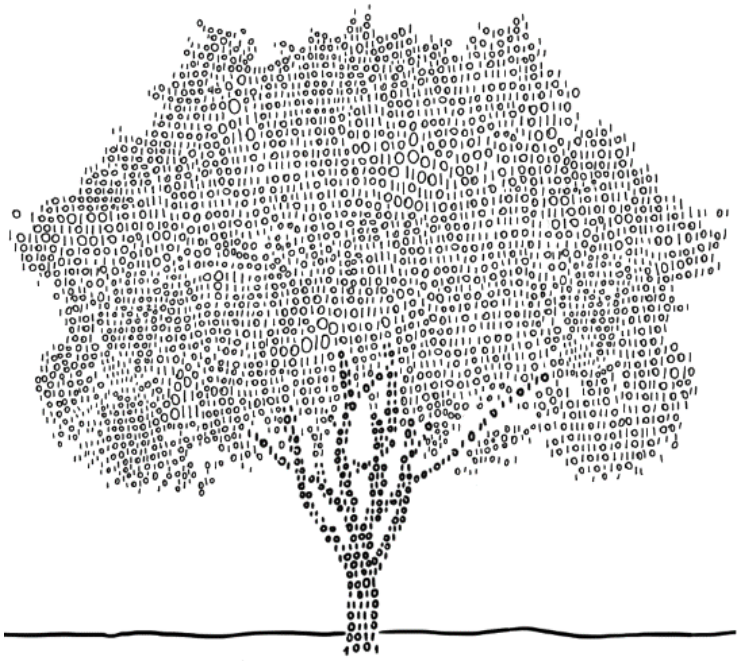
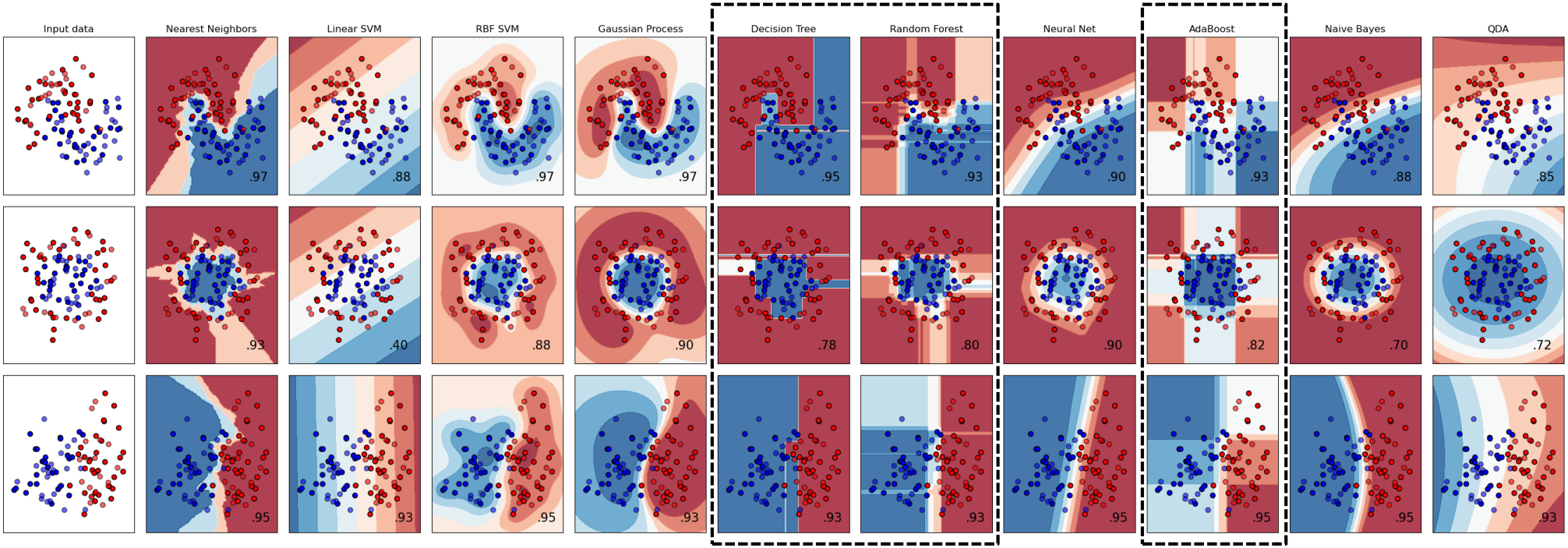
제9장. 분류(分類, Classification)

- Tree based classification -



Classification

- 3가지 다른 분포를 가진 이항분류된 자료를 다양한 분류 알고리즘으로 분류예측하여 그 경계선을 시각화
- 훈련데이터에서 학습한 모델로 시험데이터를 분류한 결정 경계선(decision boundary)을 표시
- 다양한 알고리즘은 다양한 결정 경계선을 보여주고 있으며 분류 예측 확률이 높을수록 색의 채도가 높음
  - 점선 박스로 표시된 알고리즘 종류가 tree-based classification 알고리즘



