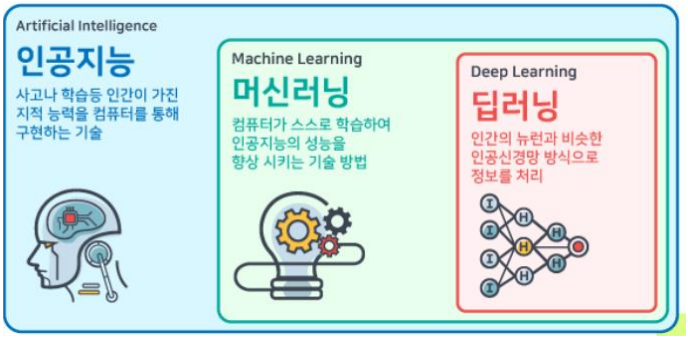
# [인공지능 개론]




## ○ 인공지능이란?

개념: 인간처럼 학습하고 생각할 수 있는 컴퓨터 시스템이다.

사전적 정의: 기계로부터 만들어진 지능 또는 시스템에 의해 만들어진 지능

## ○ 인공지능의 분류

〈기본 분류〉

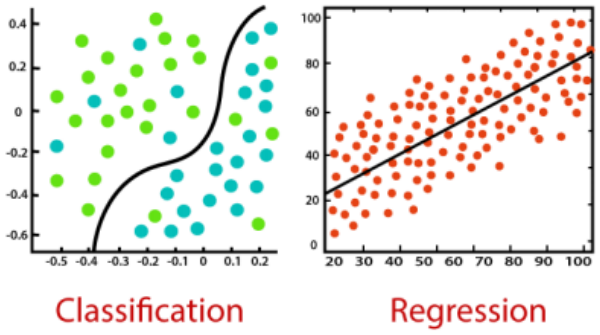
약인공지능: 특정 문제 해결을 목적으로 인간의 지능을 기계적으로 일부 모방에 구현한 인공지능

강인공지능: 특정 문제 해결을 넘어 사람처럼 생각하고, 경험해보지 않은 문제도 해결할 수 있는 수준의 인공지능

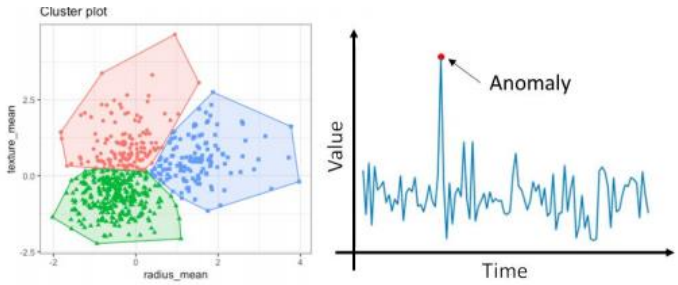
초인공지능: 인간의 지능을 넘어서 훨씬 더 높은 지능을 갖도록 구현한 인공지능

〈Task에 따른 분류〉

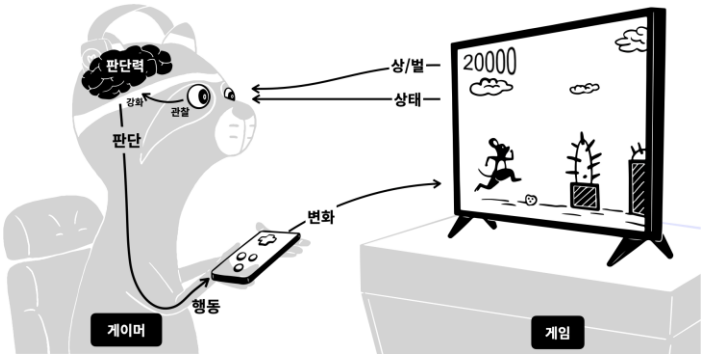
지도학습 (Supervised learning): 훈련 데이터에 원하는 타겟이 포함되어 있으며 데이터를 바탕으로 분류(classification)나 회귀(regression)시 사용한다. 분류의 경우 타겟이 범주형 혹은 이산적으로 나타나게 되고 회귀의 경우 타겟이 연속형이다.



비지도학습 (Unsupervised learning): 훈련 데이터에 원하는 출력이 포함되어있지 않으며 군집화(Clustering), 이상치 탐지(anomaly detection), 가짜 데이터 생성시 활용한다.




강화학습 (Reinforce learning): 데이터를 바탕으로 스스로 판단해 급작스러운 변화에 대응할 수 있도록 진행되는 학습이다. 고정된 dataset이 제공되기보단 환경이 제공되며 행동의 결과로 reward를 주는 방식으로 이루어진다.




