

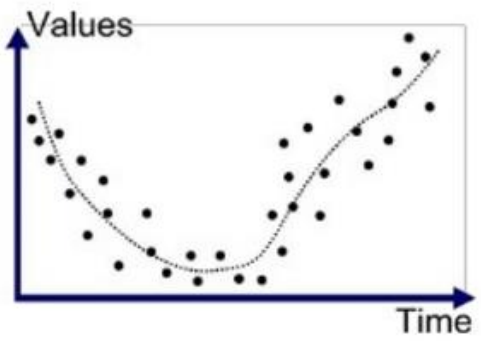
모델이 데이터를 얼마나 잘 표현하는지를 적합도라고 한다.

과소 적합 (Underfitting)



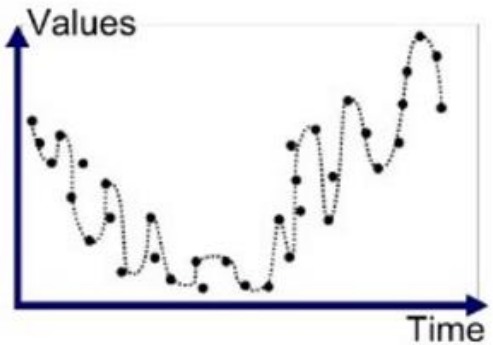
- 데이터의 분포를 제대로 반영하지 못하는 곡선을 표현
- 표현력이 좋은 모델로 변경해야 함

적합 (Fitting)



- 데이터를 잘 설명하는 곡선을 표현
- 관측 데이터의 평균적인 지점을 지나는 부드러운 곡선을 표현

과적합 (Overfitting)

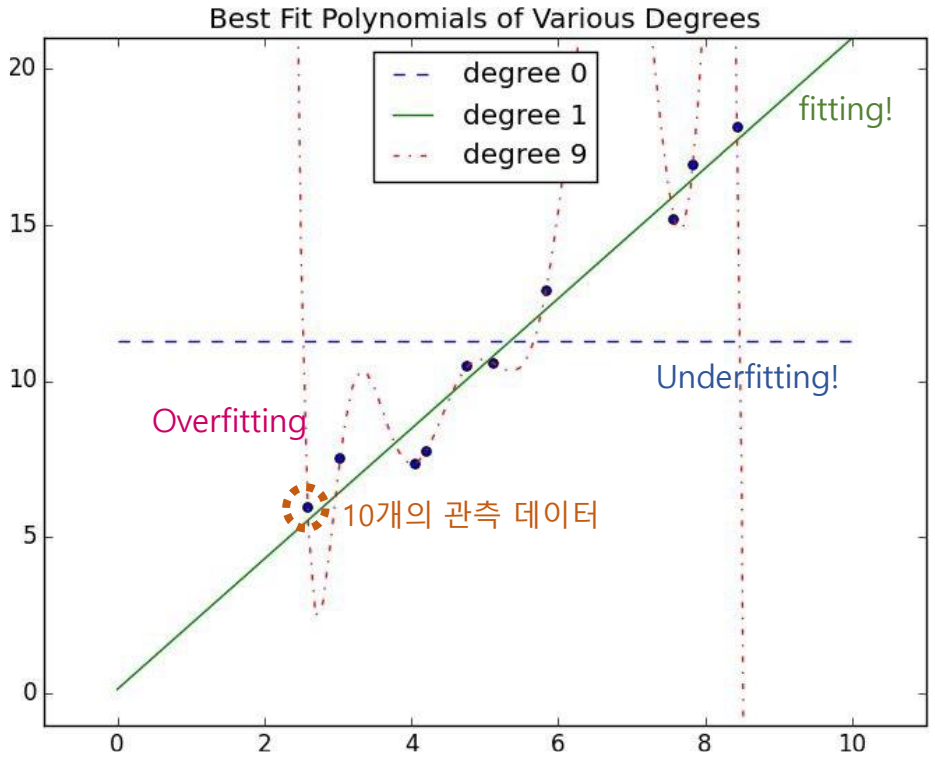


- 잡음까지 학습을 해서 모든 점을 지나는 복잡한 모양의 곡선을 표현
- 새로운 데이터에 대해 올바른 예측을 할 수 없음

모델의 적합 (fit)

10개의 관측 데이터를 3 종류의 다항식으로 학습했을 때 적합 결과

$f(x) = b + a_1 x + a_2 x^2 + \ldots + a_9 x^9$
(9차 다항식)



$f(x) = b + a_1 x$ (1차 다항식)

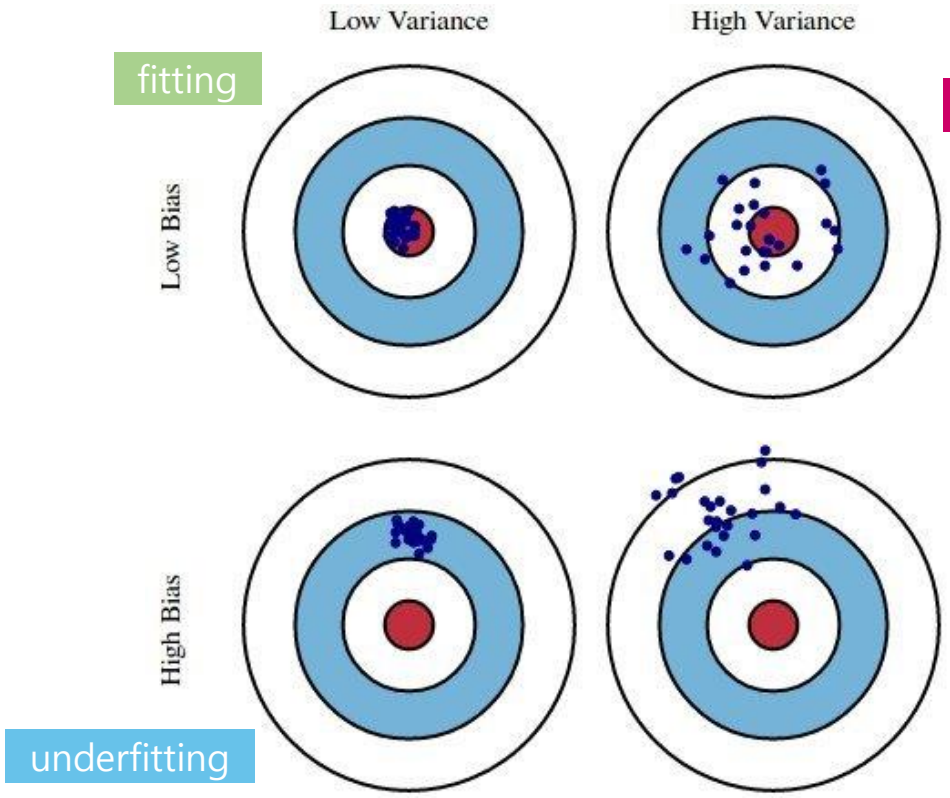
$f(x) = b$ (상수 함수)

데이터 수에 비해 모델 복잡하면 과적합이 생긴다.

