



16. 분류분석

chap16_Classification 수업내용

1. Tree 분류분석 개요
2. Tree 분류분석 특징
3. Tree 분류분석 실습
4. 앙상블 모형
 - 1) RandomForest
 - 2) XGBoost



1. Tree 분류 분석 개요

- 분류 분석?

분류 분석(classification analysis)은 다수의 속성(attribute) 또는 변수를 갖는 객체를 사전에 정해진 그룹 또는 범주(class, category) 중의 하나로 분류하여 분석하는 방법

- 의사결정나무

분류 모델링에 의해서 만들어진 규칙(rule)을 나무 모양으로 그리는 방법, 의사결정이 이뤄지는 시점과 성과를 한눈에 볼 수 있다.

- 활용분야

고객을 분류하는 변수, 규칙, 특성들을 찾아내고, 이를 토대로 미래 잠재 고객의 행동이나 반응을 예측하거나 유도하는데 활용



2. Tree 분류 분석 특징

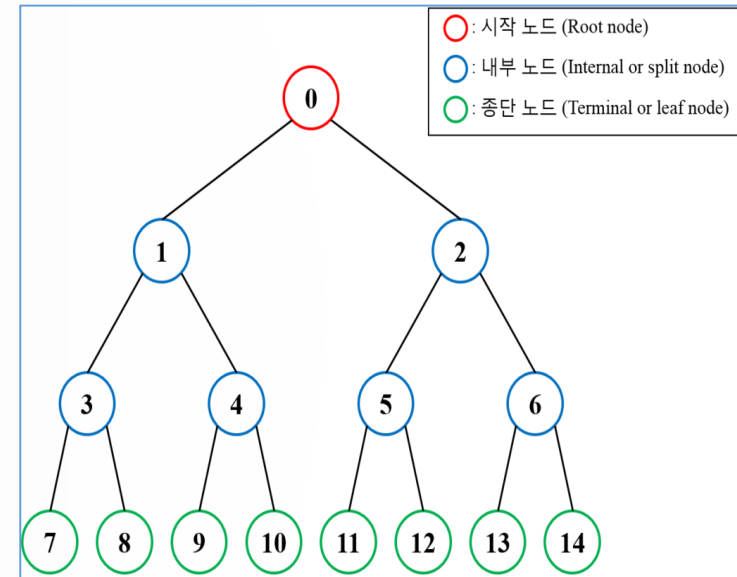
● 분류분석 특징

- 종속변수(y변수) 존재
- 종속변수 : 예측에 Focus을 두는 변수
- 규칙(Rule)을 기반으로 의사결정트리 생성
- 비모수 검정 : 선형성, 정규성, 등분산성 가정 필요 없음
- 단점 : 유의수준 판단 기준 없음(추론 기능 없음)



2. Tree 분류 분석 특징

- 트리(Tree) 구조
 - ✓ 계층 구조로 노드와 에지(edge) 집합
 - ✓ 노드는 내부 노드(internal node)와 종단 노드(leaf node) 분류
 - ✓ 모든 노드에서 들어오는 에지는 하나 (그래프와 차이점)
 - ✓ 각 노드에서 나가는 에지는 제한 없음 (주로 두 개의 에지가 있는 것으로 가정)
- 의사결정트리(Decision Tree)
 - ✓ 결정을 내리기 위해 사용하는 트리
 - ✓ 복잡한 문제를 간단한 계층 구조형태로 나누기 위한 기술
 - ✓ 훈련 데이터로부터 트리구조와 매개변수 자동으로 학습

