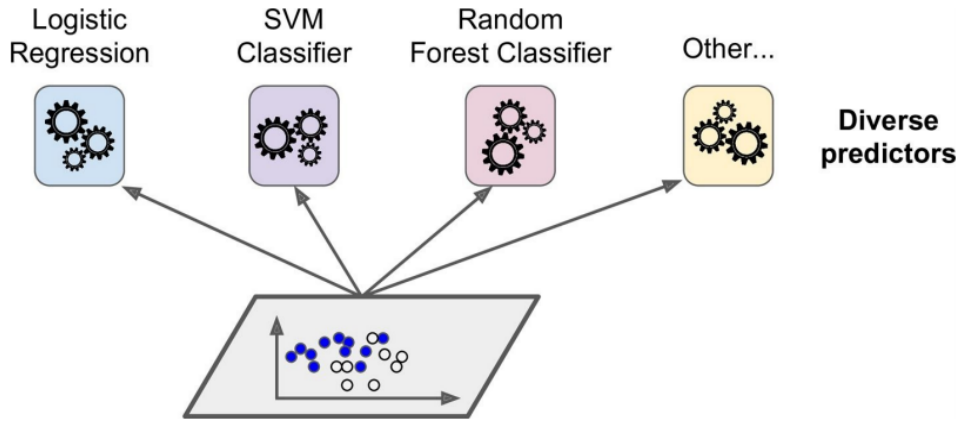
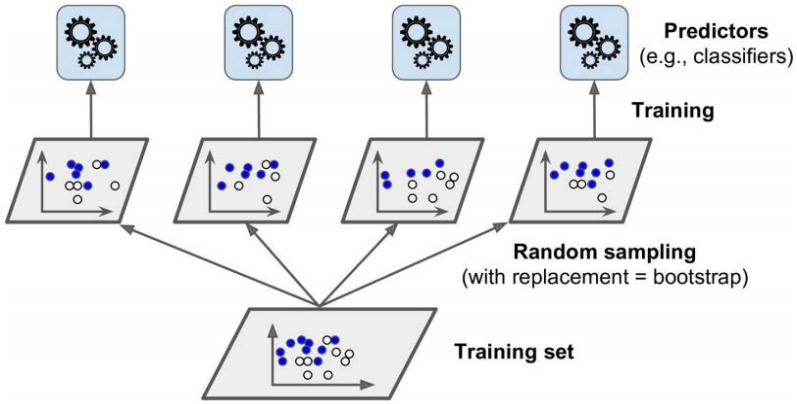
6장 데이터 분석기법(분류 모델)

- 머신 러닝
- 분류 모델
  - 의사결정트리
  - 랜덤 포리스트
  - SVM

03 앙상블 모델 (Ensemble model)

- 앙상블(ensemble) 모델은 여러 개의 분류 모델에 의한 결과를 종합하여 분류의 정확도를 높이는 방법이다.
- 여러 개의 모델을 합쳐 일반화 성능 향상 : 대중의 지혜(Wisdom of Crowd)
- 표본추출법으로 원래 데이터에서 여러 개의 훈련용 데이터 집합을 만들어 각각의 데이터 집합에서 하나의 분류기를 만들어 앙상블 하는 방법이다.
- 새로운 자료에 대해 각 분류기의 예측값들의 가중 투표(weighted vote) 등을 통해 분류를 수행한다.





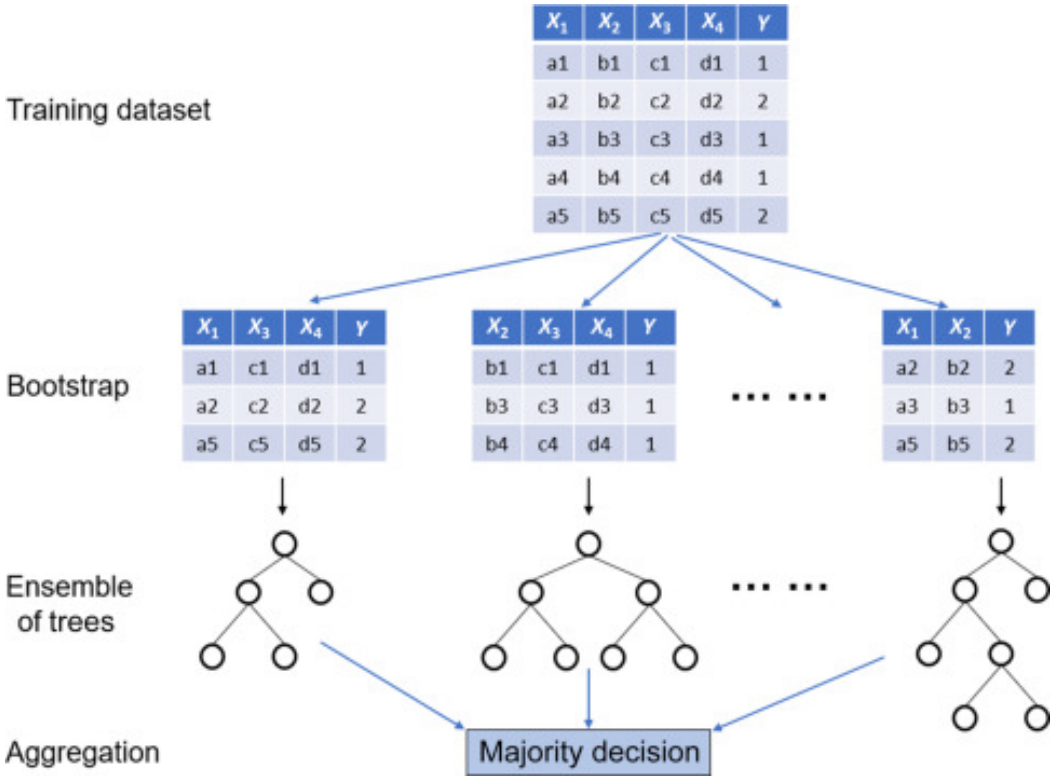
03 앙상블 모델 – 계속

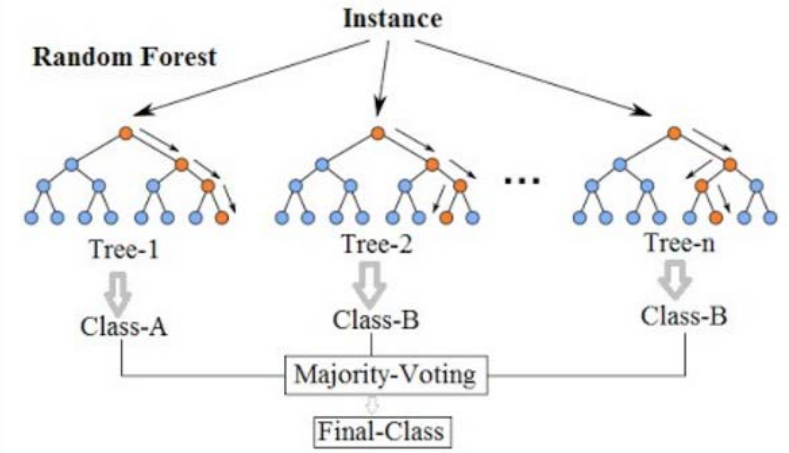
- 개별 모형과 비교한 앙상블 모델의 장점
  - 치우침이 있는 여러 모형의 평균을 취하면, 어느 쪽에도 치우치지 않는 결과(평균)를 얻게 된다.
  - 분산을 감소시킨다: 한 개 모형으로부터의 단일 의견보다 여러 모형의 의견을 결합하면 변동이 작아진다.
  - 과적합의 가능성을 줄여준다: 각 모형으로부터 예측을 결합하면 과적합의 여지가 줄어든다.