3. 편향과 분산

편향과 분산 (Bias and Variance)

편향과 분산은 오차의 종류로 모델의 상태를 표현한다.



편향 (Bias)

- 최적의 모델(optimal)과 근사 모델(approximation)의 차이

분산 (Variance)

- 데이터셋이 달라질 때마다 모델이 다르게 근사
- 근사 모델들의 분산
- 데이터셋이 무한한다면 근사 모델은 최적의 모델로 수렴한다.

참고 모델의 오차

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise}$$

$$\mathrm{E}\left[\left(t - \hat{f}(x)\right)^2\right] = \text{bias}[\hat{f}(x)]^2 + \text{var}[\hat{f}(x)] + \sigma^2$$

제곱 오차 편향 분산 관측 오차

회귀 모델에서

$f(x)$	최적의 모델
$t = f(x) + \epsilon$	관측 데이터
$\epsilon \sim \mathcal{N}(\epsilon \mid 0, \sigma^2)$	관측 오차
$\hat{f}(x)$	근사 된 모델

최적의 모델과 근사 된 모델과의 차이

$$\text{bias}[\hat{f}(x)] = \mathrm{E}\left[\mathrm{E}\left[\hat{f}(x)\right] - f(x)\right]$$

편향

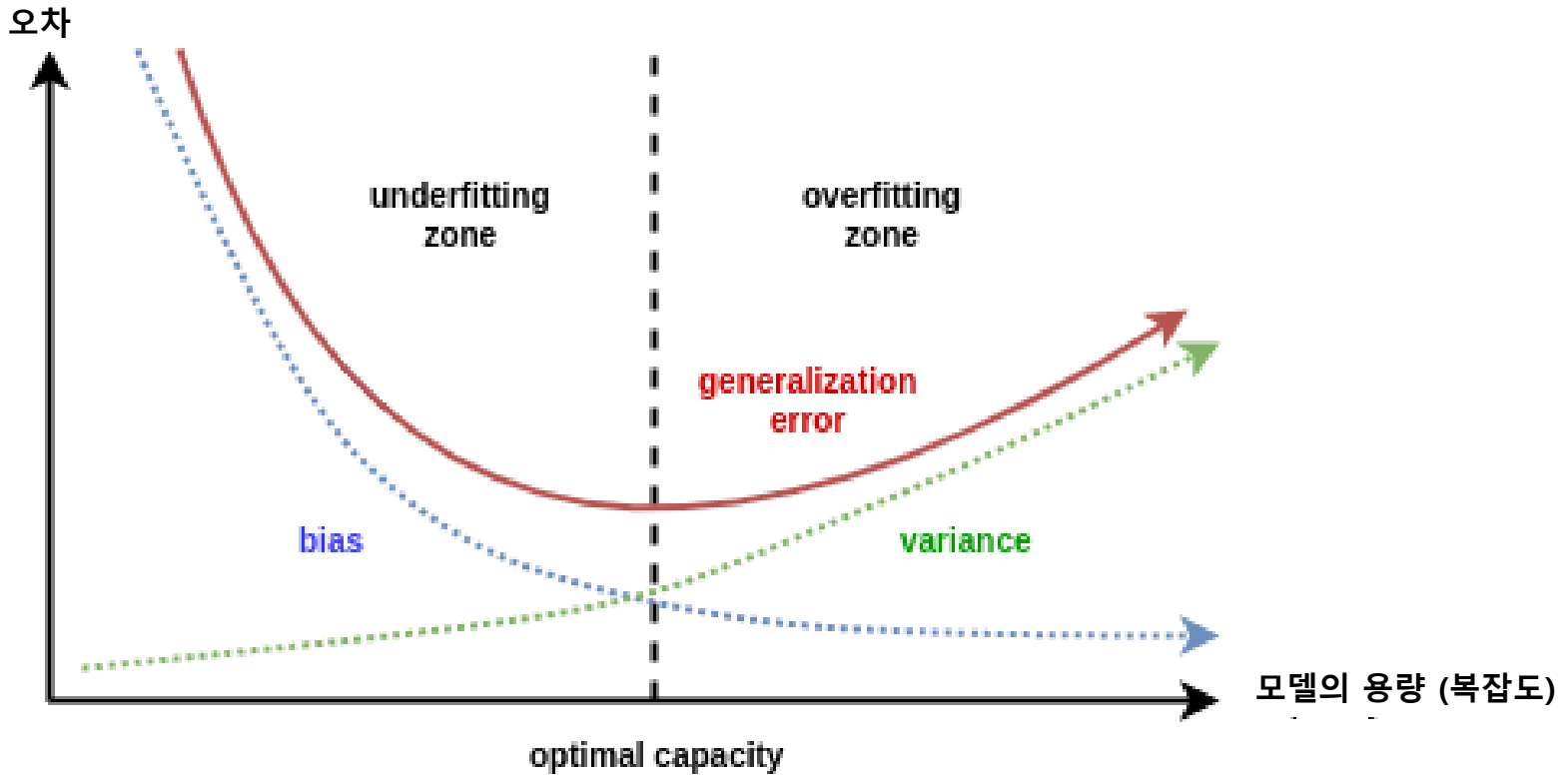
근사 된 모델의 분산

$$\text{var}[\hat{f}(x)] = \mathrm{E}\left[\left(\hat{f}(x) - \mathrm{E}\left[\hat{f}(x)\right]\right)^2\right]$$

분산

편향과 분산 (Bias and Variance)

