# 머신러닝 기초 II

## ML Session 2차시

빅데이터 분석 학회 D&A

# CONTENTS.


## 01. 머신러닝이란

- 머신러닝의 목적
- 비용함수
- 경사하강법

## 02. 분류와 회귀

- 분류와 회귀

## 03. 과적합

- 과적합이란
- 과적합 해결방법

## 04. 교차검증

- Hold-Out
- K-fold
- Stratified K-fold
- LOOCV

## 05. 평가지표

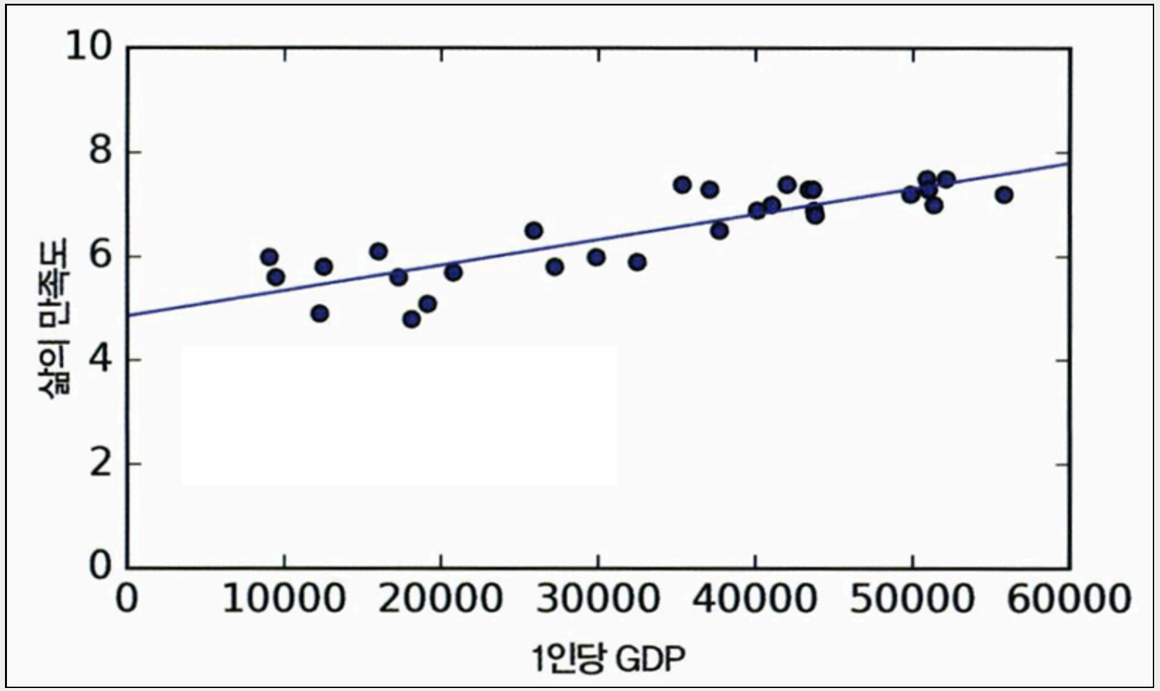
- 회귀 평가지표
- 분류 평가지표

# 머신러닝의 목적

## Task

데이터를 활용하여 현실의 문제를 해결 or 차선책을 제시

- 1인당 GDP와 삶의 만족도 간의 관계
- 대선 주자의 소득과 지지율 사이의 관계
- 고객 사진을 보고 고객의 나이를 예측




## 머신러닝 모델의 목적

- 기계에게 목적을 부여하기 위한 작업을 수행
- 비용함수 최소화
- ex) $\mathrm{cost}(W,b) = \frac{1}{N}\sum_{(x,y)\in\mathcal{D}} (y - \mathrm{prediction}(x))^2$

# 경사하강법

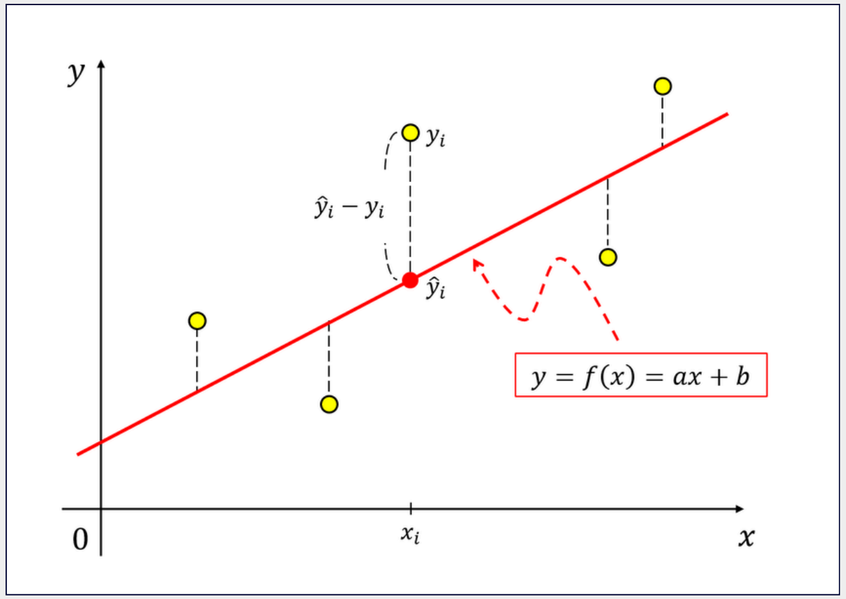
## 비용함수

머신러닝의 핵심 : 기계에게 내 의도를 전달하는 것

- 컴퓨터에게 내 의도를 전달하기 위해 알맞은 목적을 부여해야 함

비용함수(Cost Function)의 정의

- 설계자의 의도에 맞게 정의 가능
- 회귀분석에서 대표적인 비용함수 MSE
- 선형회귀식이 $\hat{y} = Wx + b$ 일 때,
- ex) $\text{cost}(W, b) = \frac{1}{N} \sum_{(x,y)\in\mathcal{D}} (y - \text{prediction}(x))^2$
- 즉, $cost(W, b)$ 가 최소가 되게 하는 $W$ 와 $b$를 구하는 것이 목표




