

# Machine learning handbook

Classifier

---

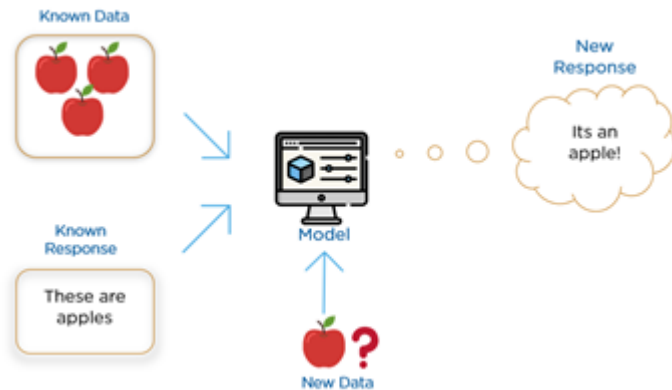
Taeyang Yang

Update: August 14, 2018

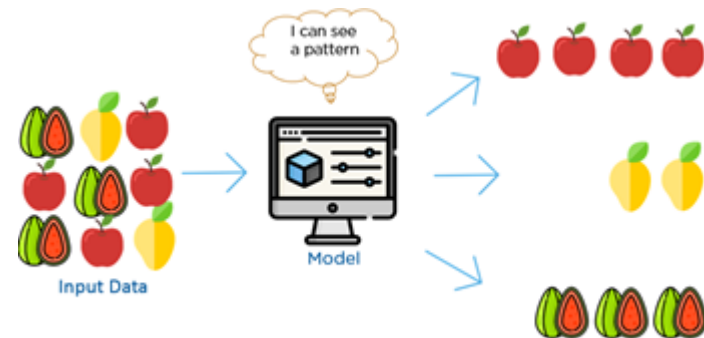
BCILAB, UNIST

# Supervised & unsupervised learning

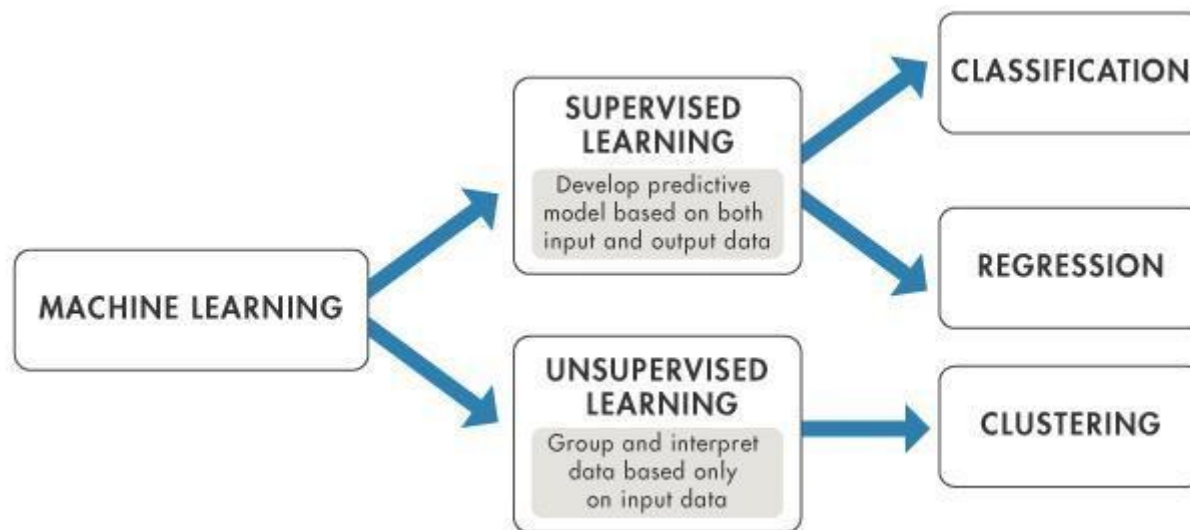
## Supervised learning



## Unsupervised learning



# Classification, Regression & Clustering

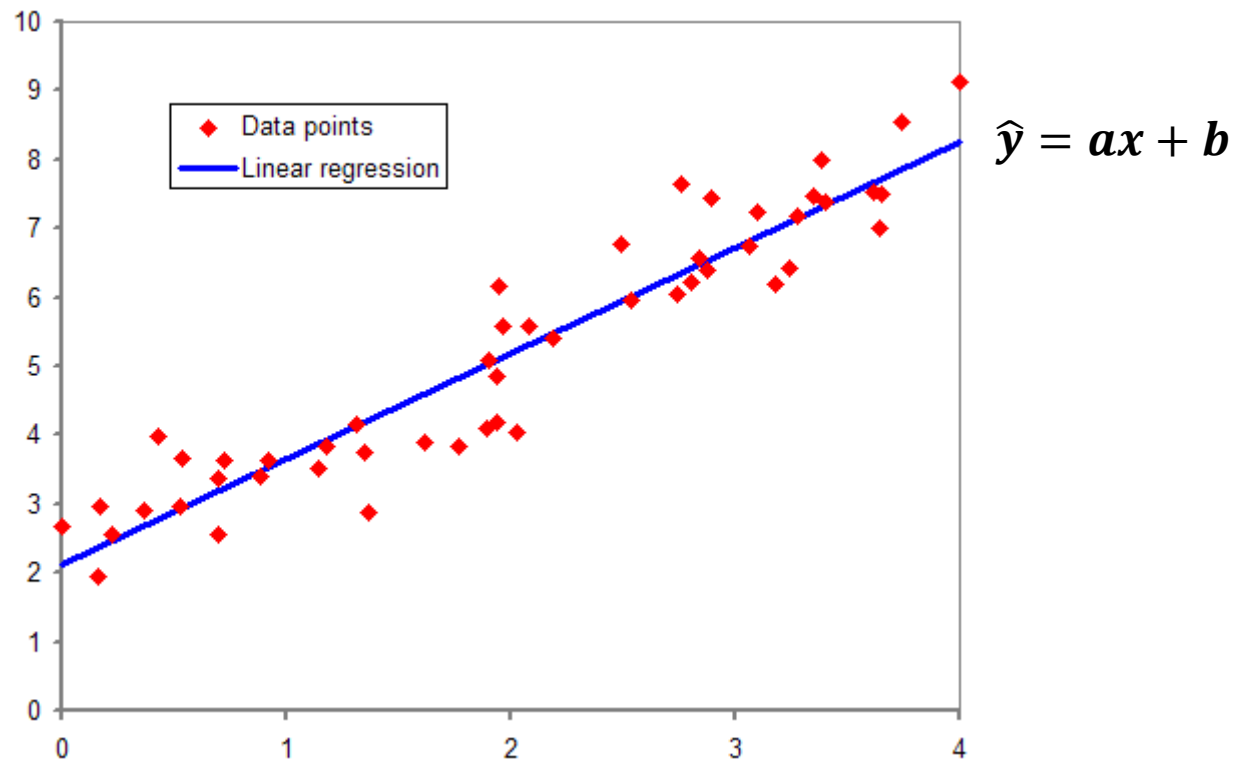


# 1

## Supervised learning

- Linear Regression
- Naïve Bayes (NB) classifier
- Linear Discriminant Analysis (LDA)
- Support Vector Machine (SVM)
- Decision Tree
- Random Forest
- Gradient boosting

# Linear Regression (overview)



# Linear Regression

- Same expression !

$$\mathbf{y} = \mathbf{ax} + \mathbf{b}$$

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

- Cost (Loss) function

$$\begin{aligned} J(\boldsymbol{\beta}) = MSE &= \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (X^{(i)}\boldsymbol{\beta} + \varepsilon - y^{(i)})^2 \end{aligned}$$

# Linear Regression

- Cost (Loss) function

$$J(\beta) = MSE = \frac{1}{2m} \sum_{i=1}^m (X^{(i)}\beta + \varepsilon - y^{(i)})^2$$

- Derivate

$$\begin{aligned}\frac{\partial J}{\partial \varepsilon} &= \frac{1}{m} \sum_{i=1}^m (X^{(i)}\beta + \varepsilon - y^{(i)}) \\ \frac{\partial J}{\partial \beta} &= \frac{1}{m} \sum_{i=1}^m (X^{(i)}\beta + \varepsilon - y^{(i)})X^{(i)}\end{aligned}$$

# Linear Regression

- Solution 1: Normal equation, **derivate=0**

$$\begin{aligned}\varepsilon m + \beta \sum_{i=1}^m X^{(i)} &= \sum_{i=1}^m y^{(i)} \\ \varepsilon \sum_{i=1}^m X^{(i)} + \beta \sum_{i=1}^m X^{(i)2} &= \sum_{i=1}^m y^{(i)} X^{(i)}\end{aligned}$$

- Matrix expression

$$\begin{bmatrix} \varepsilon \\ \beta \end{bmatrix}^T \begin{bmatrix} m & \sum_{i=1}^m X^{(i)} \\ \sum_{i=1}^m X^{(i)} & \sum_{i=1}^m X^{(i)2} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^m y^{(i)} \\ \sum_{i=1}^m y^{(i)} X^{(i)} \end{bmatrix}$$