- 앙상블 방법
  - 배깅 (bagging) : 중복을 허용하는 샘플링
  - 페이스팅(pasting) : 중복을 허용하지 않는 샘플링
  - 부스팅(boosting) : 이전 예측기의 오차를 보완해서 샘플링
  - 스태킹(stacking) : 앙상블 결과 위에 예측을 위한 모델 추가

03 앙상블 모델 – 계속

- 랜덤 포리스트(random forests) :
  - 의사결정트리의 앙상블 모델
  - 배깅의 개념과 설명변수(독립변수)의 임의 선택(random selection) 방법을 결합한 앙상블 기법
  - 매우 강력한 머신러닝 기법

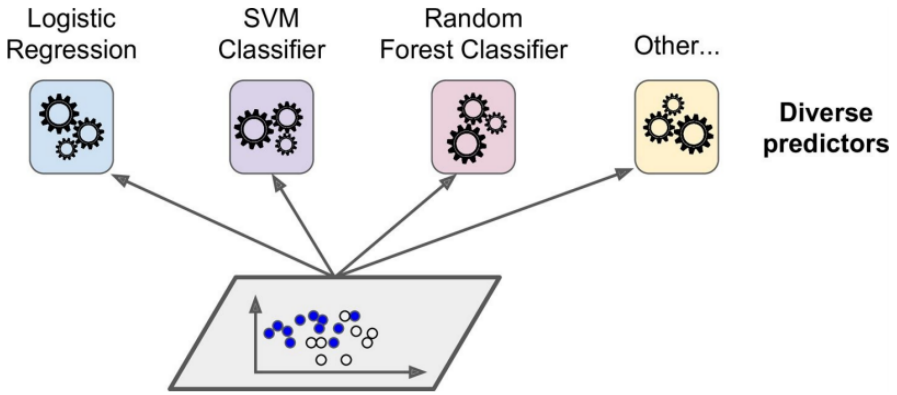




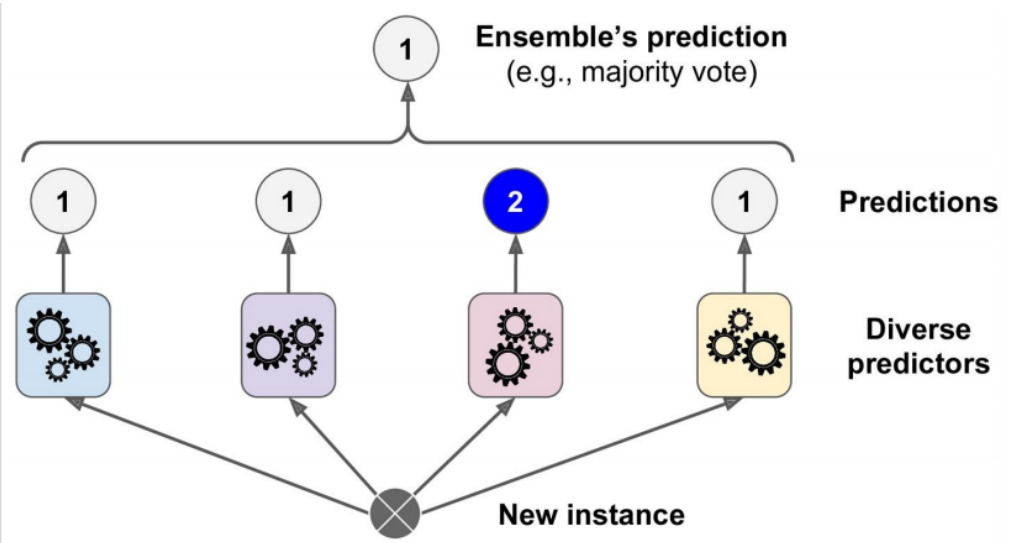
03-1 투표기반 분류기 (Voting classifiers)

- 기존 방법 :
  - 다양한 분류기를 학습.
  - 정확도 가장 높은 분류기를 선택하여 결과 예측

- 앙상블 방법 :
  - 다양한 분류기를 학습.
  - 각 분류기 예측 결과를 혼합해서 최종 예측 결과를 결정 (아래 예에서는 다수결 투표 했음)



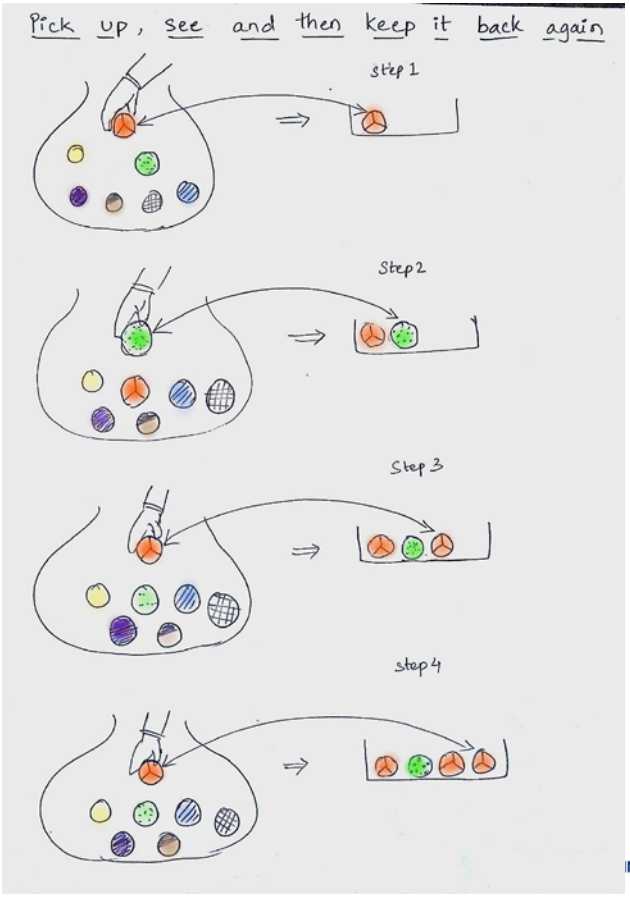
다양한 분류기로 학습



다수결 투표로 결과 예측

03-2 배깅(Bagging) 과 페이스팅(Pasting)