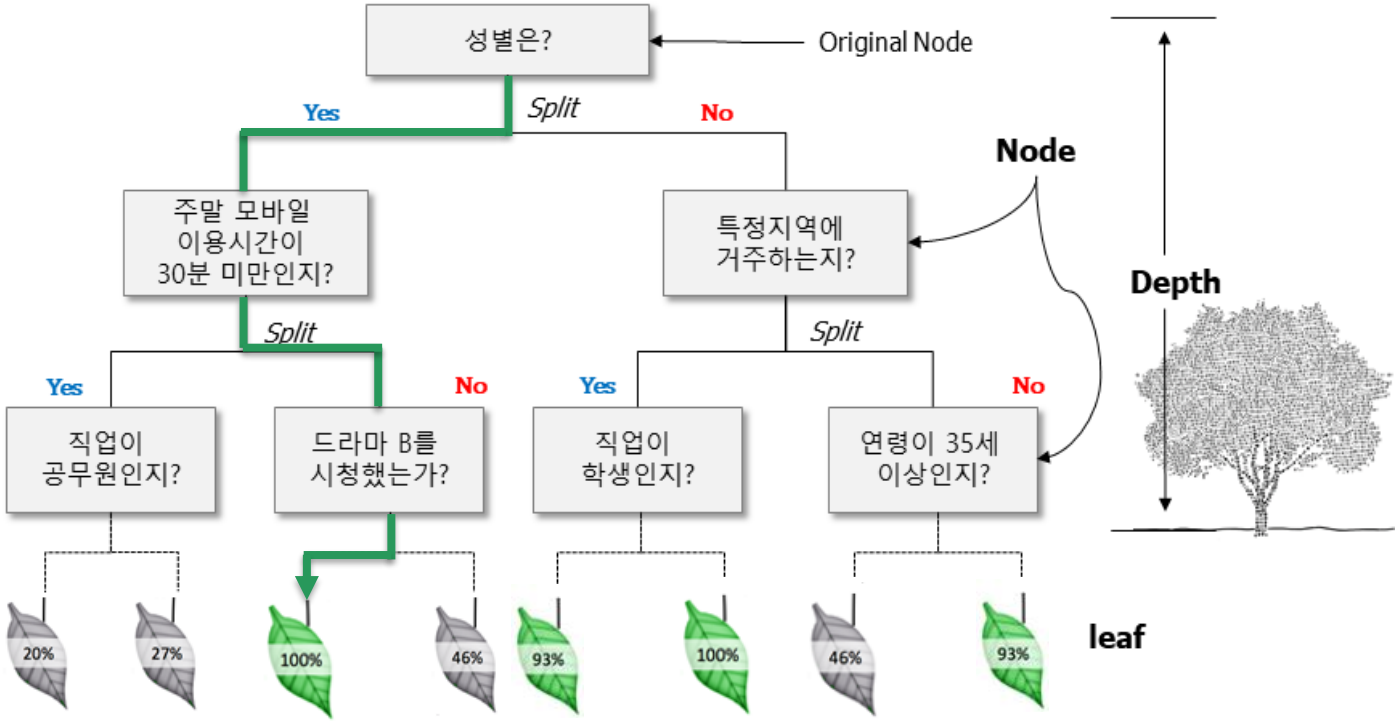
https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

목차

- 의사결정 나무(Decision Tree)
- 앙상블(Ensemble: Bagging, Random Forest)

Decision Tree

- 스무고개 놀이(20 Questions game) : 처음 질문의 '예/아니오 ' 둘 중 하나의 응답에 따라 다음 질문을 하면서 확신이 생기는 단계에 답을 맞추는 수수께기 놀이와 동일한 원리
- '남자'이고 '주말 모바일 이용시간이 30분 이상'이며 '드라마 B를 시청한' 사람은 100%확률로 상품을 구매