# Linear Regression

- Trick

$$\mathbf{w} = \begin{bmatrix} \varepsilon \\ \beta \end{bmatrix}, \mathbf{X} = \begin{bmatrix} 1 & \mathbf{x}_1^T \\ 1 & \mathbf{x}_2^T \\ \vdots & \vdots \\ 1 & \mathbf{x}_n^T \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\mathbf{X}^T \mathbf{X} = \begin{bmatrix} m & \sum_{i=1}^m x_i \\ \sum_{i=1}^m x_i & \sum_{i=1}^m x_i^2 \end{bmatrix}, \mathbf{X}^T \mathbf{y} = \begin{bmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m y_i x_i \end{bmatrix}$$

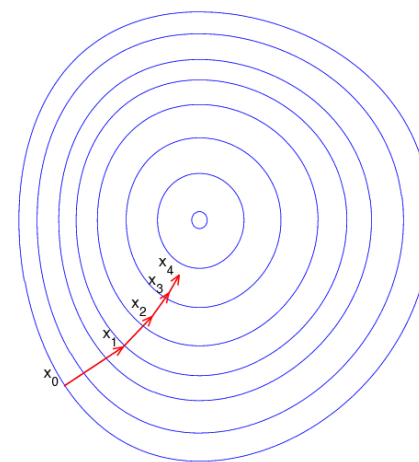
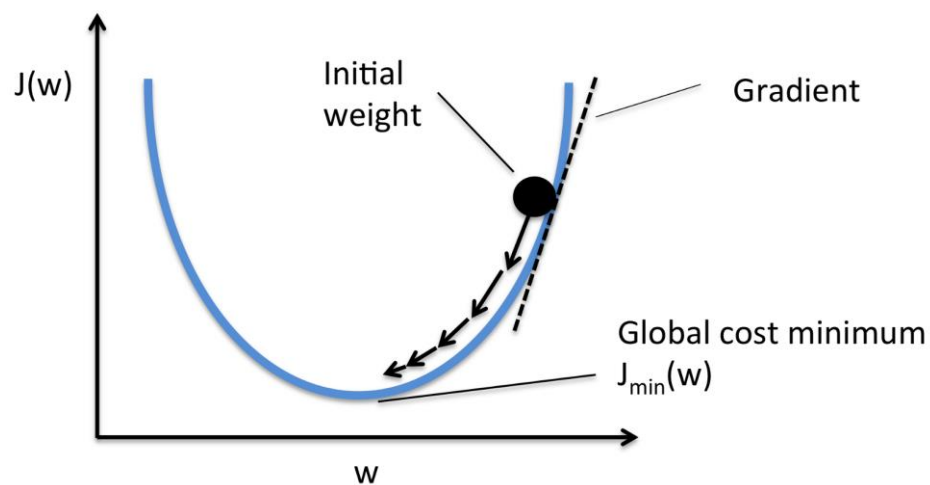
- Estimated weight

$$\begin{aligned} \mathbf{X}^T \mathbf{X} \mathbf{w} &= \mathbf{X}^T \mathbf{y} \\ \mathbf{w} &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \end{aligned}$$

# Linear Regression

- Solution 2: Gradient descent algorithm

$$\theta = \theta - \eta * \frac{\partial}{\partial \theta} J(\theta)$$



# Linear Regression: Ridge and Lasso

- Cost function에 weight에 대한 Penalty를 부여하여, Weight 들의 크기에 제한을 줌
  - 극단적인 weight 값 방지 → Overfitting 방지

- 종류

- L1 regularization (Lasso): Manhattan distance

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^n |\theta_j|$$

- L2 regularization (Ridge): Euclidean distance

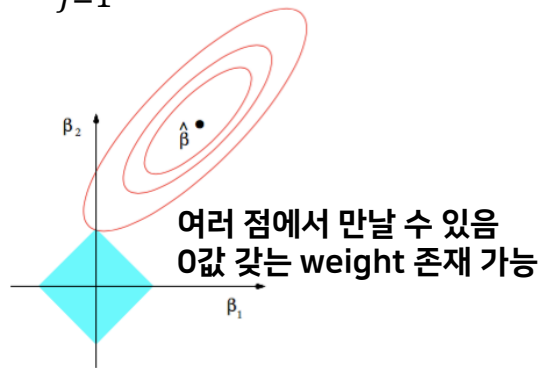
$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2} \sum_{j=1}^n \theta_j^2$$

# Linear Regression: Ridge and Lasso

## L1 regularization (Lasso)

- Unstable solution (여러 접점)
- Always on solution
- Sparse solution (weight = 0 도 존재함)
- Feature selection (weight = 0 → 덜 중요)

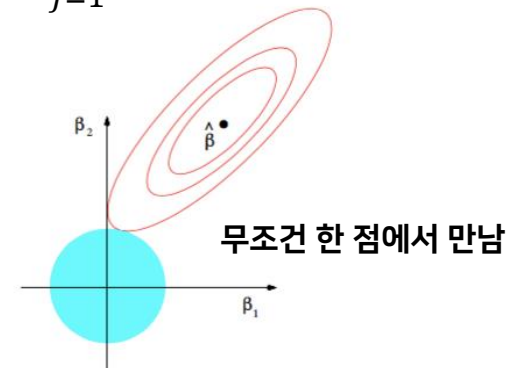
$$\sum_{j=1}^n |\theta_j| \leq s$$



## L2 regularization (Ridge)

- Stable solution
- Only one solution
- Non-sparse solution

$$\sum_{j=1}^n \theta_j^2 \leq s$$



**s가 크면 파란 면적 (weight가 가질 수 있는 범위)이 넓어짐**

# Linear regression evaluation

## MSE/RMSE ...

- Error 의 척도

$$MSE = \frac{1}{n} \sum (y - \hat{y})^2$$

$$RMSE = \sqrt{\frac{1}{n} \sum (y - \hat{y})^2}$$

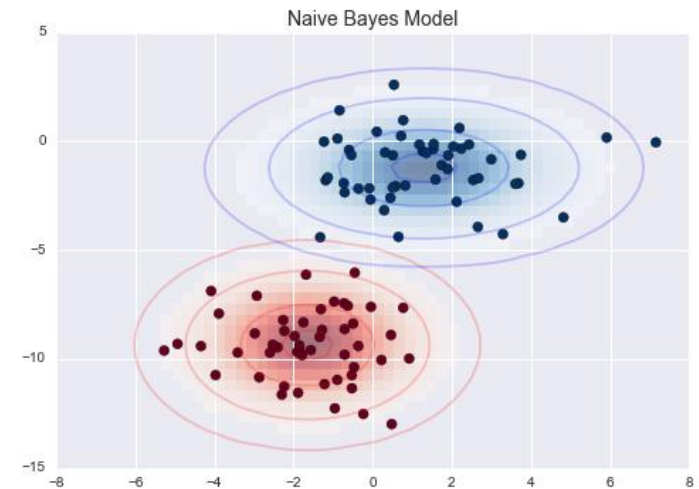
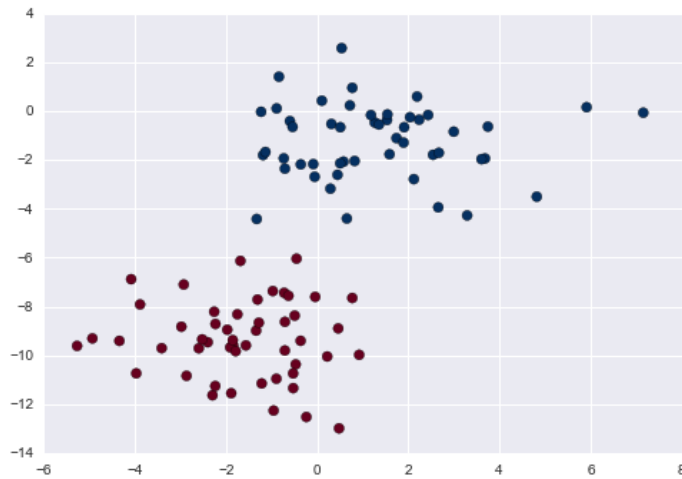
## R-squared ( $R^2$ )

- goodness of fit

$$\begin{aligned} R^2 &= 1 - \frac{\text{Explained variation by model}}{\text{Total variation}} \\ &= 1 - \frac{SS_{regr}}{SS_{total}} \\ &= 1 - \frac{\sum_i (f_i - \hat{y})^2}{\sum_i (y_i - \hat{y})^2} \end{aligned}$$

- 0~1 범위 밖의  $R^2$  (negative)의 의미
  - 모델이 Horizontal hyperplane보다 별로다

# Naïve Bayes classifier (overview)



# Naïve Bayes classifier

- Bayes theorem-based model
  - Bayes' theorem

$$\underset{\text{Posterior probability}}{p(C_k|x)} = \frac{\overset{\text{Likelihood}}{p(x|C_k)} \overset{\text{Class prior probability}}{p(C_k)}}{\underset{\text{Predictor prior probability}}{p(x)}}$$