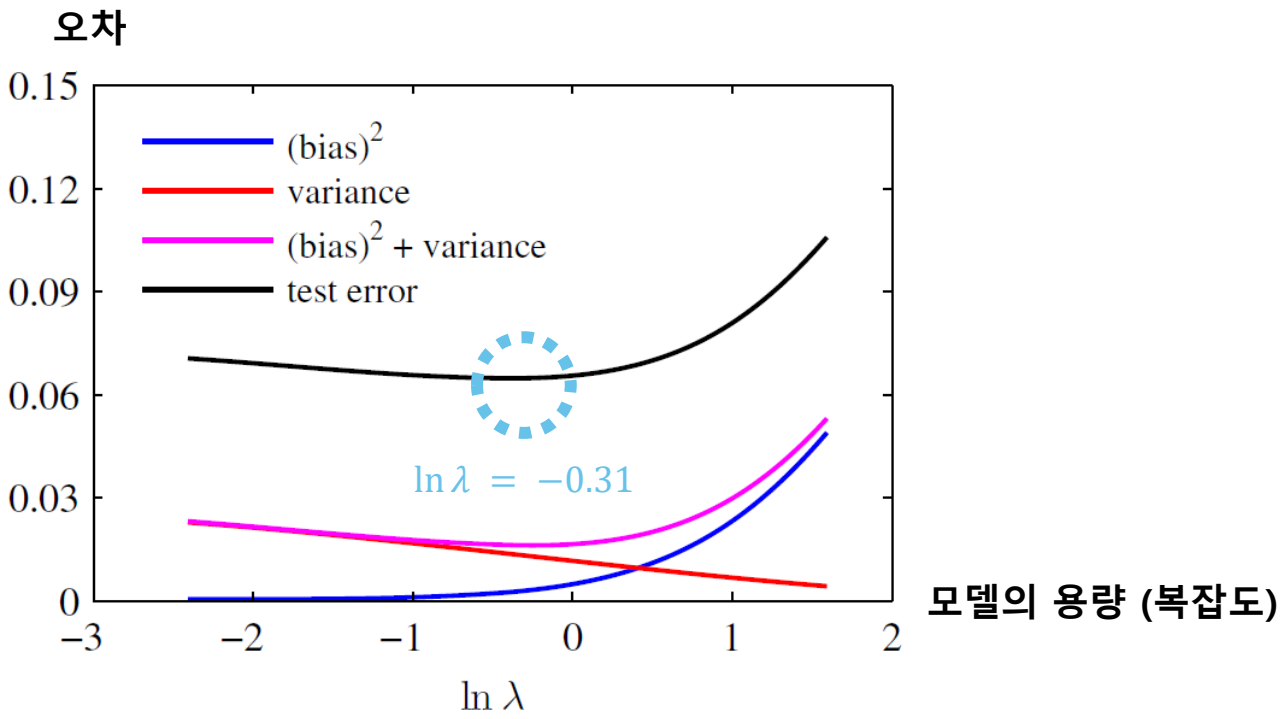
편향과 분산의 적정한 지점을 찾아야 함



- 모델의 편향이 크면 파라미터를 추가해서 모델의 복잡도를 올려줘야 한다.
- 분산이 크면 모델의 파라미터를 줄이거나 데이터를 늘려야 한다.

편향과 분산 (Bias and Variance)

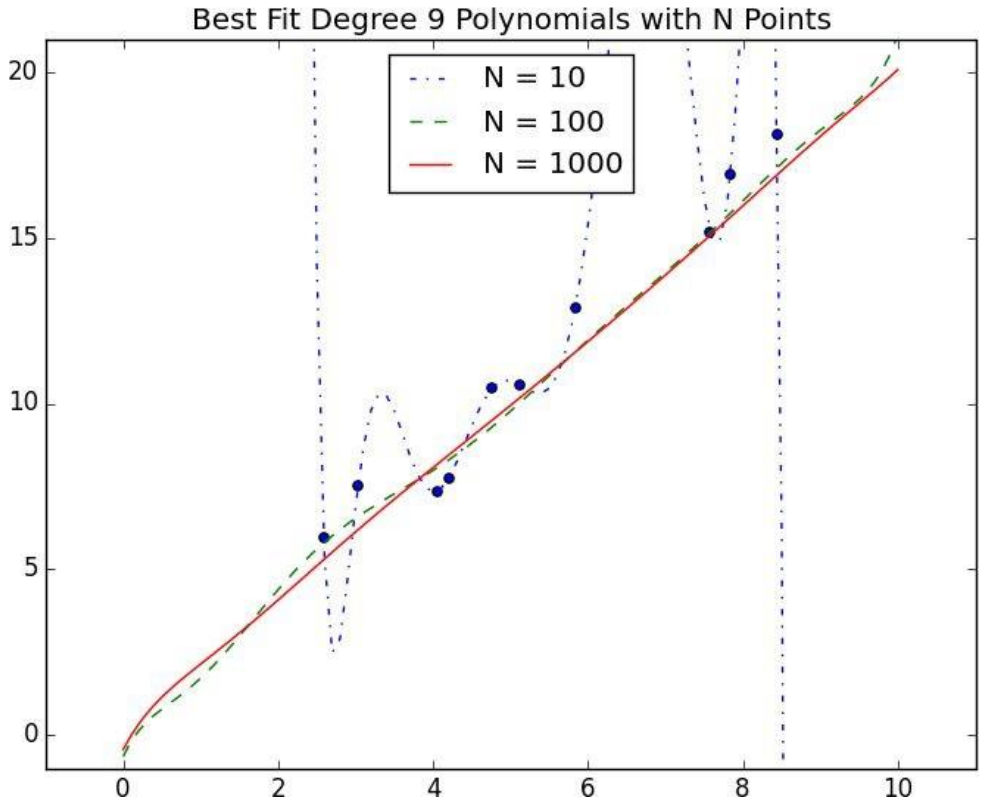
모델의 편향과 분산의 적정한 지점을 어떻게 찾을 수 있을까?



테스트 오차가 줄어들다가 다시 증가하는 지점이 모델 용량의 최적점이 됨

편향과 분산 (Bias and Variance)

