Sigmoid : 2진 분류 => 0과 1사이의 값
Softmax : 다중 분류

DNN
-특정 데이터(도메인) 에서 학습된 구조를 다른 데이터(도메인)에서도 사용할 수 있음
=>전이학습

-문제점
1) Gradient가 소실되지 않는 Activation function을 사용해야 한다
=>즉, Squashing 특성이 없는 activation function인 Relu 를 사용한다
-Relu 는 음수이면 0 이고 음수가 아니면 0이 아닌값이다 (max(0, x))
-이 Relu를 변형한 activation function 은 ELU, Leaky Relu가 있다 (음수에 살짝 값을 준다)

2)
-층(Layer)을 많이 사용하면 과적합(overfitting) 발생
=>(모델 관점)학습 과정에서 일부러 특정 노드들에 연결된 가중치들은 업데이트를 하지 않으면 된다 어떤 노드를 업데이트 시키지 않을 것인가는 무작위로 그때그때 바꿔가며 결정한다

-사용하고 싶은 모델은 파라미터의 수가 엄청 많은데 보유하고 있는 데이터의 수가 너무 적어도 과적합이 발생한다
=>(데이터 관점)현재 보유한 데이터를 증폭한다 즉, Data Augmentation을 하는 것이다

-DNN의 특별한 형태인 CNN이 존재한다

<CNN>
-컬러 이미지는 3차원의 Tensor로 표현된다 (Red, Green, Blue)
ex) Raw image 는 [332, 344, 3] 로 표현되고 Red, Green, Blue는 각각 [332, 344, 1]의 형태이다

-이미지 데이터를 neural network에 학습시키려면 N1 = 400(width) * 700(height) * 3(RGB) = 840,000 개의 input 노드가 생기고 hidden node, weight node 까지 고려하면 엄청나게 많아서 불가능하다

-따라서, 픽셀 하나하나가 변수인데 변수의 수를 줄여야 한다
=>**Feature engineering(selection)** 기법: 도메인 지식, 기계학습(decision tree, PLS 등등), transform 기법을 이용하여 적은수의 feature을 생성해서 이것을 input으로 사용한다

-이미지의 경우 인접 변수(픽셀)간 높은 상관관계를 가진다 =>**Local correlation**
-이미지의 부분적 특성(ex. 눈, 귀)은 고정된 위치에 등장하지 않음 =>**invariant feature**

-위의 두개를 한꺼번에 실현시킬 수 있는 방법이 Convolution 이다

Convolution Neural Network
-이미지 데이터의 특성을 잘 반영할 수 있는 인공신경망 모델
-2D 혹은 3D 구조를 유지하면서 학습
-일반적인 CNN은 Convolution 연산, Activation 연산, Pooling 연산의 반복으로 구성된다

1) Convolution 연산
-변수(variable)은 맨 처음 raw data의 값이고 feature는 variable 들의 combination을 의미한다
-Stride가 커지면 이미지의 특징을 놓칠 가능성이 증가해서 성능이 약화될 가능성이 있으나 feature size가 작다 => 이미지가 굉장히 클 때는 stride가 조금 큰게 좋다

-실질적으로 입력데이터와 합성곱 연산을 할 filter의 값을 back-propagation으로 추정해야 하는 것임
-위의 합성곱 연산의 결과가 담겨져 있는 것을 "Feature map" 이라고 한다

2) Activation 연산 = Relu
-Convolution을 통해 학습된 값들의 비선형 변환

Padding
-가장자리에 있는 픽셀들은 중앙에 위치한 픽셀들에 비해 Convolution 연산이 적게 수행됨
=>원 이미지 테두리에 0의 값을 갖는 픽셀 추가, 사이즈 유지

3) Pooling 연산
-Feature map의 공간적 크기를 줄이자 => 일정 영역의 정보를 축약
-Max Pooling은 이미지 분석에서 주로 사용하고 Average Pooling도 존재한다

-여기까지 하게되면 Pooled Featured map이 촘촘하게 쌓여있다. 이것을 Flattening 과정을 거쳐서 2차원, 3차원의 행렬/텐서 구조를 1차원의 벡터로 변환한다