## ○ 인공지능의 학습을 위한 단계

1. 문제설정

    - Task 결정: classification, regression, detection, segmentation, data generation

2. 데이터 레이블링(labeling) : 정답 데이터 생성

3. 데이터 전처리(preprocessing): Normalization, De-noising, feature extraction

4. 모델 학습: Training – validation, Tuning hyperparameter

5. 평가

    - Classification: accuracy, sensitivity, specificity, positive/negative predictive value, F score, Precision@10

    - Regression: mean square error, mean absolute error, correlation coefficient

    - Segmentation: Intersection of union, mean average precision, mean average recall

## ○ 데이터 종류

정형 데이터: 구글 스프레드시트 또는 마이크로소프트 엑셀과 같은 스프레드시트 프로그램에 표시할 수 있는 모든 데이터는 정형데이터이다. 해당 데이터는 행과 열로 표시될 수 있으며 각 행에는 단일 인스턴스의 속성과 연결된 데이터가 있고 각 열에는 다른 속성을 나타낸다.

비정형 데이터: 정형 데이터가 아닌 모든 데이터는 비정형 데이터로 분류될 수 있다. 현재 우리가 접하는 데이터의 80%가 텍스트, 오디오, 이미지 또는 동영상의 형태가 비정형 데이터이다.

## ○ 인공지능 학습 평가지표

| | | 실 제 | |
|---|---|---|---|
| | | True<br>(참) | False<br>(거짓) |
| 판<br>정 | Positive<br>(양성으로 판정) | TP(True Positive)<br>양성으로 판정했는데 실제로 양성인 경우 | FP(False Positive)<br>양성으로 판정했는데 실제로 음성인 경우<br>→ 음성을 양성으로 판정 |
| | Negative<br>(음성으로 판정) | TN(True Negative)<br>음성으로 판정했는데 실제로 음성인 경우 | FN(False Negative)<br>음성으로 판정했는데 실제로 양성인 경우<br>→ 양성을 음성으로 판정 |

정확도(accuracy): (TP+TN)/(TP+TN+FP+FN)

민감도(sensitivity): TP/(TP+FN) = 재현율(recall)

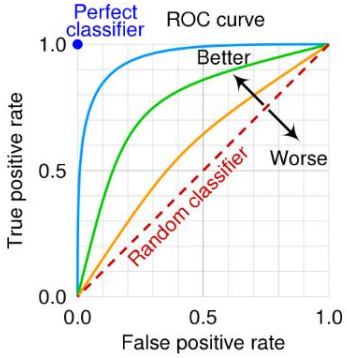
특이도(specificity): TN/(TN+FP)

양성예측도(positive predictive value): TP/(TP+FP) = 정밀도(precision)

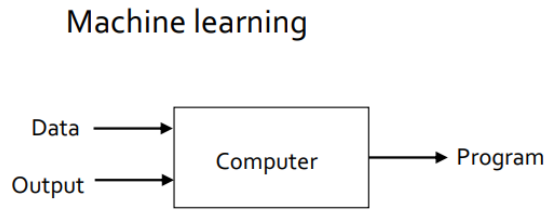
음성예측도(negative predictive value): TN/(TN+FN)

**〈ROC (Receiver operating characteristic) curve〉**

Threshold에 따른 TPR(=Recall)과 FPR(=Fall-out)을 나타낸 그래프이다. 대각선을 기준으로 좌상단으로 붙은 ROC curve 일수록 좋은 분류 성능을 나타낸다. ROC curve의 면적을 계산한 ROC-AUC (Area under the curve)를 통해 ROC curve를 정량적으로 나타낼 수 있으며 일반적으로 분류 모델 성능을 확인할 때 사용한다.

