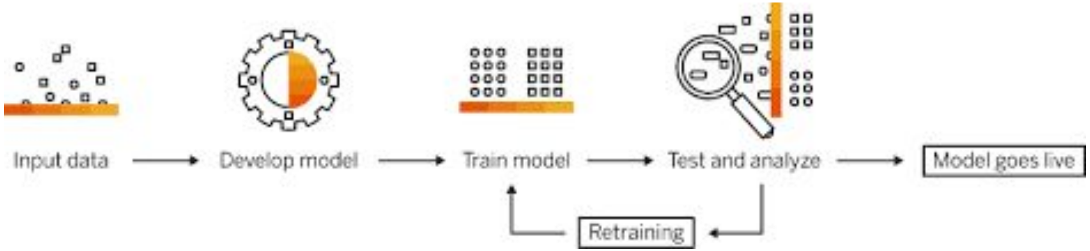
# 면접 준비

데이터 사이언티스트

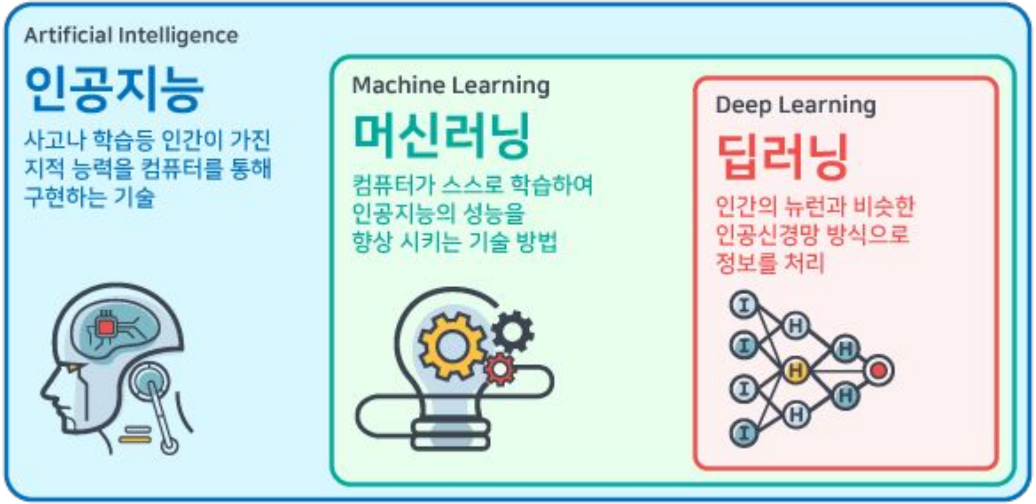
# 출처

- https://www.simplilearn.com/tutorials/data-science-tutorial/data-science-interview-questions?source=sl_frs_nav_playlist_video_clicked#basic_data_science_interview_questions
- https://www.ubuntupit.com/frequently-asked-machine-learning-interview-questions-and-answers/
-

# 0. Machine Learning

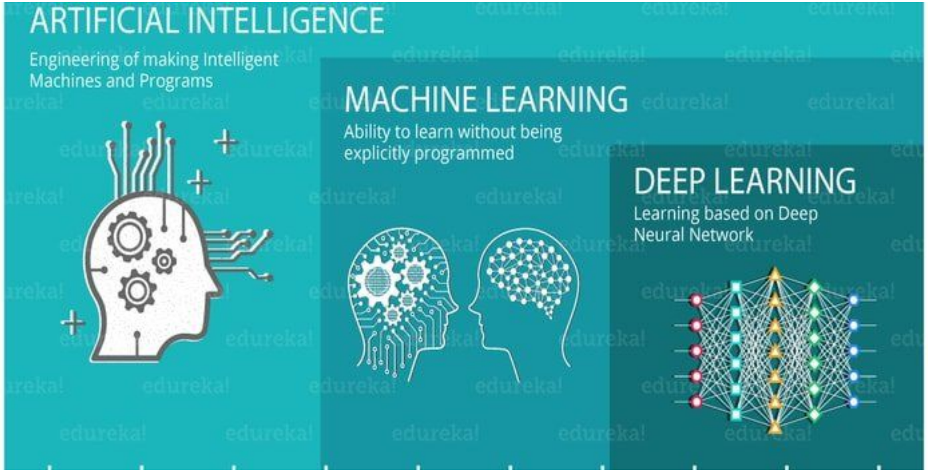



- 명시적으로 컴퓨터를 프로그래밍하는 대신, 데이터로 학습하고 개선하도록 훈련에 중점
- 대규모 데이터 세트에서 패턴과 상관관계를 찾고 분석을 토대로 최적의 의사결정과 예측을 수행하도록 훈련



# 0-1. AI vs Machine Learning vs Deep Learning

- Artificial intelligence : **인간의 뇌를 모방하는 컴퓨터로 구현**하는 것을 목표
- 딥 러닝 : 머신 러닝의 하위 개념, 머신 러닝에 해당하며 비슷한 방식으로 작동
- 차이점 : 머신 러닝 모델은 점진적으로 향상 중 약간의 안내가 필요 그러나 딥 러닝 모델은 **신경망**을 통해 예측의 정확성 여부를 스스로 판단

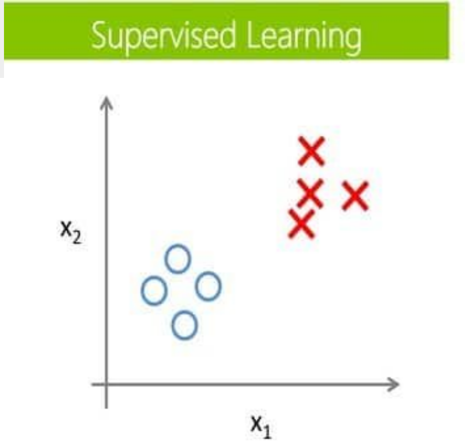
# 0-2. Machine Learning vs Data Mining

- 둘 모두 유사점이 굉장히 많다.
- 데이터 마이닝 : **데이터에서의 패턴을 추출하는 것**을 목적
- 머신러닝 : 데이터를 이용하여 **자동적으로 학습하는 기계를 만드는 것**을 목적

# 1. Supervised vs Unsupervised

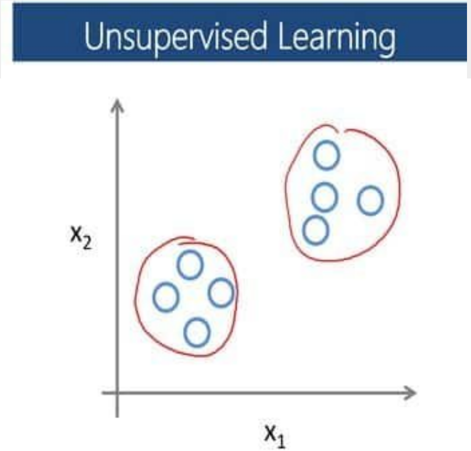
**Supervised Learning**

- **라벨링 데이터 사용**
- decision tree
- logistic regression
- support vector machine



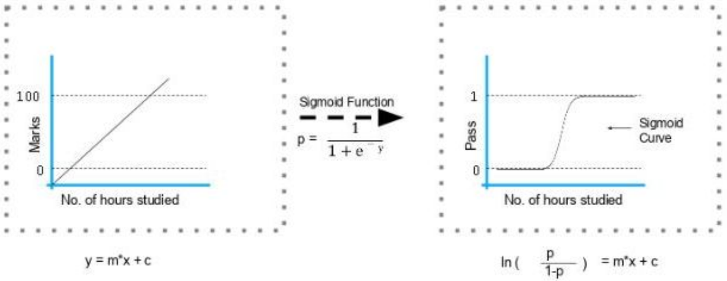

**Unsupervised Learning**

- **라벨링 데이터 사용 X**
- k-means clustering
- hierarchical clustering
- apriori algorithm