-이 1차원 벡터가 이제 우리가 알고있는 neural network의 input으로 사용한다

CNN의 하이퍼파라미터

1) Filter의 크기

2) Filter의 수 : Filter로부터 나온 Feature map의 개수임

3) Stride

4) zero padding

1 by 1 convolution의 역할 : feature map의 depth를 감소시켜 연산량을 감소시킴

Resnet

-152개 Layer를 사용하는 CNN 중에서 가장 획기적인 모델

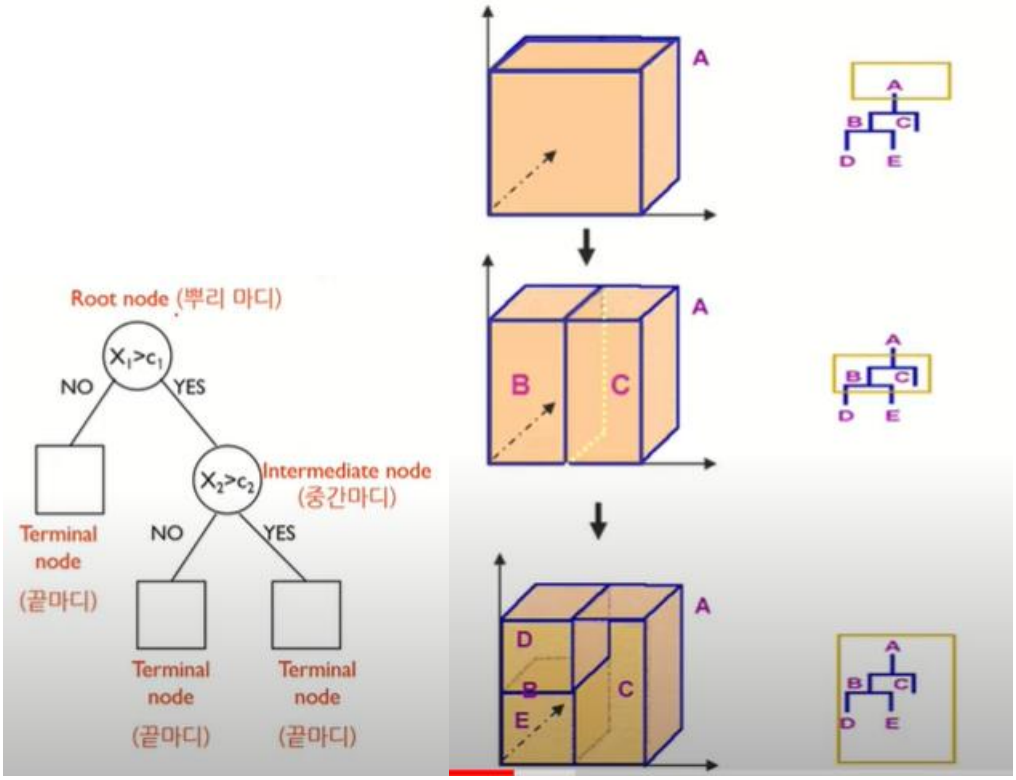
-152개 Layer에 대한 전통적인 문제인 Feature map의 값자체가 작아지는 것과 Gradiant(weight) 소실을 막기 위해 이전의 Feature 값을 더해줌으로써 Feature map의 값을 1 이상으로 계속 유지시켜준다