# Naïve Bayes classifier

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{p(x)}$$

- **Joint probability:** chain rule of conditional probability

$$\begin{aligned} p(C_k, x_1, \dots, x_n) &= p(x_1, \dots, x_n, C_k) \\ &= p(x_1|x_2, \dots, x_n, C_k)p(x_2, \dots, x_n, C_k) \\ &= p(x_1|x_2, \dots, x_n, C_k)p(x_2|x_3, \dots, x_n, C_k)p(x_3, \dots, x_n, C_k) \\ &= \dots \\ &= p(x_1|x_2, \dots, x_n, C_k)p(x_2|x_3, \dots, x_n, C_k) \dots p(x_n|C_k)p(C_k) \end{aligned}$$

- **Assumption of “naïve conditional independence” :**
  - Each  $x_i$  is **conditionally independent** of every other features  $x_j$  for  $j \neq i$  given the category  $C_k$

$$p(x_i|x_{i+1}, \dots, x_n, C_k) = p(x_i|C_k)$$

- **Posterior**

$$\begin{aligned} p(C_k|x_1, \dots, x_n) &\propto p(C_k, x_1, \dots, x_n) = p(C_k)p(x_1|C_k)p(x_2|C_k) \dots p(x_n|C_k) \\ &= p(C_k) \prod_{i=1}^n p(x_i|C_k) \end{aligned}$$



# Naïve Bayes classifier

- Maximum a posterior (MAP)

$$\hat{y} = \operatorname{argmax}_{k \in \{1, \dots, K\}} p(C_k) \prod_{i=1}^n p(x_i | C_k)$$

- What is  $p(x_i | C_k)$ ?
  - parameter estimation **based on assumption of likelihood**
    - Gaussian naïve Bayes  $\rightarrow \mu, \sigma$

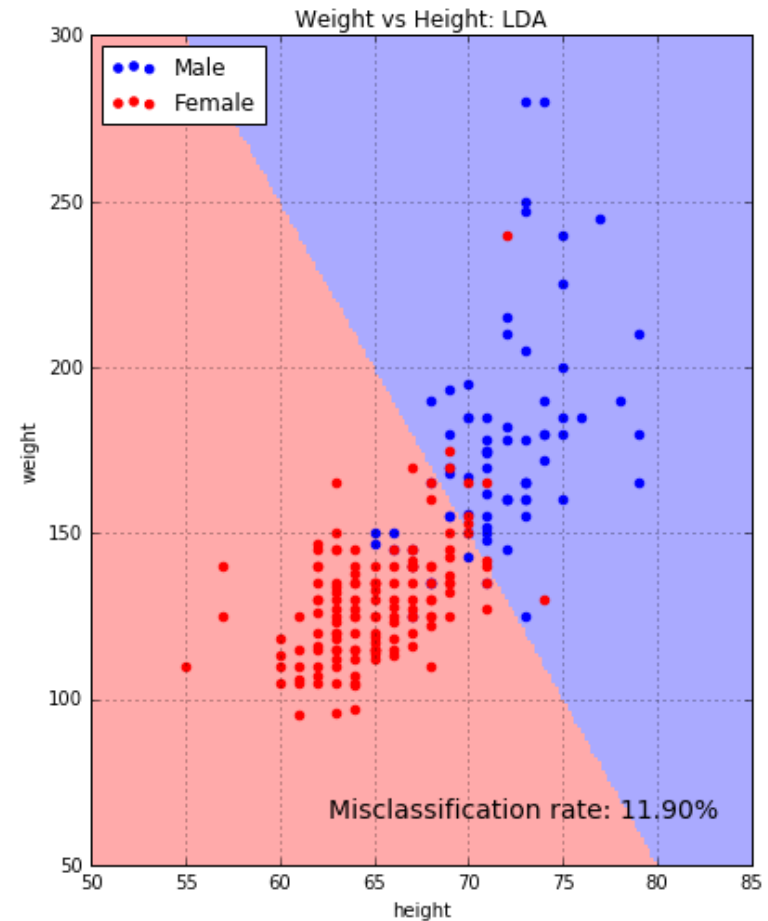
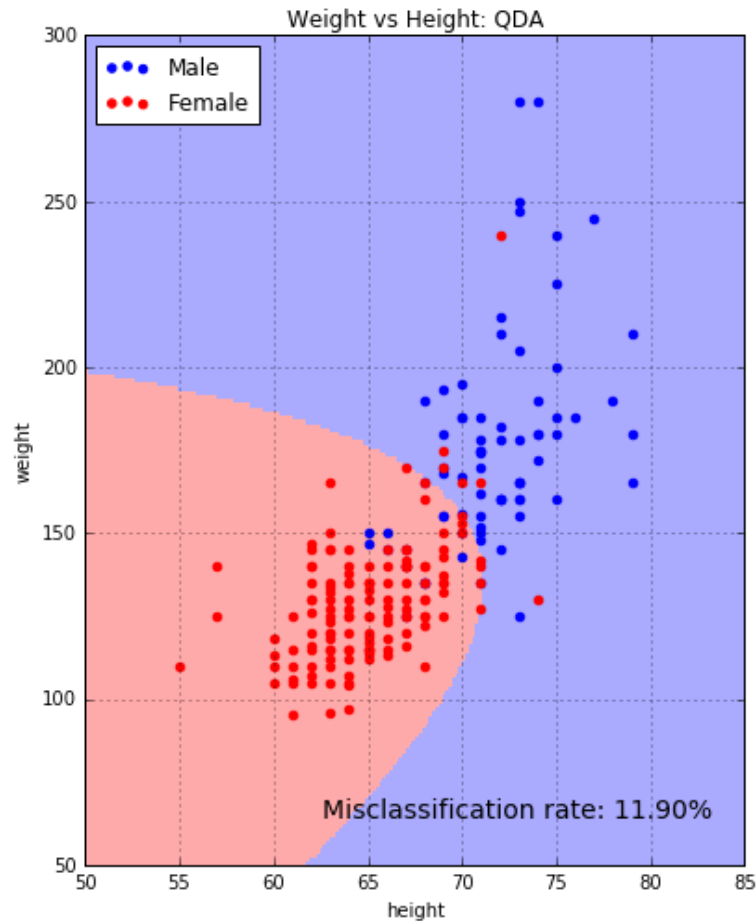
$$p(x = v | C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

- Multinomial naïve Bayes  $\rightarrow p_{ki}$

$$p(X | C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i}$$

# Linear Discriminant Analysis (overview)

Comparison of QDA and LDA



# Linear Discriminant Analysis

- 데이터 클래스를 가장 잘 분류하는 축으로 projection
  - 원래:  $P$  차원의 데이터 벡터  $X_i$

$N_1$ : the number of samples in class  $C_1$

$N_2$ : the number of samples in class  $C_2$

- 원 좌표계의 평균벡터

$$\overrightarrow{m_1} = \frac{1}{N_1} \sum_{n \in C_1} X_n$$

$$\overrightarrow{m_2} = \frac{1}{N_2} \sum_{n \in C_2} X_n$$

- Projected된 평균벡터

$$\begin{aligned}\overrightarrow{\mu_1} &= \overrightarrow{w}^T \overrightarrow{m_1} \\ \overrightarrow{\mu_2} &= \overrightarrow{w}^T \overrightarrow{m_2}\end{aligned}$$

# Linear Discriminant Analysis

- 목표: Projected 시킨 좌표계에서 아래를 만족시키는  $\vec{w}$  구하기
  - 클래스간 차이는 크고 = 평균 차이 max

$$\underset{\vec{w}}{\operatorname{argmax}}((\overrightarrow{\mu_1} - \overrightarrow{\mu_2})^2)$$

$$\overrightarrow{\mu_1} - \overrightarrow{\mu_2} = \vec{w}^T \overrightarrow{m_1} - \vec{w}^T \overrightarrow{m_2} = \vec{w}^T (\overrightarrow{m_1} - \overrightarrow{m_2})$$

- 클래스내 차이는 적게 = 클래스 내 분산 차이 min

$$\underset{\vec{w}}{\operatorname{argmin}}(s_1^2 + s_2^2)$$

$$s_1^2 = \sum_{n \in C_1} (w^T X_n - \overrightarrow{\mu_1})^2$$

$$s_2^2 = \sum_{n \in C_2} (w^T X_n - \overrightarrow{\mu_2})^2$$

# Linear Discriminant Analysis

- 목표: Projected 시킨 좌표계에서 아래를 만족시키는  $\vec{w}$  구하기
  - 평균 차이는 최대화시키고, 클래스 내 분산 차이는 최소화시키는 방법?
  - 분수 !

$$\operatorname{argmax}_{\vec{w}} \frac{(\overrightarrow{\mu_1} - \overrightarrow{\mu_2})^2}{s_1^2 + s_2^2}$$

# Linear Discriminant Analysis

- 분자항 전개

$$\begin{aligned}\overrightarrow{\mu_1} - \overrightarrow{\mu_2} &= (w^T \overrightarrow{m_1} - w^T \overrightarrow{m_2})^2 \\ &= (w^T (\overrightarrow{m_1} - \overrightarrow{m_2}))^2 \\ &= w^T (\overrightarrow{m_1} - \overrightarrow{m_2})(\overrightarrow{m_1} - \overrightarrow{m_2})^T w \\ &= w^T S_w w\end{aligned}$$

- Between class scatter matrix: 대칭행렬

$$S_w = (\overrightarrow{m_1} - \overrightarrow{m_2})(\overrightarrow{m_1} - \overrightarrow{m_2})^T$$