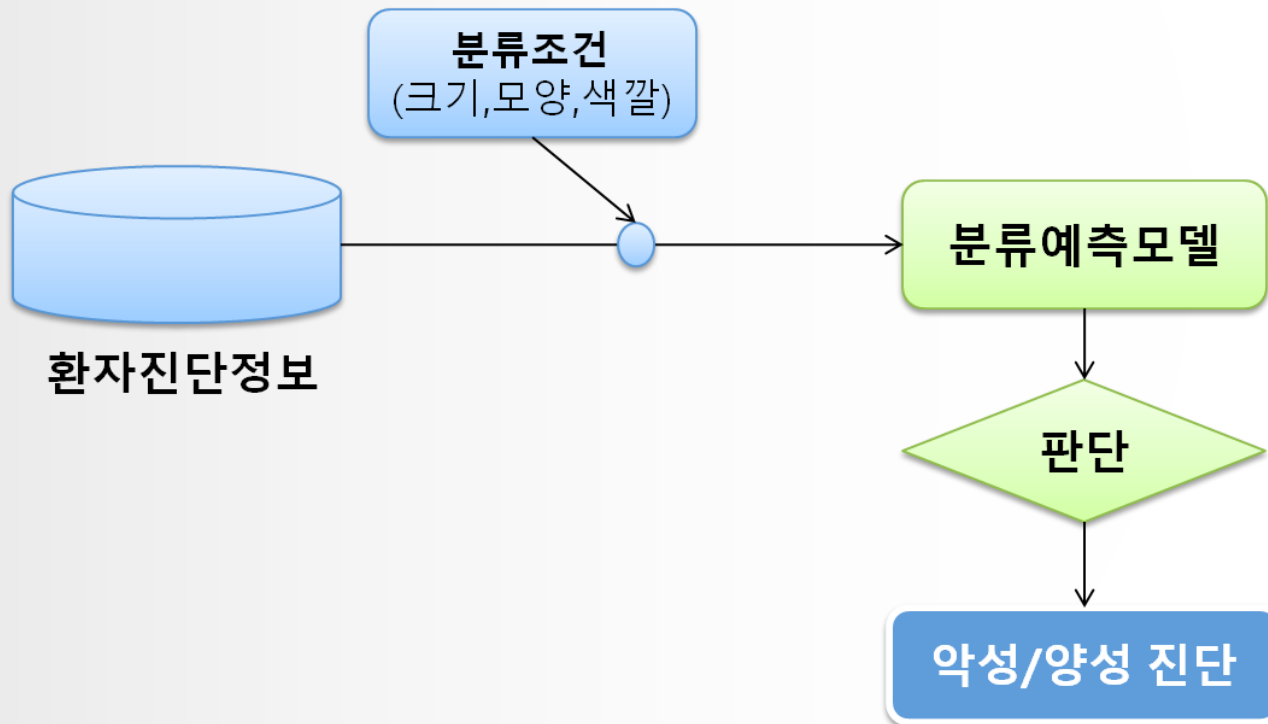


[참고] 위키백과



2. Tree 분류 분석 특징

- 의. 생명분야에서 분류분석 사례





3. Tree 분류 분석 실습

단계1 : 학습데이터와 검증데이터 샘플링

```
result <- sample(2, nrow(iris), replace=T, prob=c(0.7,0.3)) # 7:3비율
```

```
table(result)
```

```
train <- iris[result==1,]
```

```
test <- iris[result==2,]
```

```
# formula 생성 : 형식) 변수 <- 종속변수 ~ 독립변수
```

```
formula <- Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width
```



3. Tree 분류 분석 실습

단계2 : 분류모델 생성(ctree()함수 이용)

part패키지 설치

```
install.packages("rparty") # rpart()함수 제공  
library(rparty)
```

```
model = rpart(Species~., data=train) # iris의 꽃의 종류(Species) 분류  
model
```



3. Tree 분류 분석 실습

rpart()에 의해서 생성된 분류모델 결과

```
> model  
n= 105
```

```
node), split, n, loss, yval, (yprob)  
* denotes terminal node
```

- 1) root 105 68 setosa (0.35238095 0.31428571 0.33333333)
- 2) **Petal.Length**< 2.45 37 0 setosa (1.00000000 0.00000000 0.00000000) *
- 3) Petal.Length>=2.45 68 33 virginica (0.00000000 0.48529412 0.51470588)
- 6) Petal.Width< 1.75 35 2 versicolor (0.00000000 0.94285714 0.05714286) *
- 7) Petal.Width>=1.75 33 0 virginica (0.00000000 0.00000000 1.00000000) *

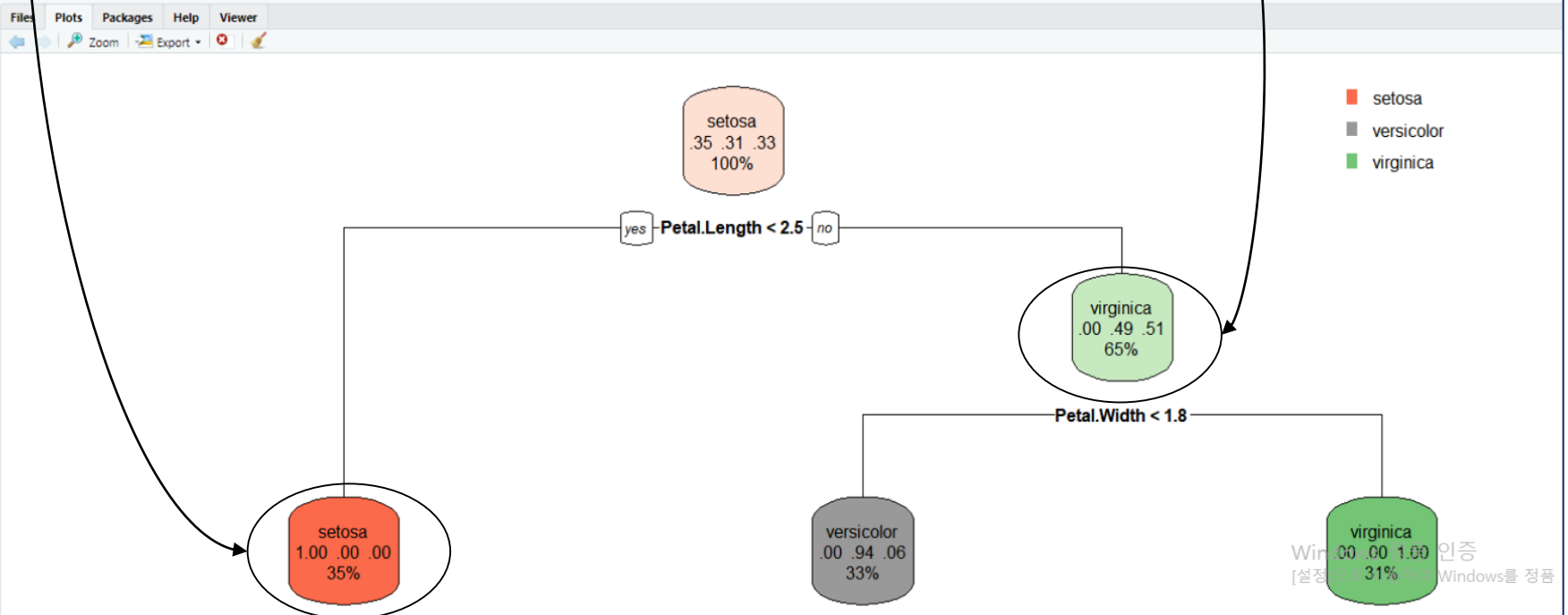
>

Petal.Length변수 -
중요변수



3. Tree 분류 분석 실습

```
1) root 105 68 setosa (0.35238095 0.31428571 0.33333333)
2) Petal.Length < 2.45 37 0 setosa (1.00000000 0.00000000 0.00000000) *
3) Petal.Length >= 2.45 68 33 virginica (0.00000000 0.48529412 0.51470588)
6) Petal.Width < 1.75 35 2 versicolor (0.00000000 0.94285714 0.05714286) *
7) Petal.Width >= 1.75 33 0 virginica (0.00000000 0.00000000 1.00000000) *
> rpart.plot(model) # rpart 모델 tree 출력
>
```