의사결정나무 모델

-데이터를 2개 혹은 그 이상의 부분집합으로 분할 => 데이터가 균일해지도록 분할

-분류 : 비슷한 범주를 갖고 있는 관측치끼리 모음

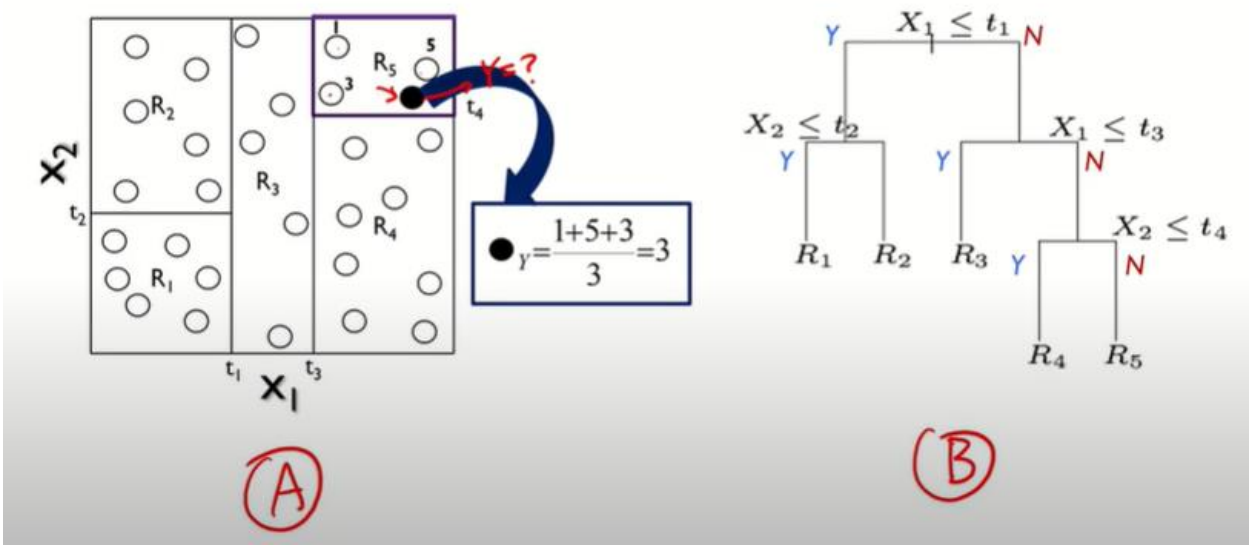
예측 : 비슷한 수치를 갖고 있는 관측지끼리 모음



=>끝마디개수3개

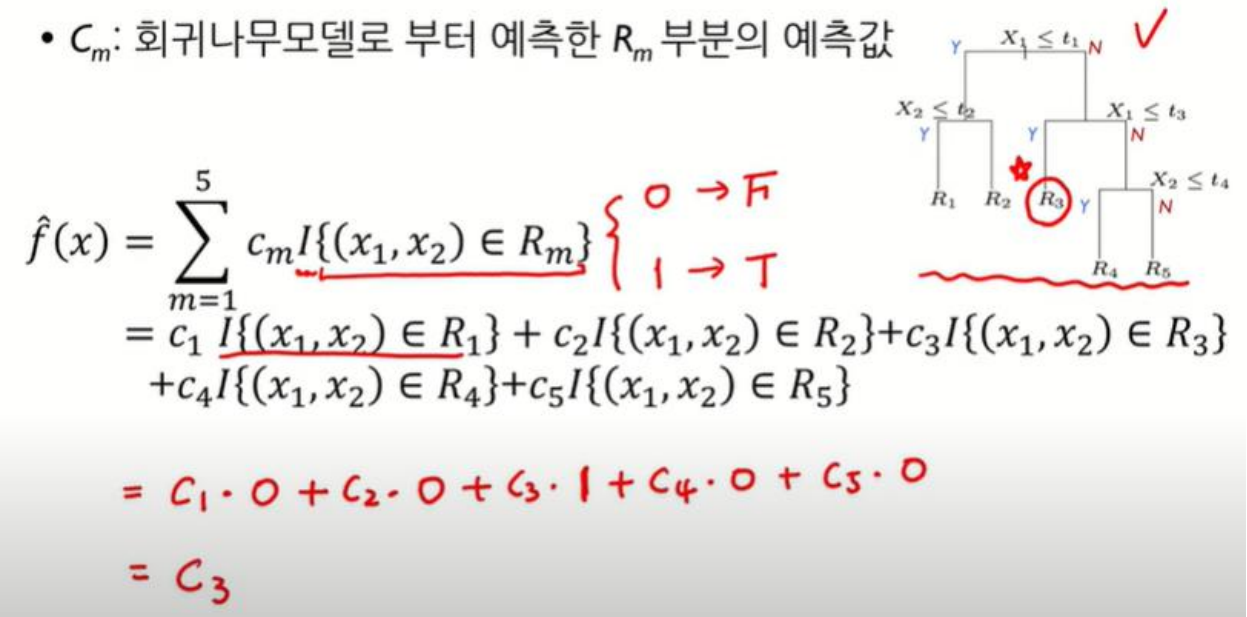
-의사결정나무는 예측나무 모델, 분류나무 모델에도 적용이 된다

1) **예측나무 모델**



-A와 B는 같은 의미이다. A의 끝마디는 5개, B의 끝마디도 5개로 같다

-R1 = 6개, R2=6개, R3=5개, R4 = 8개, R5 = 4개의 데이터를 가지고 있다



I : 중괄호안에 있는 것이 참이면 값이 1이고, 중괄호안에 있는 것이 거짓이면 값이 0이다

-(x1,x2) 가 R3 에 속한다고 가정하면 결과적으로 c3 이라는 값을 가진다

=>여기까지 나온 A, B, 수식은 모든 같은 의미를 가진다 (나타내는 표현만 다를 뿐)