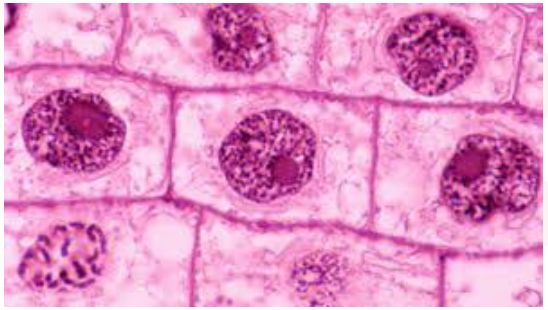


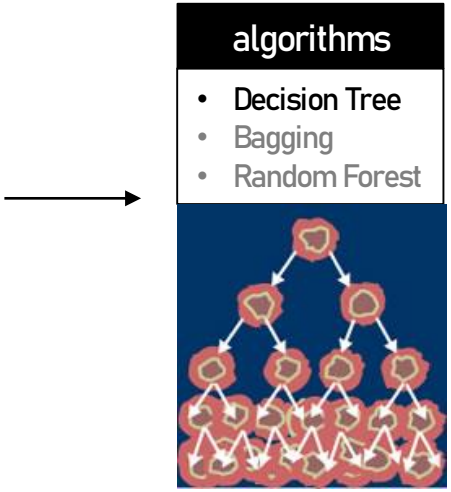
문제 및 해결 방안

- 50개 UCI Breast cancer 세포핵 이미지 자료(target의 1의 의미는 비정상, positive)
- 2개의 특성변수(세포핵 외벽의 '평탄도'와 '길이')를 사용하여 진단 정확도(Accuracy)가 높은 예측 모델을 개발

Input (X) Output (y)

worst smoothness	worst perimeter	target
0.16000	152.10	1
0.10050	97.66	0
0.13160	68.81	0
	58.36	0
	123.80	1
	93.22	0
	115.90	0
	113.70	0
0.12650	67.88	0

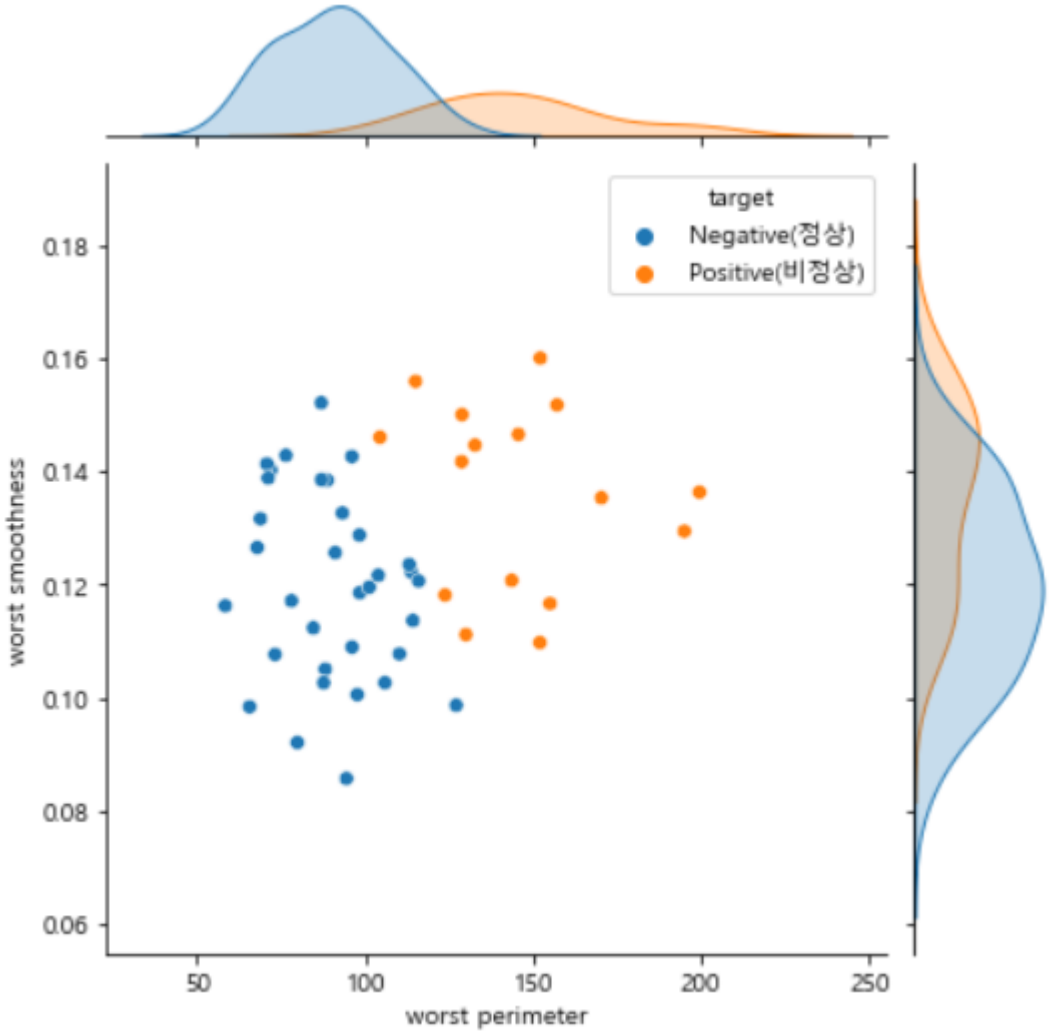




$f : X \rightarrow y$

- 모델 = 알고리즘(데이터)