# 2. Logistic regression

- 독립 변수의 선형 조합으로 종속 변수를 예측
  - 종속 변수 : dependent variable, label
  - 독립 변수 : independent variable, feature
- Logistic function ~ Sigmoid function
- Classification problem or 확률 예측
- Linear Regression의 목표는 범위가 정해지지 않은 종속 변수와 독립 변수 사이의 선형 관계를 측정하는 것이지만 Logistic Regression은 Linear Regression을 이용하여 확률을 예측



# 2-1. Linear regression vs Logistic regression

| **Linear Regression** | **Logistic Regression** |
| --- | --- |
| - Regression problems<br>- Continuous 데이터를 출력<br>- 종속 변수를 추정<br>- 직선 형태 | - Classification problems 또는 확률값 예측 문제<br>- Categorical 데이터를 출력<br>- 종속 변수의 가능성을 계산<br>- Sigmoid curve |

Q. 나이, 성별, 혈중 콜레스테롤 수치라는 3가지 위험 요인을 바탕으로 심장병으로 인한 사망 확률을 예측하고자 할때 적합한 모델은 무엇인가?

A. Logistic Regression

# 2-2. Sigmoid Function

- 시그모이드 함수는 S자형 곡선 또는 시그모이드 곡선을 갖는 수학 함수이다.

  - 로지스틱 함수

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

  - 쌍곡탄젠트 (위의 로지스틱 함수를 평행이동하고 상수를 곱한 것과 같음)

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

  - 아크탄젠트 함수

    $$f(x) = \arctan x$$

  - 오차 함수

    $$f(x) = \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$$

  - 일부 대수함수, 예를 들어:

    $$f(x) = \frac{x}{\sqrt{1+x^2}}$$

https://ko.wikipedia.org/wiki/%EC%8B%9C%EA%B7%B8%EB%AA%A8%EC%9D%B4%EB%93%9C_%ED%95%A8%EC%88%98

# 2-3. Logistic Function의 미분

$$
\begin{aligned}
\frac{d}{dx} sigmoid(x) &= \frac{d}{dx}(1+e^{-x})^{-1} \\
&= (-1)\frac{1}{(1+e^{-x})^2}\frac{d}{dx}(1+e^{-x}) \\
&= (-1)\frac{1}{(1+e^{-x})^2}(0+e^{-x})\frac{d}{dx}(-x) \\
&= (-1)\frac{1}{(1+e^{-x})^2}e^{-x}(-1) \\
&= \frac{e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1+e^{-x}-1}{(1+e^{-x})^2} \\
&= \frac{(1+e^{-x})}{(1+e^{-x})^2} - \frac{1}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} - \frac{1}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}}(1-\frac{1}{1+e^{-x}}) \\
&= sigmoid(x)(1-sigmoid(x))
\end{aligned}
$$

$$\boxed{\frac{d}{dx} sigmoid(x) = sigmoid(x)(1-sigmoid(x))}$$

http://taewan.kim/post/sigmoid_diff/

# 3. Decision Tree

- Algorithm
  1. 분기 전 데이터를 입력으로 사용
  2. 분기 전 데이터의 Entropy 계산, 분기 특징 후보들에 대한 Entropy 계산
  3. 분기 특징 후보들의 Information Gain 계산
  4. 가장 높은 Information Gain 값을 가지는 분기 특징을 선택
  5. 사전에 정의한 멈추는 조건이 될때까지 위 과정을 반복
- Entropy : 데이터가 얼마나 균일하게 분류되었는지 알려주는 척도, 즉 작을수록 잘 분류된 상태
- Information Gain : 분기 이전의 Entropy에서 분기 이후의 Entropy를 뺀 수치, 즉 높을수록 잘 분기했다고 판단
- 단점 : Overfitting 문제 >> **pre-pruning, post-pruning (가지치기)**, Random Forest

# 3-1. Decision Tree 장단점

| 장점 | | 단점 | |
|---|---|---|---|
| 결과 해석 용이 | - 직관적인 해석 가능<br>- 주요 변수와 분리기준 제시 | 비안정성 | - 데이터 수가 적을 경우 특히 불안정<br>- 과대적합 발생률 높음(가지치기 필요) |
| 비모수적 모델 | - 통계모델에 요구되는 가정에 자유로움 (e.g., 정규성 독립성, 등분산성) | 선형성 미흡 | - 전체적인 선형관계 파악 미흡 |
| 변수 간 상호작용 | - 변수 간의 상호작용을 고려하며 선형/비선형 관계 탐색 가능 | 비연속성 | - 분리 시 연속형 변수를 구간화 처리(비연속화)<br>- 분리 경계점 근처에 오류 발생 가능 |

https://heytech.tistory.com/145

# 3-2. Entropy and Information Gain (ID3)

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

| Play Golf | |
|---|---|
| Yes | No |
| 9 | 5 |

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

| | | Play Golf | | |
|---|---|---|---|---|
| | | Yes | No | |
| Outlook | Sunny | 3 | 2 | 5 |
| | Overcast | 4 | 0 | 4 |
| | Rainy | 2 | 3 | 5 |
| | | | | 14 |

**E**(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)
= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971
= 0.693

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

**G**(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)
= 0.940 – 0.693 = 0.247

# 3-3. Entropy and Information Gain (C4.5)

Information gain ratio

# 3-4. Entropy and Information Gain (CART)

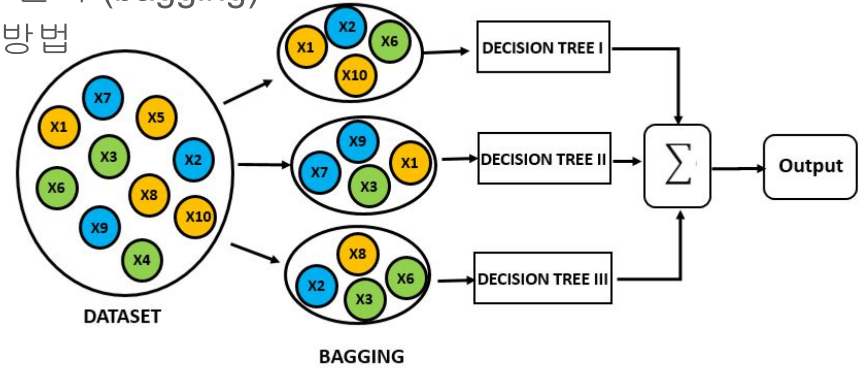
Gini index

# 3-5. Pruning

- 가지치기 역할
- pre-pruning
- post-pruning

# 4. Random Forest

- 앙상블 머신러닝 모델
- Decision Tree로 생성된 Overfitting Tree에서 일반적인 결과 출력
- Algorithm
    1. 학습 데이터에서 n개 데이터 표본 선택 (bootstrap)
    2. k개 feature 중 $\sqrt{k}$ 개를 선택
    3. Decision Tree 생성
    4. 1~3 번의 과정을 m 번 반복
    5. 테스트 데이터에서 m개의 결과 중 다수결 결과 출력 (bagging)
    - Bagging : random forest에서 사용하는 앙상블 방법
    -