데이터를 늘려주면 9차 다항식이 과적합 상태에서 적합 상태로 변환됨

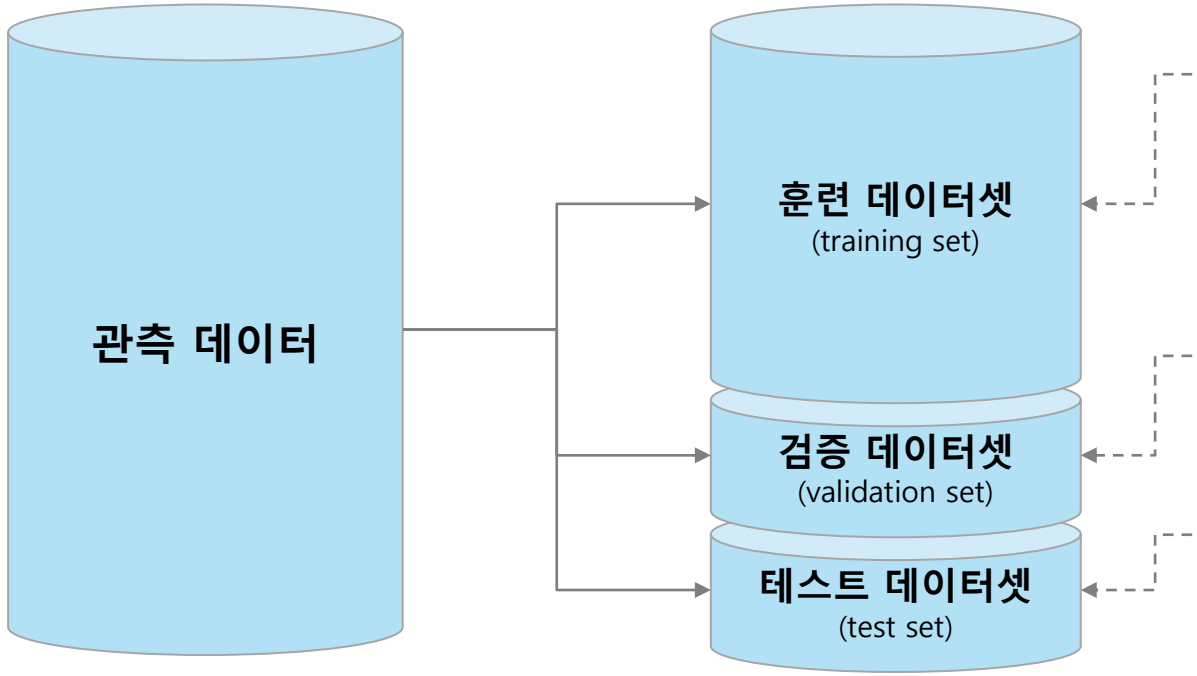


단, 모델에 편향이 있다면 데이터를 늘려줘도 소용이 없다.

4. 데이터셋 분할

데이터셋 분할 (Data Split)

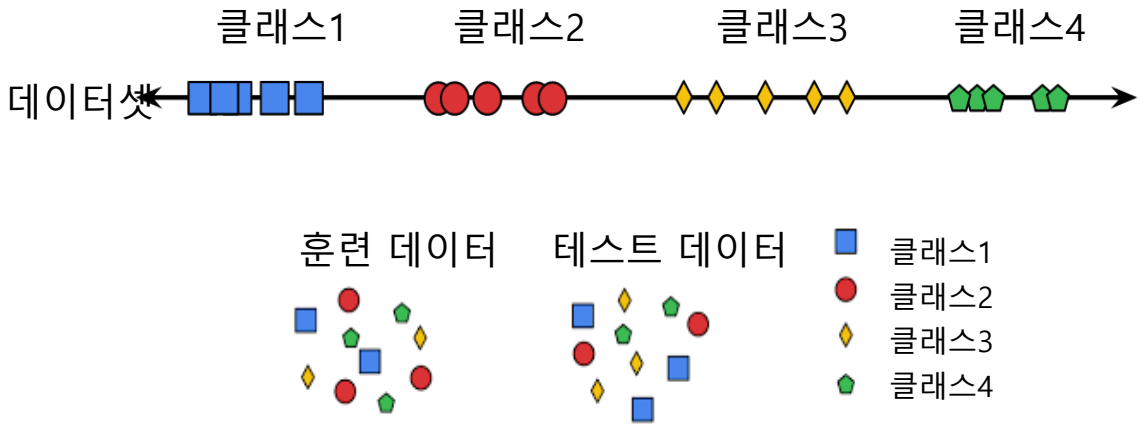
데이터셋을 분할해서 모델의 성능 평가를 하면
모델의 과적합 여부를 알아낼 수 있다.



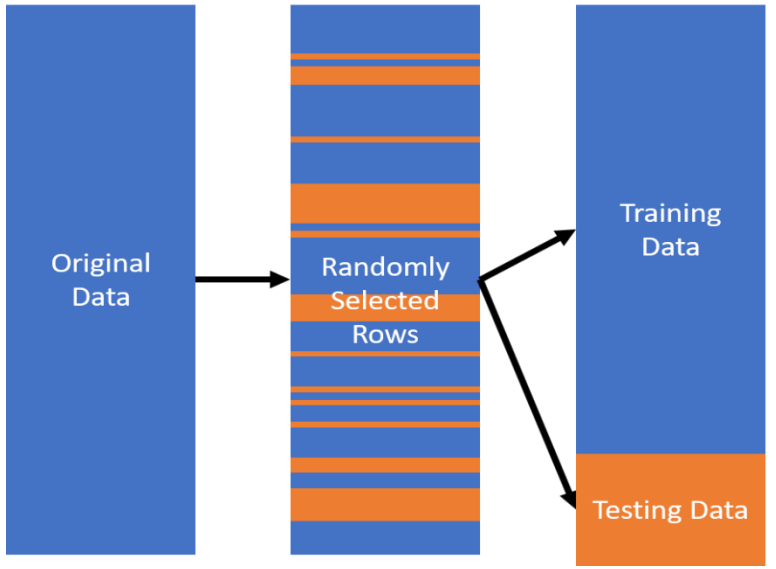
- 모델 훈련 용도
- 훈련된 모델의 성능을 확인해서 모델 및 훈련 관련 하이퍼파라미터를 튜닝하는 용도
- 하이퍼라파미터 : 훈련 전에 미리 설정하는 매개변수
- 모델의 최종 성능을 측정하기 위한 용도

데이터셋 분할 (Data Split)

데이터가 골고루 분할되어야 한다!



데이터셋을 "랜덤하게" 분할하자.



- 데이터셋 전체를 골고루 섞어서 분할하거나
- 데이터셋의 인덱스를 랜덤하게 샘플링

데이터셋 분할 (Data Split)

데이터셋 분할

```python
import random
from typing import TypeVar, List, Tuple
X = TypeVar('X')  # generic type to represent a data point

def split_data(data: List[X], prob: float) -> Tuple[List[X], List[X]]:
    """Split data into fractions [prob, 1 - prob]"""
    data = data[:]                      # Make a shallow copy
    random.shuffle(data)                # because shuffle modifies the list.
    cut = int(len(data) * prob)         # Use prob to find a cutoff
    return data[:cut], data[cut:]       # and split the shuffled list there.
```

- 데이터셋을 분할 할 때는 랜덤 선택하는 것이 중요
- 데이터를 무작위로 섞고 비율에 맞춰 분할

검증 : 1000개 데이터 분할

```python
data = [n for n in range(1000)]
train, test = split_data(data, 0.75)
```

- 1000개의 데이터를 생성해서 분할

데이터셋 분할 (Data Split)