분류를 가장 잘할 수 있는 방법



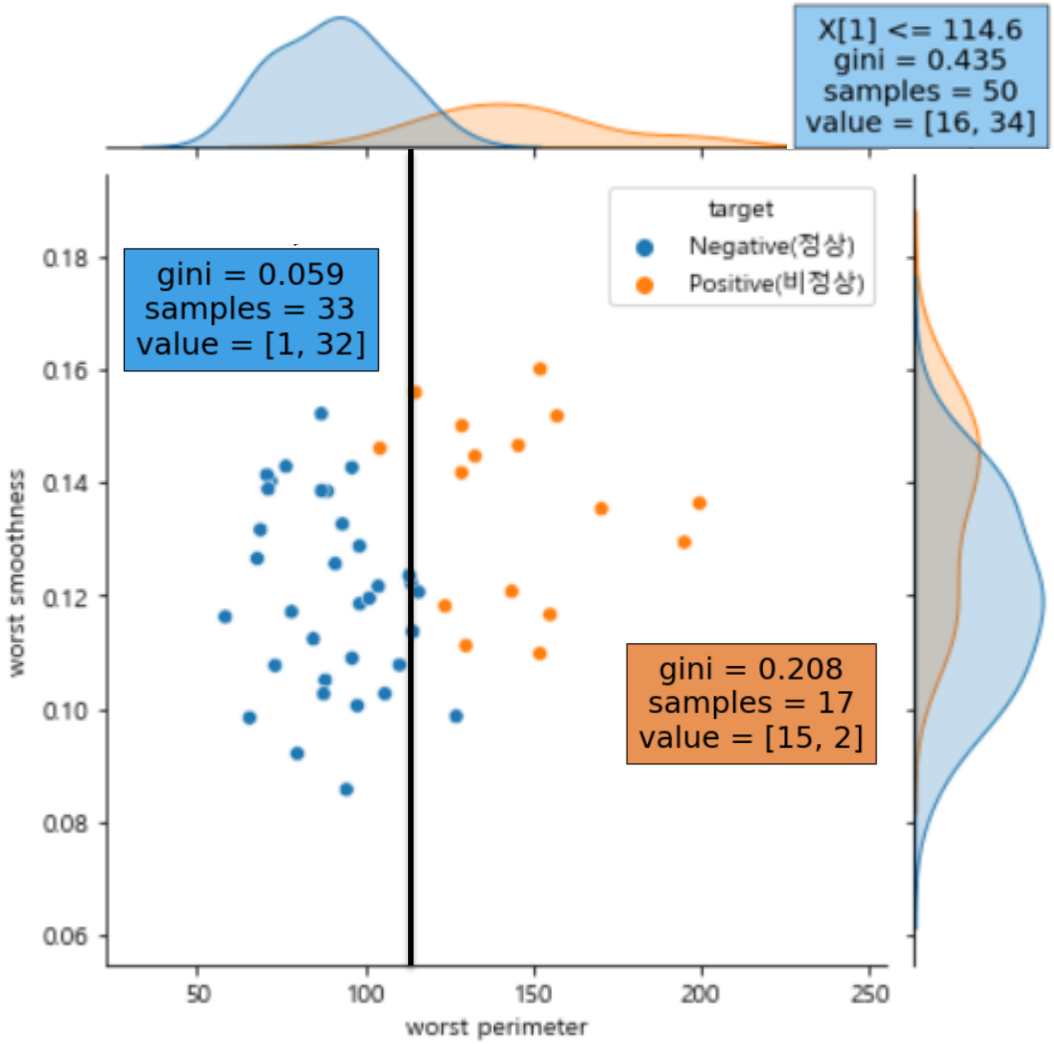
- 특성변수가 없을 경우에는 항상 정상이라고 예측하면 정확도가 68%

[Level 0 비중]

Target	건수(비율)	예측
정상	34(68%)	Negative (정상)
비정상	16(32%)	

- 세포핵 외벽 '평탄도'와 '길이' 2개의 특징이 주어지면, 과연 어떤 첫 질문(노드분리 변수와 값)을 하여야 예측을 잘 할 수 있을까 ?