https://eunsukimme.github.io/ml/2019/11/26/Random-Forest/

# 앙상블

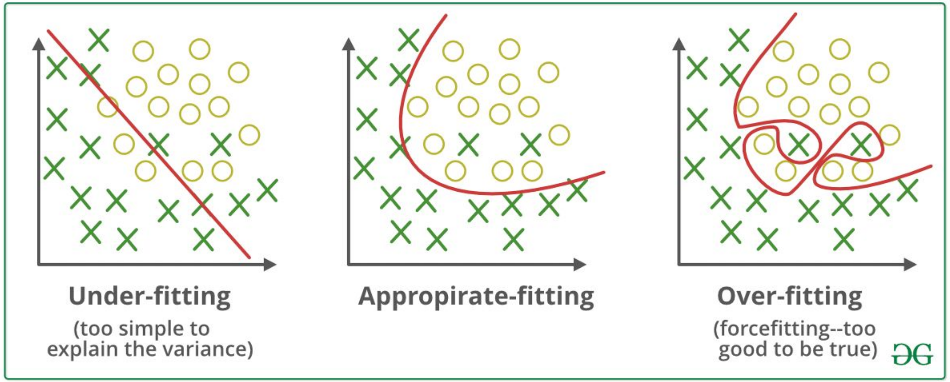
- Voting
  - 다른 알고리즘을 가진 분류기가 같은 데이터셋을 기반으로 학습되고 결합
- Bagging (bootstrap aggregating)
  - 같은 분류기가 같은 데이터셋을 기반으로 학습되고 결합 (부트스트랩)
  - 높은 bias의 underfitting 문제, 높은 variance로 인한 overfitting 문제 해결
  - categorical data는 투표, continuous data는 평균
  - 병렬적 구조
- Boosting
  - 직렬적 구조
- Stacking

https://libertegrace.tistory.com/entry/Classification-3-%EC%95%99%EC%83%81%EB%B8%94-%ED%95%99%EC%8A%B5Ensemble-Learning-Boosting

https://data-matzip.tistory.com/4

# 5. Overfitting

- 훈련 데이터에 과하게 맞추어져 훈련 데이터의 성능은 좋지만, 테스트 데이터에서는 성능이 저조
- Overfitting 방지
    1. 더 많은 데이터 확보
    2. 모델 복잡도 줄이기
    3. cross validation 방법 사용 : k-fold cross validation
    4. 정규화 사용 (LASSO, Ridge)
    5. 앙상블 학습 방법 사용
- Underfitting 방지
    1. 새로운 특성 추가
    2. 모델 복잡도 증가
    3. 정규화 계수 줄이기



# 6. Univariate, Bivariate and Multivariate analysis

- Univariate : 일변량 데이터
  - 1개의 feature
  - 평균, 중위수, 최빈값(mode), 산포도, 범위, 최대, 최소 등의 통계 분석 진행

- Bivariate : 이변량 데이터
  - 2개의 다른 feature
  - 원인과 영향을 두 변수 사이의 관계 비교를 통해 분석

- Multivariate : 다변량 데이터
  - 3개 이상의 feature

# 7. Feature Selection Method

- Filter Method
  - 각 변수들에 대해 통계적인 점수와 순위를 매기고 선택
  - Linear Discrimination Analysis
  - ANOVA
  - Chi-Square
- Wrapper Method
  - 변수의 일부만을 모델링에 사용 후, 평가 작업을 반복하여 변수 선택
  - Forward Selection
  - Backward Selection
  - Recursive Feature Elimination
- Embedded Method
  - 위의 두 방법을 결합하여 어떤 변수가 가장 크게 기여하는 지를 찾아내는 방법
  - LASSO
  - Ridge Regression
  - Elastic Net

# 7-1. Feature Selection vs Feature Extraction

- https://bioinformaticsandme.tistory.com/188

# 8. Python Print

- 3의 배수는 “fizz”
- 5의 배수는 “buzz”
- 3과 5의 배수는 “fizzbuzz”

```python
for fizzbuzz in range(51):
    if fizzbuzz % 3 == 0 and fizzbuzz % 5 == 0:
        print("fizzbuzz")
        continue
    elif fizzbuzz % 3 == 0:
        print("fizz")
        continue
    elif fizzbuzz % 5 == 0:
        print("buzz")
        continue
    print(fizzbuzz)
```

```
fizzbuzz
1
2
fizz
4
buzz
fizz
7
8
fizz
buzz
11
fizz
13
14
fizzbuzz
16
17
fizz
19
buzz
fizz
22
23
fizz
buzz
26
.
.
.

46
47
fizz
49
buzz
```

# 9. Missing Value

- Missing data 삭제 (확보한 데이터가 충분히 클때)
- 특정 값으로 채우기
- 결측값의 앞 또는 뒷 방향의 값으로 채우기
- mean, mode, medium, trimmed mean
- **Knnimputer** 를 사용하여 결측치 채울 수 있음

- Pandas, scipy 등의 라이브러리를 사용하여 쉽게 채울 수 있음 (fillna)

https://rfriend.tistory.com/262

# 10. Euclidean Distance in Python

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

```
import math

plot1 = [1,3]
plot2 = [2,5]

euclidean_distance = math.sqrt((plot1[0]-plot2[0])**2 + (plot1[1]-plot2[1])**2)
```

```
euclidean_distance
```

```
2.23606797749979
```

# 11. Dimensionality Reduction

- 원래의 차원에서 작은 차원으로 변환
- 장점
  - 데이터 압축하여 저장 공간 감소
  - 계산 시간 감소
- 종류
  - pca
  - auto-encoder
  - Linear Discriminant Analysis

# 12. Eigenvalues and Eigenvectors

- Eigenvector : 어떤 벡터에 선형변환 결과가 방향은 변하지 않고 크기만 변환되는 벡터를 의미
- Eigenvalue : Eigenvector가 변환되는 크기를 의미

**DEFINITION 1. 고윳값, 고유벡터**

임의의 $n \times n$ 행렬 $A$ 에 대하여, 0이 아닌 솔루션 벡터 $\vec{x}$ 가 존재한다면 숫자 $\lambda$ 는 행렬 $A$ 의 고윳값라고 할 수 있다.

$$A\vec{x} = \lambda\vec{x} \tag{2}$$

이 때, 솔루션 벡터 $\vec{x}$ 는 고윳값 $\lambda$ 에 대응하는 고유벡터이다.

https://angeloyeo.github.io/2019/07/17/eigen_vector.html

# 12-1. Eigenvalues and Eigenvectors 계산 과정

선형 변환 A에 대해 Eigenvalue와 Eigenvector를 구하면 다음과 같다.

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$det(A - \lambda I) = det\left(\begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix}\right) = 0$$

$$\Rightarrow (2-\lambda)^2 - 1$$

$$= (4 - 4\lambda + \lambda^2) - 1$$

$$= \lambda^2 - 4\lambda + 3 = 0$$

그러므로, $\lambda_1 = 1$, $\lambda_2 = 3$이다.

$\lambda_1 = 1$인 경우에 대해,

$$A\vec{x} = \lambda_1 \vec{x}$$

$$\Rightarrow \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$2x_1 + x_2 = x_1$$

$$x_1 + 2x_2 = x_2$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$\lambda_2 = 3$인 경우의 고유벡터는

