- regression, classification을 위한 많은 통계적 기법 중,
- Linear Regression, Logistic Regression 등은 기본적인 모델 형태를 갖춘 형태에서
  데이터에 따라 모델의 파라미터를 조정하는 방식으로 모델을 형성하였음

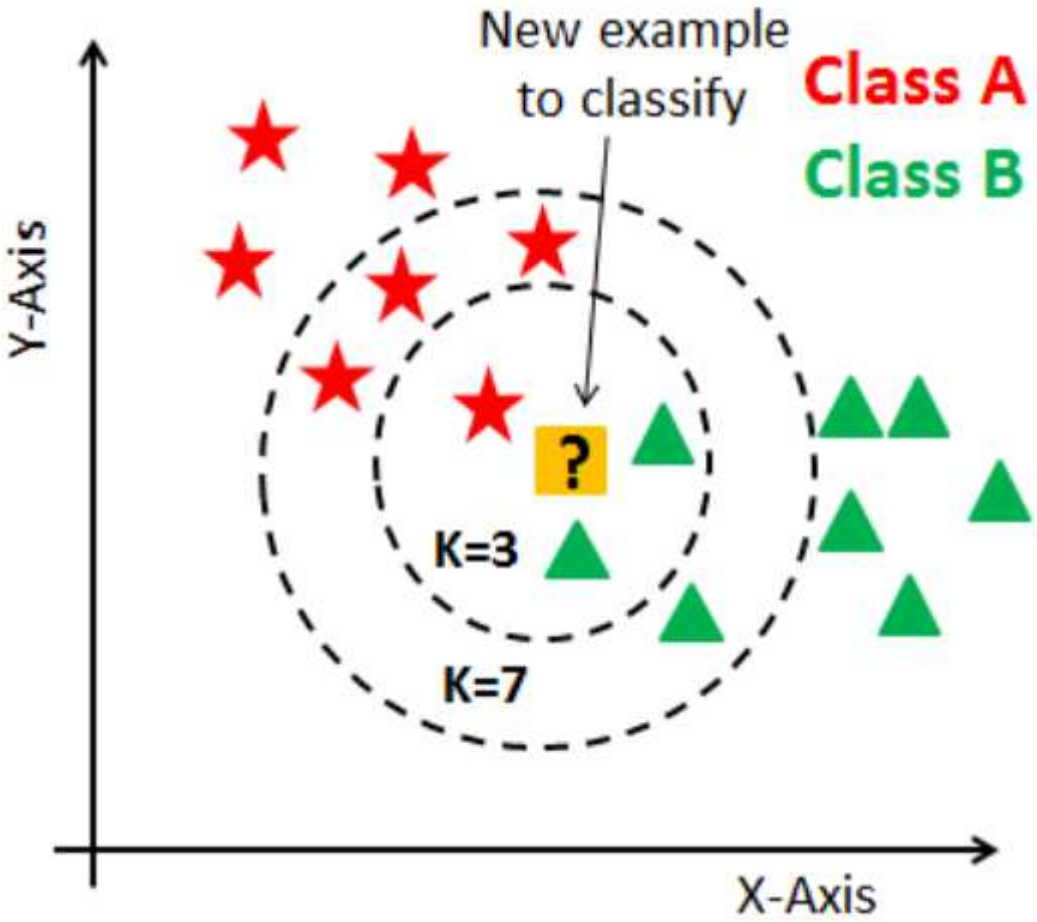
$$Y = b_0 + b_1X_1 + b_2X_2 + ... + b_pX_p + e$$

- 반면에 KNN, decision tree, 딥러닝 모델 등은 전체 모델 형태로 인한 한계가 적고, 데이터를 중심으로 유동적임

- 데이터를 중심으로 한다는 점에서,
  기존 통계에서는 데이터가 어떤 분포(정규 분포, t-분포 등)를 가지고 있는지 추론하는 과정을 전제한 후에 모수 추정을 수행했으나, bootstrap 기법은 분포 추론없이 바로 모수 추정을 진행할 수 있는 것과 유사

K-Nearest Neighbors

- 다른 값과의 유사도를 비교하여 classification / prediction 수행
- (classification) 주변 값과 비교하여, 주변 값들의 클래스가 차지하는 비율을 바탕으로 class 분류
- (prediction) 주변 값들이 공유하는 평균 값으로 예측(KNN regression)
- 학습데이터보다 변수의 개수가 많은 경우 성능이 크게 떨어짐
  '특성 선택' 및 '차원 축소 기법'을 사용하여 변수의 개수를 줄이거나 다른 알고리즘을 사용해야 함