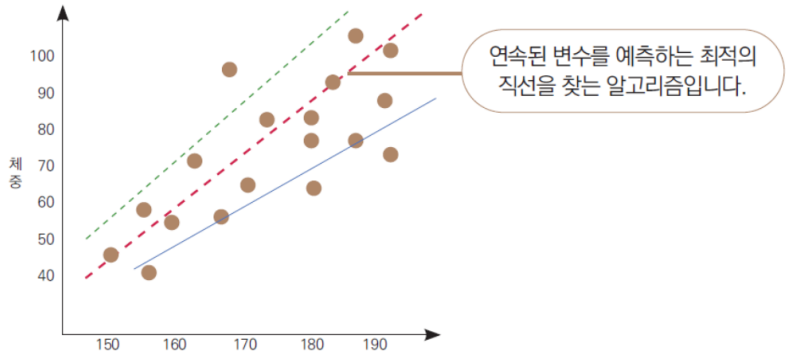


## ○ 머신러닝 모델의 종류



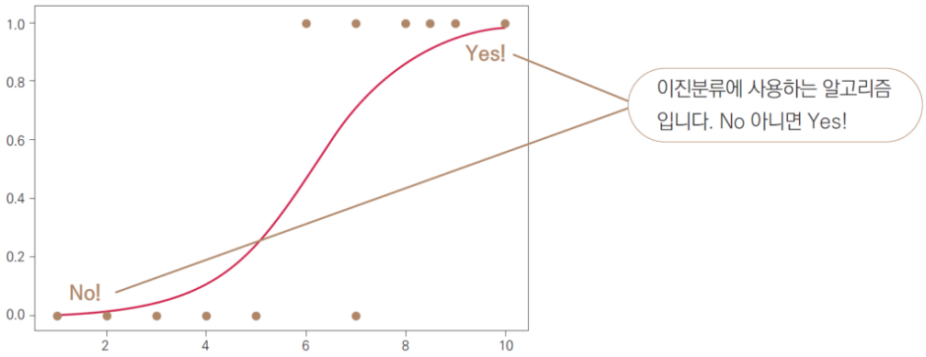

–

〈Linear regression (선형 회귀)〉



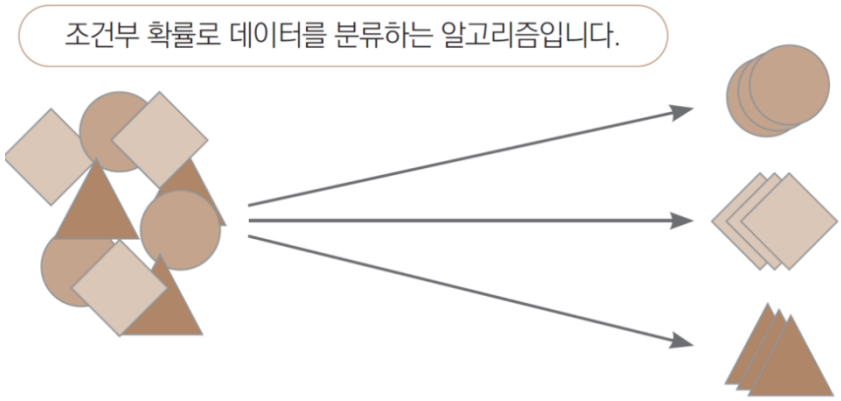

**선형 회귀는 가장 기초적인 머신러닝 모델입니다. 여러가지 데이터를 활용하여 연속형 변수인 목표 변수를 예측해 내는 것이 목적입니다.** 예를 들어 몸무게, 나이, BMI, 성별 등을 데이터로 활용하여 키와 같은 연속형 변수를 예측하는 겁니다. 연속형 변수는 165.5cm, 172.3cm, 182.9cm와 같이 연속적으로 이어질 수 있는 변수를 의미합니다. 반면 남성/여성으로 구분되는 성별은 연속형 변수가 아닙니다. 선형 회귀 모델에서는 예측할 종속변수만 연속형 변수면 족합니다. 예측하는 데 사용되는 그외 변수들은 연속형일 필요는 없습니다.

〈Logistic regression〉




로지스틱 회귀 또한 선형 회귀처럼 기본 분석 모델입니다. 로지스틱 회귀 분석은 알고리즘의 근간을 선형 회귀 분석에 두고 있어서 선형 회귀 분석과 상당히 유사하지만 다루는 문제가 다릅니다. 선형 회귀 분석은 연속된 변수를 예측하는 반면, 로지스틱 회귀 분석은 Yes/No처럼 두 가지로 나뉘는 분류 문제를 다룹니다.

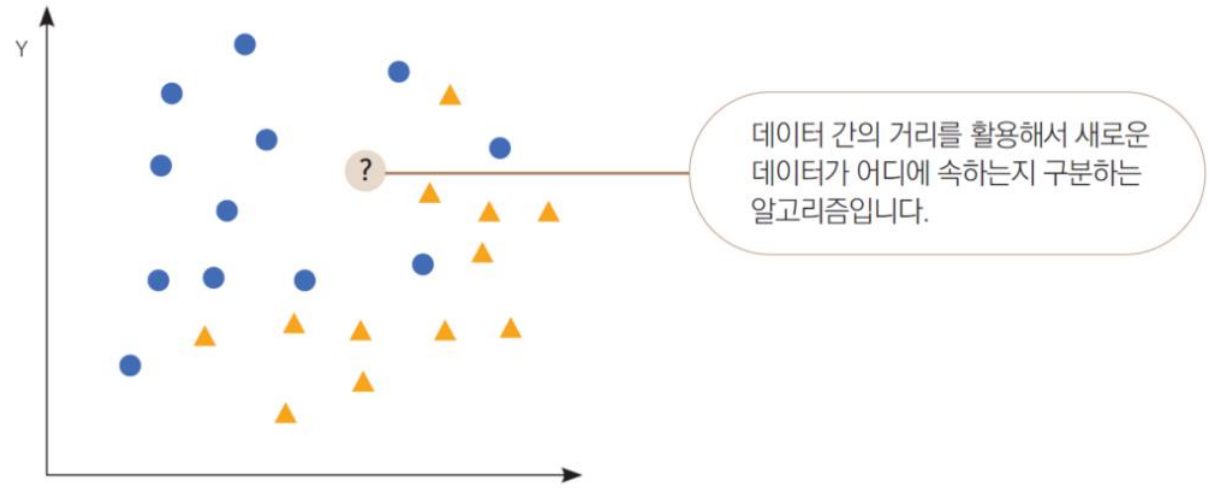
〈나이브 베이즈 (Naïve Bayes)〉




나이브 베이즈는 베이즈 정리를 적용한 조건부 확률 기반의 분류 모델입니다. 여기서 조건부 확률은 A가 일어났을 때 B가 일어날 확률을 의미합니다. 예를 들어 '무료라는 단어가 들어 있을 때 해당 메일이 스팸일 확률' 같은 겁니다. 이러한 특징으로 스팸 필터링을 위한 대표적인 모델로 꼽힙니다. 최근에는 딥러닝 같은 대안이 있어서 나이브 베이즈 모델을 쓰고자 하는 상황이 많지는 않습니다만, 그래도 스팸 메일 필터처럼 자연어 처리가 목적일 때는 여전히 나이브 베이즈 모델이 좋은 선택이 될 수 있습니다(딥러닝이 자연어 처리에 더 탁월한 모습을 보여주지만, 딥러닝보다 간단한 방법으로 자연어 처리를 원할 때).