-결과적으로 예측나무 모델에서 끝마디 마다 비슷한 값을 가지는 데이터끼리 모이게 된다

-예측나무 모델링 프로세스

1) 데이터를 M개 (끝마디가 M개) 로 분할 : R1, R2, ... RM

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

Cm : 회귀나무모델로부터 예측한 Rm부분의 예측값

2) 최상의 분할은 다음 **비용함수**를 최소로 할 때 얻어짐

$$= \min_{C_m} \sum_{i=1}^{N} (y_i - \sum_{m=1}^{M} c_m I(x \in R_m))^2$$

-실제 숫자 y와 f(x)로 부터 나온 y 값의 차이의 제곱의 합을 최소화 시키는 Cm ?

-어떤 데이터가 들어왔을 때 **해당 부분에 속해 있는 관측치들의 y값들의 평균으로 예측**하는게 가장 좋음

3) 맨 처음 최상위 분할의 분할변수(j)와 분할점(s)는 어떻게 결정?

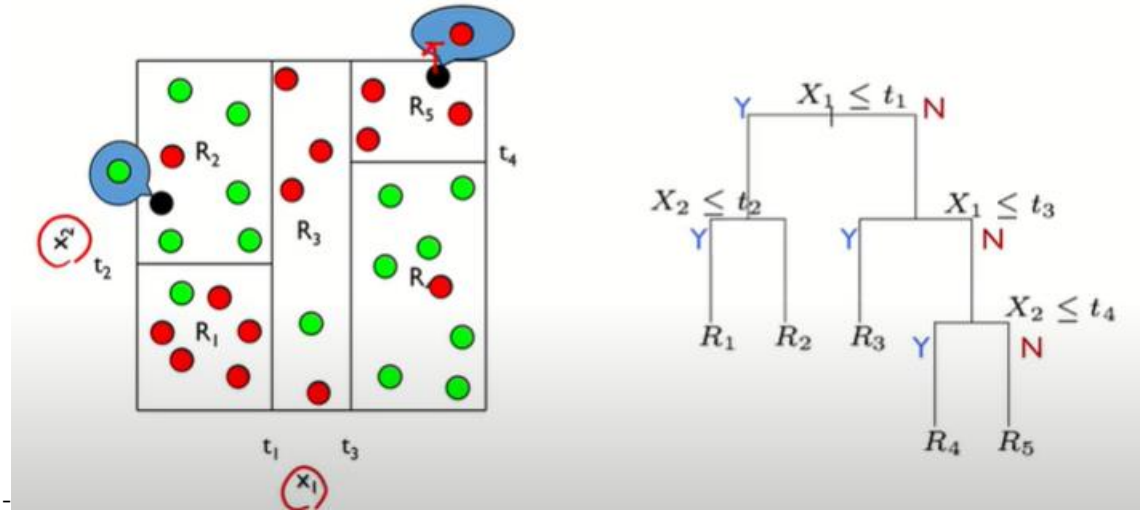
$$R_1(j,s) = \{x | x_j \leq s\}$$
$$R_2(j,s) = \{x | x_j > s\}$$

$$\underset{j,s}{\operatorname{argmin}} \left[\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2 \right]$$

-방법은 예로 들어 x1 = (2, 3), x2 = (5, 6), x3 = (1, 7) 이 있으면 x1 >=2, x1>=3, x2>=5, x2>=6, x3>=1, x3>=7 을 각각 해보고 가장 작은 값을 가지는 분할변수, 분할점을 구하면 된다