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

https://angeloyeo.github.io/2019/07/17/eigen_vector.html

# 12-2. PCA의 Eigenvalues and Eigenvectors

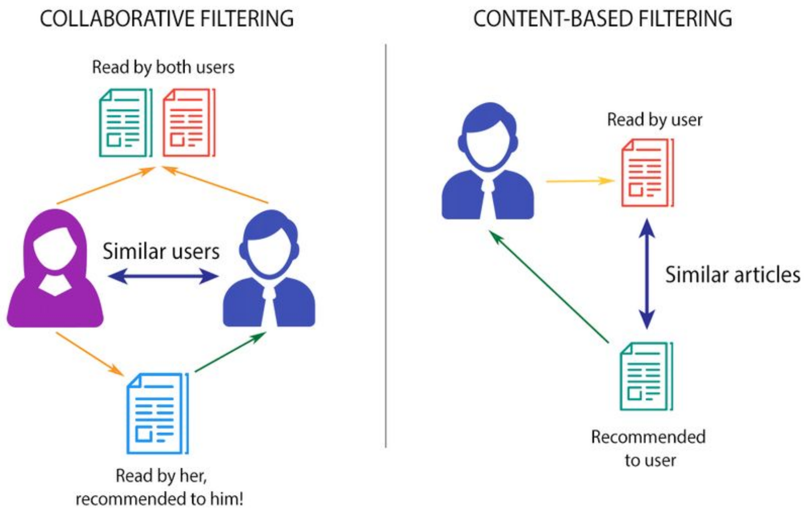
- pca

# 13. Model maintain

1. 모니터 : 제대로 작동하는지에 대한 지속적인 모니터링이 필요
2. 평가 : 새로운 알고리즘이 필요한지에 대한 여부를 판단
3. 비교 : 기존 모델과 새 모델을 비교
4. 재작성 : 성능이 더 좋은 모델로 변경

# 14. Recommender System

- 사용자가 자신의 선호도에 따라 특정 제품을 어떻게 생각할지 예측
- Collaborative Filtering
  - 다른 사용자와의 유사함에 기초
  - 비슷한 사용자가  좋아하는 아이템을 추천
  - ex) 아마존 추천 시스템..
- Content-based Filtering
  - 다른 아이템과의 유사함에 기초
  - 유사한 아이템을 추천
- 그 외
  - Hybrid Recommender System
  - Context-based Recommender System
  - …



# 15. RMSE and MSE

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# answer : y = 1 * x_0 + 2 * x_1 + 3
X = np.array([[1, 1], [1, 2], [2, 2], [2, 3]])
noise = np.array([0.00201, 0.00032, -0.0001, -0.071902])
y = np.dot(X, np.array([1, 4])) + 3 + noise

reg = LinearRegression().fit(X, y)
reg.score(X, y)

print('coefficients : ',reg.coef_)

print('intercept : ', reg.intercept_)
```

```
coefficients :  [0.99958  3.963254]
intercept :  3.056704
```

```
y_hat = reg.predict(np.array([[3, 5], [4, 5], [6, 7]]))
y_true = np.dot(np.array([[3, 5], [4, 5], [6, 7]]), np.array([1, 4])) + 3
```

```
print('y_hat : ', y_hat)
print('y_true : ', y_true)
```

```
y_hat :  [25.871714 26.871294 36.796962]
y_true :  [26 27 37]
```

```
print('rmse : ', mean_squared_error(y_true, y_hat, squared=False))
print('mse  : ', mean_squared_error(y_true, y_hat))
```