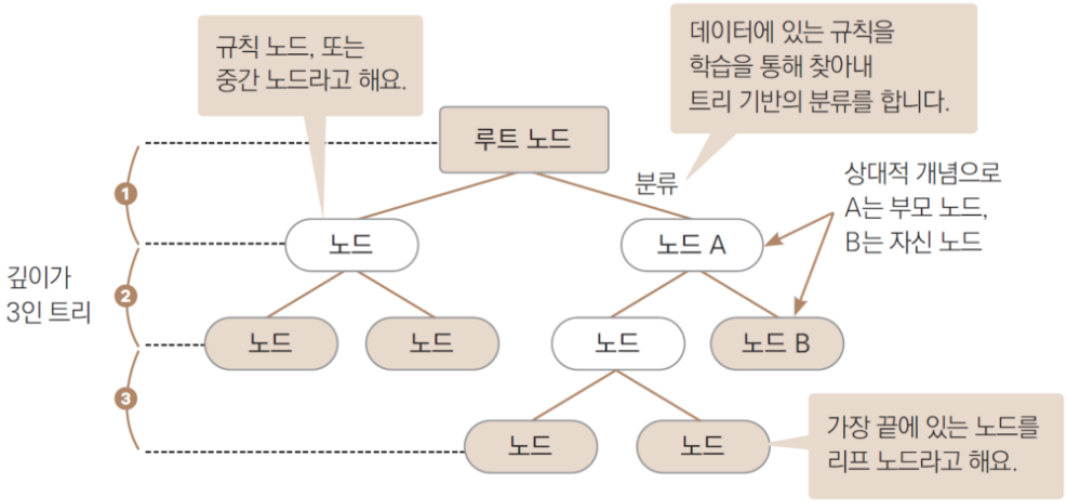
〈K-means clustering〉

K-평균 군집화는 비지도 학습의 대표적인 알고리즘 중으로 목표 변수가 없는 상태에서 데이터를 비슷한 유형끼리 묶어내는 머신러닝 기법입니다.K-최근접 이웃 알고리즘과 비슷하게 거리 기반으로 작동하며 적절한 K값을 사용자가 지정해야 합니다. 거리 기반으로 작동하기 때문에 데이터 위치가 가까운 데이터끼리 한 그룹으로 묶습니다. 이때 전체 그룹의 수는 사용자가 지정한 K개입니다.



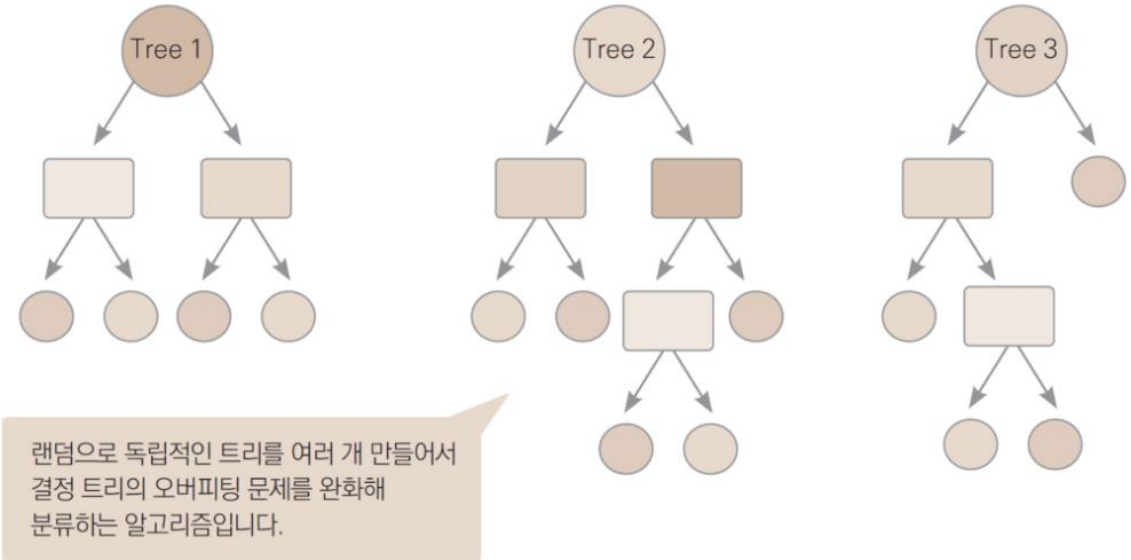
〈결정 트리(Decision Tree)〉

결정 트리는 관측값과 목푯값을 연결시켜주는 예측 모델로서 나무 모양으로 데이터를 분류합니다. 수많은 트리 기반 모델의 기본 모델이 되는 중요 모델입니다. 트리 기반의 모델은 선형 모델과는 전혀 다른 특징을 가지는데, 선형 모델이 각 변수에 대한 기울기값들을 최적화하여 모델을 만들어나갔다면, 트리 모델에서는 각 변수의 특정 지점을 기준으로 데이터를 분류해가며 예측 모델을 만듭니다. 예를 들어 남자/여자로 나눠서 각 목푯값 평균치를 나눈다거나, 나이를 30세 이상/미만인 두 부류로 나눠서 평균치를 계산하는 방식으로 데이터를 무수하게 쪼개어나가고, 각 그룹에 대한 예측치를 만들어냅니다.