# Linear Discriminant Analysis

- 분모항 전개

$$\begin{aligned} s_1^2 + s_2^2 &= \sum_{n \in C_1} (w^T X_n - \bar{\mu}_1)^2 + \sum_{n \in C_2} (w^T X_n - \bar{\mu}_1)^2 \\ &= \sum_{n \in C_1} (w^T X_n - w^T \bar{m}_1)^2 + \sum_{n \in C_2} (w^T X_n - w^T \bar{m}_2)^2 \\ &= \sum_{n \in C_1} (w^T (X_n - \bar{m}_1))^2 + \sum_{n \in C_2} (w^T (X_n - \bar{m}_2))^2 \\ &= \sum_{n \in C_1} w^T (X_n - \bar{m}_1) (X_n - \bar{m}_1)^T w + \sum_{n \in C_2} w^T (X_n - \bar{m}_2) (X_n - \bar{m}_2)^T w \\ &= w^T \left[ \sum_{n \in C_1} (X_n - \bar{m}_1) (X_n - \bar{m}_1)^T + \sum_{n \in C_2} (X_n - \bar{m}_2) (X_n - \bar{m}_2)^T \right] w \end{aligned}$$

- Within-class Scatter matrix: 대칭행렬

$$S_W = \sum_{n \in C_1} (X_n - \bar{m}_1) (X_n - \bar{m}_1)^T + \sum_{n \in C_2} (X_n - \bar{m}_2) (X_n - \bar{m}_2)^T$$

# Linear Discriminant Analysis

- “최대화”:  $\vec{w}$ 에 대해 미분한 값 = 0

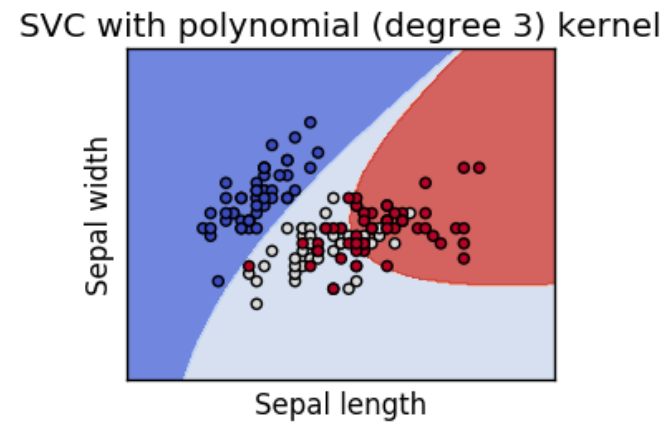
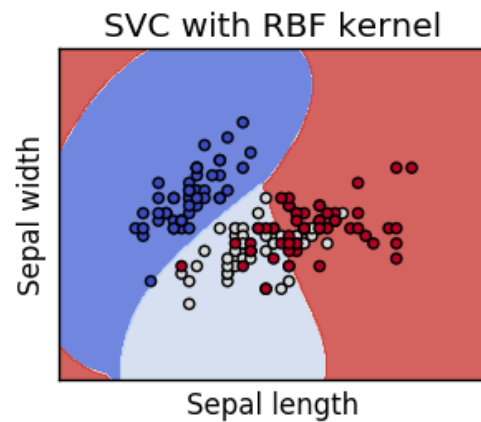
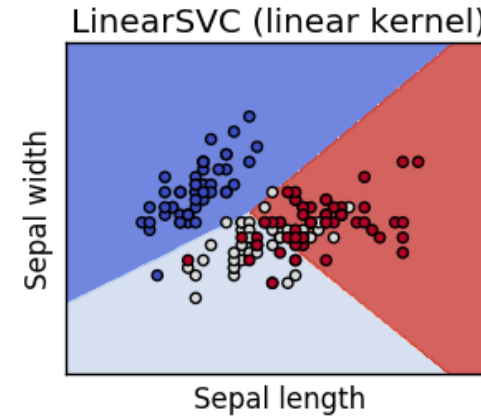
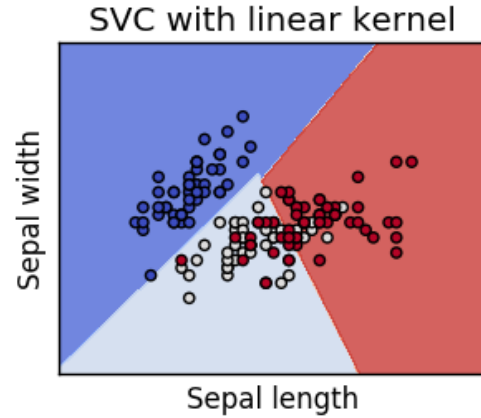
$$J(\vec{w}) = \frac{(\vec{\mu}_1 - \vec{\mu}_2)^2}{s_1^2 + s_2^2} = \frac{w^T S_B w}{w^T S_W w}$$

$$\frac{\partial J(\vec{w})}{\partial \vec{w}} = \frac{2S_B S_W w - w^T S_B w 2S_W w}{(w^T S_W w)^2} = 0$$

- $\vec{w}$ 를 찾으면 됩니다!
  - 어떻게? 아직 모르겠다 ...

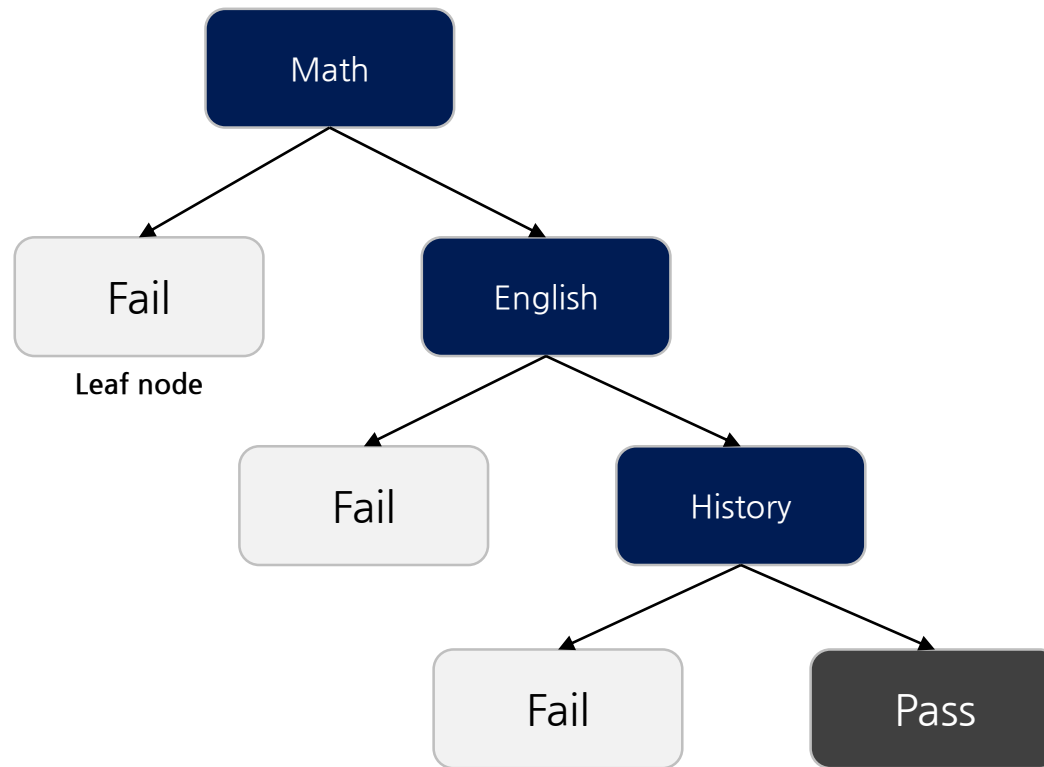


# Support Vector Machine (overview)



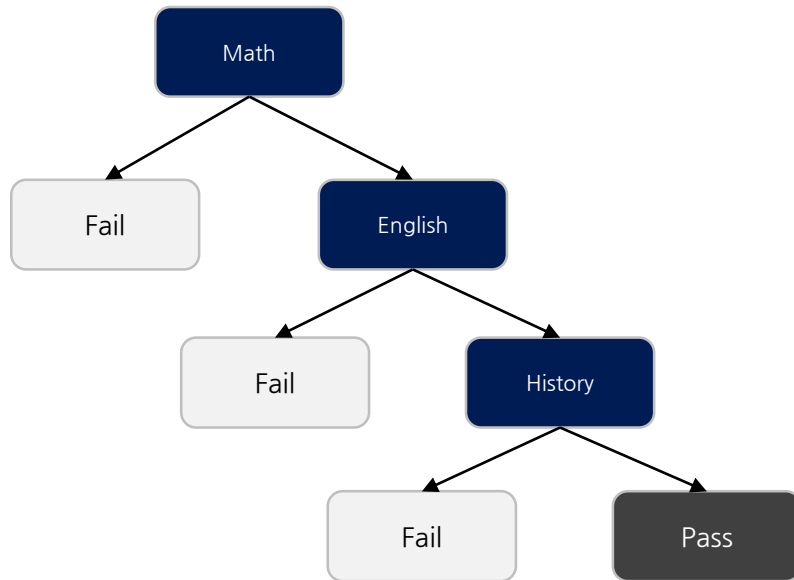
# Decision Tree (Overview)