왼쪽 노드 : 분류조건, 가장 많이 분류된 수(분류), 나머지(미분류), 분류 label (분류 비율), 단노드 여부



3. Tree 분류 분석 실습

단계3 : 분류모델 예측

```
pred <- predict(model, test) # 비율 예측  
pred # y범주 비율예측  
pred <- predict(model, test, type="class") # 분류 예측  
pred # y범주 예측
```

Species변수
로 분류모델
예측



3. Tree 분류 분석 실습

단계4 : 분류모델 평가

2) 분류모델 성능 평가

```
table(pred, test$Species)
```

pred	setosa	versicolor	virginica
setosa	13	0	0
versicolor	0	16	3
virginica	0	1	12



4. 앙상블 모형

● 앙상블 학습

- 여러 가지 우수한 학습 모델을 조합해 예측력을 향상시키는 모델
 - ✓ 장점 : 단일 모델에 비해서 분류 성능 우수
- 앙상블 알고리즘
 - ✓ 배깅(Bagging), 부스팅(Boosting)
 - RandomForest 배깅 알고리즘의 일종
- 앙상블 학습 모델 생성 절차

1. 전체 데이터에서
훈련 집합 생성



2. 각 훈련집합
모델 학습



3. 학습된 모델
앙상블 도출

※ 단점 : 모델 결과의 해석이 어렵고, 예측 시간 많이 소요



4. 앙상블 모형

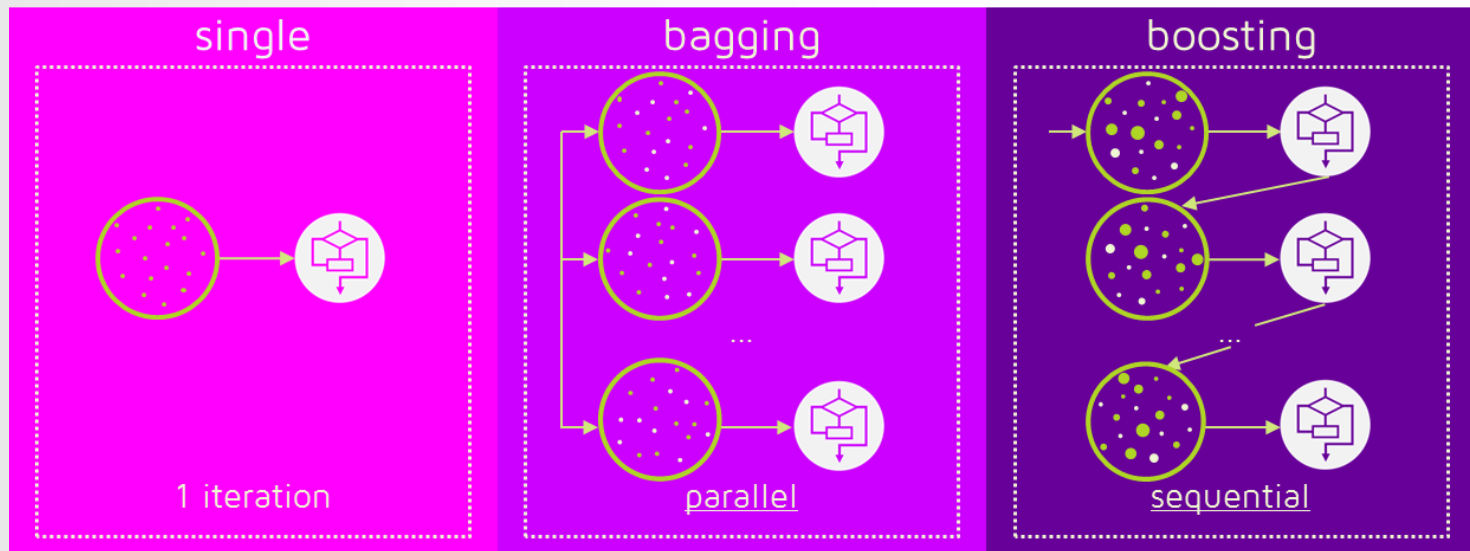
● 앙상블 학습 알고리즘 비교

분류	배깅(Bagging)	부스팅(Boosting)
공통점	전체 데이터 집합으로부터 복원 랜덤 샘플링(bootstrap)으로 훈련 집합 생성	
차이점	병렬학습 : 각 모델의 결과를 조합하여 투표 결정	순차학습 : 현재 모델 가중치 -> 다음 모델 전달
특 징	균일한 확률분포에 의해서 훈련 집합 생성	분류하기 어려운 훈련 집합 생성
강 점	과대적합에 강함	높은 정확도
약 점	특정 영역 정확도 낮음	Outlier 취약
R 패키지	randomForest	XGboost