-Cm : 해당 끝마디에 있는 y값의 평균

2) 분류나무 모델



-각 관측치 마다 반응변수 값 yi = 1, 2, ... k, 즉 K개의 클래스(범주)가 존재

-Rm : 끝노드 m에 해당하며 Nm 관측치 개수를 가지고 있음

$\hat{p}_{mk}$: 끝노드 m에서 k 클래스에 속해 있는 관측치의 비율

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

=>즉, 특정 끝노드에 속한 노드들이 속한 클래스중에서 가장 높은 확률을 가지는 클래스를 의미

$$\hat{f}(x) = \sum_{m=1}^{5} k(m) I\{(x_1, x_2) \in R_m\}$$

$$= k(1)I\{(x_1, x_2) \in R_1\} + k(2)I\{(x_1, x_2) \in R_2\} + k(3)I\{(x_1, x_2) \in R_3\}$$

$$+ k(4)I\{(x_1, x_2) \in R_4\} + k(5)I\{(x_1, x_2) \in R_5\}$$

$$k(m) = \underset{k}{\operatorname{argmax}}\, \hat{p}_{mk}$$

=>제일 큰 확률을 가진 클래스를 내뱉는다

분류 모델에서의 비용 함수(불순도 측정)

1) Misclassification rate: $\frac{1}{N_m}\sum_{i\in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{(mk)m}$

2) Gini Index: $\sum_{k\neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})$

3) Cross-entropy : $-\sum_{k=1}^{K} \hat{p}_{mk} log\hat{p}_{mk}$

분할변수 (j) 와 분할점 (s) 은 어떻게 결정할까?

-분할 변수와 분할 기준은 목표변수(y)의 분포를 가장 잘 구별해주는 쪽으로 정함 (균일하게)

-목표변수(y)의 분포를 잘 구별해주는 측도로 순수도 또는 불순도를 정의

-예를 들어 클래스 0과 클래스 1의 비율이 45% 와 55% 인 노드는 각 클래스의 비율이 90%와 10% 인 마디에 비하여 순수도가 낮다(불순도가 높다) 라고 해석

-각 노드에서 분할 변수와 분할점의 설정은 **불순도의 감소가 최대가 되도록** 선택

정보 획득량 : 특정 변수를 사용했을 때 엔트로피 감소량

$$IG(S,A) = Entropy(S) - \sum_{v\in value(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