- Decision Tree = Successive filter system

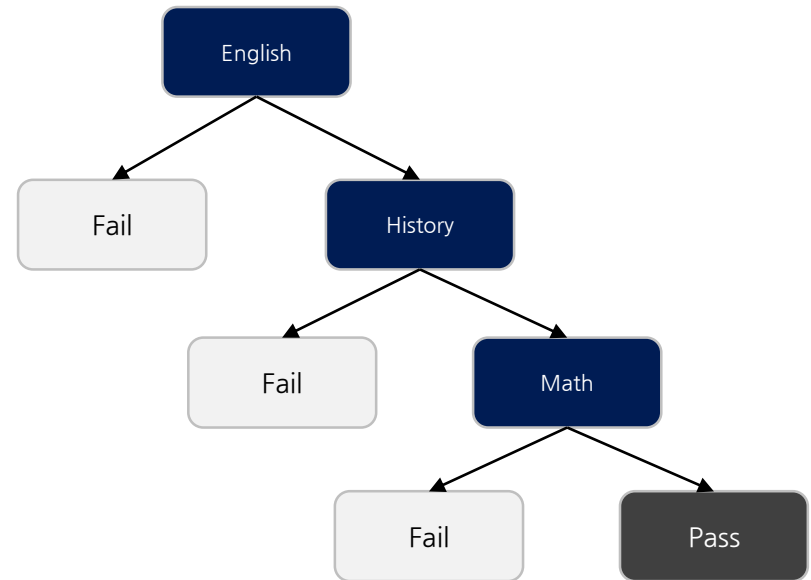


# Decision Tree

- Tree 1



- Tree 2



How can we design decision tree?


**Information gain !**

# Decision Tree: ID3 algorithm

- Information gain

$$\text{Information gain} = \text{base entropy} - \sum p(\text{case}) * \text{new entropy}$$

*Weight for each option*



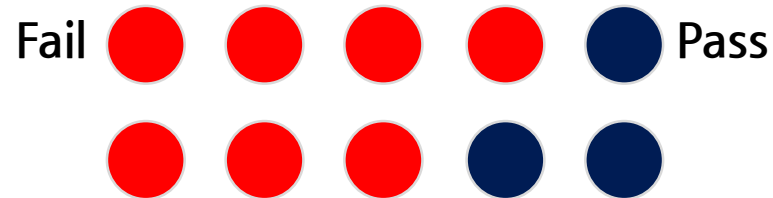
- Entropy

$$\text{Entropy}(\text{Pass}, \text{Fail}) = -p(\text{Pass}) \ln p(\text{Pass}) - p(\text{Fail}) \ln p(\text{Fail})$$

**High information gain criteria → low entropy in next node**

# Decision Tree: ID3 algorithm

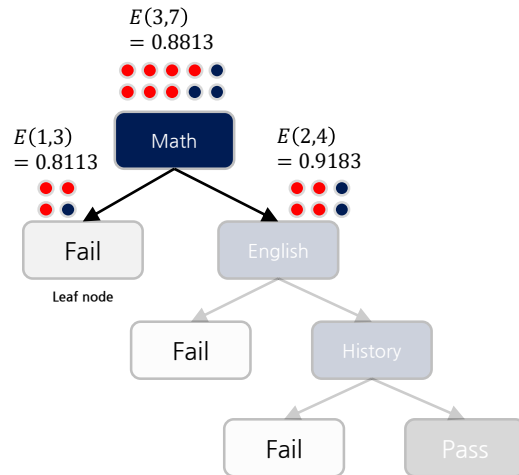
- Base entropy



$$\begin{aligned} Entropy(3,7) &= -\left(\frac{3}{10} \log_2 \frac{3}{10} + \frac{7}{10} \log_2 \frac{7}{10}\right) \\ &= 0.8813 \end{aligned}$$