최솟값을 구하면, 비용이 최소가 되므로 나의 목적과 가까운 결과를 얻을 수 있음

어떻게 최솟값을 구할까? 1. 쌩으로 미분하기 2. 경사하강법

# 경사하강법

## 쌩으로 미분하기 (해석적 계산 방법)

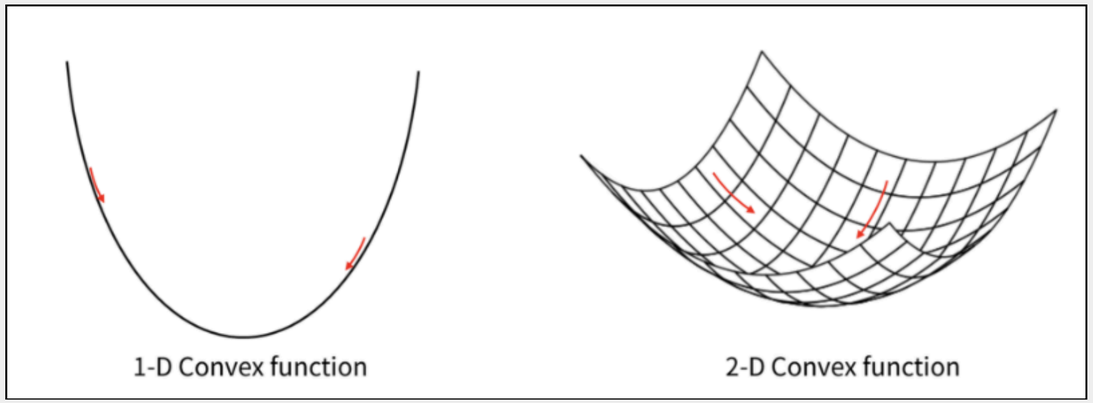
$$\text{cost}(W, b) = \frac{1}{N} \sum_{(x,y)\in\mathcal{D}} (y - \text{prediction}(x))^2$$

- 미분 결과값 : $\hat{\theta} = (X^T X)^{-1} X^T Y$ (선형회귀의 정규방정식)

위 식은 convex하기 때문에 전역 최솟값을 가짐

➡ 식의 최솟값은 1개

- BUT, 행렬곱, 역행렬 계산은 X가 많으면 계산하기 어려움
- 보다 효율적으로 계산할 수 있는 방법은 없을까?
  - 한 번에 미분하지 말고 차근차근 내려가자!
  - 경사하강법의 등장



# 경사하강법

## 경사하강법 (Gradient Descent)

- 한 지점에서 기울기를 구한 뒤, 기울기가 감소하는 방향으로 차근차근 내려가는 방법
- 매개변수를 업데이트할 때, 비용함수의 기울기를 사용하여 현재 위치에서 가장 가파른 경사 하강으로 이동함
- 최적화 과정에서 점진적으로 더 작은 손실 값을 구하는 iterative한 방법

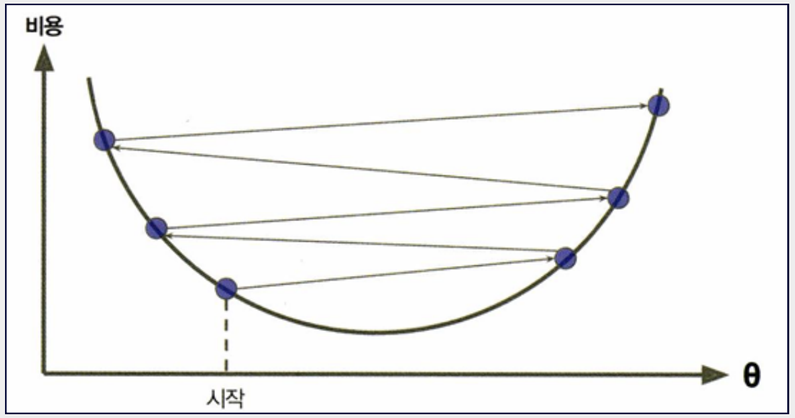
## 주의할 점

- X는 input 데이터(= 고정 값 ≠ 비용함수 공간에서 움직이는 변수 값), y는 output
- 우리가 찾는 것은 $\theta, \beta, w$
- ex) $\frac{\partial}{\partial\theta}MSE(\theta) = \frac{2}{m}\sum\left(\theta^T x^{(i)} - y^{(i)}\right)x_j^{(i)}$

# 경사하강법

## Learning rate

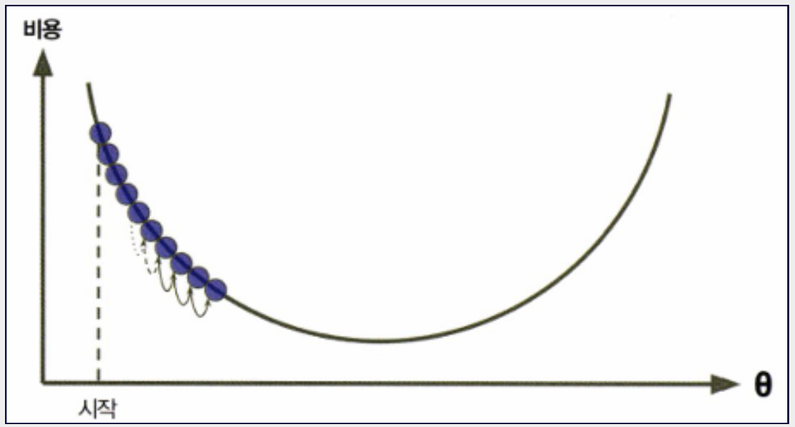
- 경사하강법 : 한 걸음씩 움직이면서 비용함수가 작아지는 지점을 찾는 방법
- 걸음마다 보폭은 어떻게 설정할 것인가? ➡ 보폭 : Learning rate




- Learning rate가 지나치게 큰 경우 (보폭이 매우 큰 경우)
- Learning rate가 지나치게 작은 경우 (보폭이 매우 작은 경우)

$$W := W - \alpha \frac{\partial}{\partial W} \mathrm{cost}(W) \cdots \alpha = \mathrm{learning\ rate}$$