```
rmse :  0.15731810838834126
mse  :  0.024748987225335153
```

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

# 15-1. Regression Metrics

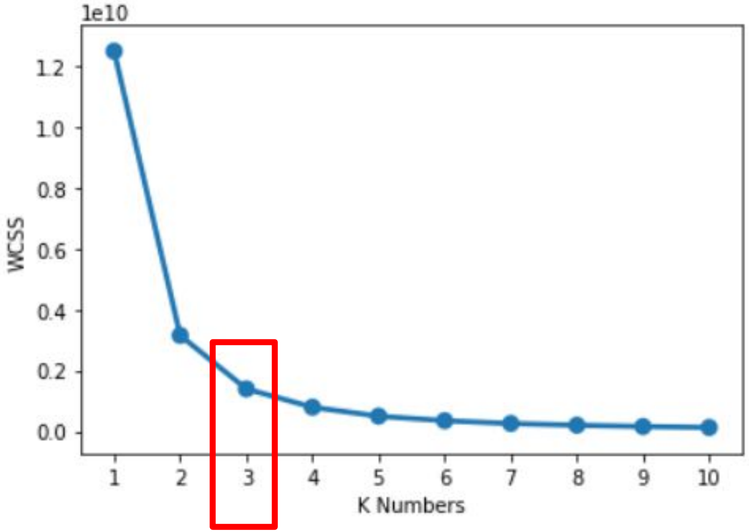
- MSE
- RMSE
- MAE
- R-Squared
- 등등.. 장단점

# 16. Select k for k-means?

- Elbow Method
  - 군집분석에서 군집수를 결정하는 방법
  - 군집내 총 제곱합 (WSS : Within cluster Sum of Squares) 을 계산하여 적절한 군집 수 설정
- WWS (Within Cluster Sum of Squares)

- **Within Cluster Sums of Squares :** $$\mathrm{WSS} = \sum_{i=1}^{N_C} \sum_{x \in C_i} d(\mathbf{x}, \bar{\mathbf{x}}_{C_i})^2$$
- **Between Cluster Sums of Squares:** $$\mathrm{BSS} = \sum_{i=1}^{N_C} |C_i| \cdot d(\bar{\mathbf{x}}_{C_i}, \bar{\mathbf{x}})^2$$

$C_i$ = Cluster, $N_c$ = # clusters, $\overline{x}_{c_i}$ = Cluster centroid, $\overline{x}$ = Sample Mean



# 16-1. WSS, BSS, TSS

- WSS
- BSS
- TSS

# 17. P-value

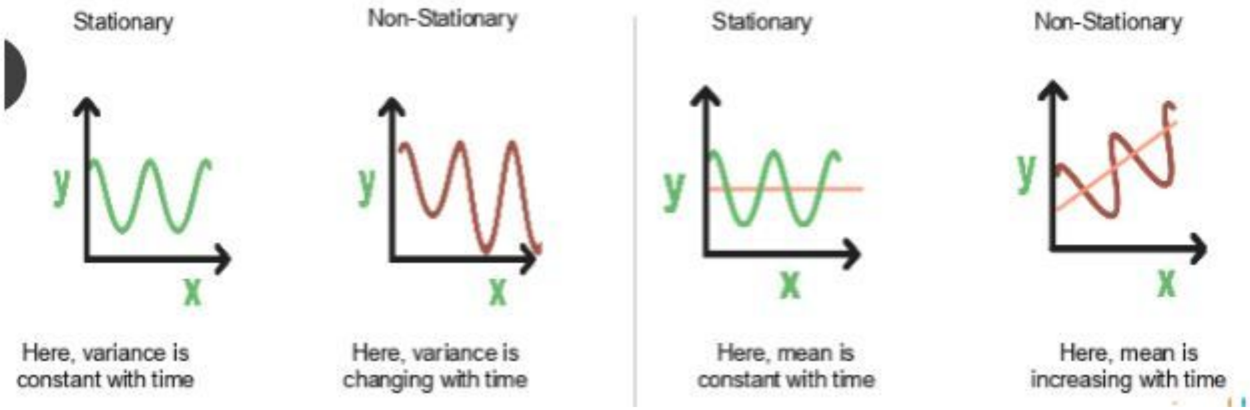
- 검정 통계량에 관한 확률로 크거나 같은 값을 얻을 수 있을 확률
- 귀무가설의 기각 여부를 결정
  - P-value < alpha : 귀무가설을 기각
  - P-value > alpha : 귀무가설을 수락
- 귀무가설 : 새로울게 없다는 가설, 똑같다는 가설
  - ex) 두 확률분포는 차이가 없다.
  - ex) 흡연 여부는 뇌혈관 질환의 발생에 영향을 미치지 않는다.

# 18. Outlier values treat

- 필요없는 데이터라면 삭제
- 다른 모델을 선택 (linear -> nonlinear)
- 데이터를 정규화
- 특이치에 강한 모델을 사용 (random forest)

# 19. Time series stationarity

- Stationarity : 시간이 변해도 일정한 분포를 따르는 경우
- 확인 방법
    - 그래프를 그려서 확인
    - 통계량의 변화를 확인
    - 통계적 검정



# 19-1. Time series stationarity

- 통계적 검정 방법

# 20. Confusion matrix

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

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

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

$$\text{F1-Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

| | | Actual | |
|---|---|---|---|
| | | Positive | Negative |
| Predicted | Positive | True Positive | False Positive |
| | Negative | False Negative | True Negative |

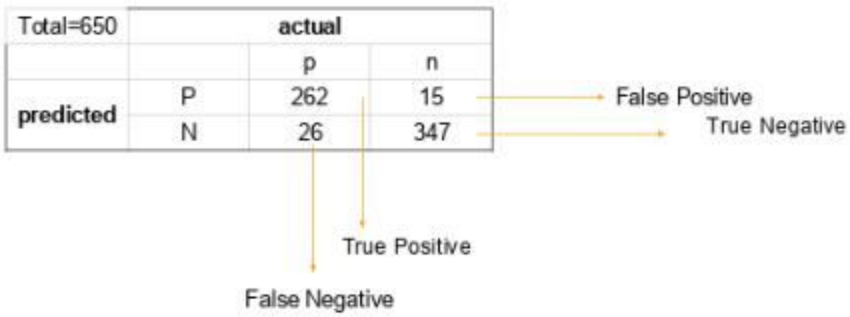
# 21. Precision and Recall Calculate

- precision = tp / (tp + fp)

  262 / 277 = 0.94

- recall = tp / (tp + fn)

  262 / 288 = 0.9



| Total=650 | actual | | |
|---|---|---|---|
| | | p | n |
| predicted | P | 262 | 15 |
| | N | 26 | 347 |

# 22. Basic SQL Query

- Order Table
  - OrderId
  - CustomerId
  - OrderNumber
  - TotalAmount
- Customer Table
  - Id
  - FirstName
  - LastName
  - City
  - Country

SQL query (모든 주문 리스트를 고객 정보와 같이 나열)

SELECT OrderNumber, TotalAmount, FirstName, LastName, City, Country

FROM Order

JOIN Customer

ON Order.CustomerId = Customer.Id

# 23. Data Imbalance Performance Matrix

- 라벨의 분포가 불균형한 경우
- Accuracy로 본다면 좋은 성능을 나타내지만 실제로 보면 좋지 못한 모델일 수 있음
  - 학습 데이터 : 99% 정상 데이터, 1% 이상 데이터
  - 모두 정상 데이터로 예측 시 Accuracy는 99% 이상일 수 있음
  - Positive를 이상 데이터로 할때
  - Precision은 낮게 나오고 Recall이 높게 나옴
  - fp(정상을 이상치로 예측) 가 높고
  - fn(이상치를 정상으로 예측) 가 낮음
- 위 경우 F1-score 를 이용

# 24. K-means clustering

- Algorithm
- 장점
- 단점
- 실제 사용 예시

# 25. Linear Regression

- Algorithm
- 장점
- 단점
    - **회귀 모형의 4가지 가정 (선형성, 독립성, 등분산성, 정규성)**
    - Categorical, binary 문제 X
- 실제 사용 예시

# 26. KNN

- Algorithm
- 장점
- 단점
- 실제 사용 예시

# 27. Association Rule

- Algorithm
- 장점
- 단점
- 실제 사용 예시

# 28. ANOVA

- 3개 이상 다수의 집단을 비교할 때 사용하는 가설검정 방법
- F 분포를 이용
- t-value ~ f-value 같은 의미를 지님

표본 평균 간 퍼진 정도

여러 표본 집단의
차이에 관한
통계적 지표

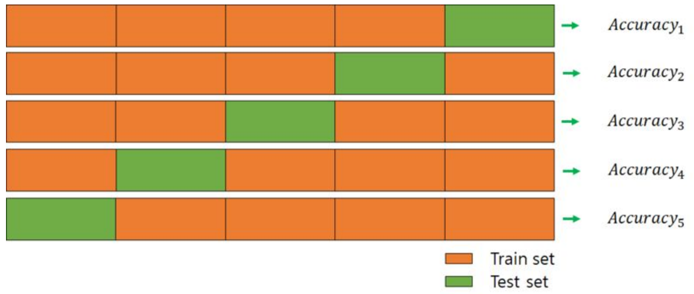
$$F = \frac{\boxed{s_{bet}^{2}}}{\boxed{s_{wit}^{2}}}$$