4. 앙상블 모형

● 앙상블 학습 알고리즘



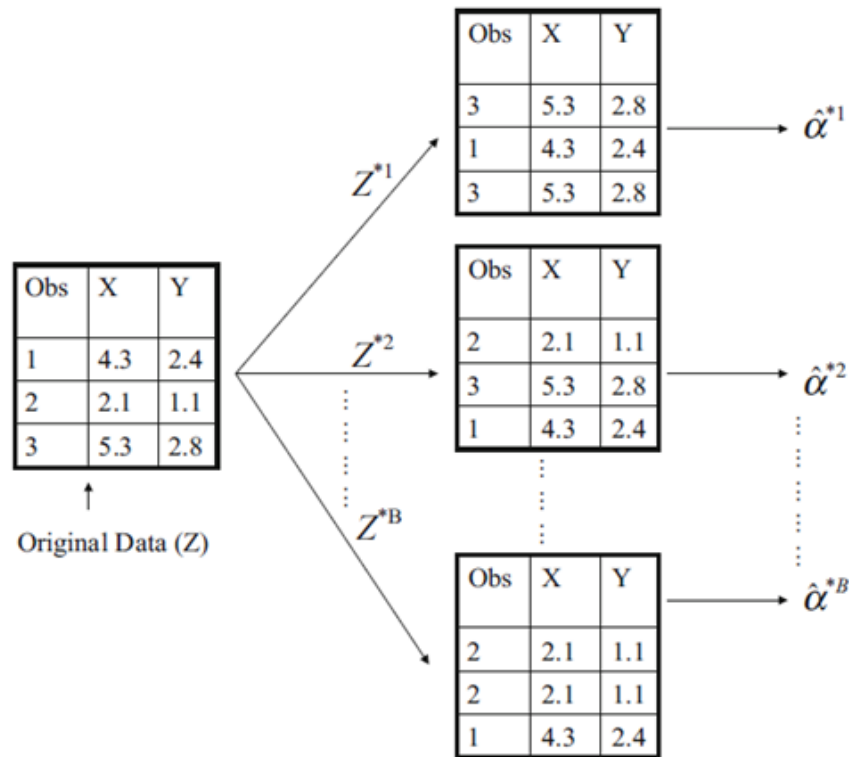
※ Boosting은 맞추기 어려운 문제를 맞추는데 초점을 맞춤

● Random Forest 사용 이유

1. 단일 의사결정트리는 과대적합(overfitting)의 위험이 큼
 2. 배깅을 이용해 각 트리의 평균, 확률, 투표를 통해 목표변수 예측
 3. 트리의 편향은 유지되고, 분산은 감소되기 때문에 정확도 높음
 4. 빅데이터 시스템에서 분산처리 시스템에 맞는 분류분석 기법
 - 여러 개의 훈련 데이터를 추출하여 트리 생성, 결합을 통해 목표변수 예측의 구조가 분산처리시스템에 적합
-
- 정리
 1. 배깅 알고리즘을 통해 임의 복원 추출되는 훈련용 데이터를 생성하고, 각 트리 생성
 2. 예측결과를 투표, 평균, 확률 등으로 결합하여 최종 모형 생성
 3. 적용되는 R 패키지 : randomForest

① 부트스트랩(Bootstrap)

- 원래의 데이터 셋으로부터 관측치를 반복적으로 추출(복원 반복 추출)하여 데이터 셋을 얻는 방법
- 데이터의 양을 임의적으로 늘리고, 데이터 셋의 분포가 고르지 않을 때 고르게 만드는 효과



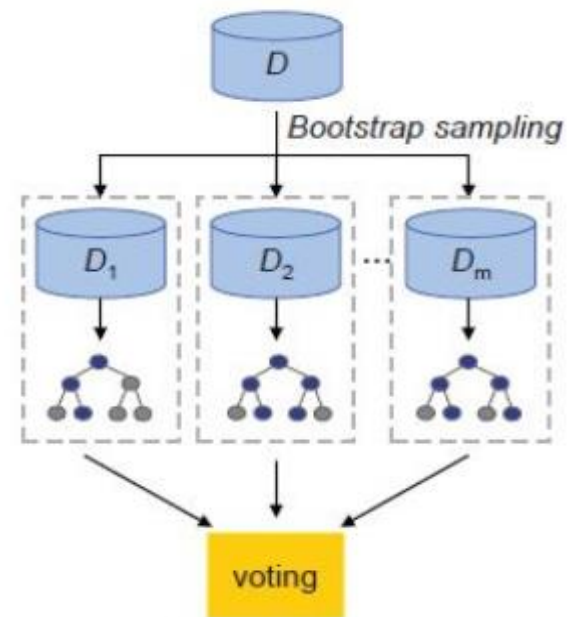
[참고 사이트]

<https://m.blog.naver.com/PostView.nhn?blogId=ysd2876&logNo=221219689884&isFromSearchAddView=true>