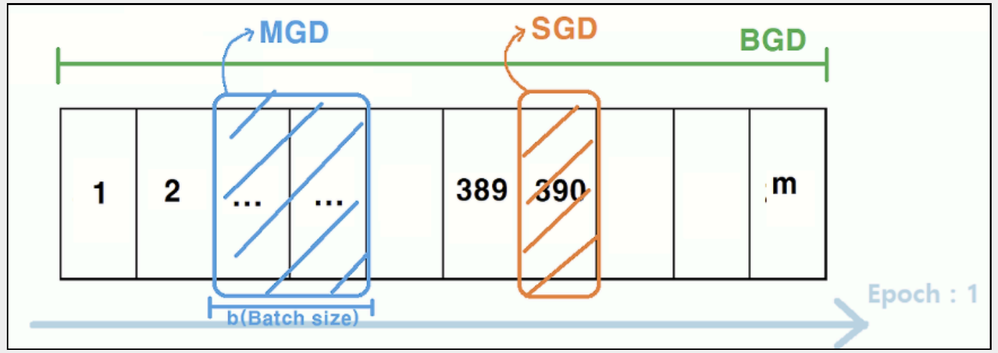
- Learning rate 설정은 설계자의 몫



# 경사하강법

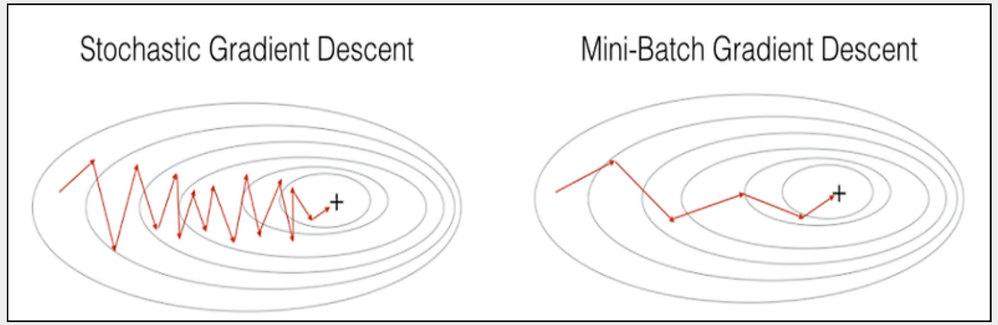
## 확률적 경사하강법 (Stochastic Gradient Descent)

- 매번 하나의 데이터를 사용하여 파라미터를 업데이트하는 방법
- 하나의 데이터는 무작위로 선택함




## 미니배치 경사하강법 (Mini-batch Gradient Descent)

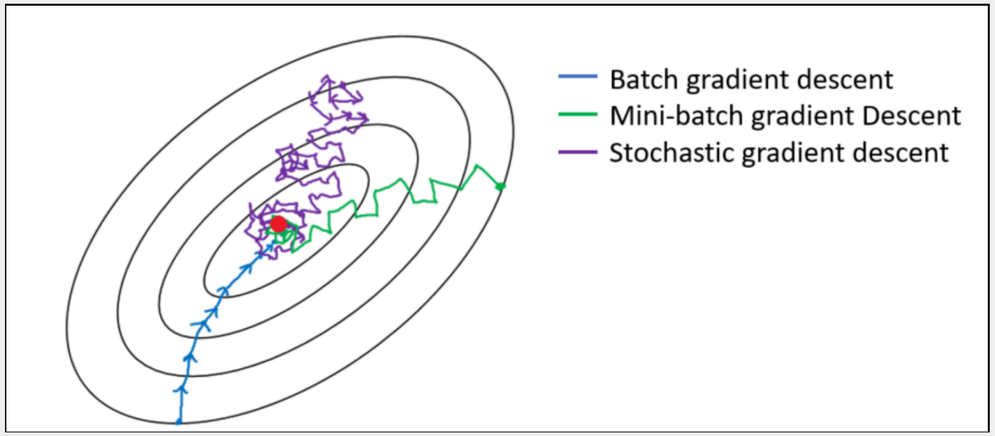
- BGD와 SGD의 절충안
- 데이터셋을 Mini batch로 나누어 각 batch에 대해 경사를 계산하고 파라미터를 업데이트하는 방법