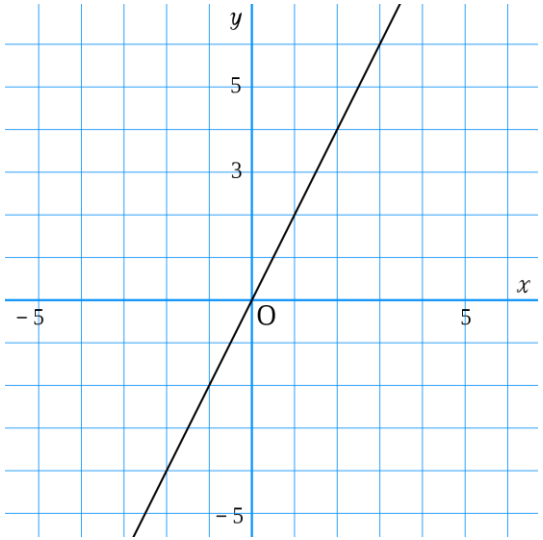
훈련 데이터셋과 테스트 데이터셋 분할

```
Y = TypeVar('Y')  # generic type to represent output variables

def train_test_split(xs: List[X],
                     ys: List[Y],
                     test_pct: float) -> Tuple[List[X], List[X], List[Y], List[Y]]:
    # Generate the indices and split them.
    idxs = [i for i in range(len(xs))]
    train_idxs, test_idxs = split_data(idxs, 1 - test_pct)

    return ([xs[i] for i in train_idxs],  # x_train
            [xs[i] for i in test_idxs],   # x_test
            [ys[i] for i in train_idxs],  # y_train
            [ys[i] for i in test_idxs])   # y_test
```

- 데이터셋 길이만큼의 인덱스 리스트 생성
- 훈련 데이터셋과 테스트 데이터셋의 인덱스를 분할
- x_train, x_test, y_train, y_test 순으로 분할된 데이터 리스트를 반환

데이터셋 분할 (Data Split)

$y = 2x$



데이터 생성 ($y = 2x$)

```
xs = [x for x in range(1000)]  # xs are 1 ... 1000
ys = [2 * x for x in xs]       # each y_i is twice x_i
x_train, x_test, y_train, y_test = train_test_split(xs, ys, 0.25)
```

비율 검증

```
# Check that the proportions are correct
assert len(x_train) == len(y_train) == 750
assert len(x_test) == len(y_test) == 250
```

X와 y 짝 검증

```
# Check that the corresponding data points are paired correctly.
assert all(y == 2 * x for x, y in zip(x_train, y_train))
assert all(y == 2 * x for x, y in zip(x_test, y_test))
```

5. 혼동 행렬

혼동 행렬 (confusion matrix)

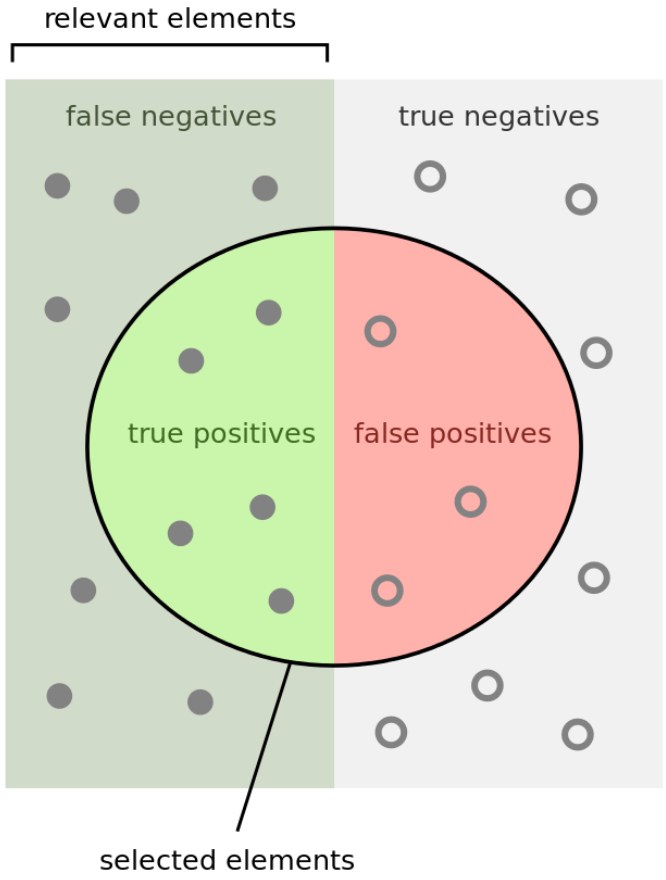
실제 클래스
(Actual Class)

예측 클래스
(Predictive Class)

	긍정 (Positive)	**부정 (Negative)**	
긍정 (Positive)	True Positive (TP)	False Positive (FP) **Type I Error**	**정밀도 (Precision)** $= \frac{TP}{TP + FP}$
부정 (Negative)	False Negative (FN) **Type II Error**	True Negative (TN)	**Negative Predictive Value (NPV)** $= \frac{TN}{TN + FN}$
	재현율 (Recall) $= \frac{TP}{TP + FN}$	**특이도 (Specificity)** $= \frac{TN}{TN + FP}$	**정확도 (Accuracy)** $= \frac{TP + TN}{TP + TN + FP + FN}$

민감도 (Sensitivity)라고도 함

정밀도와 재현율



How many selected items are relevant?

$$\text{Precision} = \frac{\text{true positives}}{\text{selected elements}}$$

How many relevant items are selected?

$$\text{Recall} = \frac{\text{true positives}}{\text{relevant elements}}$$

https://www.wikiwand.com/en/Precision_and_recall

혼동 행렬 예시

‘이름이 루크(Luke)이면 백혈병(leukemia)에 걸린다’는 판독 방법으로 100만명을 검증했다면?

	leukemia	no leukemia	total
“Luke”	70 (TP)	4,930 (FP)	5,000
not “Luke”	13,930 (FN)	981,070 (TN)	995,000
total	14,000	986,000	1,000,000

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{70 + 981{,}070}{1{,}000{,}000} = 0.98114$$

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{70}{5{,}000} = 0.014$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{70}{14{,}000} = 0.005$$

$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = 2 \times \frac{0.014 \times 0.005}{0.014 + 0.005} = 0.00736$$

성능 평가 (Performance Test)

정확도 (accuracy)

```python
def accuracy(tp: int, fp: int, fn: int, tn: int) -> float:
    correct = tp + tn
    total = tp + fp + fn + tn
    return correct / total
```

정밀도 (precision)

```python
def precision(tp: int, fp: int, fn: int, tn: int) -> float:
    return tp / (tp + fp)
```

재현율 (recall)

```python
def recall(tp: int, fp: int, fn: int, tn: int) -> float:
    return tp / (tp + fn)
```

F1 Score

```python
def f1_score(tp: int, fp: int, fn: int, tn: int) -> float:
    p = precision(tp, fp, fn, tn)
    r = recall(tp, fp, fn, tn)

    return 2 * p * r / (p + r)
```

정확도 (Accuracy)

- 전체 데이터에서 자신의 클래스를 정확히 맞춘 비율

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

정밀도 (Precision)