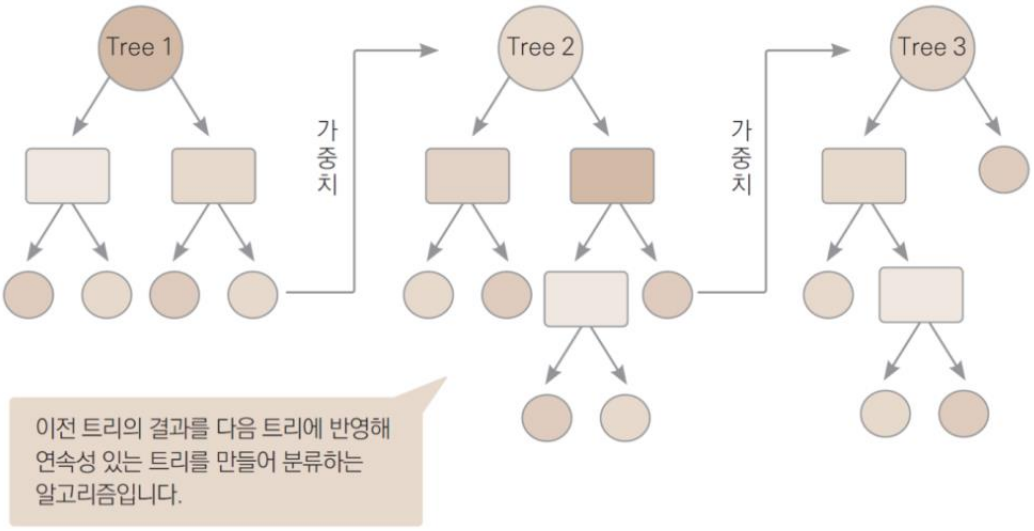

〈랜덤 포레스트(Random Forest)〉

랜덤 포레스트 모델은 결정 트리의 단점인 오버피팅 문제를 완화시켜주는 발전된 형태의 트리 모델입니다. 랜덤으로 생성된 무수히 많은 트리를 이용하여 예측을 하기 때문에 랜덤 포레스트라 불립니다. 이렇게 여러 모델(여기서는 결정 트리)을 활용하여 하나의 모델을 이루는 기법을 앙상블이라 부릅니다.



〈XG부스트(XGBoost)〉

랜덤 포레스트는 각 트리를 독립적으로 만드는 알고리즘입니다. 반면 부스팅은 순차적으로 트리를 만들어 이전 트리로부터 더 나은 트리를 만들어내는 알고리즘입니다. 부스팅 알고리즘은 트리 모델을 기반으로 한 최신 알고리즘 중 하나로, 랜덤 포레스트보다 훨씬 빠른 속도와 더 좋은 예측 능력을 보여줍니다. 이에 속하는 대표적인 알고리즘으로 XG부스트, 라이트GBM(LightGBM), 캣부스트(CatBoost) 등이 있습니다. 그중 XGBoost(eXtra Gradient Boost)가 가장 먼저 개발되기도 했고, 가장 널리 활용됩니다. XGBoost는 손실함수뿐만 아니라 모형 복잡도까지 고려합니다.




## ○ K means clustering 실습

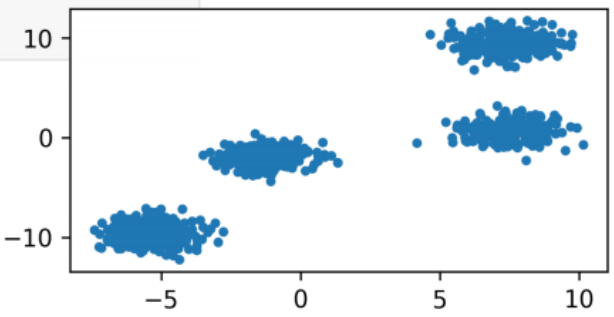
### 1. Package import & 샘플 생성

**1 pakages**

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
```

**2 샘플 생성**

```
np.random.seed(8)
x, y = make_blobs(n_samples = 1000, centers = 4, cluster_std = 0.9)

plt.figure(figsize = (4, 2), dpi = 300)
plt.scatter(x[:, 0], x[:, 1], marker = ".")
```



## 2. 학습 진행 & 학습 결과 확인

### 3 학습 진행

```
k_means = KMeans(init = "k-means++", n_clusters = 4, n_init = 12)
k_means.fit(x)
```