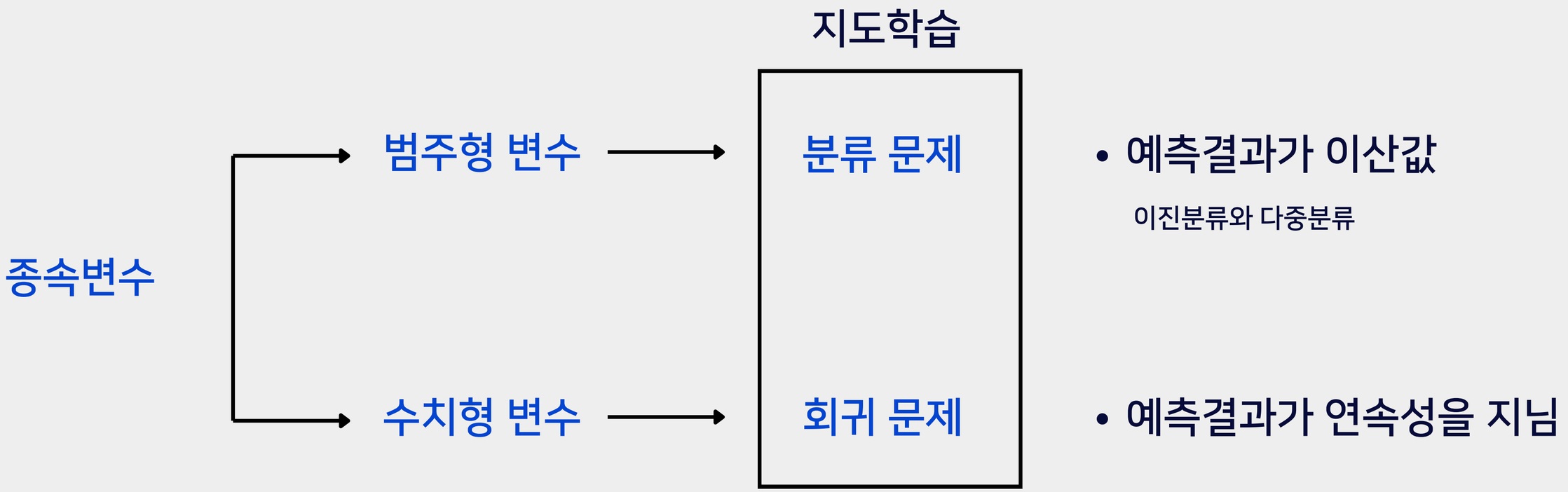
# 경사하강법

## 경사하강법 방법론 비교

| 구분 | BGD | SGD | MGD |
|---|---|---|---|
| 기울기 계산 방식 | 전체 데이터 | 1개의 데이터 | 미니배치 |
| 계산 속도 | 느림 | 매우 빠름 | 중간 |
| 기울기 안정성 | 매우 안정적 | 불안정 | 비교적 안정적 |
| 메모리 요구량 | 높음 | 매우 적음 | 중간 |
| 수렴 속도 | 느림 | 빠름 | 빠름 |
| 하이퍼파라미터 | 적음 | 많음 | 많음 |
| 스케일 조정 | 필요하지 않음 | 필요 | 필요 |



# 분류와 회귀




- 예측결과가 이산값
  이진분류와 다중분류
- 예측결과가 연속성을 지님

# 분류와 회귀

## 분류 문제 예시

- 공부시간(x)를 입력 받아 합격 여부 (y)를 예측
- 메일 발신인, 제목, 본문 내용 (x)를 입력 받아 스팸 메일 여부 (y)를 예측
- X-ray 사진과 영상 속 종양의 크기, 두께 (x)를 입력 받아 악성 종양 여부 (y)를 예측

## 회귀 문제 예시

- 공부시간(x)를 입력 받아 시험 점수 (y)를 예측
- 온도 (x)를 입력 받아 레모네이드 판매량 (y)를 예측
- 자동차 속도 (x)를 입력 받아 충돌 시 사망 확률 (y)를 예측

# 과적합이란

## 과대적합(overfitting)

- 학습이 지나치게 잘 된 상태 (train data 맞춤형 학습)
- train data를 너무 잘 학습하여 train data가 아닌 새로운 데이터에 대해서는 제대로 예측을 하지 못하는 상태

- 머신러닝 모델의 성능 저하, Kaggle 내 shake up 현상 (private 리더보드에서의 급격한 순위 변화)의 주요 원인이 됨
- 일반적으로 overfitting이 underfitting보다 더 많이 발생함
- overfitting을 항상 조심해야 함! train data로 학습한 모델의 성능 향상에 지나치게 매달리지 말 것.

## 과소적합(underfitting)

- 모델이 지나치게 단순하여 학습 데이터조차 제대로 학습하지 못하는 상태
- 모델의 복잡성 부족이나 학습 데이터 양의 부족으로 인해 이와 같은 문제가 발생

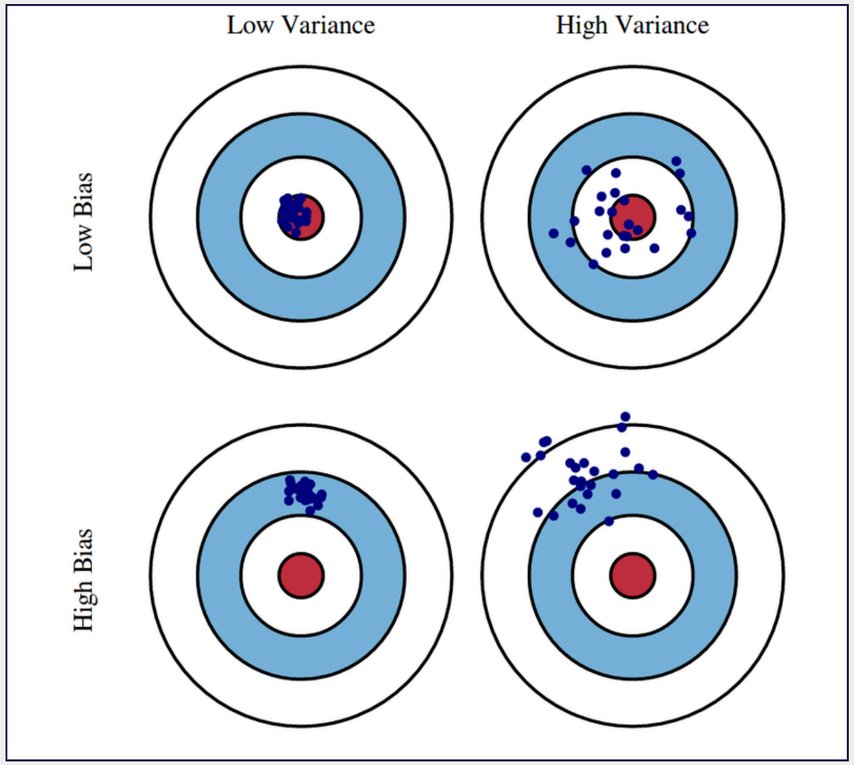
# 과적합이란

## 편향-분산 트레이드 오프 (Bias-Variance Trade off)

- 편향(Bias) : 예측값이 정답과 얼마나 멀리 떨어져 있는가
- 분산(Variance) : 예측값끼리의 차이

- Bias와 Variance는 서로 trade-off 관계
- 즉 한쪽이 올라가면, 다른 한쪽은 내려가는 시소와 같은 관계로 정의할 수 있음

- Low Bias / Low Variance : 이상적인 모델
- Low Bias / High Variance : Overfitting
- High Bias / Low Variance : Underfitting