유사도(similarity, nearness) 측정

- 두 instance vector 간 거리를 distance metric을 이용하여 계산

two records $(x_1, x_2, \ldots, x_p)$ and $(u_1, u_2, \ldots, u_p)$

- euclidean distance: 두 벡터 사이의 직선 거리

$$\sqrt{(x_1-u_1)^2+(x_2-u_2)^2+\cdots+(x_p-u_p)^2}$$

- manhattan distance: 두 벡터의 각 요소간 길이의 합

$$|x_1-u_1|+|x_2-u_2|+\cdots+|x_p-u_p|$$

스케일러 사용

- 표준화(standardization): 변수의 값이 가지는 스케일 차이가 모델 학습에 과도하게 영향을 미치는 것을 막기 위해 사용
  일반적으로 학습 전에 데이터 표준화 수행
- 값의 크기는 변형하되 변수의 값 차이가 가졌던 거리 특성은 남아있도록 함
- 이 중 z-score 형태의 변환은 정규화(normalization)라고도 부름
- 변수(attribute) 마다 다른 스케일러를 적용하여 변수의 영향력을 임의로 조절할 수 있음
  StandardScaler, MinMaxScaler 등을 변수마다 다르게 적용하는 식

$$z=\frac{x-\bar{x}}{s}$$

K 고르기

- KNN의 성능을 결정하는데 가장 중요
- 1-nearest neighborhood classifier: k=1로 설정하여, 현재 데이터와 가장 유사한 하나를 찾음
- 일반적으로 k가 너무 작으면 데이터의 noise 영향을 많이 받아 overfitting될 수 있음
  overfitting 되는 경우 새로운 데이터에 대해서는 성능이 떨어짐
- 반면 너무 크면 KNN이 가지는 지역성이라는 이점을 잃음
- 절대적인 최선의 k 값은 정해져있지 않으며, 데이터의 성질에 영향을 많이 받음
  예를 들어 noise가 거의 없는 데이터에서는 k가 작을 수록 좋음

예측시 제일 처음으로 사용된다는거...

The new predictor variable (feature) that we add via KNN is the KNN predictor for each record (analogous to the realtors' comps). Since we are predicting a numerical value, the average of the K-Nearest Neighbors is used instead of a majority vote (known as KNN regression).

In [ ]:

```
# 데이터셋 불러오기
```

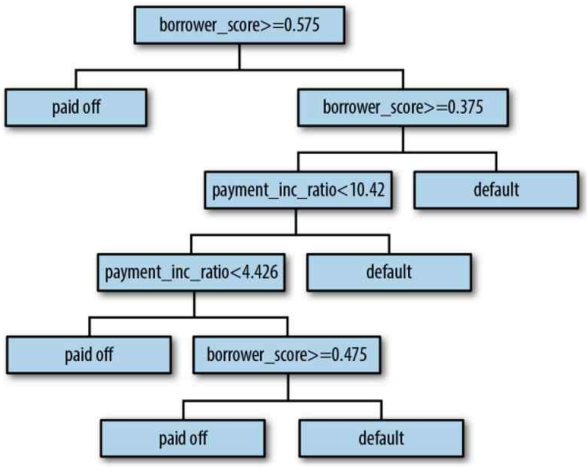
In [ ]:

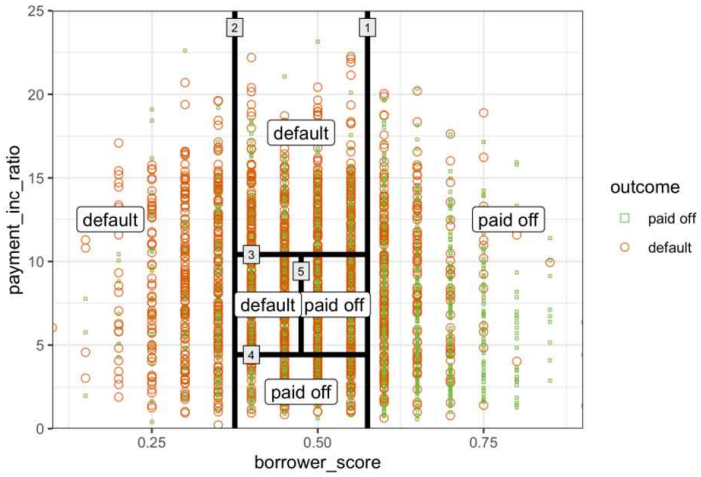
```
# knn classifier과 knn regressor
```

Tree Models

- classification / regrssion
- decision tree, tree, CART(Classification and Regression Tree)라고도 불림

- 일련의 "if-then-else"가 계층적으로 구조되어 있음(hierarchical tree)
  이 과정에서 변수간 interaction이 반영되며 simple dicision tree에서는 사용자도 이를 확인할 수 있음
- root: 첫번째 노드, leaf: 마지막 노드. 클래스가 분류되는 지점





recursive partitioning algorithm

- decision tree를 형성하기 위한 알고리즘
- straightforward and intuitive
- 변수(predictor)의 값을 기준으로, 비교적으로 높은 순도를 가진 부분(partition)을 나누는 작업