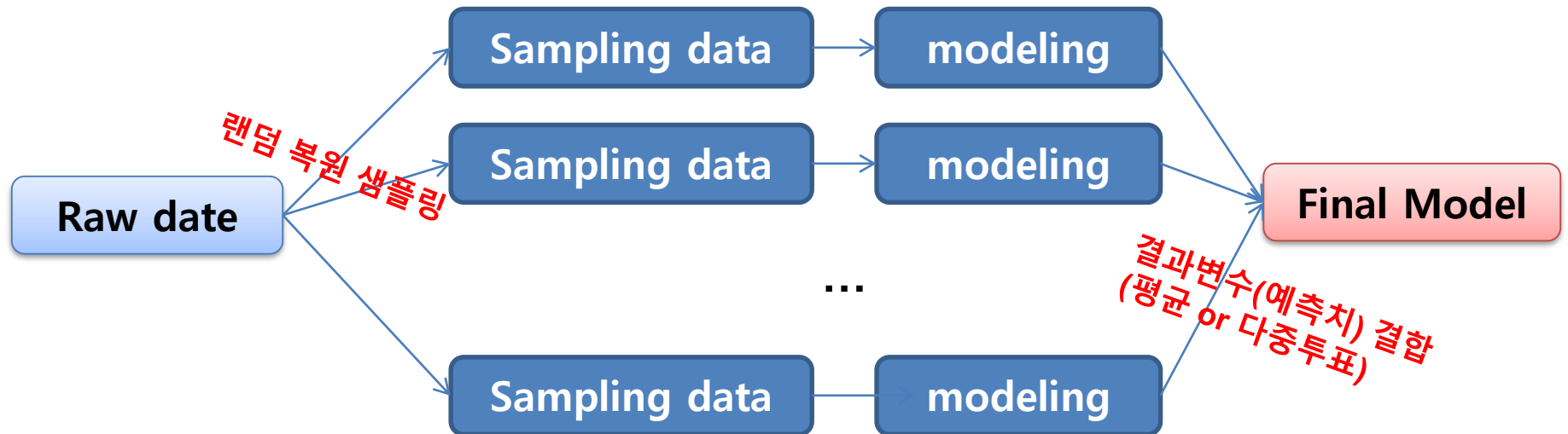
② 배깅(Bagging) 알고리즘

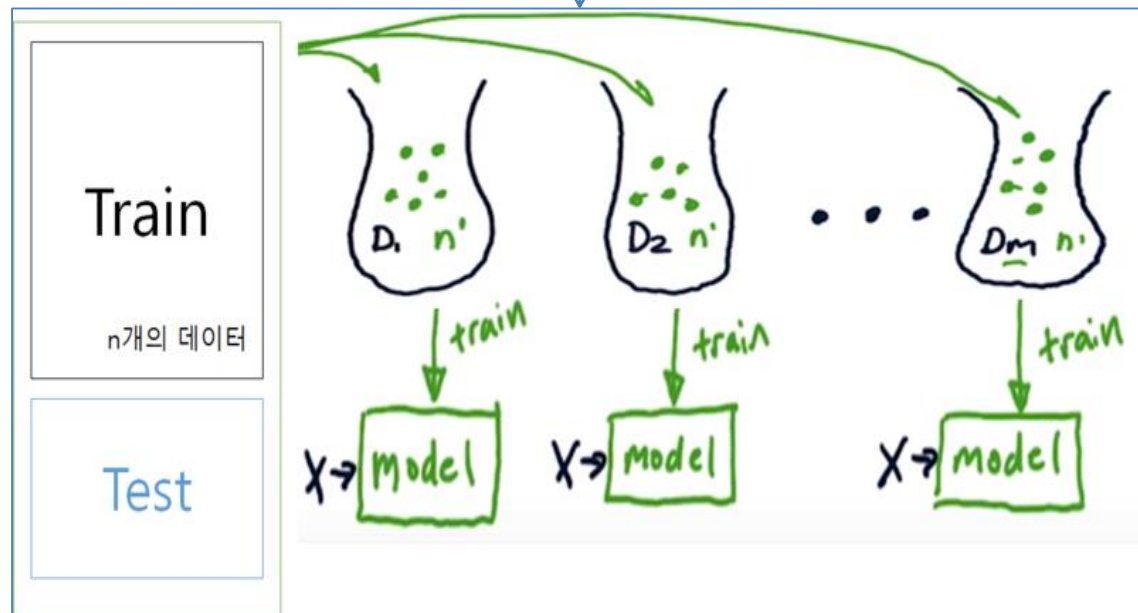
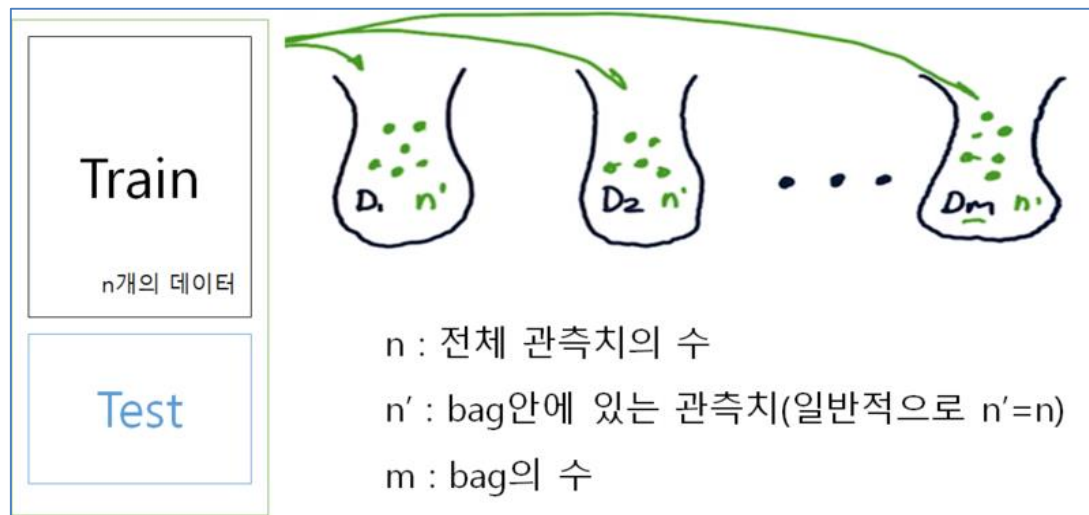
- Bagging : Bootstrap Aggregating(“주머니 통합”)
- 부트스트랩을 통해서 조금씩 서로 다른 훈련 데이터를 생성하여 모델(훈련 된 트리)을 생성하고, 결과를 결합(aggregating) 시키는 방법