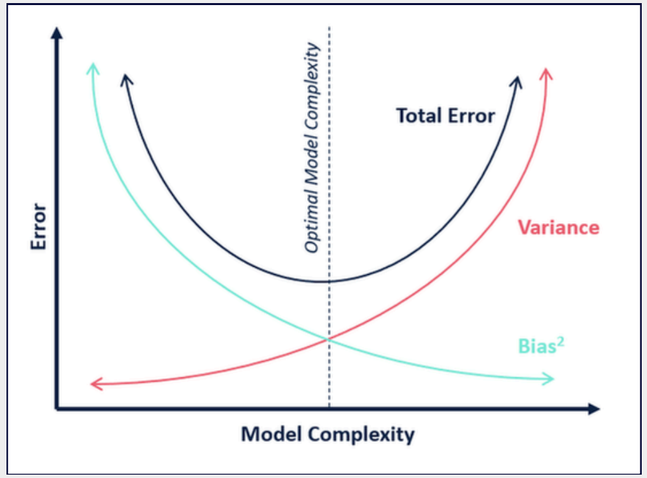


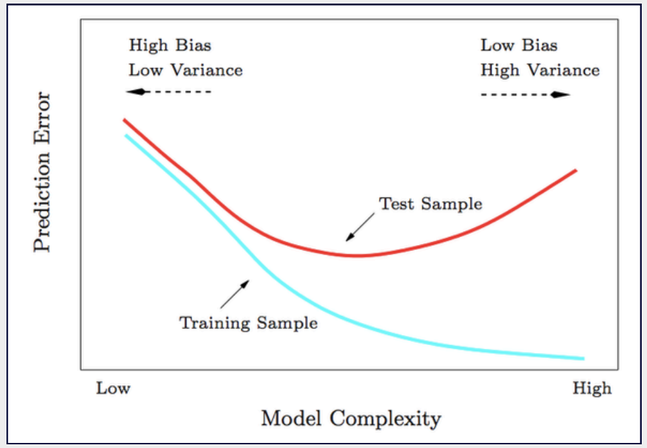
# 과적합이란

## 편향-분산 트레이드 오프 (Bias-Variance Trade off)

- Model Complexity가 증가한다는 것은 곧 Variance도 높아진다는 의미

| 모델 복잡성 | 구분 | 과적합 |
|---|---|---|
| 모델 학습↑, 모델 복잡 | Low Bias / High Variance | Overfitting (예측값 y가 크게 흔들림) |
| 모델 학습↓, 모델 단순 | High Bias / Low Variance | Underfitting (정확한 예측 X) |



# 과적합 해결방법

## Underfitting 해결방법

- 모델 복잡성 높이기 : Parameter가 더 많은 복잡한 모델을 선택함
- 모델 제약 줄이기 : 규제 Hyperparameter의 값들을 줄임

## Overfitting 해결방법

- 데이터 정규화 : 각 Feature의 단위에 상관없이 값으로 단순 비교할 수 있도록 데이터를 Scaling 하는 기법
- Dropout : 학습 과정에서 일부 Feature를 제거(Drop)하는 기법

- 앙상블 : 여러가지 단일 모델을 조합하여 일반화된 모델을 만드는 기법
- 교차검증 : 학습용 데이터를 계속 변경하여 모델을 훈련시키는 기법

# 교차검증

## 교차검증을 왜 사용해야 할까?

노이즈가 크고 불균형하게 분포되어 있는 데이터는 훈련 및 검증이 제대로 되지 않는 문제가 있음

- 데이터가 불균형하게 분포되어 있는 경우, 각 폴드에서의 클래스 분포를 고려하여 편향을 방지할 수 있음

고정된 train set과 test set을 사용하는 경우, overfitting이 발생할 수 있음

- 고정된 test set에 최적화된 과적합 모델이 탄생하는 것을 방지하기 위해, 데이터의 모든 부분을 test set으로 사용함

## 교차검증의 핵심 아이디어

**모든 데이터셋을 훈련과 평가에 이용하자!**

- 즉 train set과 test set을 변경해보자는 아이디어
- 과대적합과 과소적합에 강건하고, 모델의 정확도를 향상시킬 수 있음

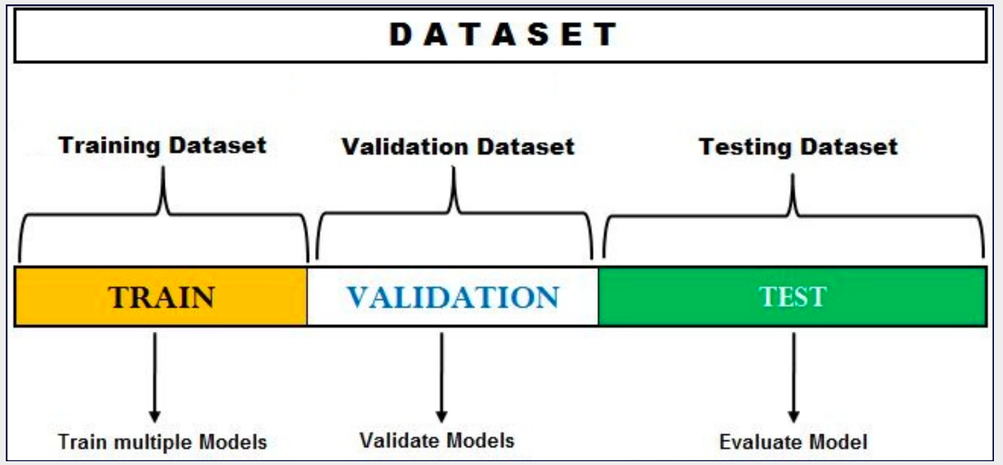
교차검증

# Hold-Out Cross Validation

## Hold-Out Cross Validation이란?

- 데이터셋을 세 가지로(train set, validation set, test set) 나누는 교차검증 방법
- train set으로 모델을 훈련시키고, validation set으로 성능을 평가한 후, test set으로 모델의 일반화 성능을 추정함

- 장점 : test set을 넣기 전, validation set으로 성능을 평가하기 때문에 모데르이 예측 성능을 측정할 수 있음
- 한계 : validation set으로 사용할 데이터는 훈련에 쓰이지 않기 때문에 데이터 자원을 낭비함