# Decision Tree: ID3 algorithm

## • Tree 1

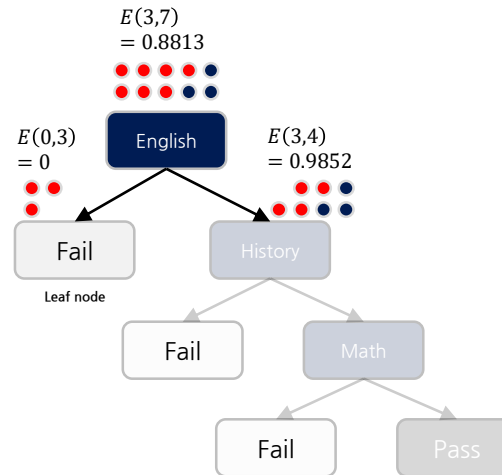


Information Gain

$$= E(3,7) - \left( \frac{4}{10} * E(1,3) + \frac{6}{10} * E(2,4) \right)$$

$$= 0.0058$$

## • Tree 2

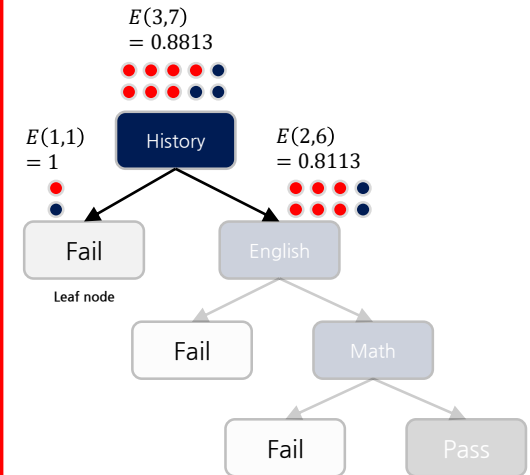


Information Gain

$$= E(3,7) - \left( \frac{3}{10} * E(0,3) + \frac{7}{10} * E(3,4) \right)$$

$$= \mathbf{0.1916}$$

## • Tree 3



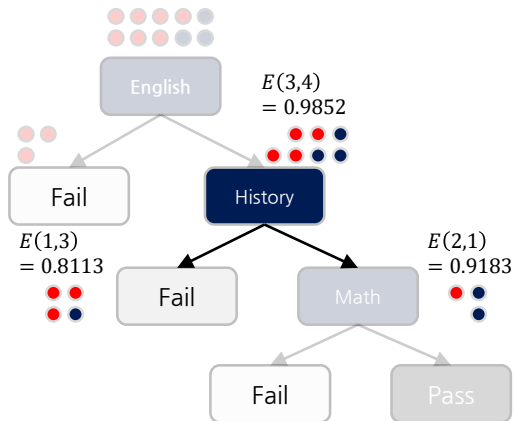
Information Gain

$$= E(3,7) - \left( \frac{2}{10} * E(1,1) + \frac{8}{10} * E(2,6) \right)$$

$$= 0.0323$$

# Decision Tree: ID3 algorithm

- Tree 2-1

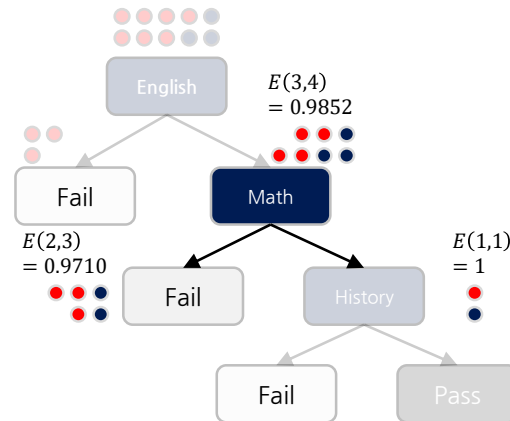


Information Gain

$$= E(3,4) - \left( \frac{4}{7} * E(1,3) + \frac{3}{7} * E(2,1) \right)$$

$$= \mathbf{0.1281}$$

- Tree 2-2



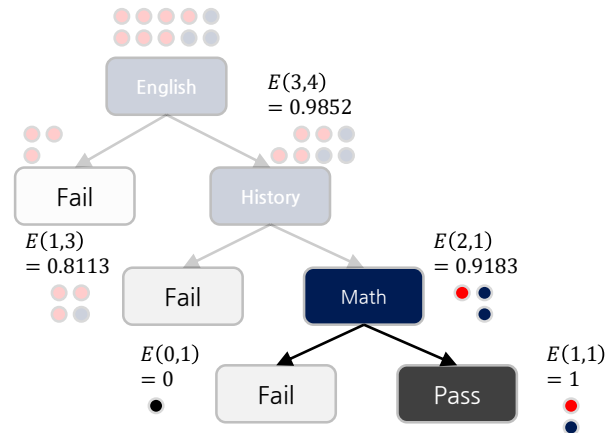
Information Gain

$$= E(3,4) - \left( \frac{5}{7} * E(2,3) + \frac{2}{7} * E(1,1) \right)$$

$$= 0.0060$$

# Decision Tree: ID3 algorithm

- Tree 2-1-1



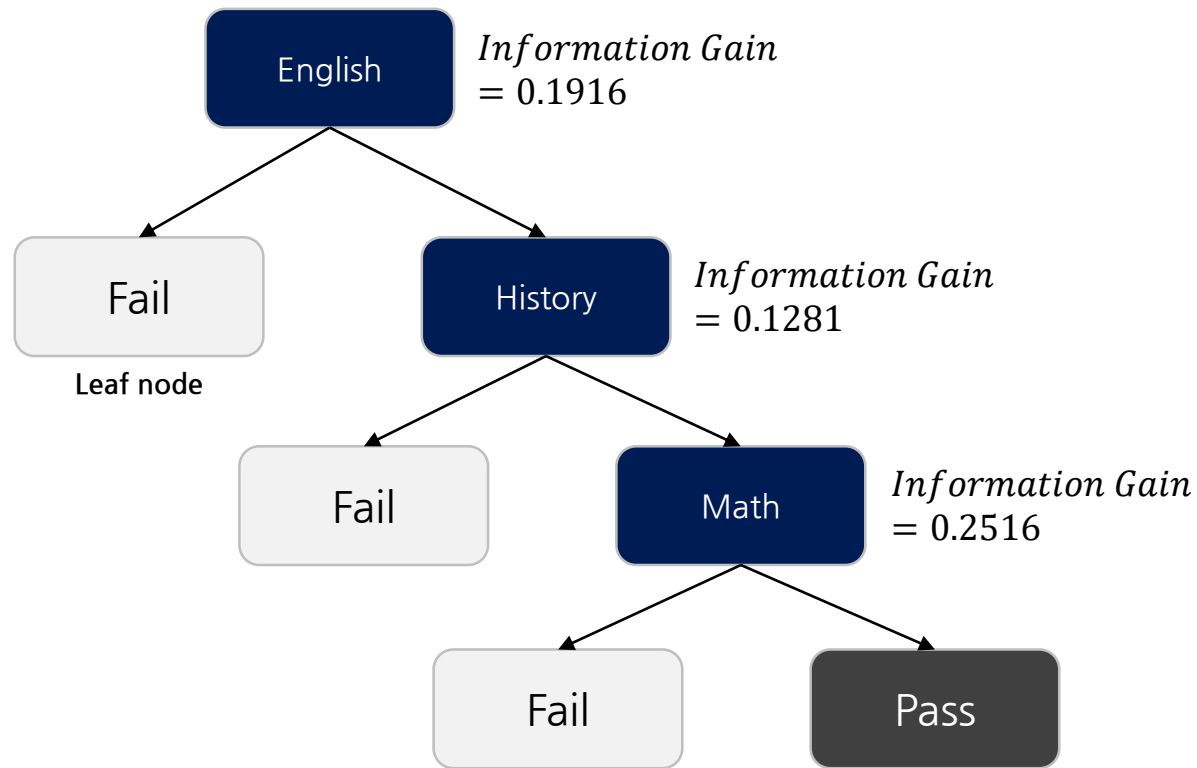
*Information Gain*

$$\begin{aligned}
 &= E(2,1) - \left( \frac{1}{3} * E(0,1) + \frac{2}{3} * E(1,1) \right) \\
 &= \mathbf{0.2516}
 \end{aligned}$$



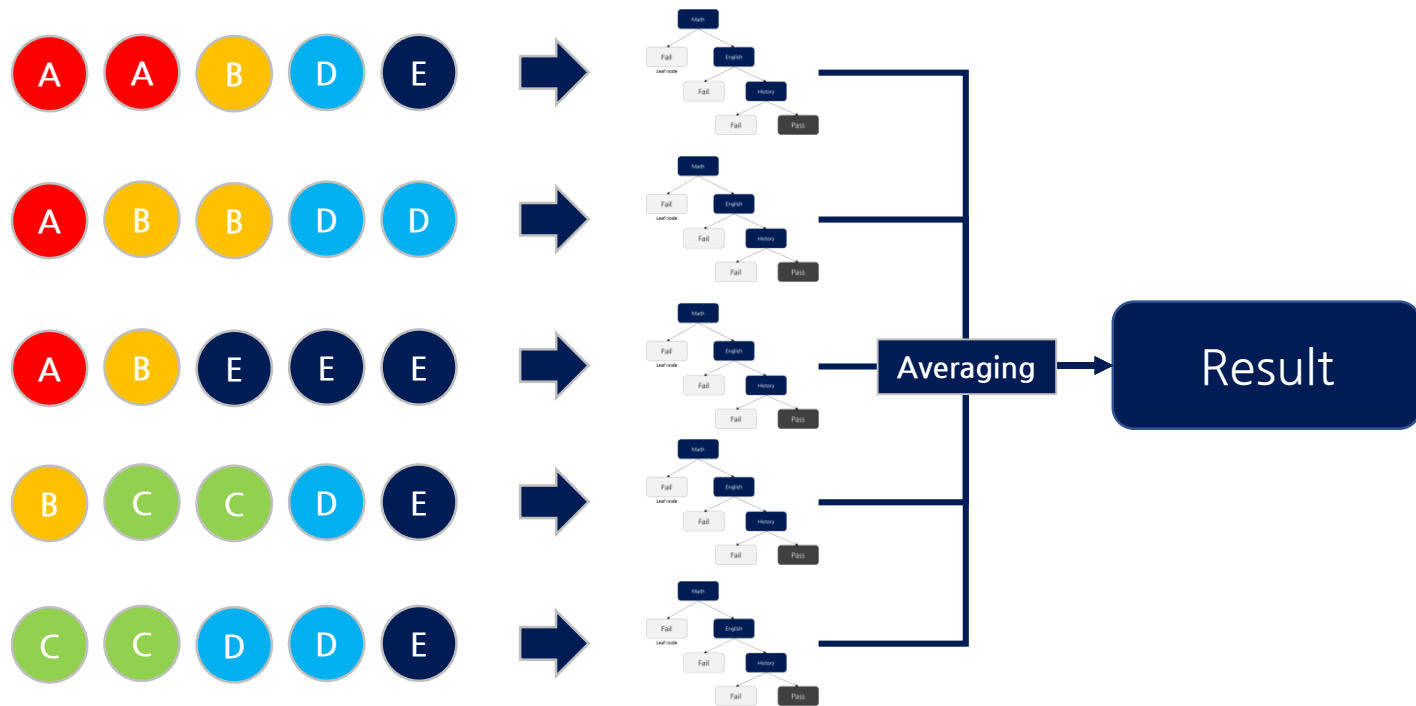
# Decision Tree: ID3 algorithm

- Final decision tree
  - Output: weight of each variables (Information gain)