- 한 partition 내 class purity(homogeneity)를 측정하는 방법
  => 불순도(impurity)를 측정하여 대체 1) Geni impurity

$$Gini\,(t,\mathcal{D}) = 1 - \sum_{l \in levels(t)} P(t=l)^2$$

2) entropy

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

- 불순도가 낮으면서도 크기가 큰 partition을 만들 수 있는 변수 Xi과 split value si를 선택하여 노드를 형성

- 오버피팅을 막기 위해서 트리구조가 너무 깊어지지 않도록 함
  (min_samples_split, min_samples_leaf) 파티션이 너무 작으면 나누지 않음 (min_impurity_decrease)
  impurity가 유의하게 낮지 않으면 새로 파티션을 나누지 않음

regression

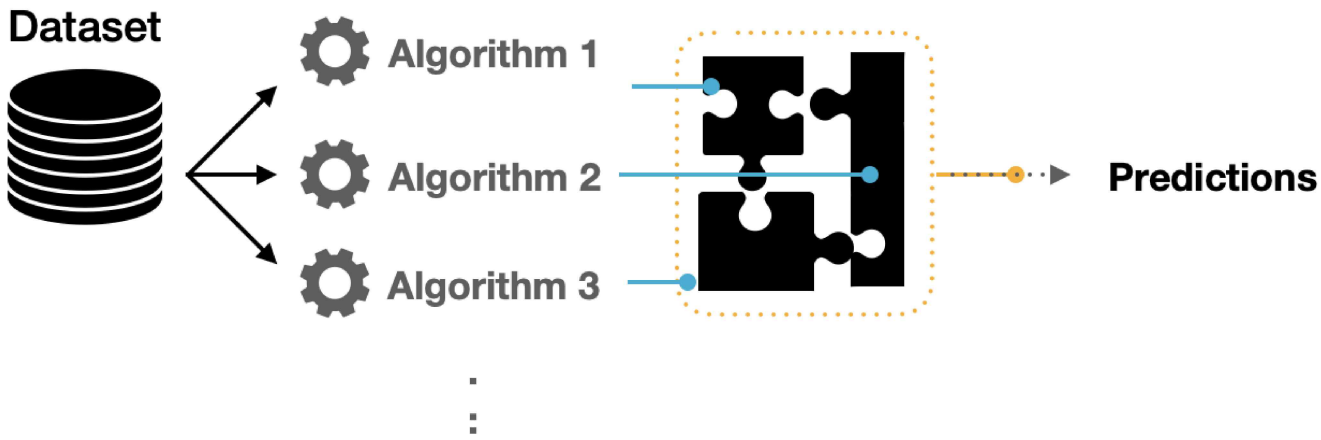
- predict continuous value
- impurity: 파티션 내 각 데이터와 파티션 평균의 차이를 이용하여 계산

In [ ]:

```
# Decision tree classifier, regressor 사용
```

Bagging and the Random Forest

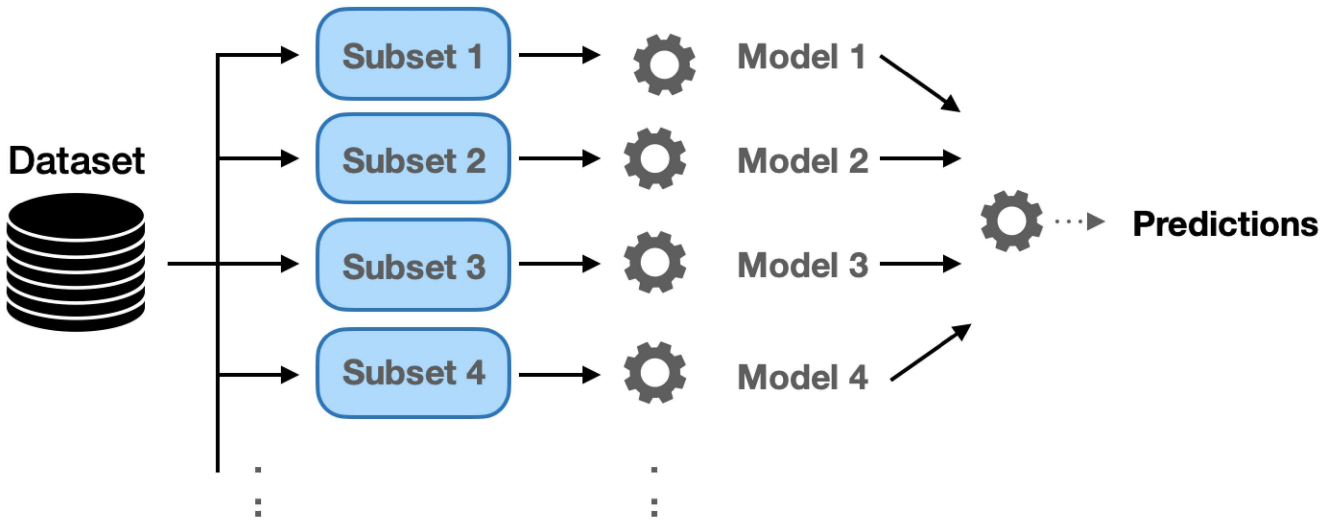
- ensemble: 여러개의 모델들의 결과물을 종합하여 최종 결론을 내리는 기법. 일반적으로 단일모델 보다 정확함
  1) 하나의 데이터셋으로 부터 여러개의 predictive model 학습
  2) 여러개의 predictive model의 결과 값들의 평균/weighted average/majority vote 등을 계산하여 최종 결과물로 사용