표본 내에서 퍼진 정도

# 28-1. ANOVA 예제

-

# 29. 모델 평가 방법

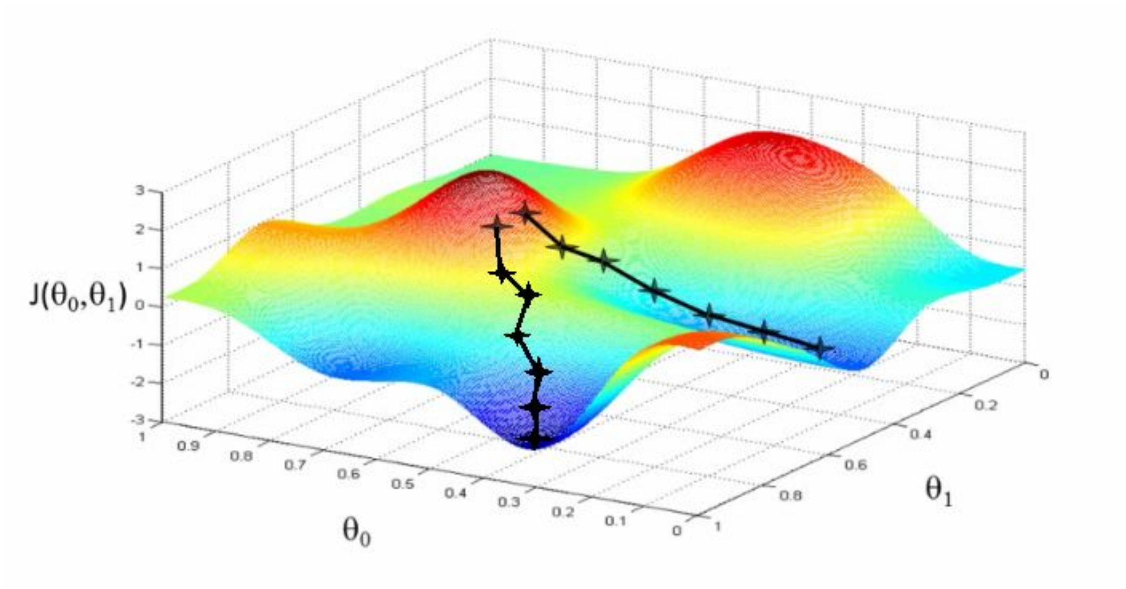



$$Accuracy = Average(Accuracy_1, \cdots, Accuracy_k)$$

- Holdout Validation
  - 초기 데이터를 일정 비율로 훈련, 검증 데이터로 구분
  - 초기 데이터를 어떻게 분리하는냐에 따라 모델 성능에 영향을 크게 끼침
- Cross Validation
  - **K-fold cross validation**
    - k개의 묶음으로 분리하고 k번 평가 후 평균값을 최종 평가 지표로 사용
  - LOOCV (Leave One Out Cross Validation)
    - 샘플 수 n개에 대해 모두 검정, 모든 샘플의 검증값을 평균하여 평가 지표로 사용
    - LpOCV의 한종류
    - 두 경우 모두 시간이 오래 걸림 (LOOCV < LpOCV)
- Bootstrapping
  - N개의 샘플에서 n번 복원 추출하여 n개의 훈련 데이터를 얻는다.
  - 원래 n개 샘플 데이터에서 추출되지 않은 데이터를 검정 데이터로 설정하고 평가한다.
  - 이 때의 검정 데이터를 **OOB (Out of Bag)**이라고 한다.

# 30. Gradient Descent method

- 최적화 알고리즘
- 매개변수를 업데이트하는데 사용
- 손실함수의 최적해를 찾기 위해 사용



# 31. Resampling

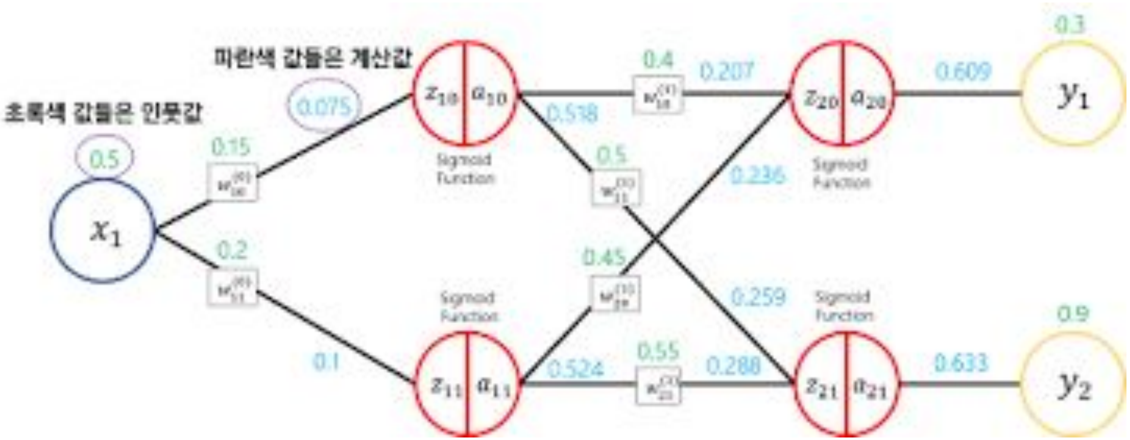
-

# 32. Bias

- Household bias
- Nonresponse bias
- Quota sampling bias
- Response bias
- **Selection bias**
- Size bias
- **Undercoverage bias**
- Voluntary response bias
- Word bias
- **Survivorship bias**

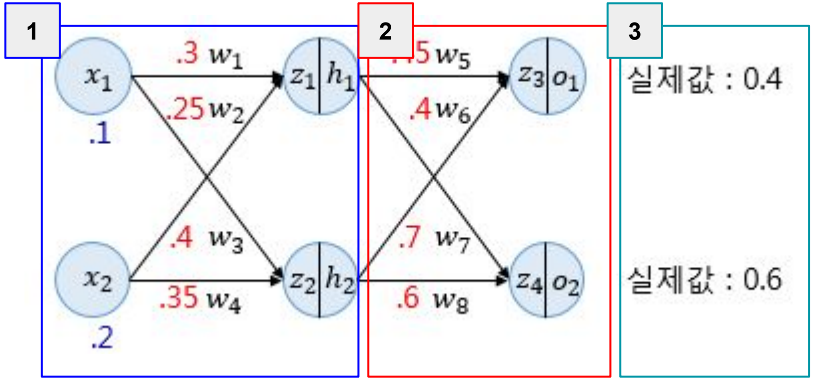
# 33. BackPropagation

- 신경망을 학습하기 위해 사용하는 방법
- gradient를 계산하며 신경망의 파라미터를 최적화




https://wikidocs.net/37406

# 33-1. BackPropagation 예제




손실 함수 : 평균 제곱 오차 사용

3

$$E_{o1} = \frac{1}{2}(target_{o1} - output_{o1})^2 = 0.02193381$$

$$E_{o2} = \frac{1}{2}(target_{o2} - output_{o2})^2 = 0.00203809$$

$$E_{total} = E_{o1} + E_{o2} = 0.02397190$$

1

$$z_1 = w_1 x_1 + w_2 x_2 = 0.3 \times 0.1 + 0.25 \times 0.2 = 0.08$$

$$z_2 = w_3 x_1 + w_4 x_2 = 0.4 \times 0.1 + 0.35 \times 0.2 = 0.11$$

$$h_1 = sigmoid(z_1) = 0.51998934$$

$$h_2 = sigmoid(z_2) = 0.52747230$$

2

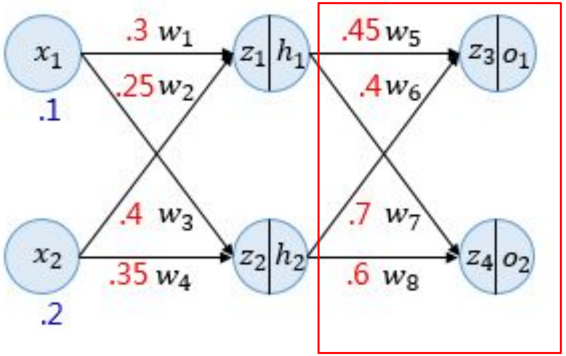
$$z_3 = w_5 h_1 + w_6 h_2 = 0.45 \times h_1 + 0.4 \times h_2 = 0.44498412$$

$$z_4 = w_7 h_1 + w_8 h_2 = 0.7 \times h_1 + 0.6 \times h_2 = 0.68047592$$

$$o_1 = sigmoid(z_3) = 0.60944600$$

$$o_2 = sigmoid(z_4) = 0.66384491$$

# 33-1. BackPropagation 예제




목표 : 손실함수에 대한 가중치 gradient 계산

$w_5, w_6, w_7, w_8$

$$\frac{\partial E_{total}}{\partial w_5} = \boxed{\frac{\partial E_{total}}{\partial o_1}} \times \boxed{\frac{\partial o_1}{\partial z_3}} \times \boxed{\frac{\partial z_3}{\partial w_5}}$$ (By chain rule)

$$\frac{\partial E_{total}}{\partial o_1} = 2 \times \frac{1}{2}(target_{o1} - output_{o1})^{2-1} \times (-1) + 0$$

$$\frac{\partial E_{total}}{\partial o_1} = -(target_{o1} - output_{o1}) = -(0.4 - 0.60944600) = 0.20944600$$

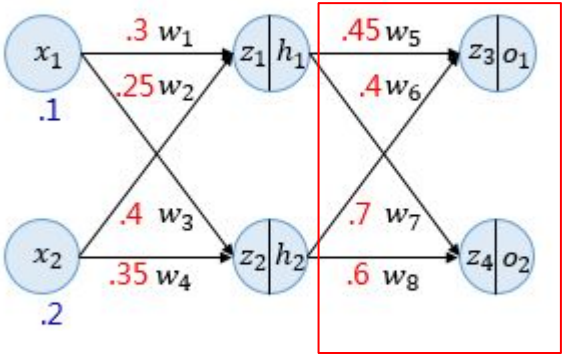
$$\frac{\partial o_1}{\partial z_3} = o_1 \times (1 - o_1) = 0.60944600(1 - 0.60944600) = 0.23802157$$

(sigmoid 미분, 2-3참고) $f(x) \times (1 - f(x))$

$$\frac{\partial z_3}{\partial w_5} = h_1 = 0.51998934$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.20944600 \times 0.23802157 \times 0.51998934 = 0.02592286$$

# 33-1. BackPropagation 예제




목표 : 손실함수에 대한 가중치 gradient 계산

$$w_5, w_6, w_7, w_8$$

$$\frac{\partial E_{total}}{\partial w_5} = \boxed{\frac{\partial E_{total}}{\partial o_1}} \times \boxed{\frac{\partial o_1}{\partial z_3}} \times \boxed{\frac{\partial z_3}{\partial w_5}}$$

(By chain rule)

가중치 업데이트

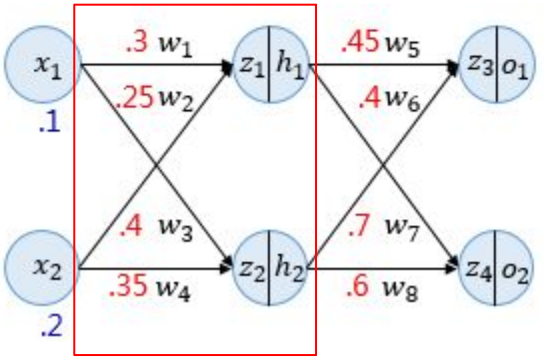
$$w_5^+ = w_5 - \alpha \frac{\partial E_{total}}{\partial w_5} = 0.45 - 0.5 \times 0.02592286 = 0.43703857$$