교차검증

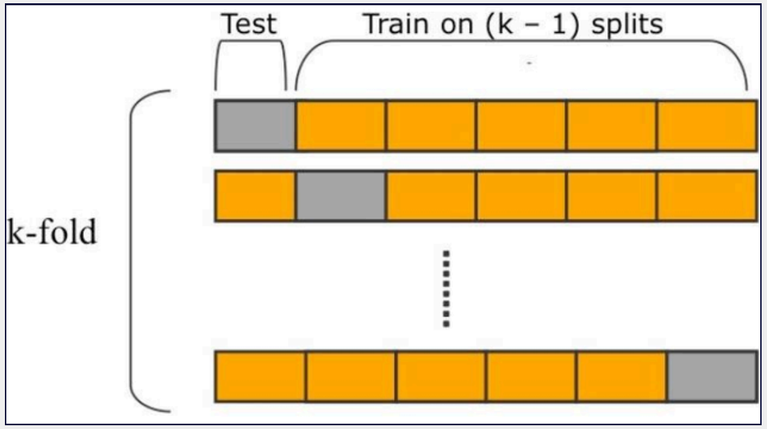
# K-fold Cross Validation

## K-fold Cross Validation이란?

- 데이터셋을 k개로 나누어, 하나의 fold를 test set으로 사용하고, 나머지를 train set으로 사용하는 교차검증 방법
- 장점 : 모든 데이터셋을 train set으로도, test set으로도 활용 가능함
- 단점 : 여전히 데이터가 편향되어 있을 경우, 편향된 데이터가 분할되지 못하고 몰릴 수 있음

## K-fold Cross Validation의 단계

1. 데이터 집합을 k개의 데이터 fold로 나눔
2. (k-1)개의 fold는 train fold로, 나머지 1개는 test fold로 지정함
3. train fold를 이용하여 모델을 훈련시키고, test fold를 이용하여 정확도를 측정함
4. 2~3번 과정을 k번 반복함 (이 때, 한 번 선정했던 test fold는 다시 test fold로 선택X)
5. 총 k개의 성능 결과가 도출되면, 이 k개의 평균을 학습 모델 성능으로 사용함

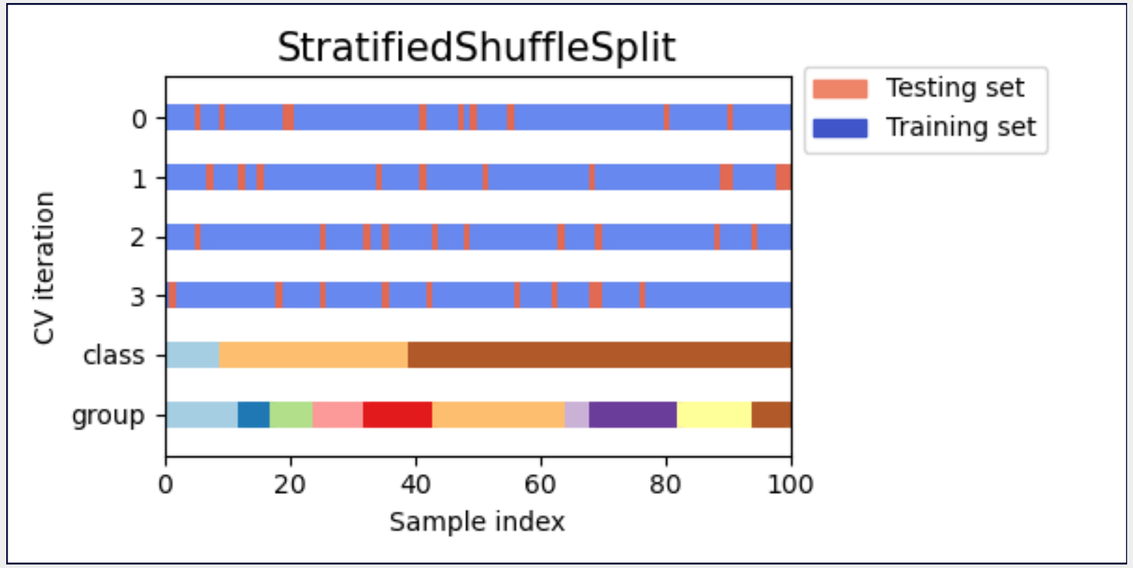


교차검증

# Stratified K-fold Cross Validation

## Stratified K-fold Cross Validation이란?

- k-fold의 경우, 데이터를 일정한 간격으로 잘라 사용하기 때문에 target 값이 편향되면 학습에 어려움이 생김
- 위와 같은 k-fold의 단점을 보완하기 위해, target 속성값의 개수를 동일하게 하여 데이터가 한 곳으로 몰리는 것을 방지함
    - 다만, 회귀의 경우 target값이 연속적인 값이기 때문에 회귀에서는 지원되지 않음

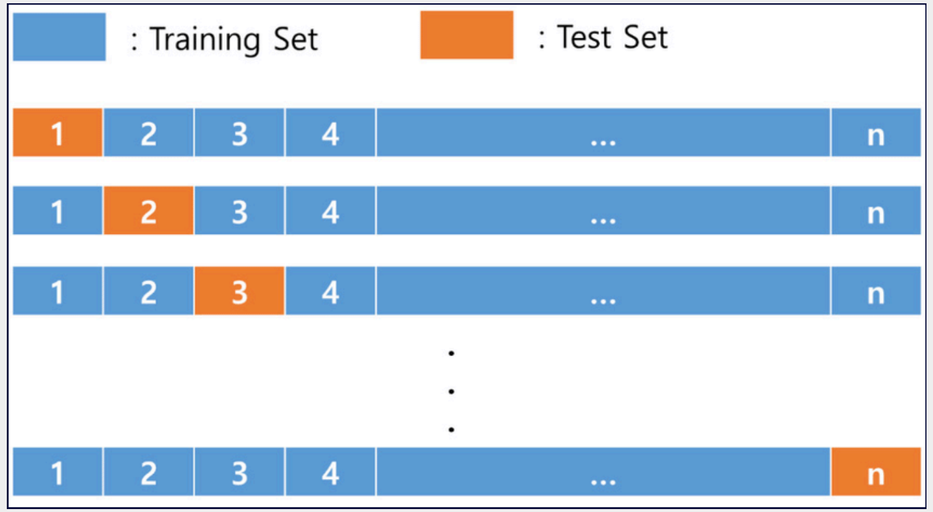


교차검증

# Leave-One-Out Cross Validation

## Leave-One-Out Cross Validation이란?

- n개의 데이터 샘플에서 한 개의 데이터 샘플을 test set으로 하고, 그 1개를 뺀 나머지를 train set으로 두고 모델을 검증함
- 장점 : 훈련에 거의 데이터셋의 전부를 사용하기 때문에 모델 성능에 대한 신뢰를 할 수 있고 편향되지 않은 추정치를 제공함
- 단점 : 계산 비용이 높음