# Random Forest (Overview)

- Random Forest = Ensemble of decision trees



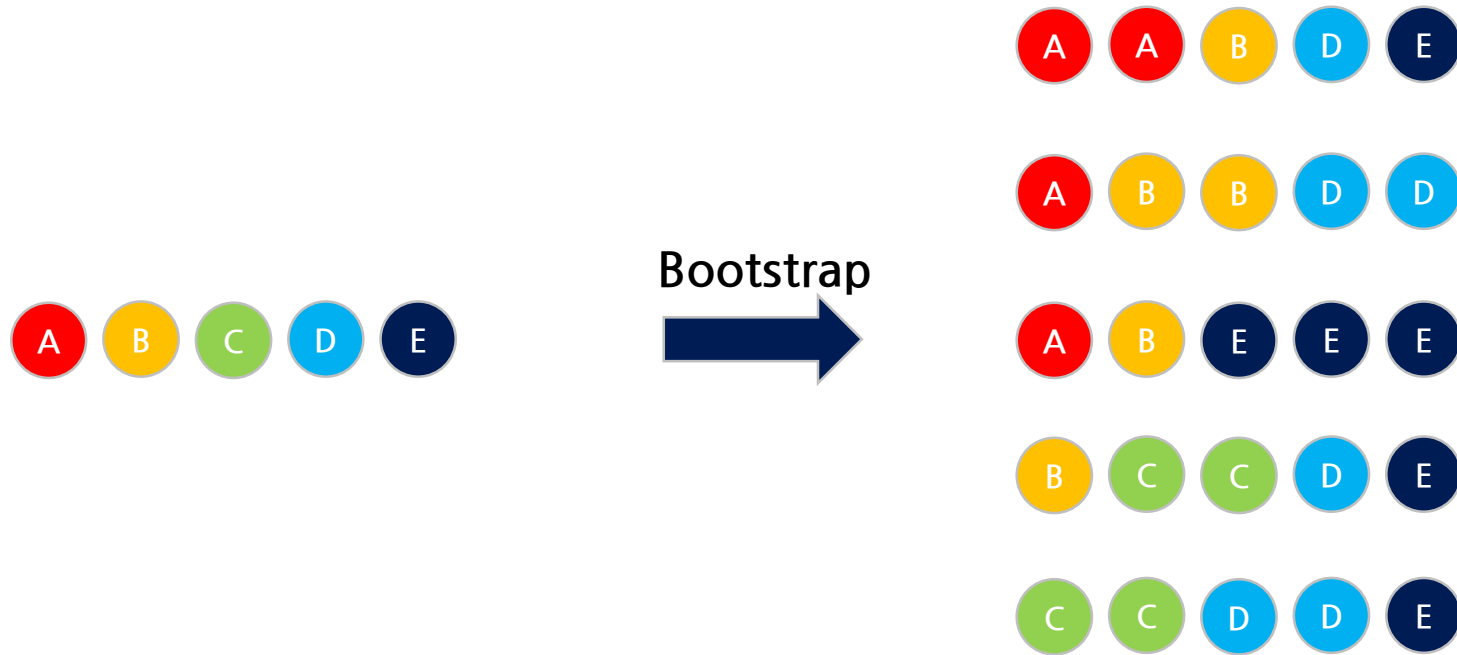
# Random forest

- Ensemble of decision trees



# Random forest: Bootstrap

- Bootstrap



Same size, Overlap permission → Different combination

# Gradient boosting: Decision Tree

- Decision Tree

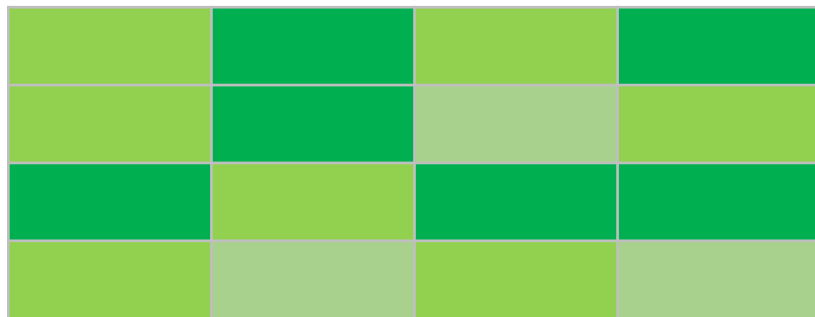


**Important parameter: max\_features**

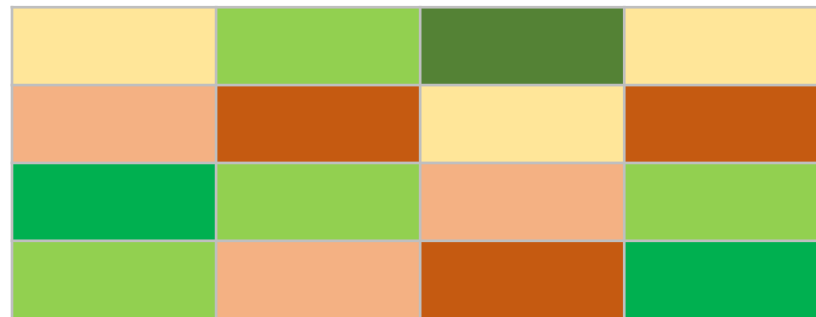
The number of variables in each tree

# Gradient boosting: Decision Tree

High max\_features

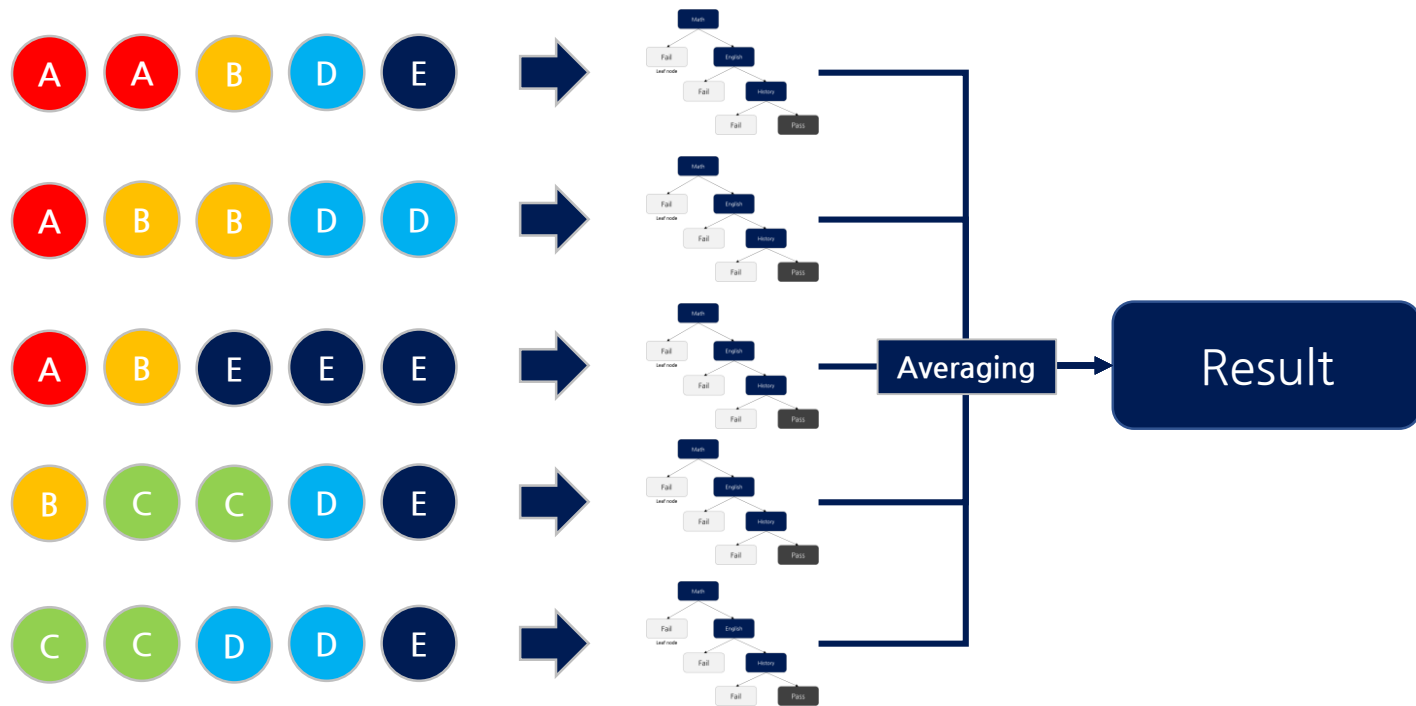


Low max\_features



# Gradient boosting: Ensemble

- Ensemble



# Gradient boosting



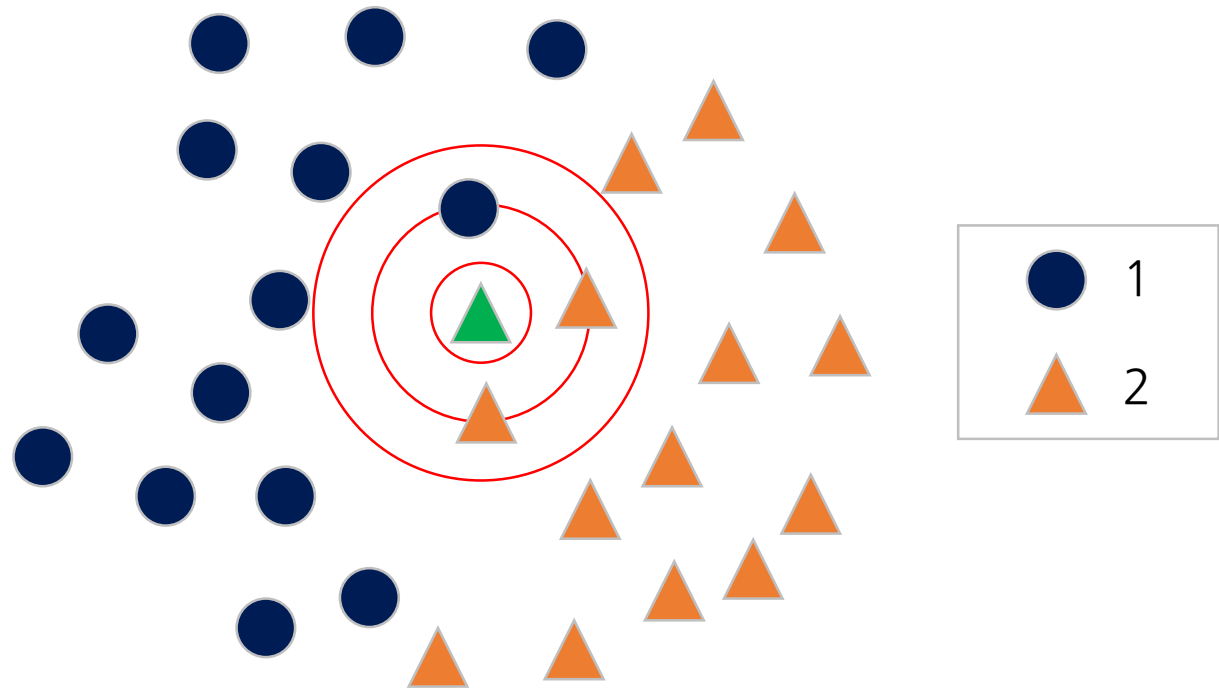
# 2

## Unsupervised learning

- K-nearest neighborhood
- K-means clustering

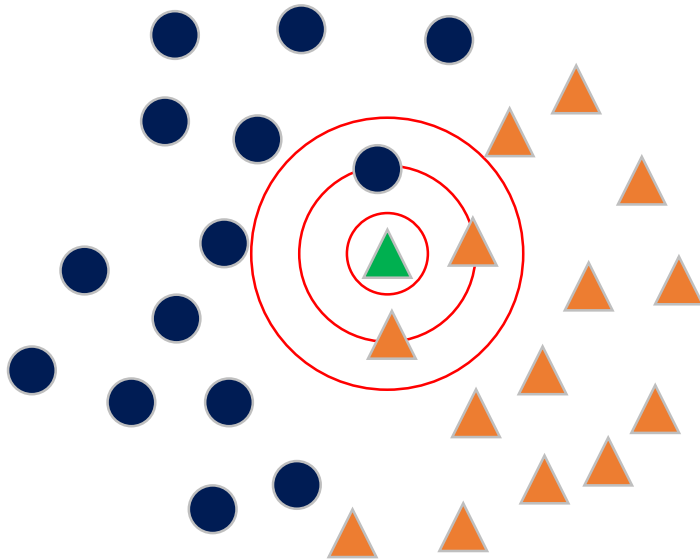
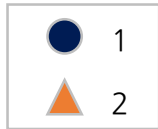
# K-nearest neighborhood (Overview)