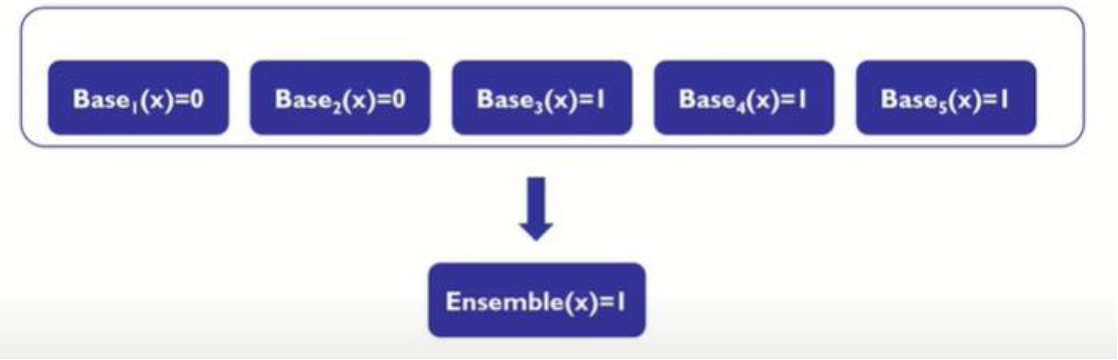
개별 트리 모델의 단점

-나무의 최종노드 개수를 늘리면 과적합 위험

랜덤 포레스트 모델

-앙상블 모델의 하나의 예시

-여러 Base 모델들의 예측을 다수결 법칙 또는 평균을 이용해 통합하여 예측 정확성을 향상시키는 방법



-Base 모델들이 서로 독립적이어야 앙상블 모델은 Base 모델보다 우수한 성능을 보여줌

-Base 모델의 성능이 무작위 모델 (확률 : 0.5) 보다는 좋아야 함

-랜덤 포레스트 모델은 Base 모델로 의사결정 나무모델을 사용하는 것이다

=>데이터의 크기가 방대한 경우에도 모델을 빨리 구축할 수 있어서

-의사결정나무모델은 앙상블 모델의 Base 모델로써 활용도가 높다

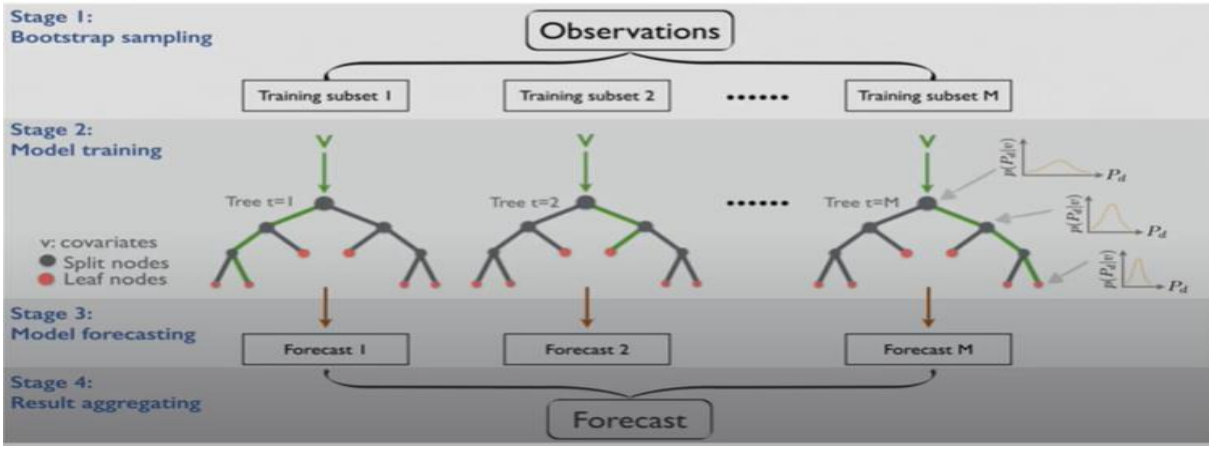
-다수의 의사결정나무모델에 의한 예측을 종합하는 앙상블 방법이다

=>일반적으로 하나의 의사결정나무모델 보다 높은 예측 정확성을 보여줌

핵심 아이디어

1) 여러 개의 Training data를 생성하여 각 데이터마다 개별 의사결정나무모델 구축 => Bagging

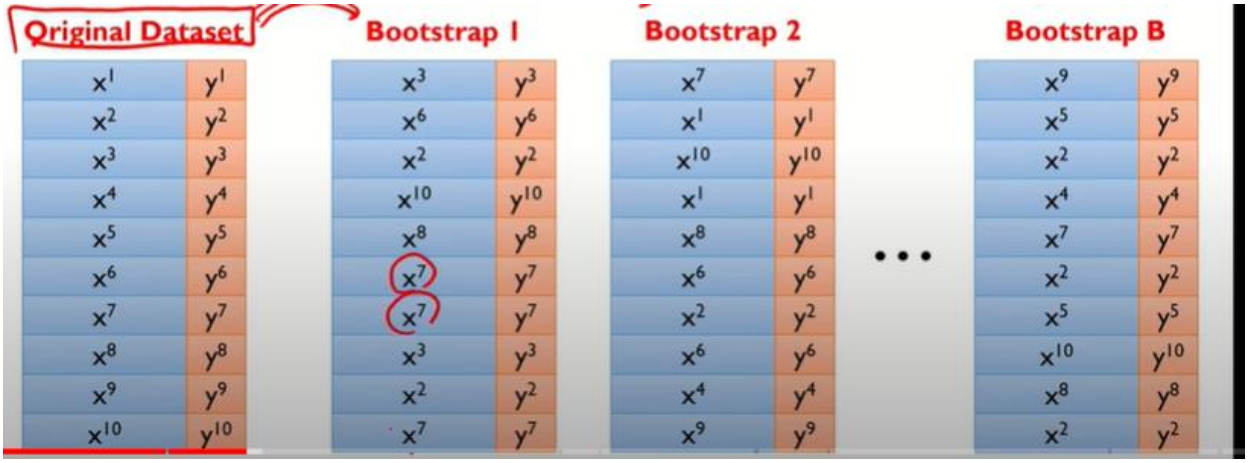
-**Bagging** (Bootstrap Aggregating) : **각각의 bootstrap 샘플로 부터 생성된 모델을 합침**