- True라고 예측한 것 중에서 실제 True인 것의 비율

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

재현율 (Recall)

- 실제 True인 것 중에서 True라고 예측한 것의 비율

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

F1 Score

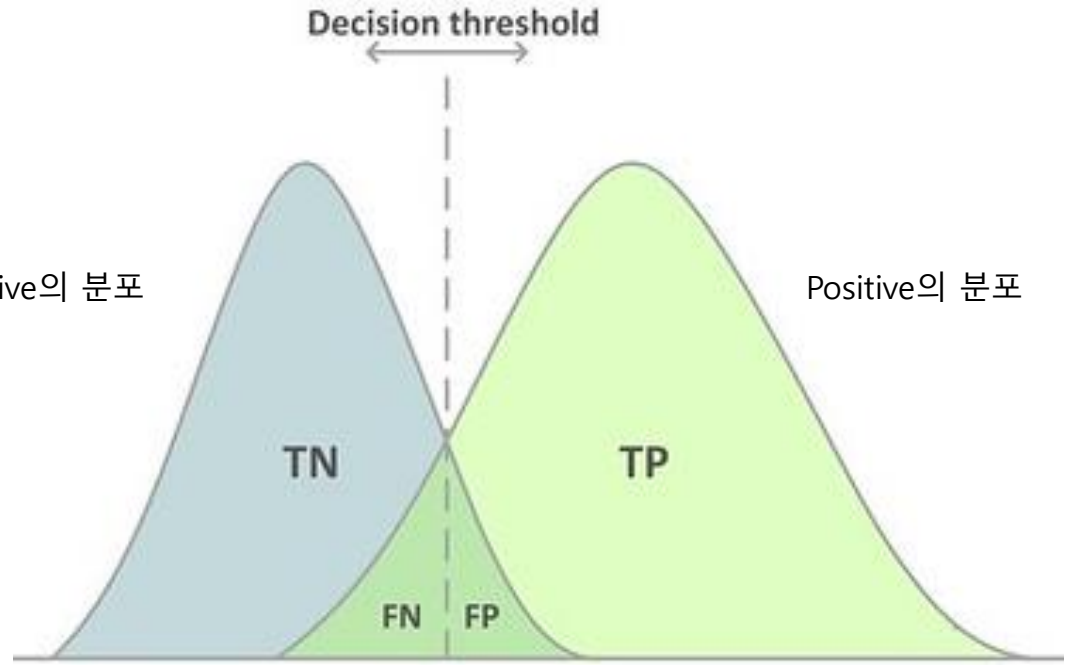
- 정밀도와 재현율의 조화평균

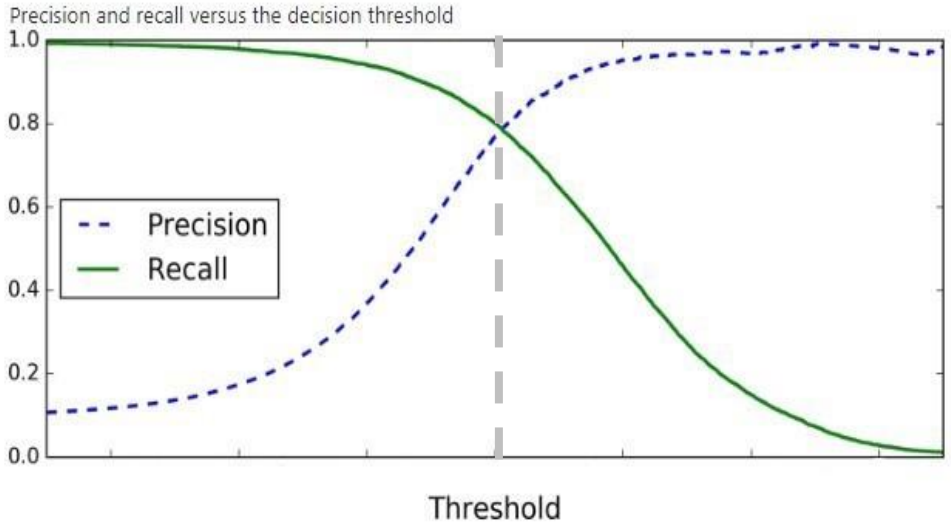
$$\text{F1 score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

* 조화 평균은 주어진 수들의 역수의 산술 평균의 역수

정밀도와 재현율 (Precision and Recall)

정밀도와 재현율이 적정한 값을 갖도록 판별 기준을 정해야 한다.