$$\frac{\partial E_{total}}{\partial w_6} = \frac{\partial E_{total}}{\partial o_1} \times \frac{\partial o_1}{\partial z_3} \times \frac{\partial z_3}{\partial w_6} \rightarrow w_6^+ = 0.38685205$$

$$\frac{\partial E_{total}}{\partial w_7} = \frac{\partial E_{total}}{\partial o_2} \times \frac{\partial o_2}{\partial z_4} \times \frac{\partial z_4}{\partial w_7} \rightarrow w_7^+ = 0.69629578$$

$$\frac{\partial E_{total}}{\partial w_8} = \frac{\partial E_{total}}{\partial o_2} \times \frac{\partial o_2}{\partial z_4} \times \frac{\partial z_4}{\partial w_8} \rightarrow w_8^+ = 0.59624247$$

# 33-1. BackPropagation 예제




실제값 : 0.4

실제값 : 0.6

목표 : 손실함수에 대한 가중치 gradient 계산

$$w_1, w_2, w_3, w_4$$

$$\frac{\partial E_{total}}{\partial w_1} = \boxed{\frac{\partial E_{total}}{\partial h_1}} \times \boxed{\frac{\partial h_1}{\partial z_1}} \times \boxed{\frac{\partial z_1}{\partial w_1}}$$

(By chain rule)

$$\frac{\partial E_{total}}{\partial h_1} = \frac{\partial E_{o1}}{\partial h_1} + \frac{\partial E_{o2}}{\partial h_1}$$

$$\frac{\partial E_{o1}}{\partial h_1} = \frac{\partial E_{o1}}{\partial z_3} \times \frac{\partial z_3}{\partial h_1} = \frac{\partial E_{o1}}{\partial o_1} \times \frac{\partial o_1}{\partial z_3} \times \frac{\partial z_3}{\partial h_1}$$

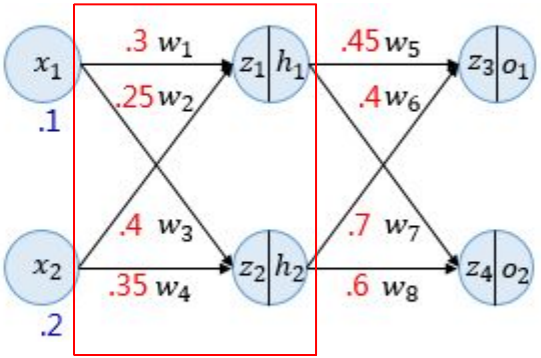
$$= -(target_{o1} - output_{o1}) \times o_1 \times (1 - o_1) \times w_5$$

$$= 0.20944600 \times 0.23802157 \times 0.45 = 0.02243370$$

$$\frac{\partial E_{o2}}{\partial h_1} = \frac{\partial E_{o2}}{\partial z_4} \times \frac{\partial z_4}{\partial h_1} = \frac{\partial E_{o2}}{\partial o_2} \times \frac{\partial o_2}{\partial z_4} \times \frac{\partial z_4}{\partial h_1} = 0.00997311$$

$$\frac{\partial E_{total}}{\partial h_1} = 0.02243370 + 0.00997311 = 0.03240681$$

# 33-1. BackPropagation 예제




목표 : 손실함수에 대한 가중치 gradient 계산

$$w_1, w_2, w_3, w_4$$

$$\frac{\partial E_{total}}{\partial w_1} = \boxed{\frac{\partial E_{total}}{\partial h_1}} \times \boxed{\frac{\partial h_1}{\partial z_1}} \times \boxed{\frac{\partial z_1}{\partial w_1}}$$

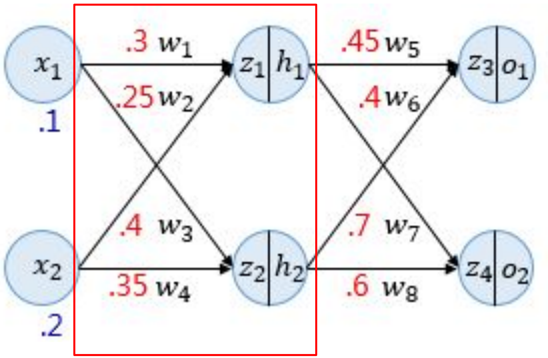
(By chain rule)

$$\boxed{\frac{\partial h_1}{\partial z_1} = h_1 \times (1 - h_1) = 0.51998934(1 - 0.51998934) = 0.24960043}$$

$$\boxed{\frac{\partial z_1}{\partial w_1} = x_1 = 0.1}$$

$$\frac{\partial E_{total}}{\partial w_1} = 0.03240681 \times 0.24960043 \times 0.1 = 0.00080888$$

# 33-1. BackPropagation 예제




목표 : 손실함수에 대한 가중치 gradient 계산

$$w_1, w_2, w_3, w_4$$

$$\frac{\partial E_{total}}{\partial w_1} = \boxed{\frac{\partial E_{total}}{\partial h_1}} \times \boxed{\frac{\partial h_1}{\partial z_1}} \times \boxed{\frac{\partial z_1}{\partial w_1}}$$ (By chain rule)

가중치 업데이트

$$w_1^+ = w_1 - \alpha \frac{\partial E_{total}}{\partial w_1} = 0.1 - 0.5 \times 0.00080888 = 0.29959556$$

$$\frac{\partial E_{total}}{\partial w_2} = \frac{\partial E_{total}}{\partial h_1} \times \frac{\partial h_1}{\partial z_1} \times \frac{\partial z_1}{\partial w_2} \rightarrow w_2^+ = 0.24919112$$

$$\frac{\partial E_{total}}{\partial w_3} = \frac{\partial E_{total}}{\partial h_2} \times \frac{\partial h_2}{\partial z_2} \times \frac{\partial z_2}{\partial w_3} \rightarrow w_3^+ = 0.39964496$$