1단계 노드분리 변수와 분리 값의 결정



- 첫번째 질문은 세포핵 외벽 '길이'가 '114.6'이하인지 아닌지로 데이터를 구분하여 예측

Worst perimeter	Target	비중(%)	예측
<= 114.6	비정상	1/33(3%)	
	정상	32/33(97%)	정상
> 114.6	비정상	15/17(88%)	비정상
	정상	2/17(12%)	

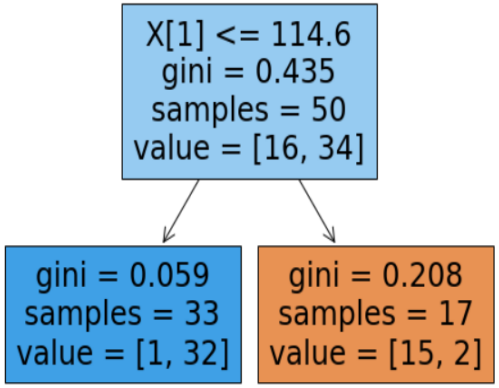
- 컴퓨터가 이해할 수 있도록 gini를 계산하여 노드 분리를 자동으로 하게 함

지니 지수(Gini Index)

- 지니지수는 정보의 불확실성이라는 추상적인 개념을 구체적으로 계량화한 지표
- 의사결정나무는 모든 변수를 대상으로 지니지수(Impurity, Uncertainty)가 낮은 순으로 데이터 분류(Node Splitting)를 순차적으로 수행하면 불확실성이 감소되어 예측 성능이 높아짐

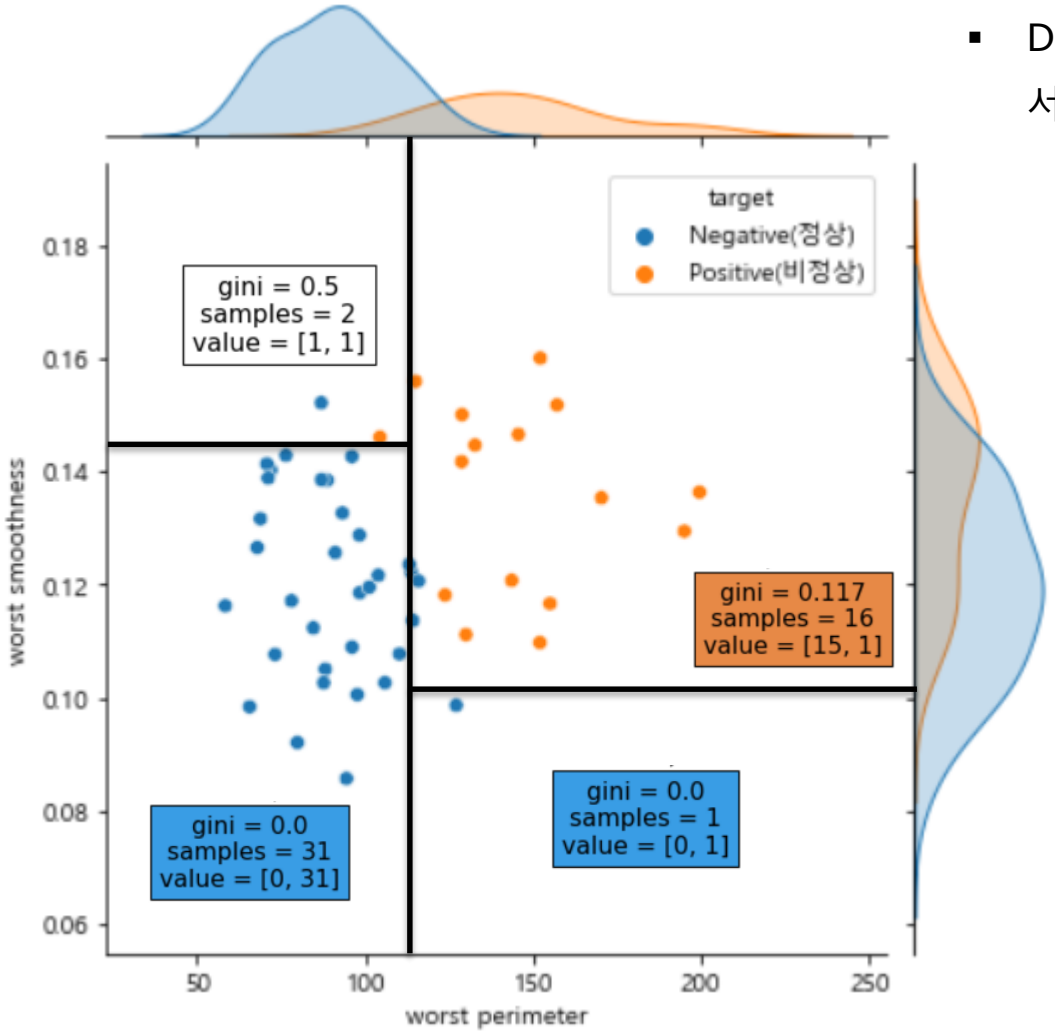
$$\text{지니지수 } (Gini\ Index) = \sum_{C=1}^{2} P_C * (1 - P_C) = 2P_1 * (1 - P_1)$$

Worst perimeter	Target	비중(%)	노드 gini	*Depth gini
<= 114.6	비정상	1/33(3%)	**0.059** = 2(3% x 97%)	**0.110** = 0.059*(33/50) + 0.208*(17/50)
	정상	32/33(97%)		
> 114.6	비정상	15/17(88%)	**0.208** = 2(88% x 12%)	
	정상	2/17(12%)		