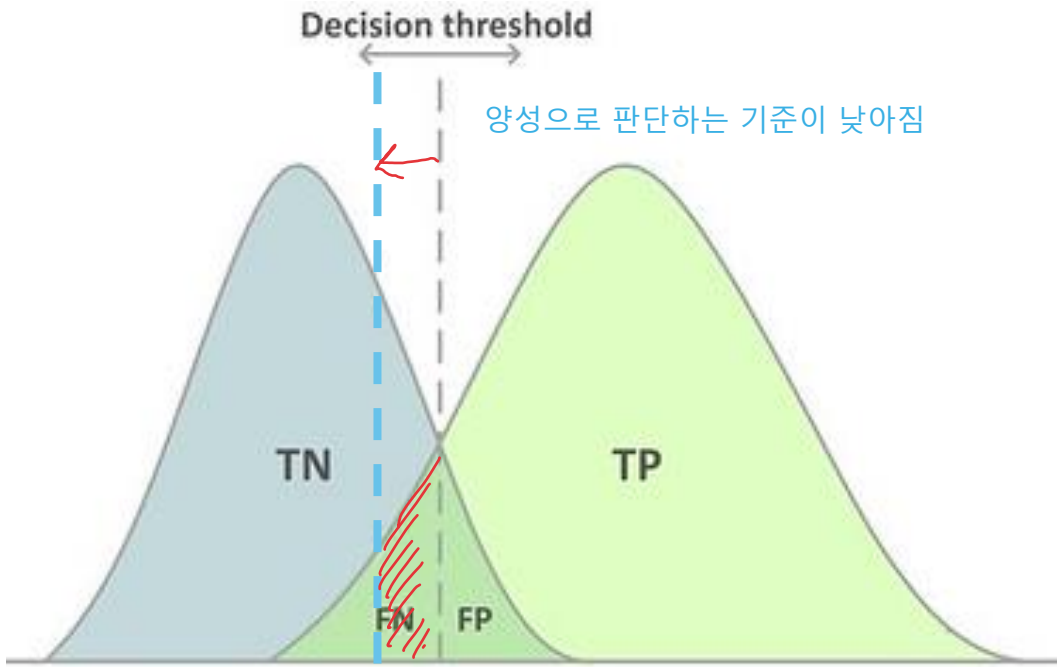




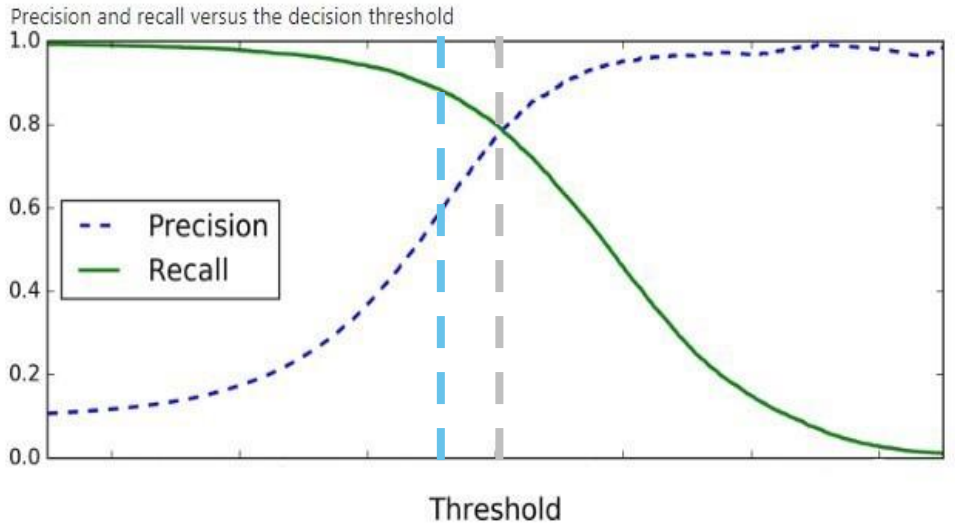
$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

정밀도와 재현율 (Precision and Recall)

양성(Positive)으로 판별하는 기준이 낮다면?



그래프의 면적으로 판단해보면
늘어난 FP 영역이 늘어난 TP 영역보다
크기 때문에 정밀도는 내려가게 됨



정밀도는 낮아지고 recall은 올라감

그래프의 면적으로 판단해보면
TP 영역은 늘어나고 FN 영역은 줄어들므로
재현율은 올라가게 됨

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

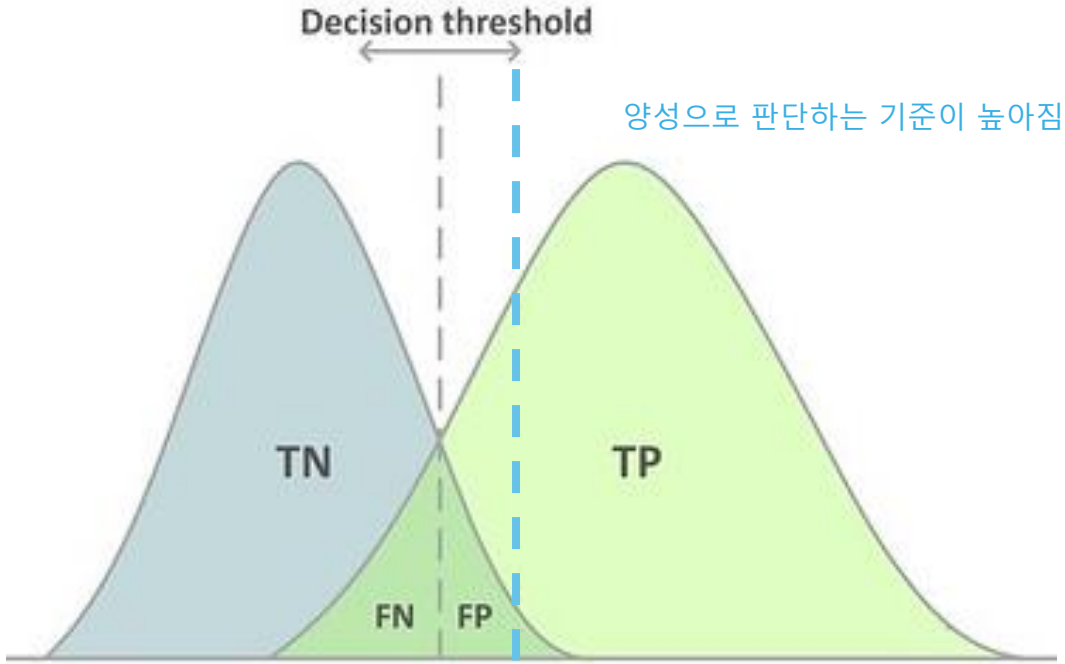
정밀도는 내려감 ⬇

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

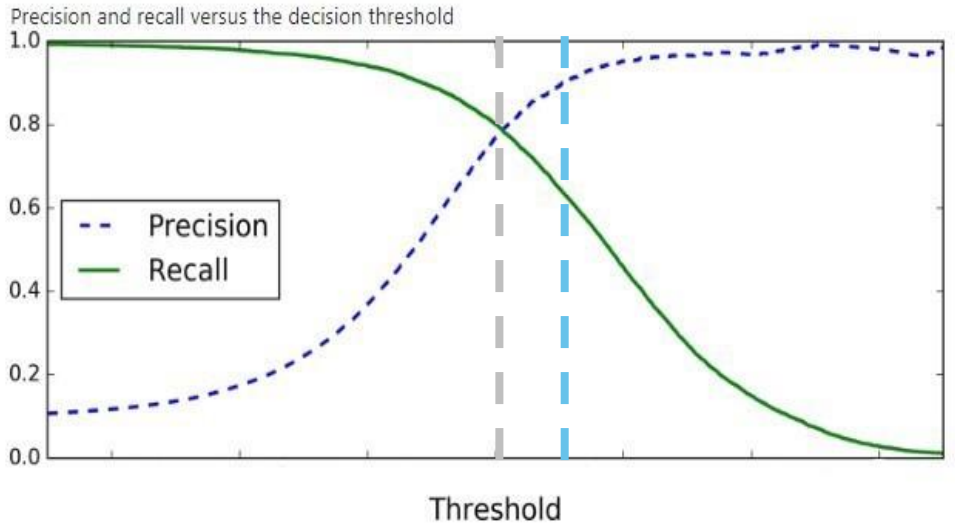
재현율은 올라감 ⬆

정밀도와 재현율 (Precision and Recall)

양성(Positive)으로 판별하는 기준이 높다면?