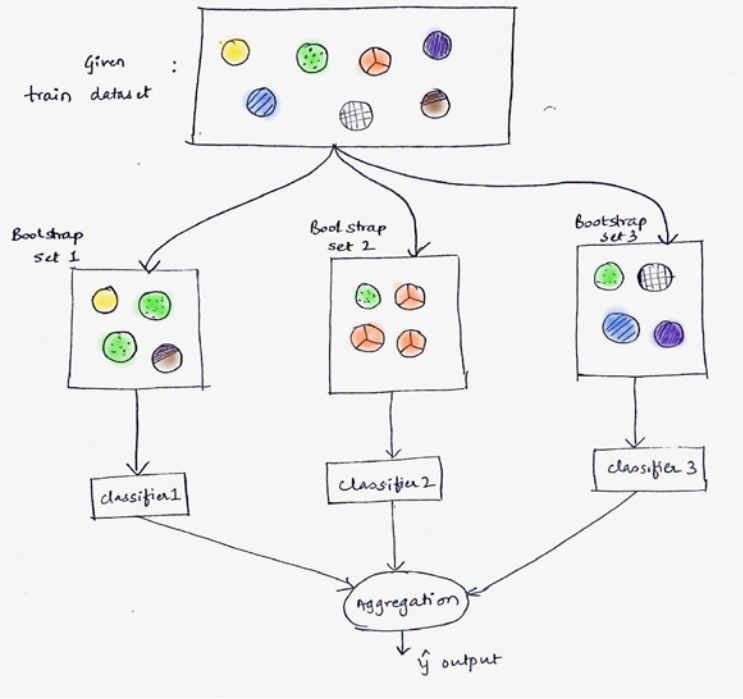
- 모델이 같아도 다른 학습 데이터 셋을 사용하여 학습하면 다른 분류기가 만들어진다.
  ➔ 학습 데이터 셋을 다양하게 해서 여러 개 분류기 만들어 앙상블을 수행
- 원래의 학습 데이터 셋에서 새로운 학습 데이터 셋을 샘플링하는 방법
  - 배깅(bagging) : 중복을 허용하는 샘플링. Bootstrap Aggregation의 줄임말
    통계학에서는 중복을 허용하는 샘플링을 부트스트래핑(bootstrapping)이라고 함
  - 페이스팅(pasting) : 중복을 허용하지 않는 샘플링



- 배깅은 중복 추출 방법을 사용하기 때문에 같은 데이터가 한 데이터셋에 여러 번 추출될 수도 있고, 어떤 데이터는 추출되지 않을 수도 있다.
- 데이터가 충분히 큰 경우, 각 데이터가 하나의 붓스트랩 표본에서 제외될 확률은 36.78%이다

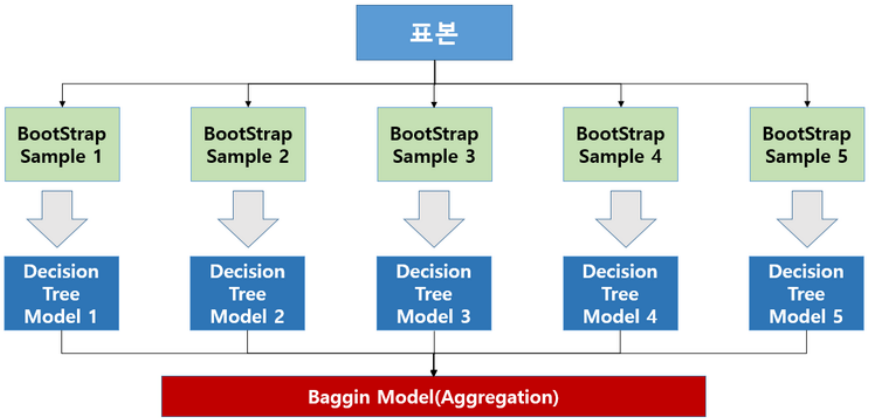
$$\left(\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}=e^{-1}=0.3678\right).$$

03-3 배깅: adabag 패키지의 bagging() 함수 이용 예



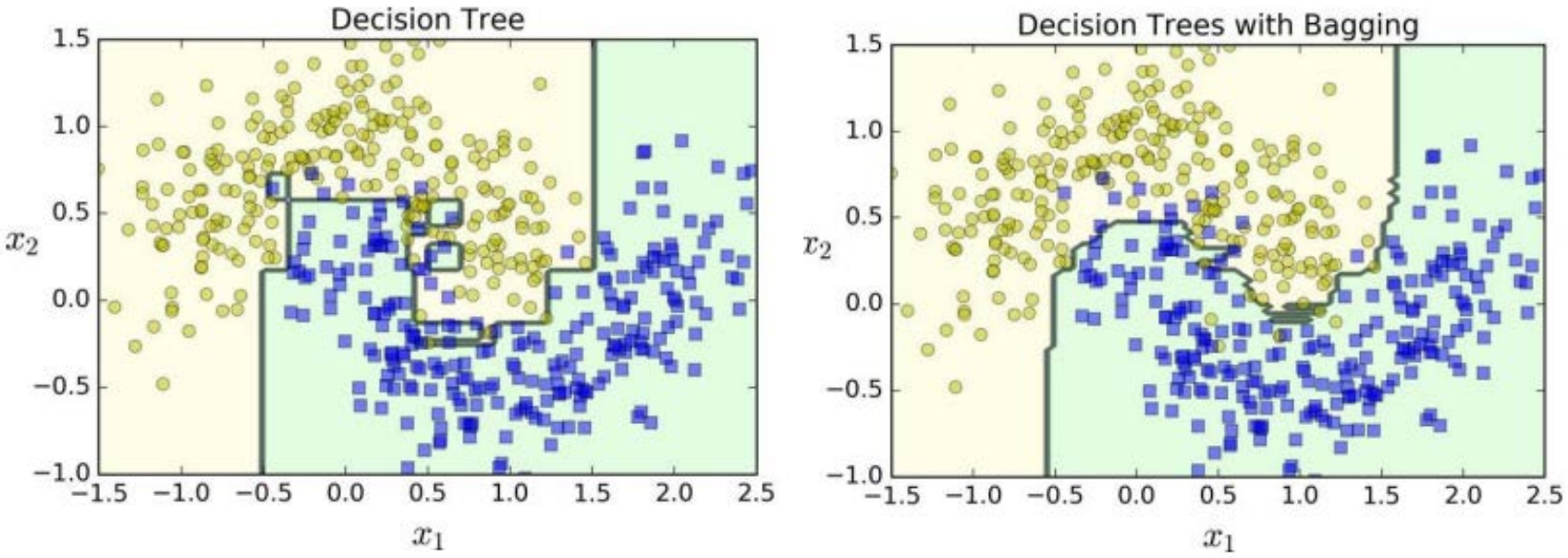

```
> library(adabag)
> data(iris)
> iris.bagging <- bagging(Species~., data=iris, mfinal=10)
> # mfinal= 반복수 또는 트리의 수(디폴트=100)
> iris.bagging$importance    # 변수의 상대적인 중요도
Petal.Length Petal.Width Sepal.Length Sepal.Width
    76.70895    23.29105      0.00000     0.00000
```

```
> pred <- predict(iris.bagging, newdata=iris)
> table(pred$class, iris[,5])
            setosa versicolor virginica
setosa          50          0         0
versicolor       0         45         1
virginica        0          5        49
```



bagging() 함수는 기본적으로 의사결정나무 함수인 rpart() 함수를 이용하기 때문에 control 옵션에 rpart.control() 함수를 이용하여 각각 의사결정나무의 옵션(maxdepth, minsplit, cp 등)을 설정할 수 있다

03-3 배깅: 성능 비교



- **한 개 결정트리 ←→ 50개 결정트리의 앙상블**
  - 결정경계가 더 부0드러운 것은?
  - 복잡도가 낮은 것은? 분산이 작은 것은? 편향이 큰 것은?
  - 일반화가 잘 되는 것은?
  - 예측하는데 시간이 더 걸리는 것은?
  - 계산시간이 문제가 안되면 어떤 것을 선택하겠는가?