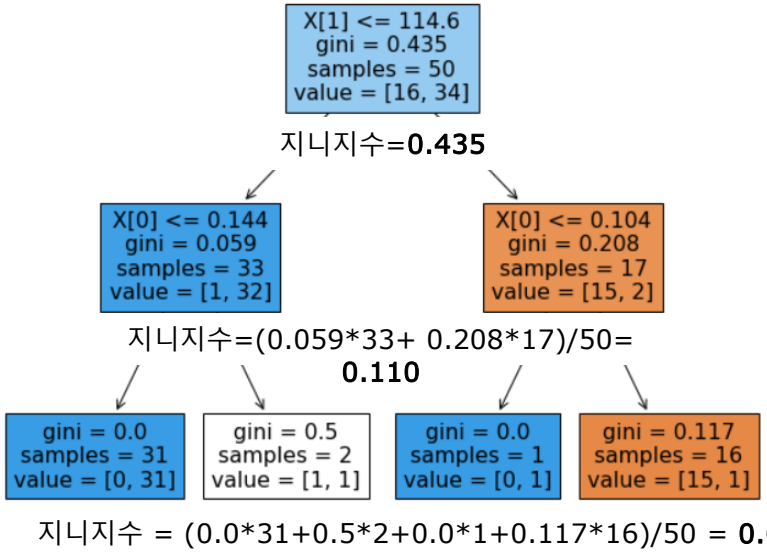


* 각 노드의 데이터 개수를 가중치 평균

2단계 노드분리 변수와 분리 값의 결정



- Depth 2에서는 Depth 1보다 지니 지수가 감소되면서 데이터를 분할(노드분리와 분리값 결정)



예측	정상	정상과 비정상	정상	비정상
	100%	50%	100%	88%

데이터와 모델 비교

- 총 568개의 샘플과 특성변수(수치형) 30개로 구성된 Input(X)과 Output(y) 자료를 바탕으로 의사결정 나무 → Bagging → Random Forest 모델의 정확도(Accuracy)를 비교 · 평가

Input (X)

Output (y)

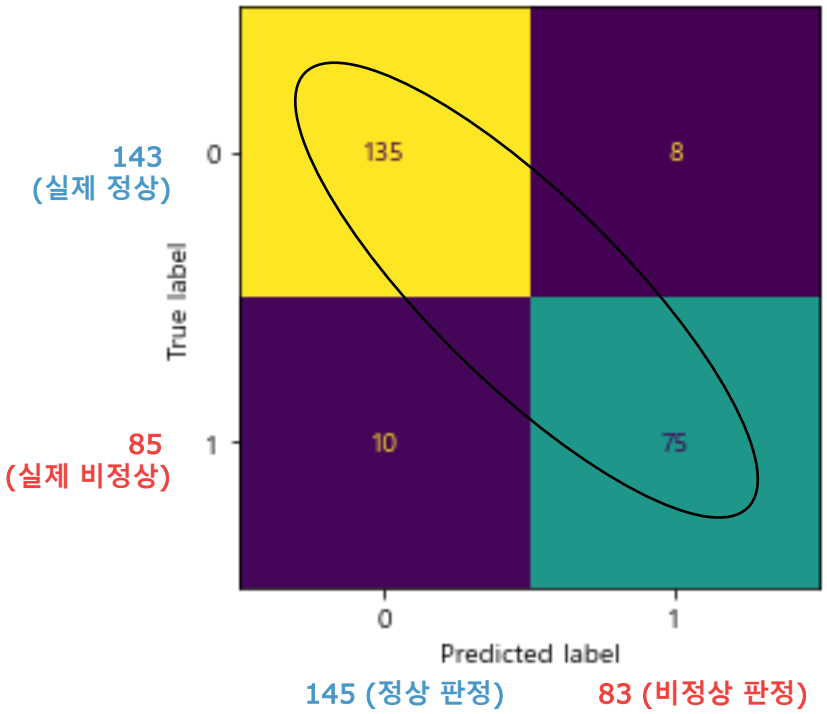
	worst smoothness	worst symmetry	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	...	worst fractal imension	target
0	0.16220	0.4601	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	0.11890	0
1	0.12380	0.2750	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	0.08902	0
2	0.14440	0.3613	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	0.08758	0
3	0.20980	0.6638	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	0.17300	0
4	0.13740	0.2364	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	0.07678	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	0.14100	0.2060	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	0.07115	0
565	0.11660	0.2572	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	0.06637	0
566	0.11390	0.2218	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	0.07820	0
567	0.16500	0.4087	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	0.12400	0
568	0.08996	0.2871	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	0.07039	1