-**bootstrapping** : 샘플링 하는 기법

=>**각 데이터셋은 복원 추출을 통해 원래 데이터의 수만큼의 크기를 갖도록 샘플링**

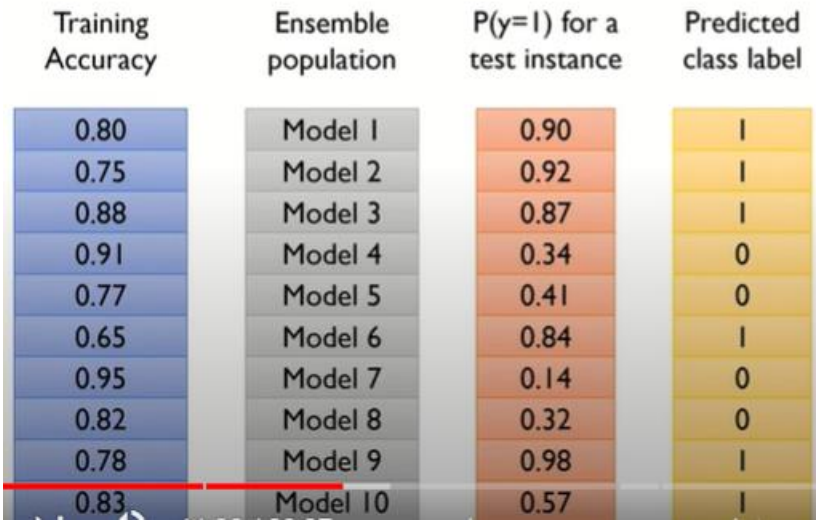
=>개별 데이터셋을 붓스트랩셋이라 부름



=>즉, original data 에서 복원 추출(중복 가능) 하며 원래 데이터의 크기만큼의 데이터를 각 모델마다 뽑는다

-**Result Aggregating** : 개별 모델들을 합치는 과정

For classification problem



$$\sum_{j=1}^{n} I(\hat{y}_j = 0) = 4$$ => 위의 도표에서 yi = 0 이 몇번 나오는지 ?

$$\sum_{j=1}^{n} I(\hat{y}_j = 1) = 6$$ => 위의 도표에서 yi = 1 이 몇번 나오는지 ?

$$Ensemble(\hat{y}) = \underset{i}{\operatorname{argmax}} \left(\sum_{j=1}^{n} I(\hat{y}_j = i), i \epsilon \{0,1\} \right)$$ =>argmax 이므로 1번 클래스가 선정!

2) 의사결정나무모델 구축 시 변수 무작위로 선택 => Random subspace

-의사결정나무의 분기점을 탐색할 때, 원래 변수의 수보다 적은 수의 변수를 임의로 선택하여 해당 변수들만을 고려함

=>즉 예로들어 x1, x2, x3, x4, x5의 변수가 있다면 일반적인 의사결정나무에서는 x1, x2, x3, x4, x5 를 전부 고려하여 분할 변수를 선택해야 한다. 하지만 Random subspace 는 각 모델마다 랜덤하게 x1, x2, x3, x4, x5 중 선택한다. 랜덤하게 선택한 변수 중에서 이제 일반적인 의사결정나무에서의 과정을 실시한다. **왜냐하면 랜덤 포레스트모델의 조건중, Base 모델은 서로 독립적이어야 한다는 점 때문**

=>여기까지 하면 랜덤 포레스트의 핵심 아이디어인 Diversity, Random을 확보한 것이다

-랜덤 포레스트 모델 각각의 개별 tree는 과적합 될 수 있다

-Random forest는 tree수가 충분히 많을 때 Strong Law of Large Numbers에 의해 **과적합 되지 않고 그 에러는 limiting value에 수렴됨**

$$\text{Generalization error} \leq \frac{\bar{\rho}(1 - s^2)}{s^2}$$

$\bar{\rho}$: Decision tree 사이의 평균 상관관계
s : 올바로 예측한 tree와 잘못 예측한 tree수 차이의 평균

-개별 tree의 정확도가 높을수록 s는 증가한다

-Bagging과 **Random subspace** 기법은 각 모델들의 독립성, 일반화, 무작위성을 최대화 시켜 **모델간의 상관관계 p를 감소시킨다**

-즉, 개별 tree의 정확도, 독립성이 높을수록 random forest의 성능이 높아짐

-Out of bag (OOB) : bagging을 사용할 경우 붓스트랩셋에 포함되지 않는 데이터들을 검증 집합으로 사용함