양성으로 판단하는 기준이 높아짐



그래프의 면적으로 판단해보면
즐어든 TP 영역보다 줄어든 FP 영역보다
크기 때문에 정밀도는 올라가게 됨

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

정밀도는 올라감 ⬆

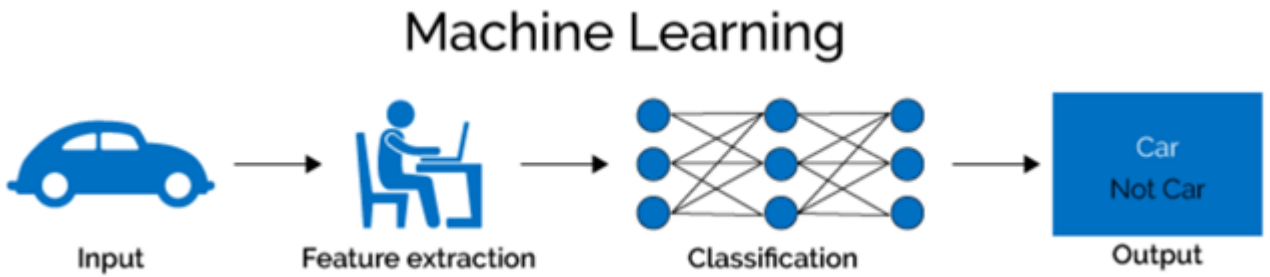
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

재현율은 낮아짐 ⬇

그래프의 면적으로 판단해보면
TP 영역은 줄어들고 FN 영역은 늘어나므로
재현율은 낮아지게 됨

6. 특징 선택

특징 추출과 선택 (Feature Extraction and Selection)

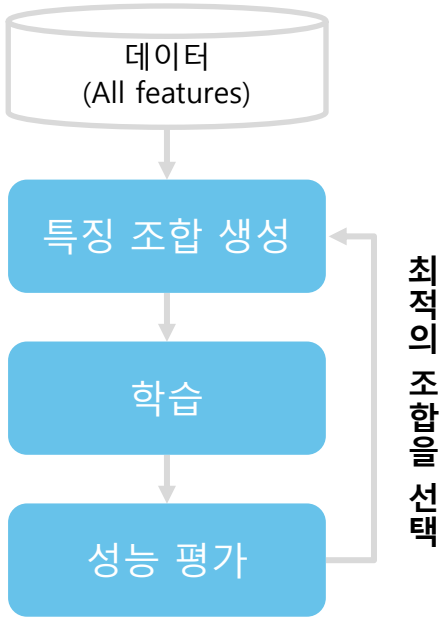


- **특징 추출** (Feature Extraction)
- 원래의 특징에서 새로운 특징을 생성하는 방법
- 데이터에 내재되어 있는 눈에 보이지 않는 특징을 추출하는 방법
- 차원 축소, 잠재 변수 학습

- **특징 선택** (Feature Selection)
- 원래의 특징에서 부분 집합을 선택하는 방법
- 특징은 많은데 샘플은 적은 경우에 사용
- 모델 해석이 쉬워지고 학습 시간이 줄어듦
- 과적합을 막을 수 있고 모델의 분산을 줄어들어 강건(robust) 해짐

특징 선택 (Feature Selection)

레퍼 방법 (Wrapper Method)



방법

- 특징의 조합을 생성해서 모델을 학습하고 성능을 평가
- 가장 성능이 좋은 조합을 선택
- 알고리즘 : recursive feature elimination(RFE), sequential feature selection(SFS), genetic algorithm, Univariate selection, Exhaustive, mRMR(Minimum Redundancy Maximum Relevance)

장점

- 특정 모델에 최적의 성능을 제공

단점

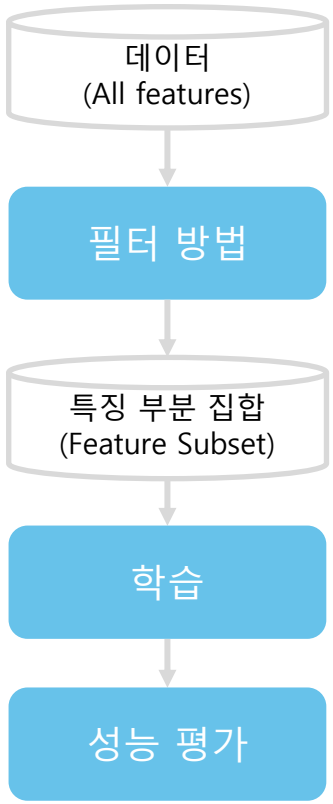
- 굉장히 많은 시간 비용이 필요하기 때문에 경험적인 방법과 접목해서 실행함
- 훈련 데이터셋에 과적합 될 수 있음

원소가 3개 일 경우
$\{x_1, x_2, x_3\}$
$\{x_1\}, \{x_2\}, \{x_3\}$
$\{x_1, x_2\}, \{x_2, x_3\}, \{x_1, x_3\}$
$\{x_1, x_2, x_3\}$
위의 7개의 원소에 대해서 성능평가 가능.

특징 선택 (Feature Selection)

독립변수, 종속변수와의 상관관계를 따짐

필터 방법 (Filter Method)



방법

- 전처리 과정에서 통계적 방법으로 특징을 선택
- 종속변수와 독립변수 간의 피어슨 상관계수를 가장 많이 사용
- 통계적 방법 : correlation coefficient, information gain, chi-square test, fisher score, variance threshold

장점

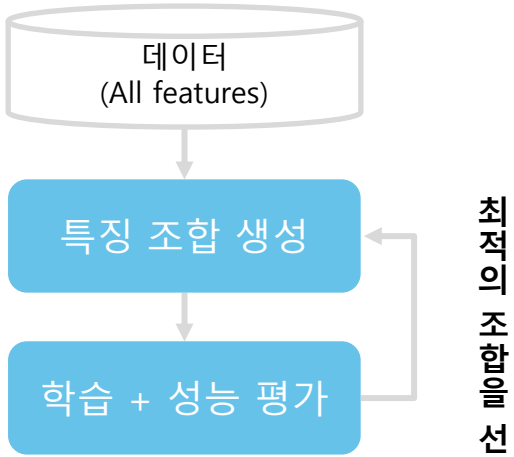
- 시간이 매우 단축됨

단점

- 모델에 맞는 특징이 아닌 일반화된 특징을 선택
- 모델 성능을 최적화된 특징 선택이 아닐 수 있음
- 따라서, 필터 방법을 적용한 후 래퍼 방식을 적용하기도 함

특징 선택 (Feature Selection)

임베드 방식 (Embedded Method)



방법

- 모델 자체에서 특징 선택을 하는 기능이 있는 경우
- 라소 회귀(Lasso Regression), 리지 회귀(Ridge Regression), 의사결정 트리 (Decision Tree) 알고리즘에 특징 선택 과정이 포함되어 있음

장점

- 특징 선택을 위한 전처리 과정이 사라짐

단점

- 특정 모델에서만 적용되는 방법

Thank you!