1. D개의 전체데이터가 있다.
2. 전체데이터에서 D개와 같은 갯수의 데이터를 복원추출하여 D_1 (주머니) 생성
3. D_1 데이터를 학습하여 모델(트리) 생성
4. 위 2번, 3번과 같은 방법으로 D_m (주머니) 생성, 모델(트리) 생성
5. m개 트리의 평균을 통해 예측
 - 양적반응변수(회귀트리) : 각 트리 평균 예측
 - 질적반응변수(분류트리) : 각 트리 voting 예측



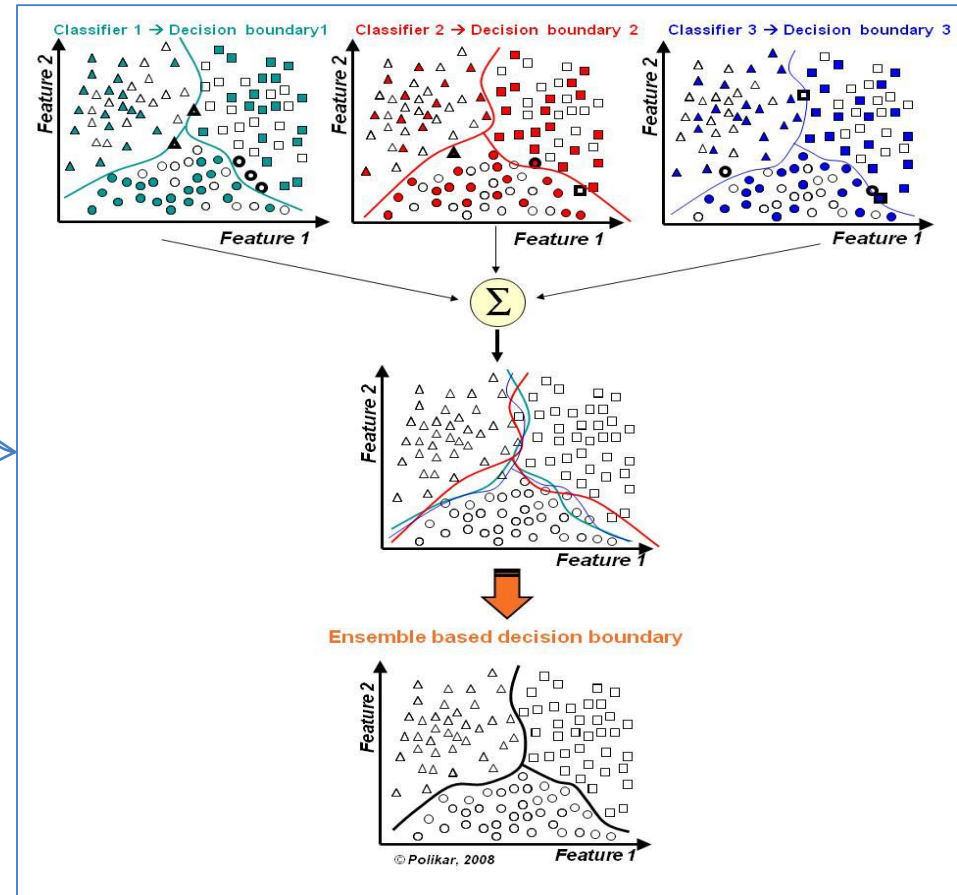
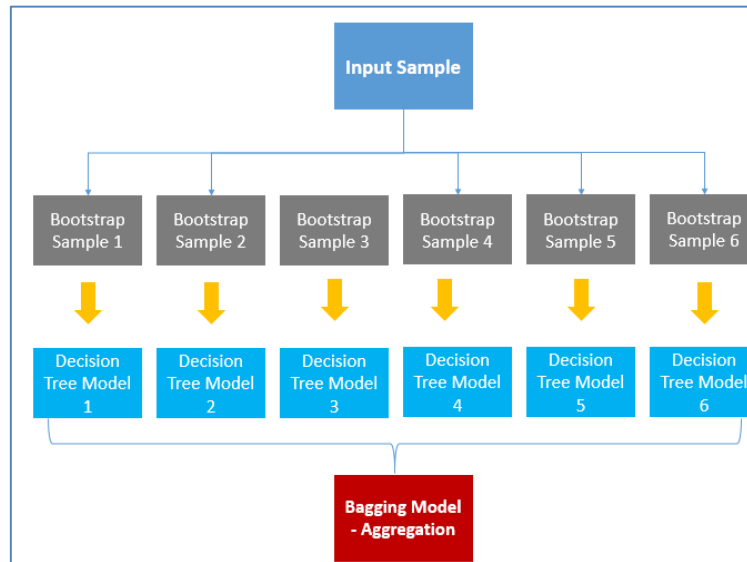
- 원 데이터로 부터 n번 랜덤 복원 샘플링을 하고 각 샘플의 모델링을 통해서 나온 결과변수(예측치)들을 결합하여 최종 모델을 생성
- 각 샘플의 결과변수(예측치)들을 결합하는 방법은 결과변수가 연속형이면 평균(average), 범주형이면 다중 투표(majority vote) 사용





[참고 사이트]