03-3 배깅: oob 평가 (out-of-bag evaluation)

- 배깅에서 몇몇 샘플은 여러 결정트리에 중복되어 샘플링 되고, 어떤 샘플들은 전혀 샘플링 되지 않을 수 있다.
- 배깅에서 데이터가 충분히 큰 경우, 각 데이터가 하나의 붓스트랩 표본에서 제외될 확률은 36.78%이다

$$(\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}=e^{-1}=0.3678).$$

- (ex) 1,000개 샘플에서 한번에 하나씩 무작위로 1,000번 꺼냈을 때 (bagging)
  한번도 선택이 되지 않은 샘플의 개수는?

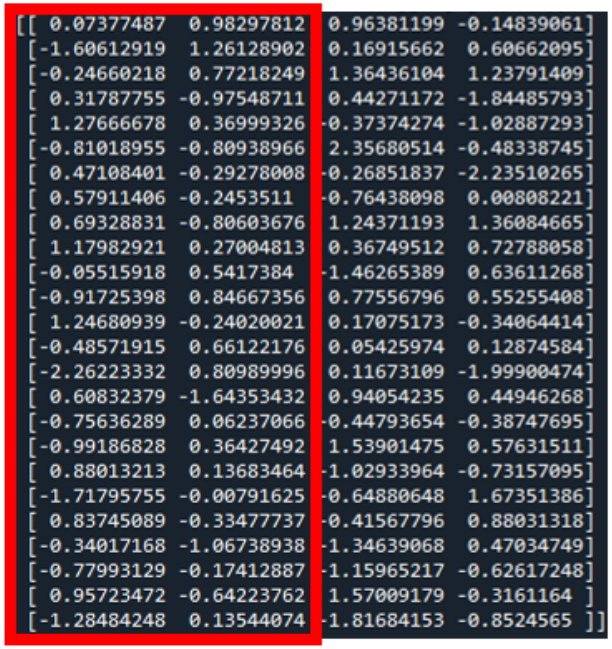
  한번 샘플링할 때 어떤 샘플이 선택되지 않을 확률 : $1-\frac{1}{1000}$

  1,000번 샘플링할 때 어떤 샘플이 선택되지 않을 확률 : $\left(1-\frac{1}{1000}\right)^{1000}\approx 0.36$ 63,2%만 선택된다.

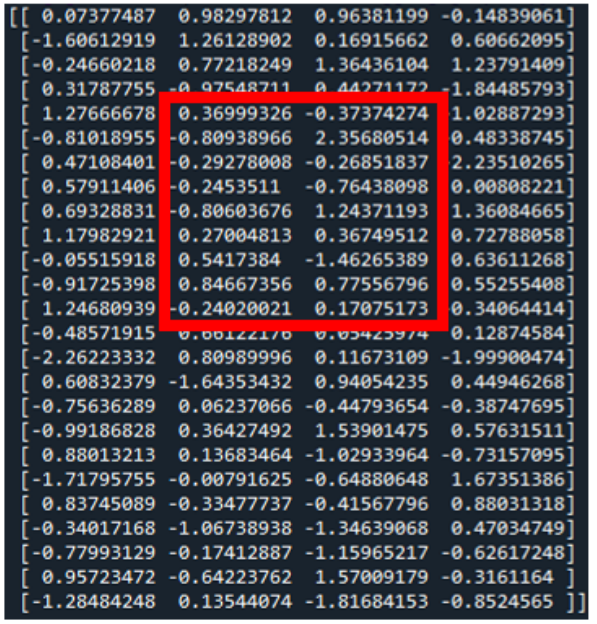
- 학습 샘플의 약 63%만이 각 결정트리에 샘플링 되어 할당되고, 나머지 약 37%는 샘플링 되지 않는다. 이 들을 oob 샘플이라고 부른다. 단, 각 결정트리에 대한 37%의 oob 샘플들이 동일하지 않다.
- 학습에 한번도 선택이 안된 샘플(out-of-bag instances)들을 이용하여 평가에 사용함.

03-3 배깅: 랜덤 서브스페이스와 랜덤패치

- 랜덤 서브스페이스(random subspace method)
  - 특징(설명 변수)이 많은 경우에 사용
  - 특징/설명변수에 대하여 배깅 (feature or attribute bagging)
  - 단, 샘플에 대하여는 배깅을 안하고 모든 샘플을 사용
- 랜덤패치(random patch method)
  - 샘플과 특징 모두를 배깅하는 방법



랜덤 서브스페이스에 의하여 생성된 데이터 셋

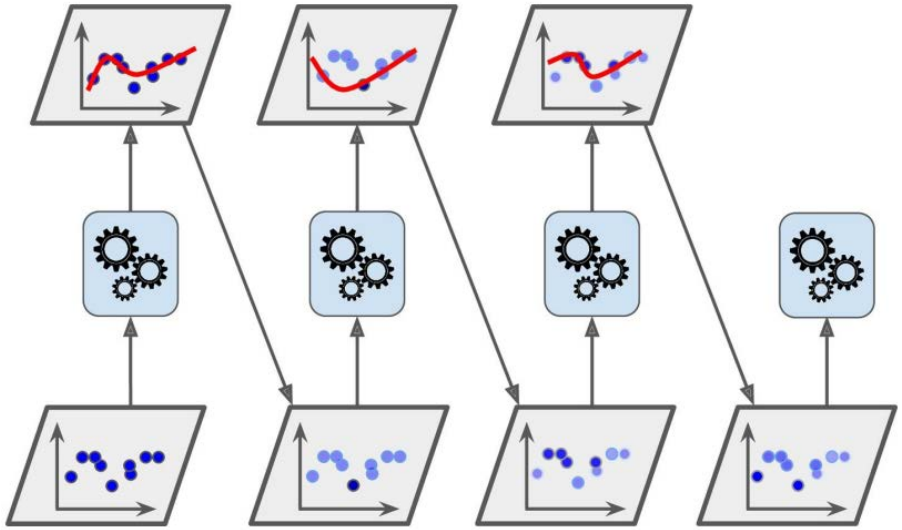


랜덤 패치에 의하여 생성된 데이터 셋

03-4 부스팅 (Boosting)

- 가설 부스팅 (Hypothesis Boosting)
- 약한 학습기를 여러 개 연결해서 강한 학습기를 만드는 앙상블 방법

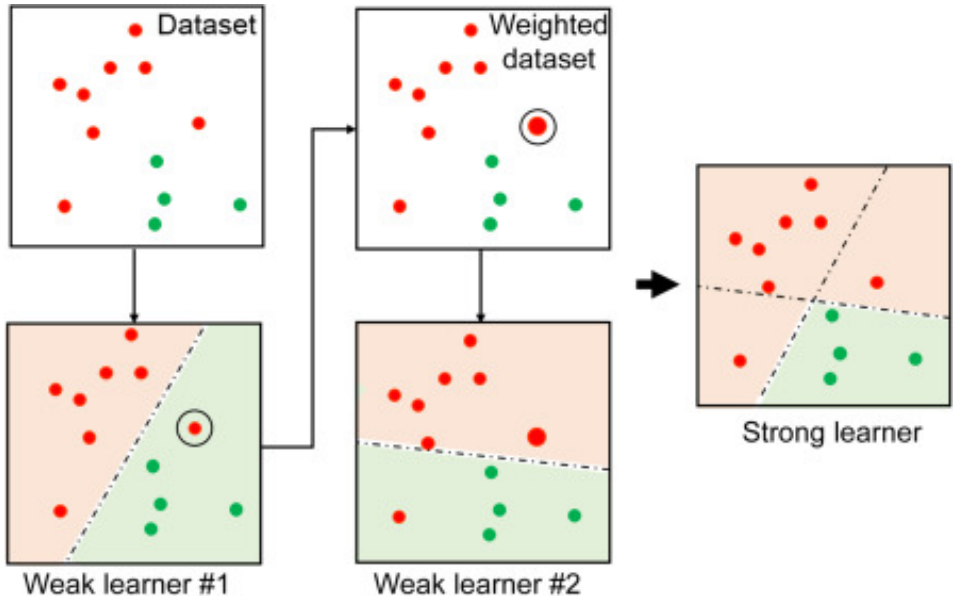
  ➔ 앞 모델을 보완해 나가도록 모델을 학습하여 연결해 나감
- 단점: 연속해서 모델을 학습해야 하므로 병렬화 하지 못함
- AdaBoost (Adpative Boosting), Gradient Boosting 이 있음