Bagging

- bootstrap aggregating
- 동일한 데이터셋에서 모델을 학습시키지 않고 / bootstrap resample에 학습시킴
- 각 predictor의 중요도와 관계가 반영되기 때문에,
  feature와 record가 많은 데이터셋에서 predictive model을 만들 때 도움이 됨
  1) 학습데이터로부터 복원 추출을 수행하여 bootstrap resample 생성(the bag. 이 과정에서 인스턴스가 중복될 수 있음)
  2) 새로만든 학습데이터 bootstrap resample에 모델 학습
  3) 여러개의 predictive model의 결과 값들의 평균/weighted average/majority vote 등을 계산하여 최종 결과물로 사용

$$\hat{f} = \frac{1}{M}\left(\hat{f}_1(\mathbf{X}) + \hat{f}_2(\mathbf{X}) + \cdots + \hat{f}_M(\mathbf{X})\right)$$



- random forest tree: decision tree + bagging

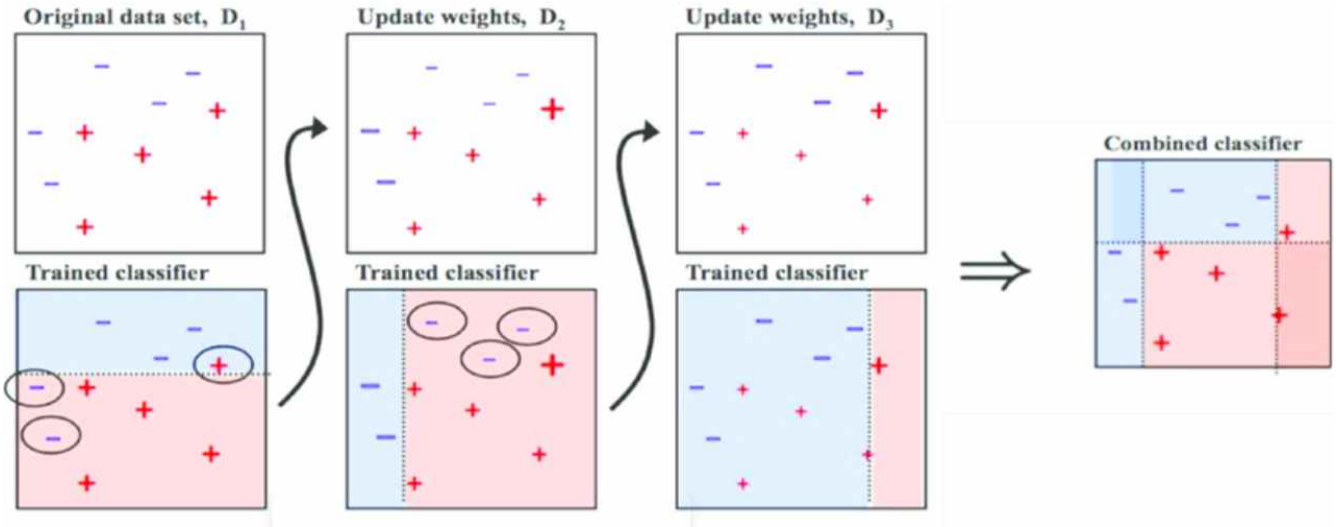
In [ ]:

```
# ensemble, BaggingClassifier, RandomForestClassifier
```

Boosting

- 병렬 형태로 모델을 학습/사용했던 Bagging과 달리 직렬 형태를 가짐(연속된 모델)
  1) 데이터에 대해 모델(1) 학습 수행 2) 모델(1)의 학습 결과를 반영하여 모델(2) 학습
  3) ...

- adaboost: 이전 모델이 분류 실패한 데이터에 대해 가중치를 조정하여 데이터 분포를 변형한 후 모델 학습
- gradient boost: 이전 모델의 residual을 고려하여 residual을 줄여나가는 방향으로 학습(오차 보정)



In [ ]:

```
# adaboost, gradient boost
```