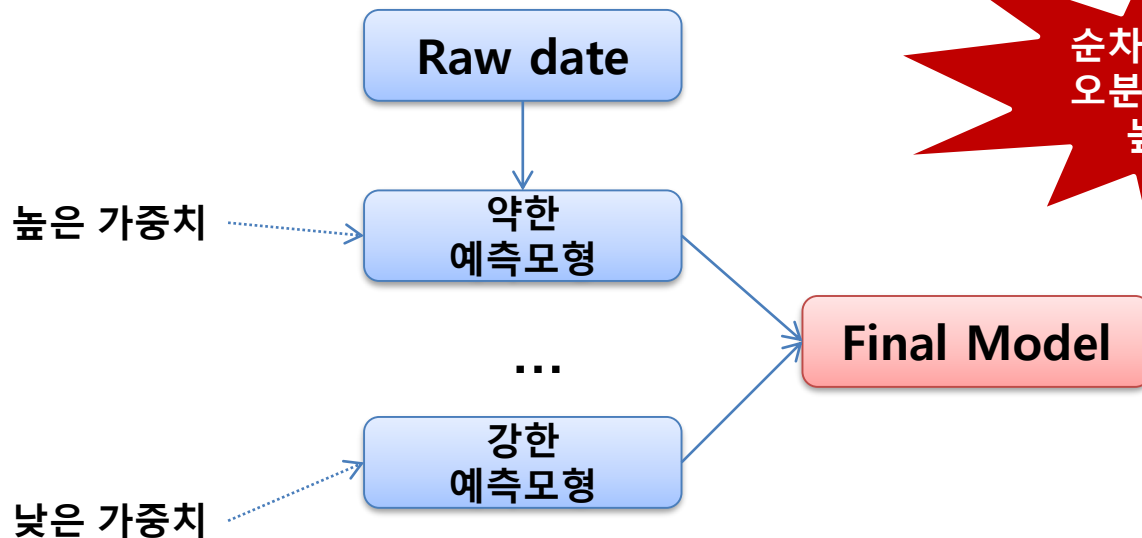
<https://m.blog.naver.com/PostView.nhn?blogId=ysd2876&logNo=221219689884&isFromSearchAddView=true>



참고 사이트 : <https://swallow.github.io/bagging-boosting>

③ 부스팅(boosting) 알고리즘

- 잘못 분류된 객체들에 집중하여 새로운 분류규칙을 생성하는 단계를 반복하는 알고리즘(순차적 학습)
- 약한 예측모형들을 결합하여 강한 예측모형을 만드는 알고리즘
- 오 분류된 개체는 높은 가중치, 정 분류된 개체는 낮은 가중치를 적용하여 **예측모형의 정확도를 향상시키기** 위한 방법





1) Random Forest 알고리즘

■ 특징

- ✓ 여러 개의 결정 트리를 임의적으로 학습하는 방식(앙상블 학습방법 : 배깅 유형)
- ✓ 회귀분석, 분류분석 모두 가능
- ✓ 별도 튜닝(스케일 조정) 과정 없음
- ✓ 분류, 회귀 등에서 가장 많이 사용 학습방법
- ✓ 단일 tree 모델 단점 보완(성능, 과대적합)
- ✓ 대용량 데이터 셋으로 처리시간 증가(단점)
- ✓ 멀티코어 프로세스 이용 병렬처리 가능



■ 차이점

- ✓ 배깅 : 샘플 복원추출 시 모든 설명변수 사용
- ✓ 랜덤포레스트 : a개의 설명변수만 복원 추출

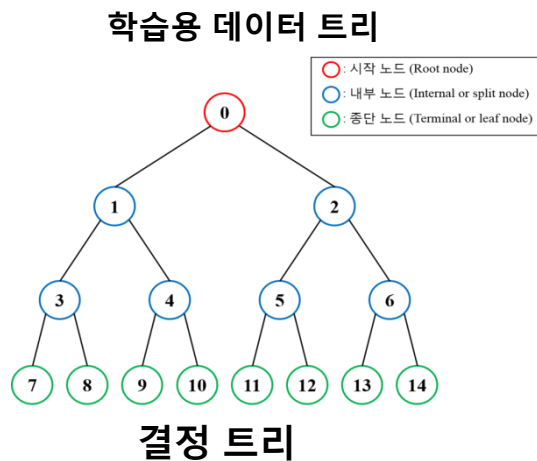
■ 설명변수 갯수 : 전체 변수 p 의 제곱근= \sqrt{p} (예: 15개 변수라면 4개 정도)

■ 랜덤포리스트는 일반적으로 배깅보다 성능 우수

(설명변수가 많을 경우, 대체로 변수간 상관성이 높은 변수가 섞일 확률이 높는데 그 가능성을 제거)

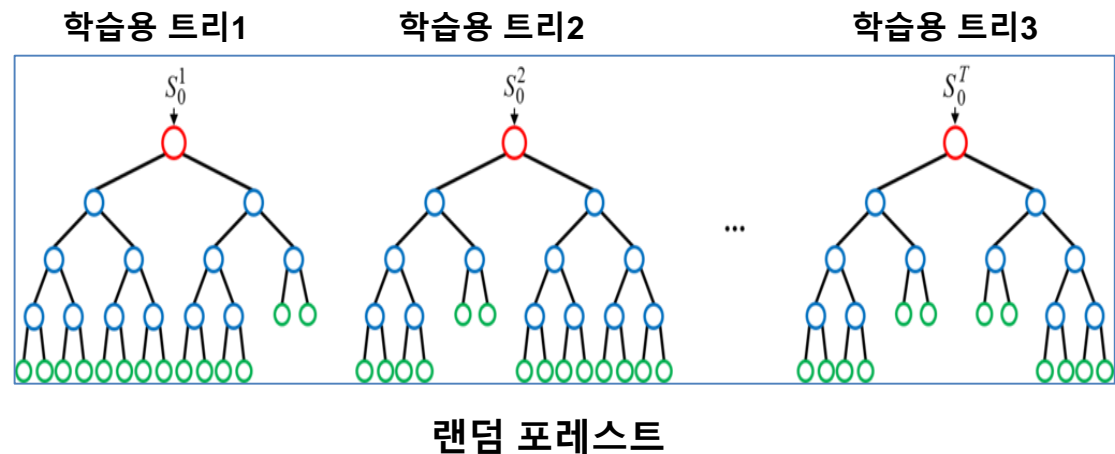
● Decision tree

1. 동일한 하나의 데이터 집합에서 한 개의 훈련용 데이터를 생성
2. 한 번의 학습을 통해서 하나의 분류 트리 생성 및 목표변수 예측
3. 생성된 분류모델을 검정데이터에 적용하여 목표변수 예측