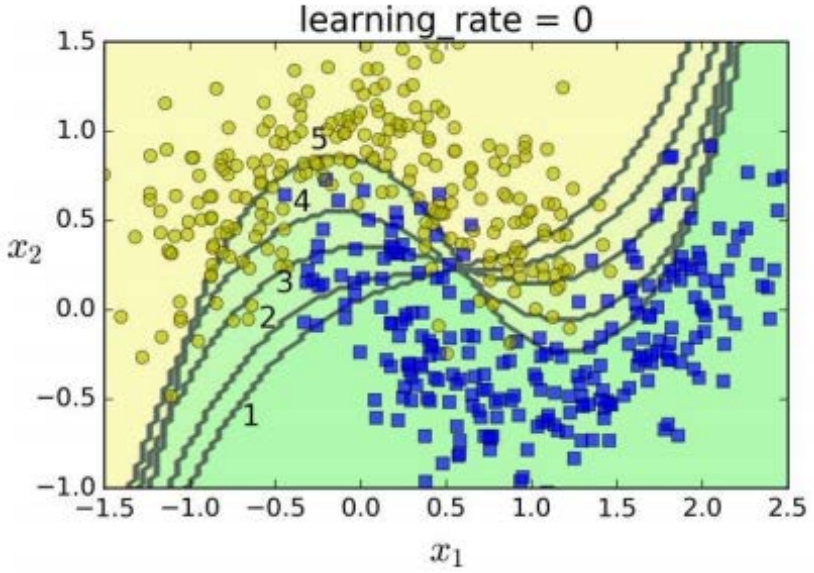
AdaBoost: 이전 모델이 놓친 샘플에 가중치를 크게 해서 다시 학습

03-4 부스팅 (Boosting) - AdaBoost

- Combining **weak learners** (decision trees)
- Assigning **weights to incorrect values**
- **Sequential tree growing** considering past mistakes



AdaBoost: 이전 모델이 놓친 샘플에 가중치를 크게 해서 다시 학습

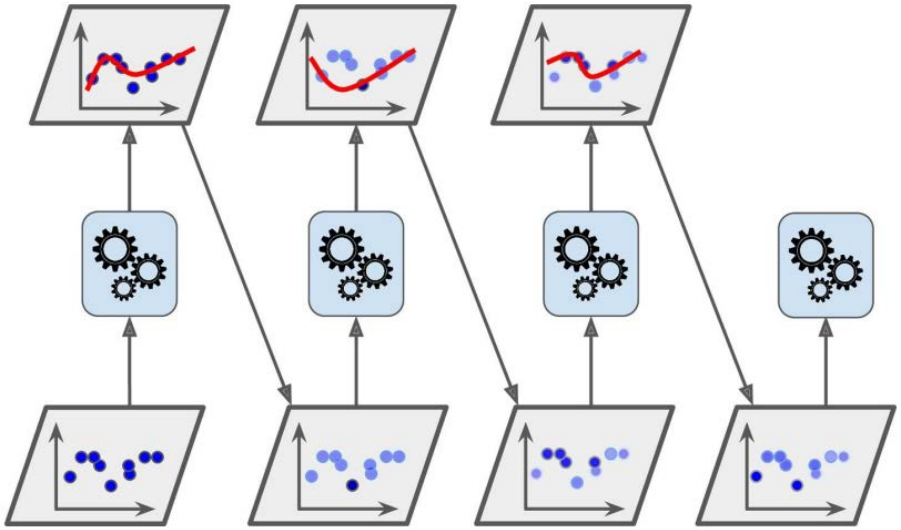


- 1번째 모델에서 틀린 샘플에 대해 가중치 주어 2번째 모델을 학습.
- 계속하면 오차가 작은 방향으로 경계를 찾아감

03-4 부스팅 (Boosting): boosting() 함수 이용 예

```
> library(adabag)
> data(iris)
> boo.adabag <- boosting(Species~., data=iris, boos=TRUE, mfinal=10)
> boo.adabag$importance
 Petal.Length  Petal.Width Sepal.Length  Sepal.Width
    61.458478    19.826149     6.216103   12.499270
```

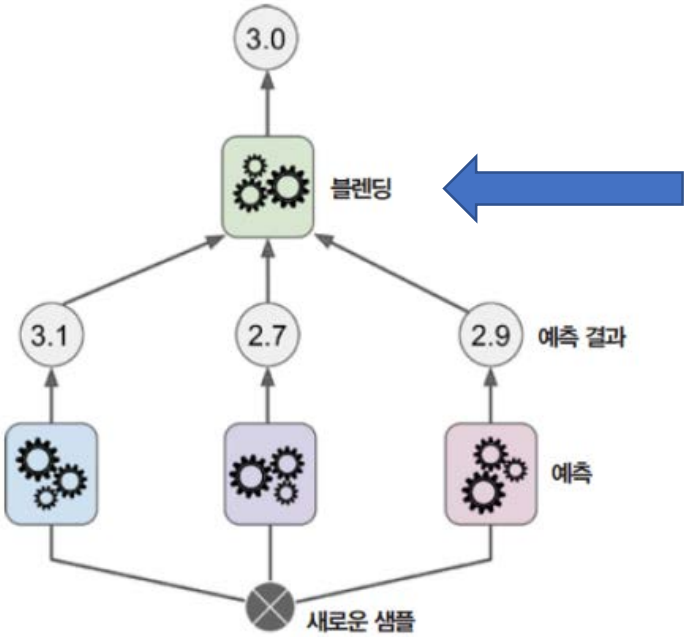
```
> pred <- predict(boo.adabag, newdata=iris)
> tb <- table(pred$class, iris[,5])
> tb
            setosa versicolor virginica
 setosa         50          0         0
 versicolor      0         50         0
 virginica       0          0        50
```



AdaBoost sequential training with instance weight update

03-5 스태킹 (Stacking)

- 스태킹 기법은 서로 다른 종류의 단일 모델들을 혼합한다는 특징이 있다.
- 예측기를 혼합하는 방법을 학습하여 결정한다. 블렌터(Blender) 혹은 메타학습기(Meta Learner)라고 부른다.
- 블렌더는 개별 모델이 예측한 데이터를 다시 meta data set으로 사용해서 학습한다.