```
KMeans(n_clusters=4, n_init=12)
```

### 4 학습 결과 확인

```
k_means_labels = k_means.labels_
print('k_means_labels: ', k_means_labels)

k_means_cluster_centers = k_means.cluster_centers_
print('k_means_cluster_centers: ', k_means_cluster_centers)
```

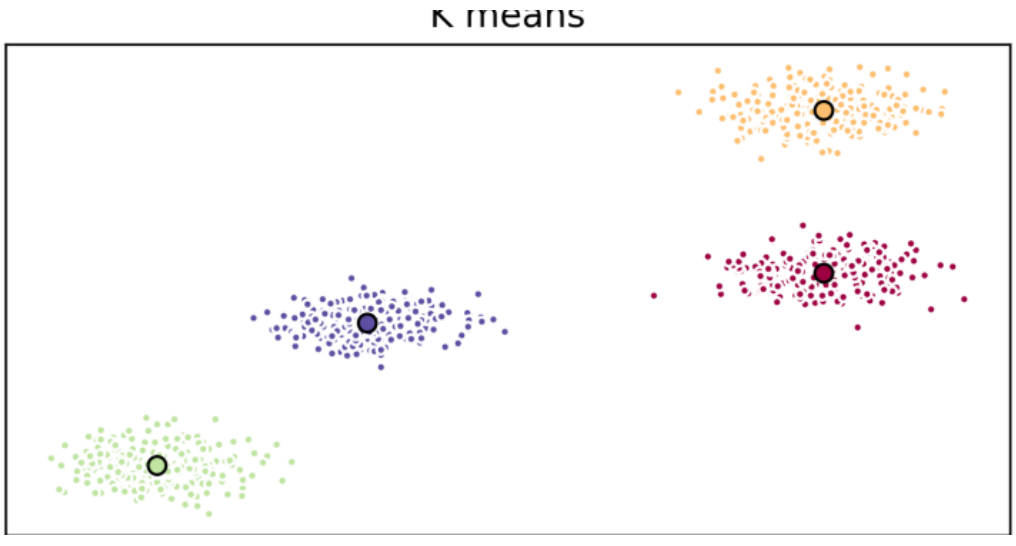
클러스터링 학습 결과