평가지표

# 회귀 평가지표

회귀 모델의 target값은 연속적인 값 → 실제 값과 예측 값의 차이가 작을 수록 해당 모델의 성능이 좋다는 것을 의미함

## MSE(Mean Squared Error) : 평균 제곱 오차

실제값과 예측값의 차이를 제곱하여 평균을 내는 방법

- 장점 : 잔차의 값이 음수가 될 수 있는 경우를 방지하고, 오차의 민감도를 높임
- 단점 : 예측 변수와 단위가 다르면, 잔차를 제곱하기 때문에 이상치에 민감함

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

## RMSE(Root Mean Squared Error) : 평균 제곱근 오차

MSE에 루트를 씌워 직관적으로 오차를 구하는 방법

- 장점 : 예측 변수와 단위가 같으며, 잔차를 제곱해서 생기는 값의 왜곡이 덜함
- 단점 : 실제 값에 대해 underestimates인지 overestimates인지 파악하기 어려움

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{n}}$$

평가지표

# 회귀 평가지표

## MAE(Mean Absolute Error) : 평균 절대 오차

실제값과 예측값의 차이를 절대값으로 변환하여 평균을 내는 방법

- 장점 : 잔차의 값이 음수가 될 수 있는 경우를 방지하고, 예측 변수와 단위가 같음
- 단점 : 잔차에 절댓값을 씌우기 때문에, 실제 값에 대해 underestimates인지 overestimates인지 파악하기 어려움

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|$$

## MAPE(Mean Absolute Percentage Error) : 평균 절대 비율 오차

MAE를 비율(%)로 표현한 방법

- 장점 : 직관적이고, 비율 변수이기 때문에 다른 평가지표에 비해 비교에 용이함
- 단점 : 실제 값에 대해 underestimates인지 overestimates인지 파악하기 어려움

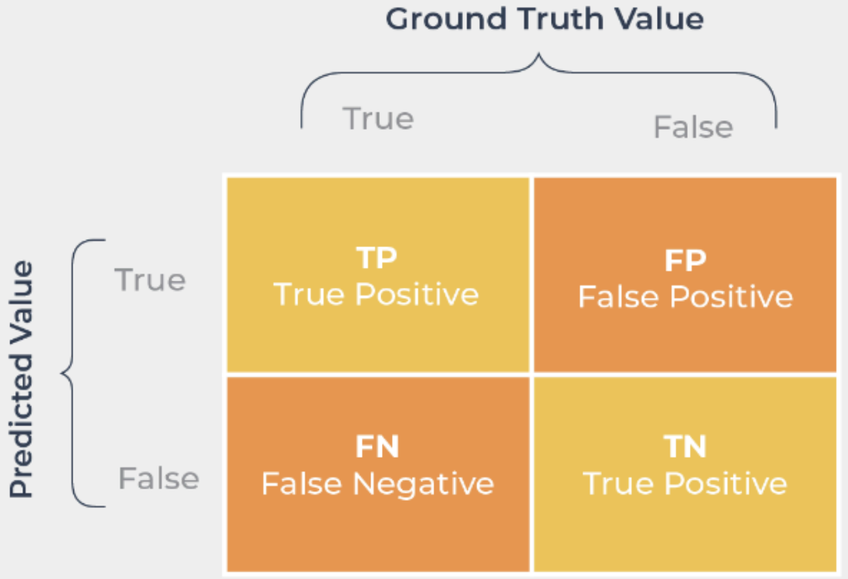
$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{f}(x_i)}{y_i}\right|$$

# 분류 평가지표

분류 모델의 target값은 카테고리 → 실제 값과 예측 값이 일치하는 수가 많을수록 모델의 성능이 좋다는 것을 의미함

## Confusion Matrix(오차행렬)

- True Positive (TP) : 실제 True인 정답을 True라고 예측 (정답)
- False Positive (FP) : 실제 False인 정답을 True라고 예측 (오답)
- False Negative (FN) : 실제 True인 정답을 False라고 예측 (오답)
- True Negative (TN) : 실제 False인 정답을 False라고 예측 (정답)



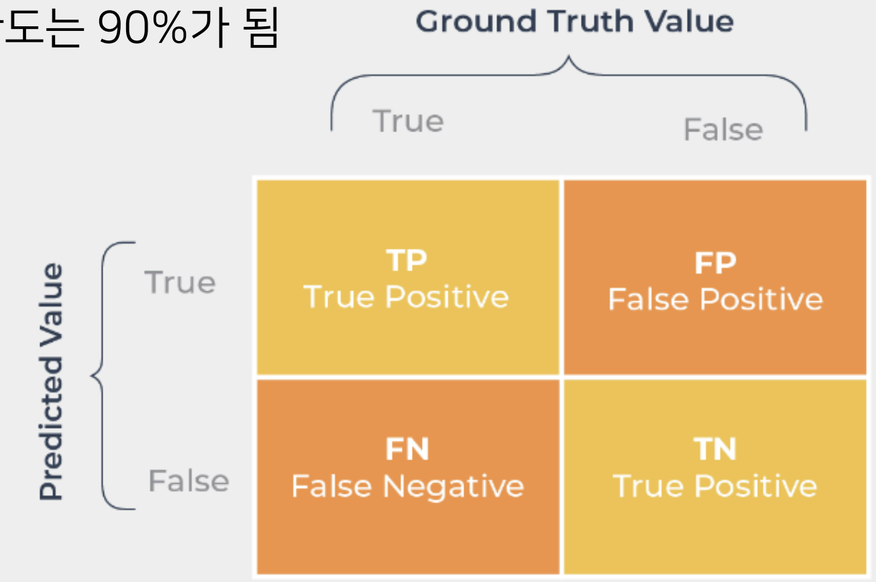
# 분류 평가지표

## 정확도(Accuaracy)

전체 데이터 중, 정확히게 예측한 데이터의 비율

- 불균형 데이터의 경우 정확한 평가지표가 될 수 없음
- ex) 양성과 음성의 비율이 1:9인 경우, 모두 음성이라고 모델이 예측한다면 정확도는 90%가 됨