고정된 방법대신 학습으로 결정

03-5 스태킹

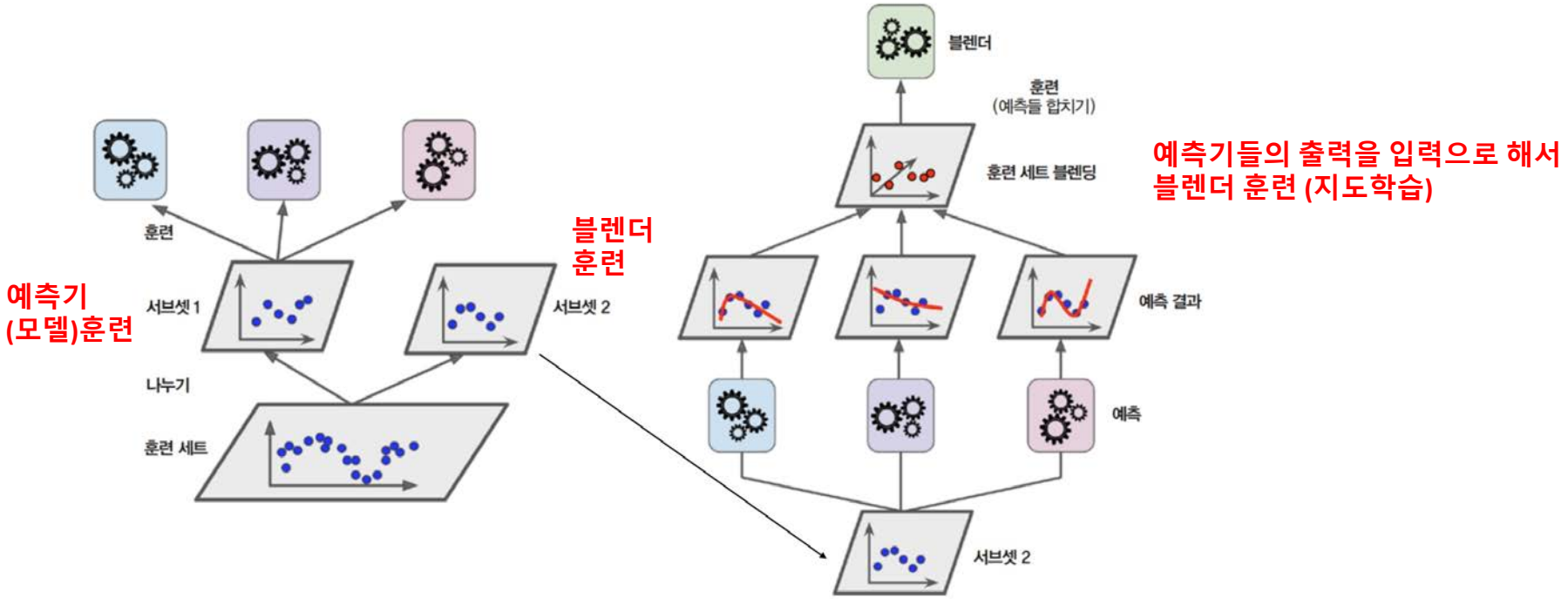
- **블렌더 훈련과정**

① 전체 트레이닝 데이터 셋을 겹치는 데이터가 없게, 2개 그룹으로 나눈다

② 여러가지 다양한 N개의 Base Learner들을 첫 번째 데이터 그룹으로 학습시킨다.

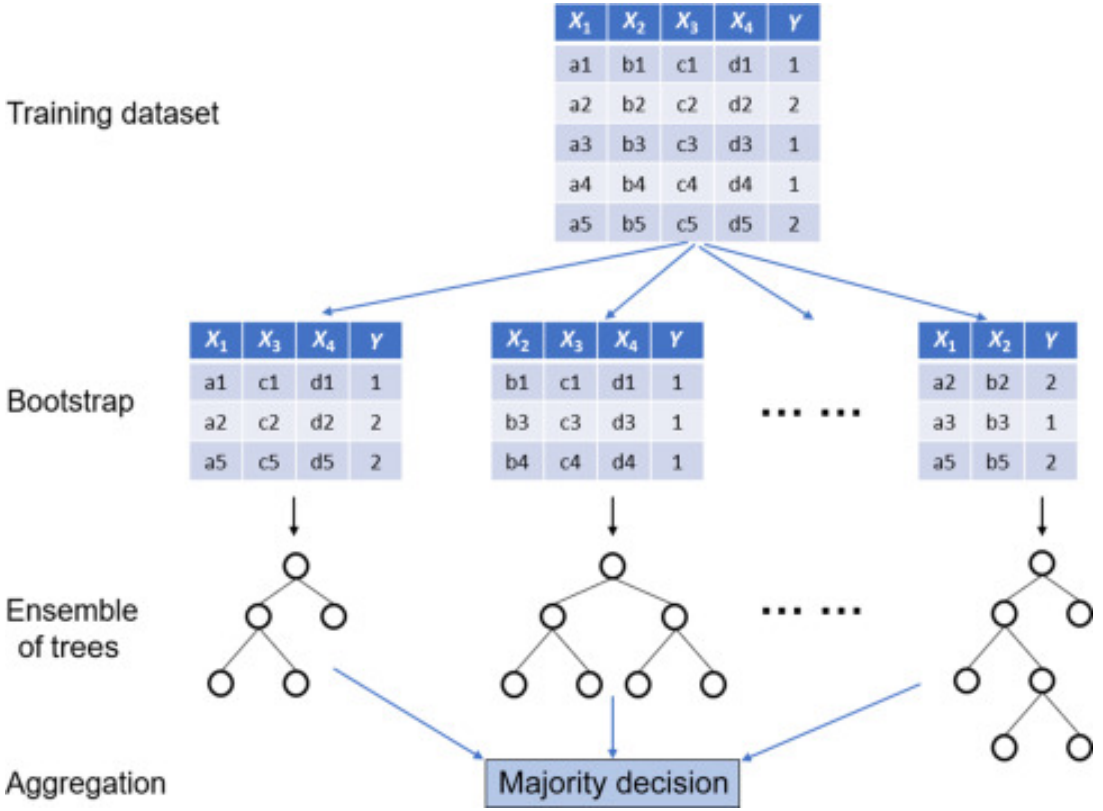
③ 학습 시킨 Base learner를 통해, 2 번째 데이터 그룹을 테스트 셋으로 받아들여, 예측값을 출력한다.

④ 출력된 N개의 예측값들을 Input으로 하고 (N-dimension의 트레이닝 데이터 생성), 2 번째 데이터 그룹의 실제값(target value)을 output으로 하여 메타학습기(블렌더)를 학습시킨다.



03-6 랜덤 포리스트 (Random Forests)

- 랜덤포리스트(random forest)는 배깅에 랜덤 과정을 추가한 방법이다.
- 원래 트레이닝 자료로부터 붓스트랩 샘플을 추출하고, 각 붓스트랩 샘플에 대해 트리를 형성해 나가는 과정은 배깅과 유사하나, 각 노드마다 모든 예측변수 안에서 최적의 분할(split)을 선택하는 방법 대신 예측변수들을 임의로 추출하고, 추출된 변수 내에서 최적의 분할을 만들어 나가는 방법을 사용한다.
- 새로운 자료에 대한 예측은 분류(classification)의 경우는 다수결(majority votes)로, 회귀(regression)의 경우에는 평균을 취하는 방법을 사용한다.