$$\frac{\partial E_{total}}{\partial w_4} = \frac{\partial E_{total}}{\partial h_2} \times \frac{\partial h_2}{\partial z_2} \times \frac{\partial z_2}{\partial w_4} \rightarrow w_4^+ = 0.34928991$$

# 34. A/B Test

- 대조군과 실험군으로 나누어서 효과를 비교하는 방법
- 모델의 최종 효과를 검증하는 최종 수단
  - 오프라인 평가로는 모델의 과적합 위험을 모두 제거 힘듬
  - 오프라인 평가로는 지연, 데이터 손실, 레이블 손실 등과 같은 상황 반영 어려움
- 실험군 : 새로운 모델
- 대조군 : 기존 모델
- 사용자는 랜덤으로 정해야 샘플의 무편향성을 유지 가능

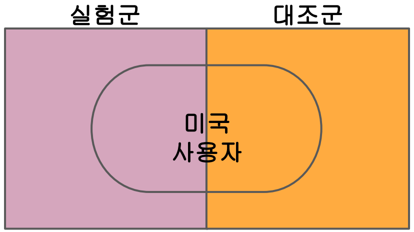
# 34-1. A/B Test 예제

- “미국 사용자”에 대해 새로운 콘텐츠 추천 모델 A를 적용시켜 보고자 한다. 현재 사용자에게 적용되고 있는 추천 알고리즘 모델은 B이다. 실험군/대조군 분류 방법중 정확한 방법을 골라라

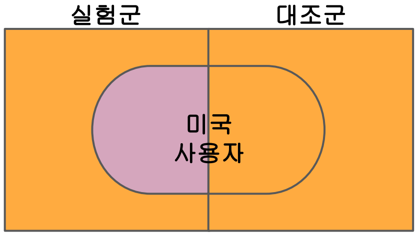
1. User_id에 기반하여 끝자리가 홀수인 사용자들에 대해 실험군과 대조군으로 나누고, 실험군에 대해 A를 적용. 그리고 대조군에는 B를 적용
2. User_id 끝자리가 홀수인 미국 사용자들을 실험군으로 나머지 사용자들은 대조군으로 분류
3. User_id 끝자리가 홀수인 미국 사용자들은 실험군으로 User_id 끝자리가 짝수인 사용자들을 대조군으로 분류
4. **User_id 끝자리가 홀수인 미국 사용자들은 실험군으로 User_id 끝자리가 짝수인 미국 사용자들을 대조군으로 분류**

# 34-1. A/B Test 예제

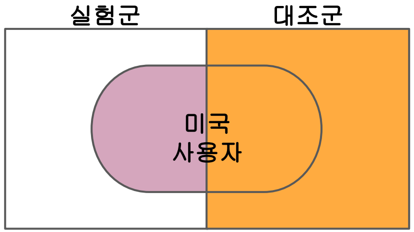
1. User_id에 기반하여 끝자리가 홀수인 사용자들에 대해 실험군과 대조군으로 나누고, 실험군에 대해 A를 적용. 그리고 대조군에는 B를 적용
2. User_id 끝자리가 홀수인 미국 사용자들을 실험군으로 나머지 사용자들은 대조군으로 분류
3. User_id 끝자리가 홀수인 미국 사용자들은 실험군으로 User_id 끝자리가 짝수인 사용자들을 대조군으로 분류
4. **User_id 끝자리가 홀수인 미국 사용자들은 실험군으로 User_id 끝자리가 짝수인 미국 사용자들을 대조군으로 분류**



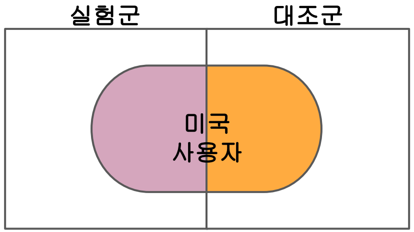

해시 공간




희석된 분할 방안




편차가 있는 분할 방안

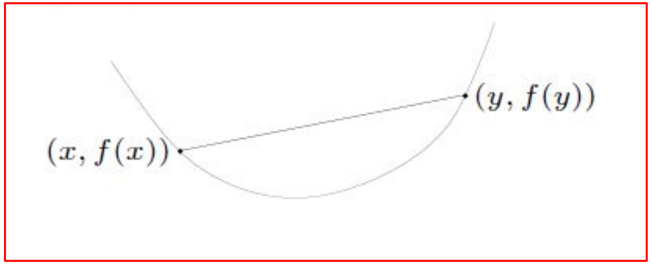



**정확하고 편차가 없는 분할 방안**

# 35. Convex Function

- 임의의 두 점을 직선으로 연결했을 때, 이 직선 위의 임의의 점은 해당 컨벡스 함수 아래에 위치하지 않는다.
- Logistic regression 문제가 convex 최적화 문제
- 컨벡스 최적화 문제는 모든 국소 최솟값이 전역 최솟값이므로 쉽게 풀수 있는 문제로 간주됨
- Non-convex 문제 : pca
    - 그러나 pca는 svd를 사용하여 전역 최솟값을 구할 수 있음

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$



# Popular Machine Learning Algorithms

https://www.ubuntupit.com/machine-learning-algorithms-for-both-newbies-and-professionals/

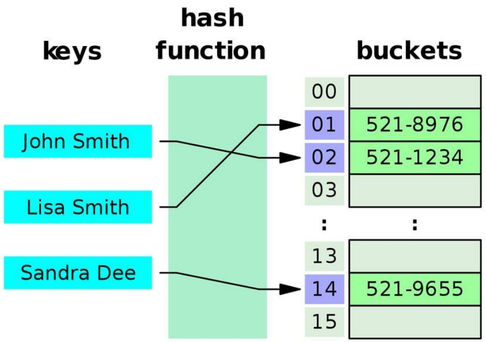
# List

1. **Naive Bayes**
2. **Support Vector Machine**
3. **Linear Regression**
4. **Logistic Regression**
5. **K-Nearest Neighbor**
6. **K-means Clustering**
7. **Decision Tree**
8. **Random Forest**
9. CART
10. Apriori Algorithm
11. **Principal Component Analysis**
12. **CatBoost**
13. Iterative Dichotomiser 3
14. Hierarchical Clustering
15. **Back Propagation**
16. **AdaBoost**
17. Deep Learning
18. **Gradient Boosting Algorithm**
19. Hopfield Network
20. C4.5

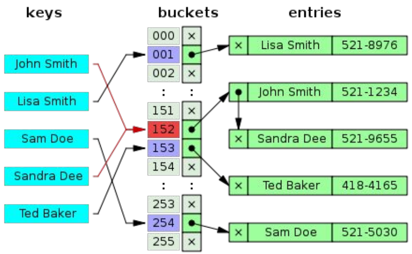
# Computer Structures

# 01. Hash Table

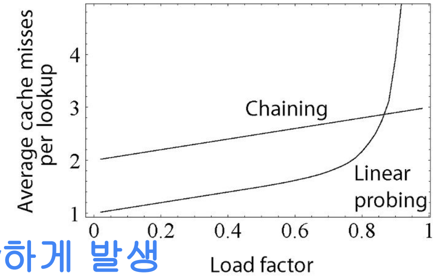
- key를 value에 매핑한 데이터 구조
- **Hash Function** : key값으로 저장되어 있는 주소를 산출하는 함수
- Hash Function을 이용하여 삽입, 검색 속도 빠름
- Python에서는 Dictionary로 구현
- **Hash Collision** 문제 발생 가능성 있음
    - Bucket 은 유한하므로 비둘기집 원리에 의해 중복되는 경우가 발생



# 02. Hash Collision




- 서로 다른 키를 가진 레코드들이 하나의 버킷에 매핑되는 경우
- **bucket overflow** : Collision 버킷에 충분한 공간이 없어 추가 할 수 없는 상태
- Chaining (Open Hashing, Closed Addressing)
    - 버켓 내에 연결리스트(Linked List)를 할당
    - 해시 충돌 발생 시 연결리스트로 데이터들을 연결하는 방식
    - 데이터의 주소값 바뀌지 않음
    - **장점 : 복잡한 계산 사용 X, 해시테이블이 채워질수록 성능저하가 Linear하게 발생**




- Open Addressing (Closed Hashing)
    - 해시 충돌 발생 시 다른 버켓에 데이터를 삽입하는 방식
    - 선형 탐색(Linear Probing): 해시충돌 시 다음 버켓, 혹은 몇 개를 건너뛰어 데이터를 삽입
    - 제곱 탐색(Quadratic Probing): 해시충돌 시 제곱만큼 건너뛴 버켓에 데이터를 삽입
    - 이중 해싱(Double Hashing): 해시충돌 시 다른 해시함수를 한 번 더 적용한 결과를 이용
    - **장점 : 체이닝처럼 포인터 필요 X, 추가적인 저장공간 X, 삽입,삭제시 오버헤드가 적다. 데이터가 적을 때 더 유리**