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




## 특이도(Specificity)

Negative로 예측한 것 중, 진짜 Negative의 비율

- Negative에 집중한 평가지표

$$\text{Specificity} = \frac{TN}{FP + TN}$$

평가지표

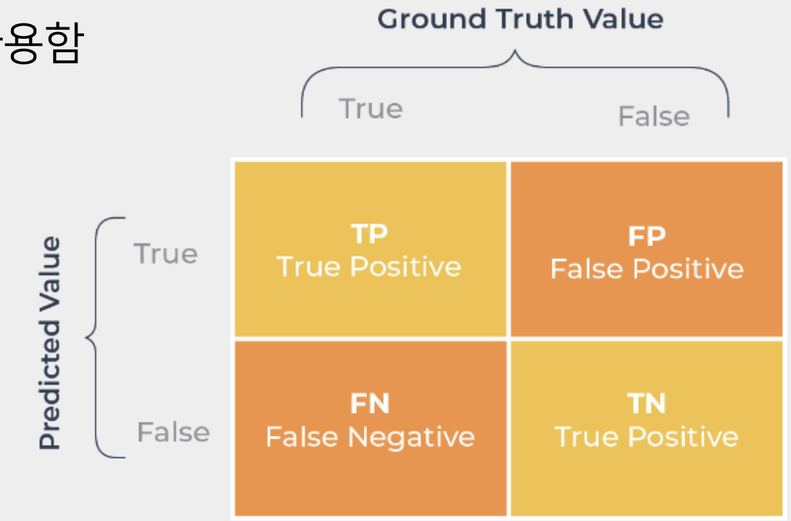
# 분류 평가지표

## 정밀도(Precision)

Positive로 예측한 것 중, 진짜 Positive의 비율

- Positive에 집중한 평가지표
- 실제 Negative 데이터를 Positive로 잘못 판단하면 업무상 큰 영향이 있는 경우에 사용함
- ex) 스팸 메일 판정(스팸 메일로 예측한 것 중 스팸 메일의 비율)

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




## 재현율(Recall) = 민감도(Sensitivity)

진짜 Positive인 것들중, 올바르게 Positive로 예측한 비율

- Positive에 집중한 평가지표
- 실제 Positive 데이터를 Negative로 잘못 판단하면 업무상 큰 영향이 있는 경우에 사용함
- ex) 암환자 판정(실제 암환자 중에 양성이라고 예측한 비율)

$$\text{Recall}(=\text{Sensitivity}) = \frac{TP}{TP + FN}$$

# 분류 평가지표

## F1-score

Precision과 Recall을 이용하여 조화평균을 구한 지표

- 정밀도와 재현율이 어느 한쪽으로 치우치지 않는 수치를 나타낼 때 높은 값을 가짐

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

## 임계값(P/N 클래스 구분 기준) 변경에 따른 정밀도와 재현율의 변화 관계

임계값을 높일 경우 : 양성으로 예측하는 기준이 엄격해짐 ( = 음성으로 예측되는 샘플이 많아짐)

- 정밀도 → 높아짐 / 재현율 → 낮아짐

임계값을 낮출 경우 : 양성으로 예측하는 기준이 낮아짐 ( = 양성으로 예측되는 샘플이 많아짐)

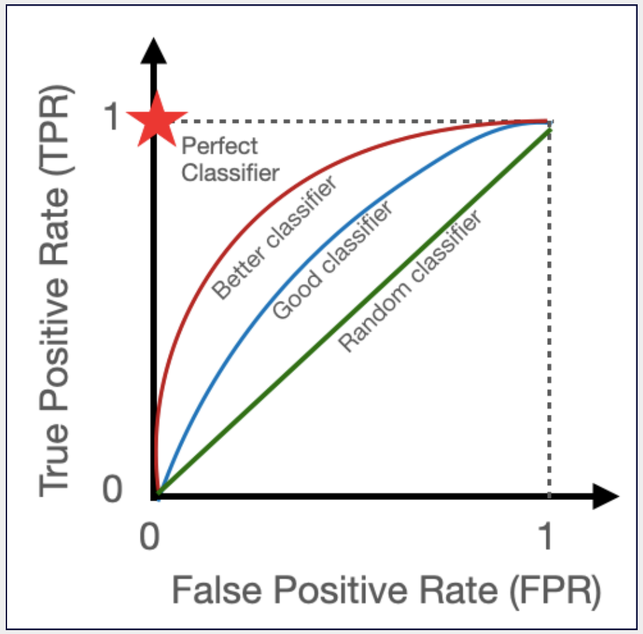
- 정밀도 → 낮아짐 / 재현율 → 높아짐

→ 즉 임계값을 변화시켰을 때 재현율과 정밀도는 음의 상관관계를 가짐

# 분류 평가지표

## ROC/AUC

FPR(False Positive Rate) : 1-특이도(Specificity) → 실제 음성 중 양성으로 잘 못 예측한 비율
TPR(True Positive Rate) : 재현율(Recall) → 실제 양성 중 양성으로 맞게 예측한 비율




### ROC (Receiver Operating Characteristic)

- 모든 임계값에서 분류 모델의 성능을 보여주는 그래프

### AUC (Area Under the Curve)

- ROC 곡선 아래의 영역
- AUC가 높다는 것은 클래스를 구별하는 모델의 성능이 좋다는 것을 의미함
- AUC는 0~1 사이 값을 가짐

# 과제

## 과제 안내

D&A_2024_ML_2주차_과제.ipynb에 있는 문제를 풀고
9월 30일 (월) 23:59까지 홈페이지에 제출해주세요.

# THANK YOU

## ML Session 2차시

빅데이터 분석 학회 D&A