- Basic idea of K-nearest neighborhood
  - Follow the majority of k neighborhoods

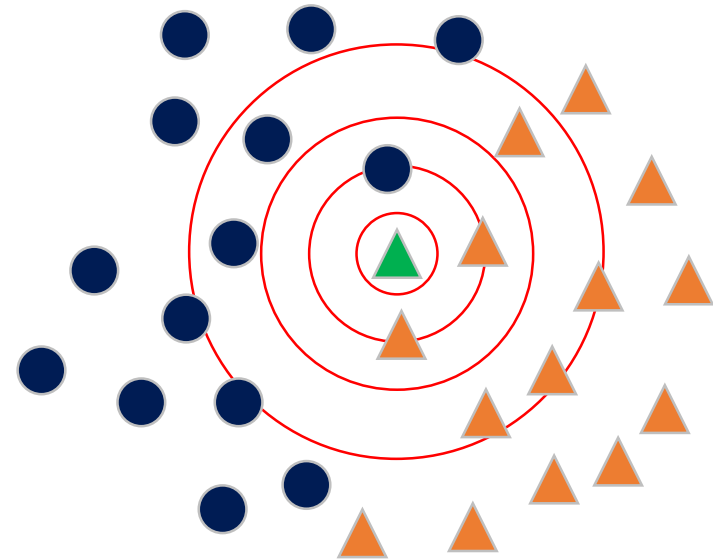
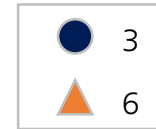


# K-nearest neighborhood

K=3

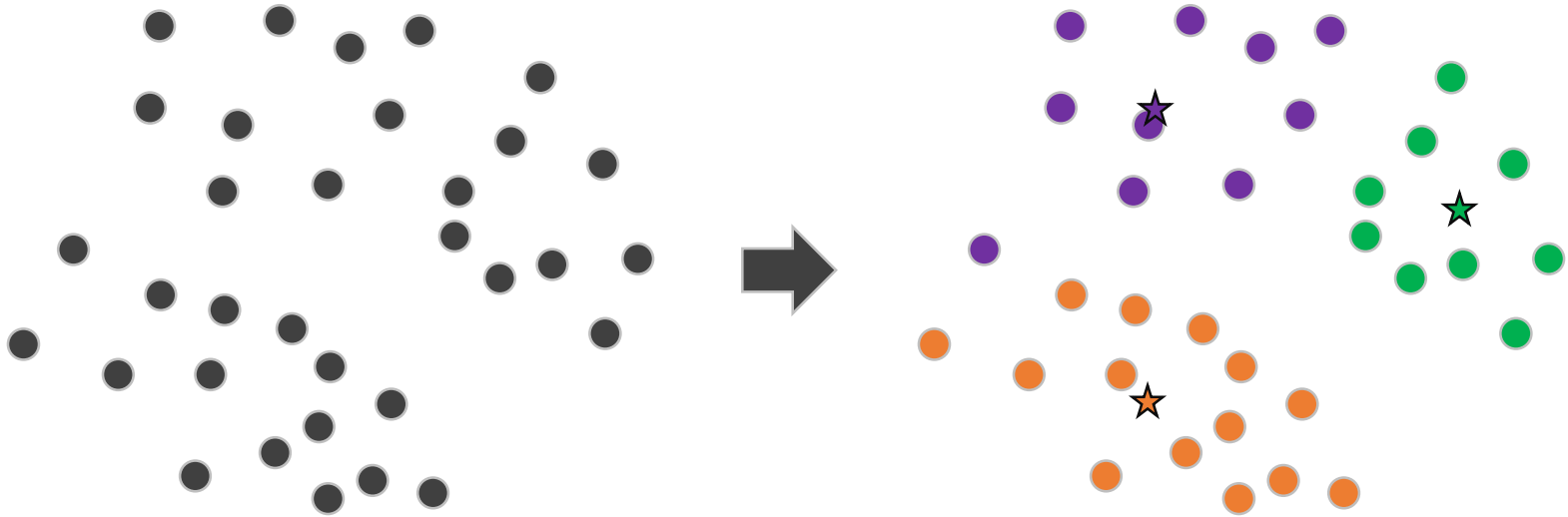


K=10



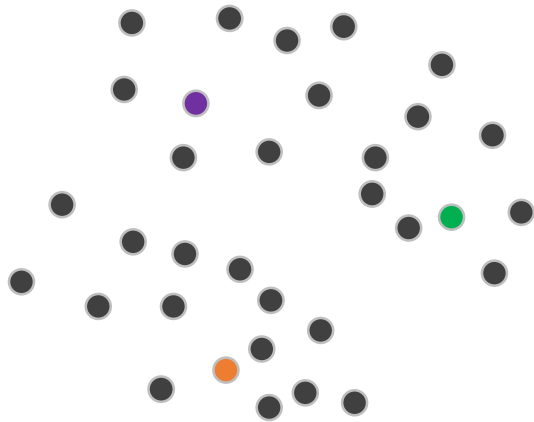
# K-means clustering (Overview)

- K-means clustering
  - Step 0: Assign random k sample as mean of cluster
  - Step 1: Calculate distance each sample and each mean of cluster
  - Step 2: Assign each sample in subsets which distance is the smallest
  - Step 3: Calculate the mean of k cluster
  - Step 4: Repeat step 1-3 until no change

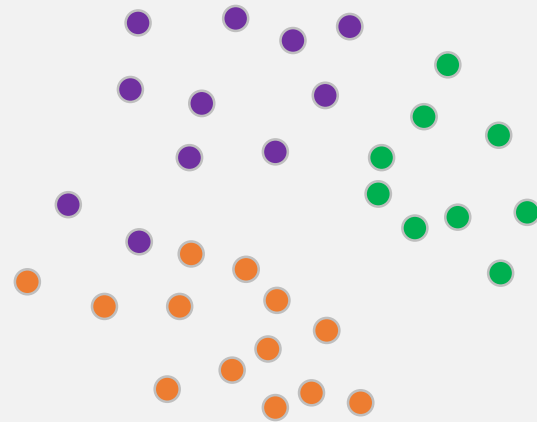


# K-means clustering

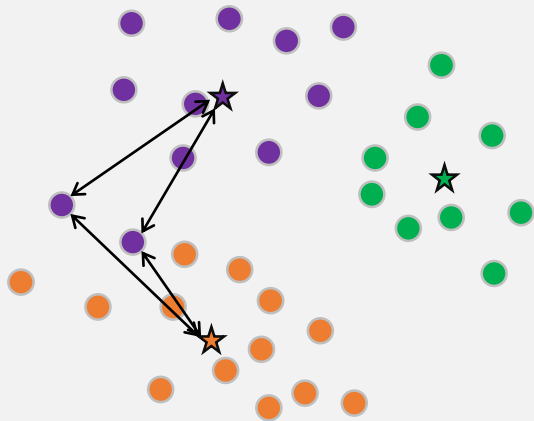
*Step 0*



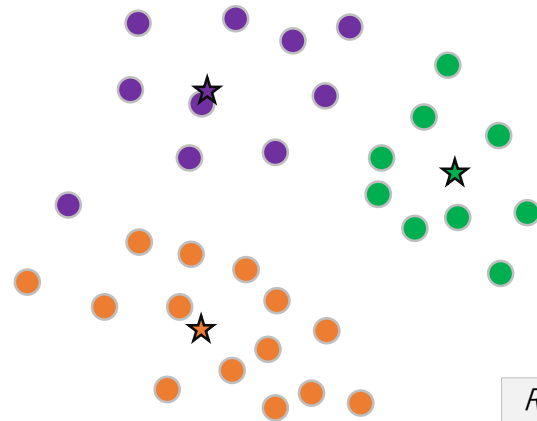
*Step 1-2*



*Step 3*



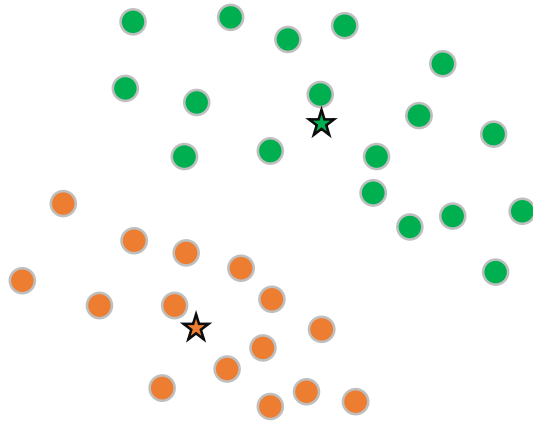
*Step 4*



*Repeated  
steps*

# K-means clustering

K=2



K=3

