앙상블(Ensemble)

http://kocw-n.xcache.kinxcdn.com/data/document/2020/hoseo/parksungbum0908/7.pdf
http://contents2.kocw.or.kr/KOCW/document/2017/chungbuk/najonghwa/13.pdf

의사결정나무의 문제점을 ctree와 다른 방식으로 보완하기 위하여 개발된 방법
주어진 자료로부터 예측 모형을 여러 개 만들고, 이것을 결합하여 최종적인 예측 모형을 만드는 방법

배깅(Breiman, 1996) → 부스팅 개발 → 랜덤포레스트(Random Forest)

앙상블에서 사용되는 기법: 배깅, 부스팅, 랜덤포레스트

1. 배깅(Bagging)

- 불안정한 예측모형에서 불안전성을 제거함으로써 예측력을 향상
- Where 불안정한 예측모형: 데이터의 작은 변화에도 예측 모형이 크게 바뀌는 경우
- Bootstrap AGGregatING의 준말
- 주어진 자료에 대하여 여러 개의 부트스트랩(bootstrap)자료를 만들고, 각 부트스트랩 자료에 예측 모형을 만든 다음, 이것을 결합하여 최종 예측 모형을 만드는 방법
- 부트스트랩자료: 주어진 자료로부터 동일한 크기의 표본을 랜덤 복원 추출로 뽑은 것

실습.

==========

```
install.packages("party")
install.packages("caret")

library(party)
library(caret)

# data sampling
data1 <- iris[sample(1:nrow(iris), replace=T),]
data2 <- iris[sample(1:nrow(iris), replace=T),]
data3 <- iris[sample(1:nrow(iris), replace=T),]
data4 <- iris[sample(1:nrow(iris), replace=T),]
data5 <- iris[sample(1:nrow(iris), replace=T),]

# 예측모형 생성
citree1 <- ctree(Species~., data1)
citree2 <- ctree(Species~., data2)
citree3 <- ctree(Species~., data3)
citree4 <- ctree(Species~., data4)
citree5 <- ctree(Species~., data5)
```

```
# 예측수행
predicted1 <- predict(citree1, iris)
predicted2 <- predict(citree2, iris)
predicted3 <- predict(citree3, iris)
predicted4 <- predict(citree4, iris)
predicted5 <- predict(citree5, iris)

# 예측모형 결합하여 새로운 예측모형 생성
newmodel <- data.frame(Species=iris$Species,
predicted1,predicted2,predicted3,predicted4,predicted5)
head(newmodel)
newmodel

# 최종모형으로 통합
funcValue <- function(x) {
  result <- NULL
  for(i in 1:nrow(x)) {
    xtab <- table(t(x[i,]))
    rvalue <- names(sort(xtab, decreasing = T) [1])
    result <- c(result,rvalue)
  }
  return (result)
}

newmodel

# 최종 모형의 2번째에서 6번째를 통합하여 최종 결과 생성
newmodel$result <- funcValue(newmodel[, 2:6])
newmodel$result


# 최종결과 비교
table(newmodel$result, newmodel$Species)
```

==========

결과

```
> table(newmodel$result, newmodel$Species)

            setosa versicolor virginica
  setosa        50          0         0
  versicolor     0         48         4
  virginica      0          2        46
> 
```

2. 부스팅(Boosting)

예측력이 약한 모형만 만들어지는 경우, 예측력이 약한 모형들을 결합하여 강한 예측 모형을 만드는 방법
where 예측력이 약한 모형: 랜덤하게 예측하는 것보다 더 좋은 예측력을 가진 모형

3. 랜덤포레스트(Random Forest)

2001년 Breiman에 의해 개발.
배깅과 부스팅보다 더 많은 무작위성을 주어서 약한 학습 모델을 만든 다음, 이것을 선형 결합하여 최종학습기를 만드는 방법
예측력이 매우 높음.
입력 변수 개수가 많을 때는 배깅이나 부스팅과 비슷하거나 더 좋은 예측력을 보여주어 많이 사용된다.

실습.

==========

```
head(iris)

# 70% training데이터, 30% testing데이터로 구분
idx <- sample(2, nrow(iris), replace=T, prob=c(0.7, 0.3))

trData <- iris[idx == 1, ]
nrow(trData)
teData <- iris[idx == 2, ]
nrow(teData)

library(randomForest)

# 랜덤포레스트 실행 (100개의 tree를 다양한 방법(proximity=T)으로 생성)
RFmodel <- randomForest(Species~., data=trData, ntree=100, proximity=T)
RFmodel


# 시각화
plot(RFmodel, main="RandomForest Model of iris")
```

```
# 모델에 사용된 변수 중 중요한 것 확인
importance(RFmodel)

# 중요한 것 시각화
varImpPlot(RFmodel)

# 실제값과 예측값 비교
table(trData$Species, predict(RFmodel))

# 테스트데이터로 예측
pred <- predict(RFmodel, newdata=teData)

# 실제값과 예측값 비교
table(teData$Species, pred)

# 시각화
plot(margin(RFmodel, teData$Species))
====================
```