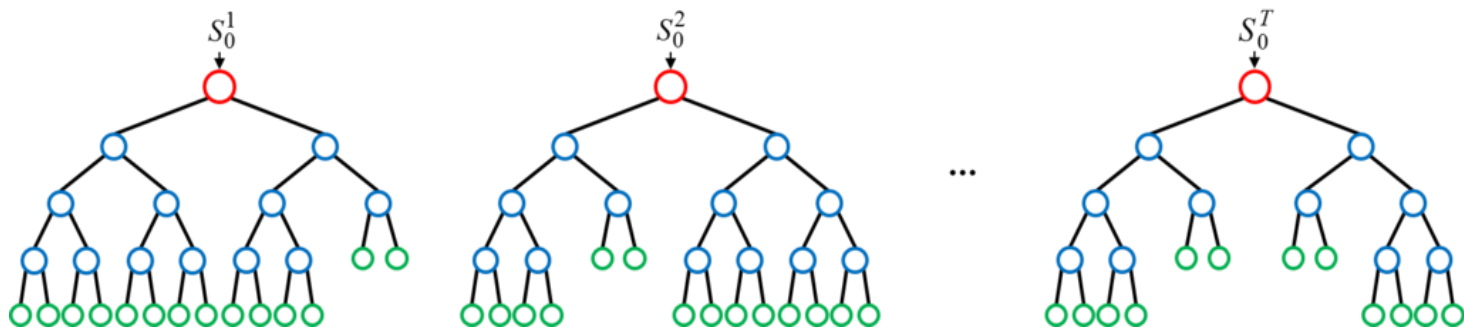
● Random Forest

1. 동일한 하나의 데이터 집합에서 임의복원 샘플링을 통해 여러 개의 훈련용 데이터를 생성
2. 여러 번의 학습을 통해 여러 개의 트리 생성하고, 이를 결합하여 최종적으로 목표변수 예측
3. 분류모델을 검정데이터에 적용



배깅에 의한 랜덤 포레스트 훈련 과정 3단계

1. 부트스트랩 방법을 통해 T 개의 훈련 데이터 집합을 생성한다.
2. T 개의 기초 분류기(트리)들을 훈련시킨다.
3. 기초 분류기(트리)들을 하나의 분류기(랜덤 포레스트)로 결합한다.



배깅을 이용하여 T 개의 결정트리들로 구성된 랜덤 포레스트를 학습하는 과정

S_0 : 전체 학습 데이터 집합, S_0^T : 결정트리를 위해 배깅을 통해 임의로 선택된 학습 데이터

- 가장 적절한 Tree와 설명변수 갯수

1. bootstrap의 수 m 은 어느 정도 커야할까?

답: m 이 100이상이면 충분하지만 검정오차가 안정화될 만큼 큰 값을 사용(예:400개 이상)

2. 랜덤포리스트의 설명변수 갯수 a 는 얼마가 적당한가?

답: 회귀트리 : $1/3p$, 분류트리 : p 의 제곱근= $\sqrt{\text{전체변수 개수}}$
변수간 상관성에 따라 최적의 a 값이 다를 수 있다.

● Random Forest 사용 이유

1. 단일 의사결정트리는 과대적합(overfitting)의 위험이 큼
 2. 배깅을 이용해 각 트리의 평균, 확률, 투표를 통해 목표변수 예측
 3. 트리의 편향은 유지되고, 분산은 감소되기 때문에 정확도 높음
 4. 빅데이터 시스템에서 분산처리 시스템에 맞는 분류분석 기법
 - 여러 개의 훈련 데이터를 추출하여 트리 생성, 결합을 통해 목표변수 예측의 구조가 분산처리시스템에 적합
-
- 정리
 1. 배깅 알고리즘을 통해 임의 복원 추출되는 훈련용 데이터를 생성하고, 각 트리 생성
 2. 예측결과를 투표, 평균, 확률 등으로 결합하여 최종 모형 생성
 3. 적용되는 R 패키지 : randomForest





2) XGBoost 알고리즘

- 여러 개의 결정 트리를 임의적으로 학습하는 방식
 - ✓ 앙상블 학습방법(부스팅 유형)
- 순차적 학습 방법 : 약한 분류기를 강한 분류기로 생성
 - ✓ 분류정확도 우수, Outlier 취약
 - ✓ 케글(Keggle.com) : 도전 데이터 과학자 에서 5년 연속 1위
- 다양한 속성으로 모델 생성
 - objective = "binary:logistic": binary:logistic : y변수 이항
 - max_depth = 2: tree 구조가 간단한 경우 : 2
 - nthread = 2 : cpu 사용 수 : 2
 - nrounds = 2 : 실제값과 예측값의 차이를 줄이기 위한 반복학습 횟수
 - eta = 1 : 학습률을 제어하는 변수(Default: 0.3), 오버 피팅을 방지