03-6 랜덤 포리스트: randomForest() 함수 이용

```
> library(randomForest)
```

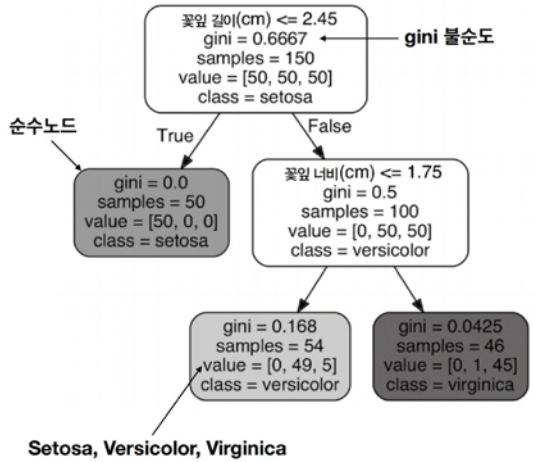
```
> f = randomForest(Species~., data=iris)
```

```
> f
Call:
 randomForest(formula = Species ~ ., data = iris)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 2

        OOB estimate of  error rate: 4%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          3        47        0.06
```

꽃받침길이/너비 꽃잎길이/너비

		Sepal.Leng	Sepal.Wid	Petal.Leng	Petal.Widt	Species
1						
2	1	5.1	3.5	1.4	0.2	setosa
3	2	4.9	3	1.4	0.2	setosa
4	3	4.7	3.2	1.3	0.2	setosa
52	51	7	3.2	4.7	1.4	versicolor
53	52	6.4	3.2	4.5	1.5	versicolor
54	53	6.9	3.1	4.9	1.5	versicolor
149	148	6.5	3	5.2	2	virginica
150	149	6.2	3.4	5.4	2.3	virginica
151	150	5.9	3	5.1	1.8	virginica

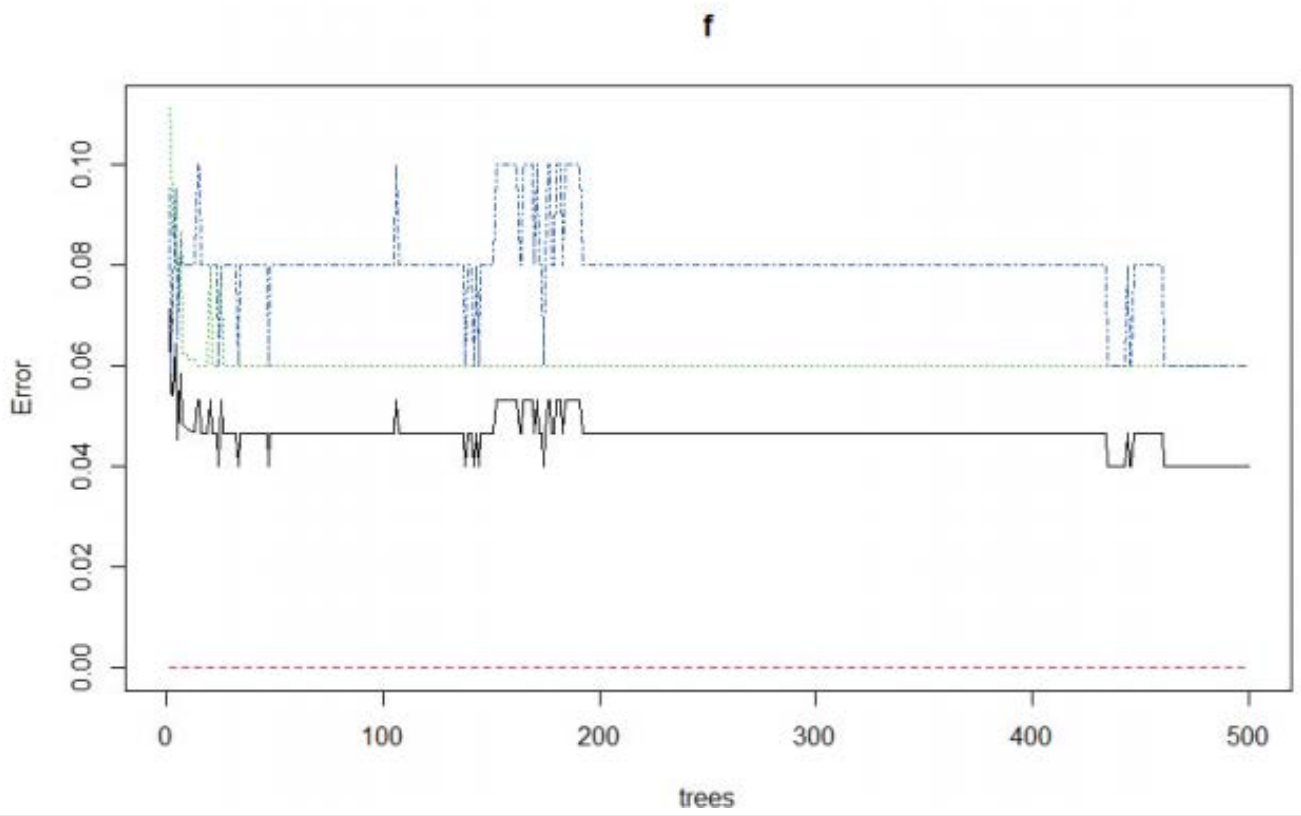


- 500개의 결정 트리를 생성
- 오류율에 대한 OOB(out-of-bag) 추정치를 제공한다. 랜덤포리스트에서는 별도의 검증용 데이터를 사용하지 않더라도 붓스트랩 샘플과정에서 제외된(out-of-bag) 자료를 사용하여 검증을 수행하여 오류율을 제공한다.
- 훈련 집합에 대한 혼동 행렬을 보면 6개 샘플을 잘못 분류함
  → 정확률은 144/150=96%

03-6 랜덤 포리스트: 시각화-plot() 함수 이용

```
> f = randomForest(Species~., data=iris)
> plot(f)
```

plot() 함수는 트리 수에 따른 설명 변수의 오분류율을 나타낸다



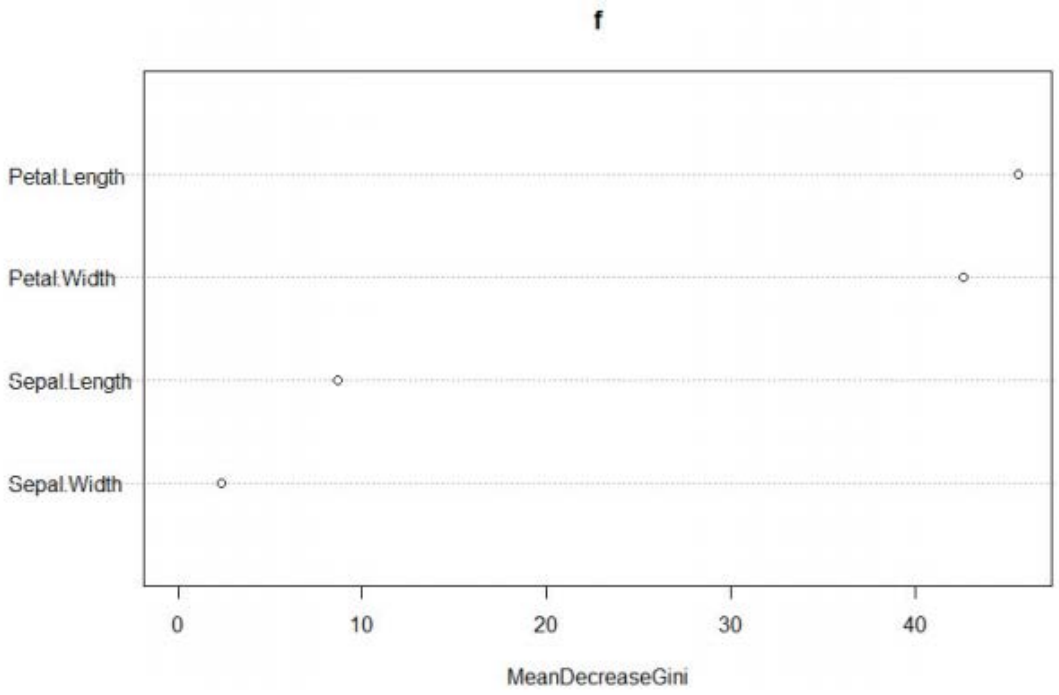
가로축은 결정 트리의 개수를 나타냄
세로축은 랜덤 포리스트의 오분류류율 (빨강, 파랑, 녹색은 각각 setosa, versicolor, virginica의 오분류율. 검정은 평균 오분류율)을 나타냄