```
k_means_labels:  [0 1 3 2 0 0 3 2 3 2 2 2 0 0 3 1 3 1 1 0 1 0 0 0 2 3 3 2 0 2 1 3 1 0 1 0 0 0
 0 2 2 2 0 0 3 0 2 3 2 3 1 1 3 0 0 2 1 3 2 3 2 3 3 3 3 3 1 2 1 1 1 1 0 1 2
 3 3 3 3 1 3 0 0 3 2 1 3 3 1 2 0 0 1 0 1 1 2 3 2 0 3 2 3 1 1 0 2 1 1 2 1 2
 2 1 0 1 3 0 3 3 1 3 2 0 2 3 3 0 2 0 1 1 0 3 2 3 1 3 1 2 0 0 0 2 2 3 3 0 0
 3 1 1 3 3 2 2 3 0 2 2 1 2 3 0 0 3 3 1 0 1 2 3 3 0 2 0 2 0 1 3 1 3 2 0 3 3
 3 0 3 1 2 1 3 2 0 3 2 3 2 3 1 1 2 3 1 1 1 3 3 2 2 3 2 2 1 0 2 0 1 1 2 1 1
 0 1 3 2 0 1 0 2 2 2 3 3 0 3 0 2 1 0 3 1 2 2 0 3 3 2 2 1 1 3 1 3 2 2 0 2 3
 1 1 3 1 2 3 3 3 3 0 2 0 0 1 3 3 2 1 1 2 3 2 2 1 3 2 2 2 1 2 1 0 2 1 0 3 0
 2 1 0 1 1 3 0 2 3 0 2 3 3 2 0 2 3 0 1 2 3 1 3 0 0 3 3 0 0 0 3 1 0 0 2 0 1
 1 3 1 3 2 1 2 3 3 0 3 0 0 3 0 2 3 2 1 1 2 3 0 3 1 2 0 0 1 1 3 2 3 3 3 0 3
 1 3 0 1 0 1 2 0 3 3 1 1 0 1 2 2 1 0 2 1 1 3 0 0 2 2 3 3 1 3 2 1 0 0 3 3 1
 3 2 2 3 0 1 2 0 1 2 3 0 0 1 1 0 1 0 1 2 2 0 0 3 0 2 0 0 2 0 3 3 1 1 3 3 1
 2 0 1 2 0 0 1 2 3 1 2 3 2 1 2 1 0 0 1 3 2 0 0 1 1 1 0 2 0 1 1 2 0 3 3 0 3
 0 2 1 3 3 3 3 3 2 1 2 3 0 2 2 1 2 0 1 2 3 2 0 3 2 2 3 3 3 1 2 2 2 0 1 2 0
 1 0 0 0 3 2 1 1 2 0 3 1 1 0 1 0 0 3 0 2 0 0 0 0 2 3 1 2 2 0 3 1 1 2 3 0 3
 0 3 3 0 2 3 0 1 3 0 0 2 1 1 2 3 0 0 0 3 0 0 0 1 1 1 2 1 1 2 2 2 2 2 1 1 2
 2 0 3 1 3 0 3 2 2 3 2 1 1 2 3 0 0 3 2 0 0 1 3 2 1 2 2 2 1 0 2 1 0 0 2 1 1
 3 2 1 1 2 1 3 1 0 3 0 1 1 3 0 1 1 2 3 1 2 2 3 0 1 1 2 3 0 3 0 2 1 3 2 0 1
 0 0 1 2 3 1 2 3 2 3 3 2 1 1 1 2 1 0 0 0 3 1 2 1 0 0 0 0 3 2 0 1 2 0 0 0 0
 0 1 0 3 0 3 0 1 2 0 1 2 1 0 0 1 2 1 2 0 1 3 0 0 2 0 3 3 0 1 1 0 3 0 2 2 2
 0 2 1 3 3 2 1 3 3 1 3 1 3 1 0 1 0 3 1 1 3 3 2 2 2 3 3 2 3 1 1 0 1 0 2 2 0
 0 0 1 3 0 3 0 1 3 2 3 3 2 0 2 0 3 1 1 1 1 0 0 0 2 1 0 2 3 1 3 2 1 2 2 0 2
 3 2 2 3 2 3 1 3 3 2 2 2 3 2 3 2 2 1 3 3 2 2 3 2 3 0 2 2 0 1 1 1 3 1 3 2 3
 0 1 1 0 0 0 2 1 3 0 2 1 2 3 1 2 2 1 2 1 1 0 0 2 3 3 2 2 1 3 3 0 1 3 1 3 1
 2 0 2 0 3 3 1 0 2 0 2 0 0 3 1 2 1 2 1 3 3 3 2 3 0 0 1 3 1 0 0 2 1 2 3 2 1
 2 1 0 0 2 1 2 0 1 1 2 0 1 3 0 3 0 1 2 0 0 0 2 3 1 3 1 0 3 3 2 2 2 3 2 3 0
 0 3 0 2 0 0 3 1 1 0 3 2 0 0 3 1 2 1 1 1 1 3 2 1 0 2 1 2 3 3 1 3 0 1 1 2 1
 2]
k_means_cluster_centers:  [[ 7.43677253  0.6966818 ]
 [ 7.44207714  9.40391707]
 [-5.35905231 -9.6306923 ]
 [-1.32469127 -1.9919531 ]]
```

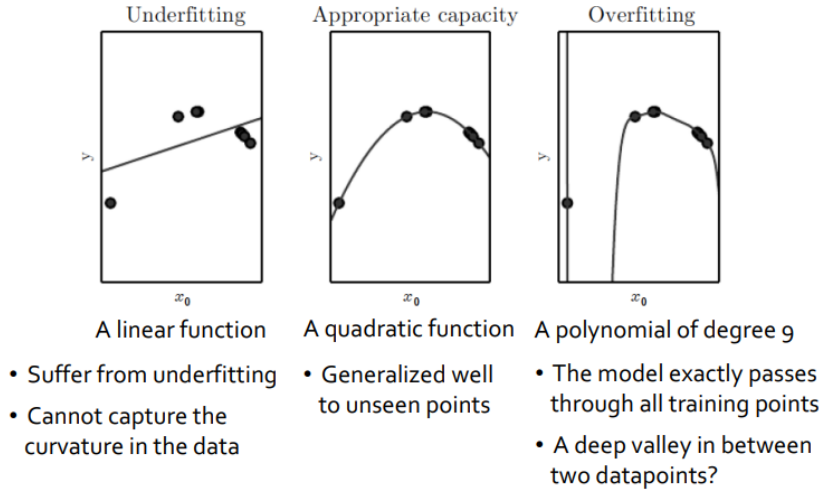
## 3. K means clustering 시각화

### 5 K means clustering 시각화

```
fig = plt.figure(figsize = (6, 4), dpi = 300)

colors = plt.cm.Spectral(np.linspace(0, 1, len(set(k_means_labels))))

ax = fig.add_subplot(1, 1, 1)

k = 0
ax.plot(x[(k_means_labels == k), 0], x[(k_means_labels == k), 1], '*', markerfacecolor = colors[0], marker = '.')
ax.plot(k_means_cluster_centers[k][0], k_means_cluster_centers[k][1], 'o',
        markerfacecolor = colors[0], markeredgecolor = 'k', markersize = 6)

k = 1
ax.plot(x[(k_means_labels == k), 0], x[(k_means_labels == k), 1], '*', markerfacecolor = colors[1], marker = '.')
ax.plot(k_means_cluster_centers[k][0], k_means_cluster_centers[k][1], 'o',
        markerfacecolor = colors[1], markeredgecolor = 'k', markersize = 6)

k = 2
ax.plot(x[(k_means_labels == k), 0], x[(k_means_labels == k), 1], '*', markerfacecolor = colors[2], marker = '.')
ax.plot(k_means_cluster_centers[k][0], k_means_cluster_centers[k][1], 'o',
        markerfacecolor = colors[2], markeredgecolor = 'k', markersize = 6)

k = 3
ax.plot(x[(k_means_labels == k), 0], x[(k_means_labels == k), 1], '*', markerfacecolor = colors[3], marker = '.')
ax.plot(k_means_cluster_centers[k][0], k_means_cluster_centers[k][1], 'o',
        markerfacecolor = colors[3], markeredgecolor = 'k', markersize = 6)