그래프에서 모델 오류가 안정적인 상태를 보이기 시작하는 시점의 tree개수로 실행

```
# 중요한 것 시각화
```

varImpPlot(RFmodel)의 그래프에서 Petal.Width, Petal.Length가 중요변수

예측정확성

margin()함수: 두 값의 차이

https://www.rdocumentation.org/packages/ggplot2/versions/3.3.3/topics/margin

랜덤포레스트 방식은 기존의 의사결정 트리 방식에 비해서 많은 데이터를 이용하여 학습을 수행하기 때문에 비교적 예측력이 뛰어나고, 과적합(overfitting)문제를 해결할 수 있다.

랜덤포레스트 모델은 기본적으로 원 데이터(raw data)를 대상으로 복원추출 방식으로 데이터의 양을 증가시킨 후 모델을 생성하기 때문에 데이터의 양이 부족해서 발생하는 과적합의 원인을 해결할 수 있다.

각각의 분류모델에서 예측된 결과를 토대로 투표방식(voting)으로 최적의 예측치 선택

실습 (랜덤포레스트 기본 모델 생성)

randomForest패키지
randomforest()함수
where
formula: y ~ x형식으로 반응변수와 설명변수 식
data: 모델 생성에 사용될 데이터 셋
ntree: 복원 추출하여 생성할 트리 수 지정
mtry: 자식 노드를 분류할 변수 수 지정
na.action: 결측치(NA)를 제거할 함수 지정
importance: 분류모델 생성과정에서 중요 변수 정보 제공 여부
https://www.rdocumentation.org/packages/randomForest/versions/4.6-14/topics/randomForest

1단계: 패키지 설치 및 데이터 셋 가져오기

```
install.packages("randomForest")
library(randomForest)
data(iris)
```

randomForest 패키지

2단계: 랜덤포레스트 모델 생성

```
model <- randomForest(Species ~ ., data = iris)
model
```

'Number of trees: 500': 학습데이터로 500개의 포레스트(Forest)가 복원 추출방식으로 생성
'No. of variables tried at each split: 2': 두 개의 변수 이용하여 트리의 자식 노드가 분류되었다는 의미

실습 (파라미터 조정 – 트리 개수 300개, 변수 개수 4개 지정)

```
model2 <- randomForest(Species ~ ., data = iris,
                       ntree = 300, mtry = 4, na.action = na.omit)
model2
```

na.action속성: NA 처리 방법 지정. 여기서는 na.omit로 속성을 지정하여 NA 제거

실습 (중요 변수를 생성하여 랜덤포레스트 모델 생성)

1단계: 중요 변수로 랜덤포레스트 모델 생성

```
model3 <- randomForest(Species ~ ., data = iris,
                       importance = T, na.action = na.omit)
```

2단계: 중요 변수 보기

```
importance(model3)
```

importance()함수: 분류모델을 생성하는 과정에서 입력 변수 중 가장 중요한 변수가 어떤 변수인가를 알려주는 역할
https://www.rdocumentation.org/packages/randomForest/versions/4.6-

14/topics/importance

MeanDecreaseAccuracy: 분류정확도를 개선하는데 기여한 변수를 수치로 제공
MeanDecreaseGini: 노드 불순도(불확실성)를 개선하는데 기여한 변수를 수치로 제공

3단계: 중요 변수 시각화

```
varImpPlot(model3)
```

더 알아보기 (엔트로피(Entropy): 불확실성 척도)

```
x1 <- 0.5; x2 <- 0.5
e1 <- -x1 * log2(x1) - x2 * log2(x2)
e1

x1 <- 0.7; x2 <- 0.3
e2 <- -x1 * log2(x1) - x2 * log2(x2)
e2
```

엔트로피가 작으면 불확실성이 낮아진다.
불확실성이 낮아지면 그만큼 분류정확도가 향상된다고 볼 수 있다.

실습 (최적의 파라미터(ntree, mtry)찾기)

1단계: 속성값 생성

```
ntree <- c(400, 500, 600)
mtry <- c(2:4)
param <- data.frame(n = ntree, m = mtry)
param
```

2단계: 이중 for()함수를 이용하여 모델 생성

```
for(i in param$n) {
  cat('ntree =', i, '₩n')
  for(j in param$m) {
    cat('mtry =', j, '₩n')
    model_iris <- randomForest(Species ~ ., data = iris,
                               ntree = i, mtry = j, na.action = na.omit)
    print(model_iris)
  }
}
```

9개의 모델이 생성된 결과에서 오차 비율(OOB(Out of Bag) estimate of error rate)을 비교하여 최적의 트리와 변수 결정