569 rows × 31 columns

Decision Tree 오차 행렬

- 총 568개 중 340개 샘플(train)로 학습・훈련하여
- 228개(test)를 대상으로 실제 정상·비정상의 자료를 판정하면 210개를 맞추는 92.11%의 정확도(Accuracy)

오차 행렬(Confusion Matrix)



정확도(Accuracy)= (135+75)/228 = 92.11%

Python script (Scikit learn Estimator API)

```
from sklearn.datasets import load_breast_cancer
X, y = load_breast_cancer(return_X_y=True)
```
1. 데이터 준비

```
from sklearn.tree import DecisionTreeClassifier
```
2. 알고리즘 선택

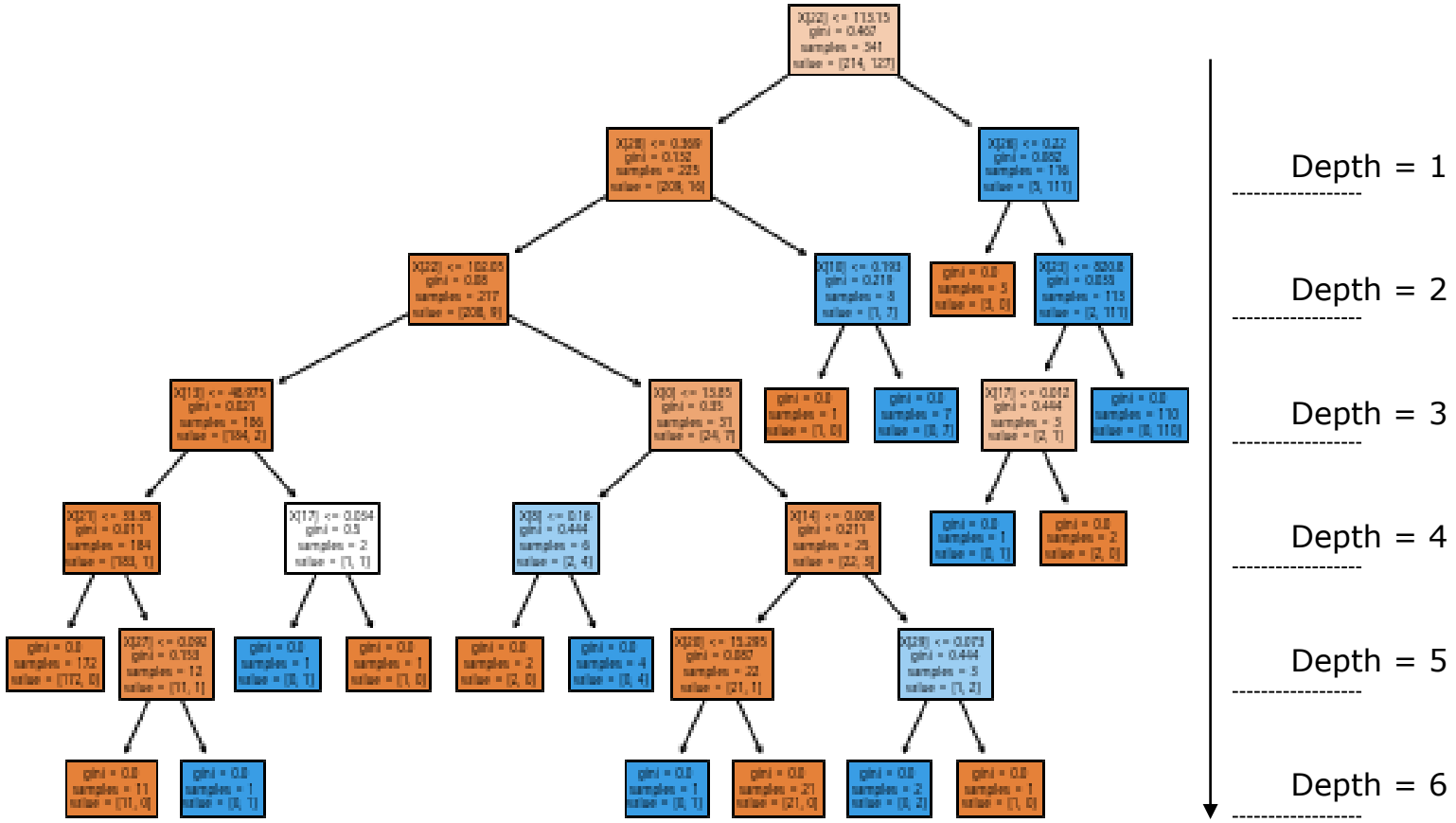
```
model = DecisionTreeClassifier()
```
3. 알고리즘 객체화

```
model.fit(X_train, y_train)
```
4. 학 습

```
model.predict(X_test, y_test)
```
5. 예 측