ax.set_title('K means')
ax.set_xticks(())
ax.set_yticks(())
plt.show()
```

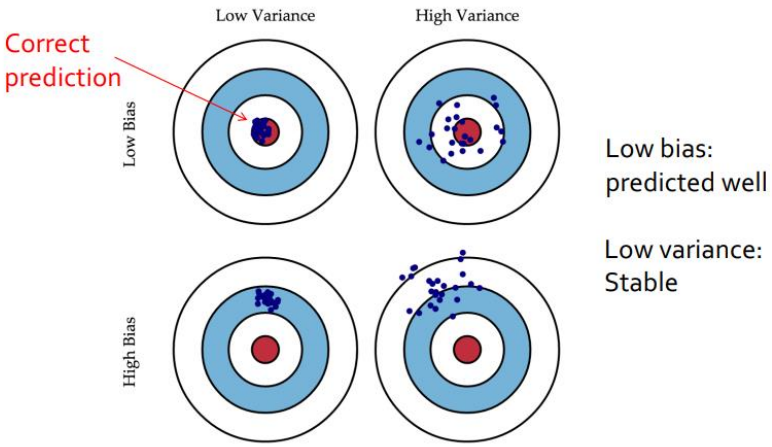


## ○ Overfitting and underfitting




위 이미지에서 왼쪽 그래프와 같이 너무 단순한 모델을 생성하여 학습 데이터와 잘 맞지 않을 때 모델이 **과소적합(Underfitting)**되었다고 한다. 이럴 때는 모델의 복잡도를 늘려줌으로써 문제를 해결해야 합니다. 반대로 오른쪽처럼 너무 복잡한 모델을 생성하는 바람에 학습 데이터에는 굉장히 잘 맞지만 새로운 데이터에는 잘 맞지 않는 현상을 **과적합(Overfitting, 과대적합)**이라고 한다.

위에서 살펴본 바와 같이 왼쪽 그래프와 오른쪽 그래프는 시험 데이터와의 오차가 발생하는 이유가 다르다. 전자는 훈련 데이터도 제대로 근사를 못할 만큼 모델이 단순한 것이 문제였다. 이런 경우에 발생하는 오차를 **편향(Bias)**이라고 한다. 후자는 훈련 데이터에 대해서는 근사를 매우 잘하지만 새로운 데이터가 들어왔을 때 제대로 근사하지 못한다는 문제가 있었다. 이런 경우에 발생하는 오차는 **분산(Variance)**이라고 한다.

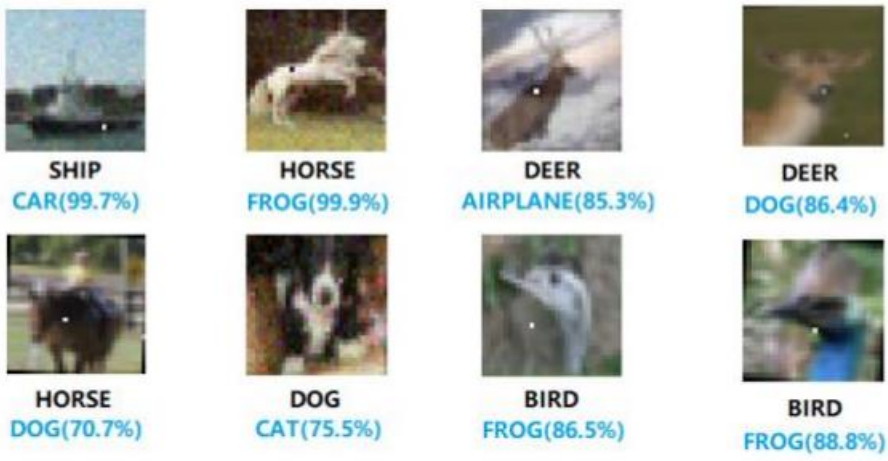



고분산(High variance)은 과적합(overfitting)을 의미한다

- 모델 클래스가 불안정하다
- 모델 복잡도가 증가하면 분산이 증가한다
- 더 많은 훈련 데이터를 통해 분산이 감소한다.

## ○ 인공지능의 한계점

〈블랙박스 알고리즘〉




흔히 AI 기반 학습 알고리즘은 설명이 가능하지 않고 블랙박스 형태라는 단점이 존재한다. High risk 결정에서는 설명력도 정확도만큼이나 중요해졌으며, Saliency map, SHAP (Explainable AI, XAI)와 같이 post-hoc explainability를 제공하는 기술이 생겼다.

그러므로 인공지능 연구를 진행할 때는 알고리즘의 설명력, 편향, 신뢰의 문제에 주의를 기울여야한다. 블랙박스 알고리즘이 실제 사회에서 사용되기 위해서는 많은 경우 설명력 보강이 필요하며, 노이즈와 데이터 가변성에도 대처 가능한 알고리즘을 개발하도록 노력해야한다. AI가 다양한 사회 서비스에서 인간 결정을 돕거나 대체함에 따라 윤리적 의사결정이 확보되도록 점검해야한다.