03-6 랜덤 포리스트: 시각화-varImpPlot() 함수 이용

```
> f = randomForest(Species~., data=iris)
```

```
> varImpPlot(f)
```

varImpPlot() 함수는 설명 변수의 중요도를 나타낸다



각 변수의 중요도를 나타내는 그림으로, 해당 변수로부터 분할이 일어날 때 불순도(impurity)의 감소가 얼마나 일어나는지를 나타내는 값이다(불순도의 감소가 클수록 순수도가 증가함). 지니 지수(Gini index)는 노드의 불순도를 나타내는 값이다.

03-6 랜덤 포리스트: treesize() 함수 이용

```
> f = randomForest(Species~., data=iris)
```

```
> treesize(f)
  [1] 11  6  4  7 12  7  9  8  6 12  8  6 10  7  7 12 12 10  5  9  9 12 12  9 13  8 12  8 11 10  8 11  8  9  5  8  9
 [38] 10  7 13  6  8 10  6  7 10  8  7  8  8  9 15  8 11  9 10  8 15  8  9  9  9 11 12  8 10 10 11 14  9  8  7  9  5
 [75]  6  9  7  8  9  5  8 11  7  7 10  9  9  7 11  6  7  7 13  7  7  7  7 11  6  5  6 12 11  9  7  8  7  6  8 11  9
[112]  8 10  7  7  7  4  9  8  7  7  8 11  6  9 11 11 10  5  5  9  9  9  8  9 11  9  8 16  9  9  6  7  7  6 10  6  5
[149]  9 10 10  9 14  7  8  9  7  6  9 10  7  5  6 10  8  8 12  6 12  9  4  9  7 12  8  8 13  6  5 10  9  7  9 13  7
[186]  6  9  6 14 10  9 10  6 10  8  7  9  8  9  8  8 13  7  8 10  9  4  7  9 11  8  7 10  6  7 13 10  8  9 10 10  6
[223]  9 12  6  9  6  4 10  8  9 11  9 13  9  9  9  9 10  9 12 10  9  7 10 11  8  7 12 11  7 15 14 12  8 10  5 11 13
[260] 14  7  7 10  9  8  7 11 10  6  7 13  8  8 12  9 11 13  5 10 10 11  8  8 12  7 10  9  7  7  9  9  8  5  6  8  7
[297] 14  9  8  9  4  7  6  9  7  8  7  6  6  9  9 10  9  8 12  9  4  4 12  9  8  8 11 11  8 14  8  6  7  8  8  7 10
[334]  7 10  9  8 11  9  8 12  9  9 12 10  8 11  9  6  9  9  8  9  6 10 11  7  5  6  6  7 10  7  8 11 10  6 11  6  8
[371]  8  6  7  8 10  7 10 11  5  8 11 10 10  9  7  9 10 11 14 10  7  9  5  9  7  9  9  8  7  9 10  3  6  6 10 12  7
[408]  9  8  9 11  7  9 10 10 10  4  6  7  8 10  8 10 11  8  6  7  8 11  9 10 11  8 13  7  6  7 10  5 12 13  8  7  8
[445]  6 12  6  8  8  7  7 12 12  6 12  7 10  4  8  8  9  7  7  8  8 12  6  7 14  5  5 11  7 10  8 11  6 10  8  8  9
[482] 10  9  8 10 12 10 11  8 11  9 10 10 11  7 10 10  8  6  7
```

treesize() 함수는 결정 트리 각각의 리프 노드의 개수를 알려준다. 즉, 1번과 2번 결정 트리는 11개와 6개의 리프 노드를 가지고 있다.

03-6 랜덤 포리스트: predict() 함수 이용

```
> f = randomForest(Species~., data=iris)
```

```
> newd = data.frame(Sepal.Length = c(5.11, 7.01, 6.32), Sepal.Width = c(3.51,
3.2, 3.31), Petal.Length = c(1.4, 4.71, 6.02), Petal.Width = c(0.19, 1.4, 2.49))

> predict(f, newdata = newd)
         1          2          3
    setosa versicolor  virginica
Levels: setosa versicolor virginica
```

분류 정보만이 출력된다.

```
> predict(f, newdata = newd, type = 'prob')
  setosa versicolor virginica
1      1      0.000     0.000
2      0      0.976     0.024
3      0      0.002     0.998
```

type='prob' 옵션을 설정하면 확률을 출력한다. 예를 들어, 두번째 샘플이 setosa, versicolor, virginica에 속할 확률이 0.0, 0.976, 0.024임을 나타낸다.

```
> predict(f, newdata = newd, type ='vote', norm.votes = FALSE)
  setosa versicolor virginica
1    500          0         0
2      0        488        12
3      0          1       499
```

득표수를 확인하려면, type='vote', norm.votes=FALSE 설정

03-6 랜덤 포리스트: hyper parameter

하이퍼 매개변수 (hyper parameter): 모델의 구조나 학습 방법을 제어하는 데 사용하는 매개 변수를 나타낸다
- ntree: 결정 트리의 개수 (기본값은 500)
- nodesize: 리프 노드에 도달한 샘플의 최소 개수 (크게 설정할수록 작은 결정 트리)
- maxnodes: 리프 노드의 최대 개수

```
> small_forest=randomForest(Species~., data=iris, ntree=20, nodesize=6, maxnodes=12)
> treesize(small_forest)
 [1] 6 5 6 7 4 8 6 8 9 7 7 5 9 6 5 9 8 7 4 5

> small_forest
Call:
 randomForest(formula = Species ~ ., data = iris, ntree = 20, nodesize = 6, maxnodes = 12)
               Type of random forest: classification
                     Number of trees: 20
No. of variables tried at each split: 2

        OOB estimate of  error rate: 6%
Confusion matrix:
           setosa versicolor virginica class.error
setosa         50          0         0        0.00
versicolor      0         47         3        0.06
virginica       0          6        44        0.12
```

랜덤포리스트는 R 패키지 {party}의 cforest() 함수를 이용할 수도 있다.

요약

- **Random Forests**
  - ✓다양한 앙상블 모델
  - ✓randomForest() 함수를 이용한 모델 생성/예측
  - ✓결과 분석

기말 프로젝트

- 기간: 2020/11/30(월) – 12/16(수)
- 주어진 데이터에 대한 분류 모델 생성 및 성능 평가
  - ✓Logistic Regression
  - ✓Decision Trees
  - ✓Random Forests
  - ✓Support Vector Machines (SVMs)

- 자세한 사항은 14주차 실습 영상에서 설명함.