모델의 과적합(Overfitting)

- 의사결정 나무 특성상 자료의 레이블이 완전히 분류될 때까지 노드 분리(node splitting)를 하면서 훈련하기 때문에 과적합(Overfitting)이 발생하기 쉬움 → Depth를 줄여 일반화 모델 필요



콩도르세의 배심원 정리

- 평범한 판단력을 가진 배심원이 모여 독립적으로 판결하게 되면 항상 모든 재판에서 올바른 판결이 가능
- 집단 지성(The wisdom of crowds)

Nicolas de Condorcet's Jury Theorem
(20 Sep 1792 – 8 Jul 1794)

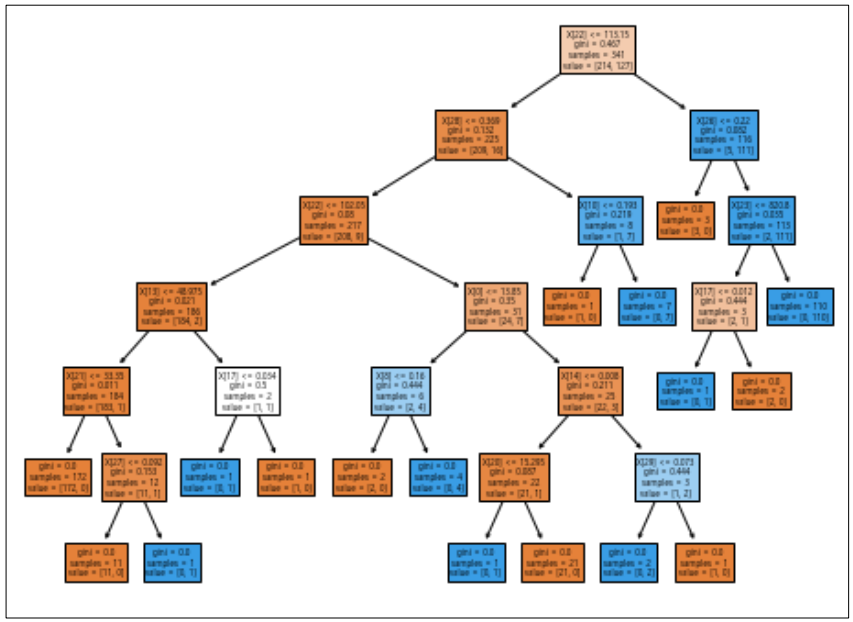
✓ 판단력이 55%인 배심원 5명이
✓ 각각 독립적 판단하면,
✓ 재판회수 200회 부터 항상 올바른 판결(50% >)


```
import numpy as np
jurors_proba = 0.55 ; num_jurors = 5; num_trals = 1000
num_decisions=(np.random.rand(num_trals, num_jurors)<jurors_proba)*1
total_trials = np.arange(1, num_trals*1).reshape(-1, 1)
cum_right_ratio=np.cumsum(num_decisions, axis=0)/total_trials
plt.figure(figsize=(20,8)); plt.plot(cum_right_ratio)
plt.plot([0, num_trals], [0.5, 0.5], 'k--', label='50% Justice')
```

여러 개의 의사결정 나무

- 복잡한(Depth가 높은) 의사결정 나무로 예측하기 보다는 여러 개의 단순한(Depth가 낮은) 의사결정 나무를 만들어 모든 예측 결과를 다수결로 총합하여 판단(단독 판사 판결 vs 배심원 판결)

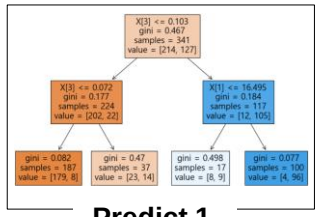
Prediction : 0



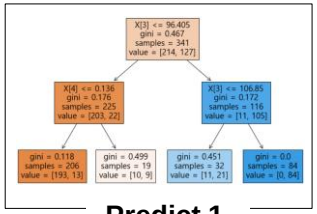
VS

Prediction : 1

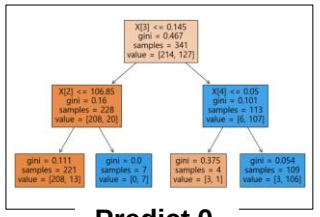
1 : 0
Three : Two



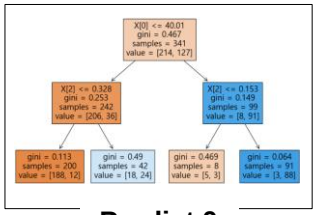
Predict 1



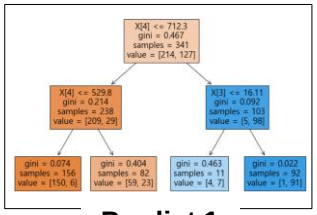
Predict 1



Predict 0

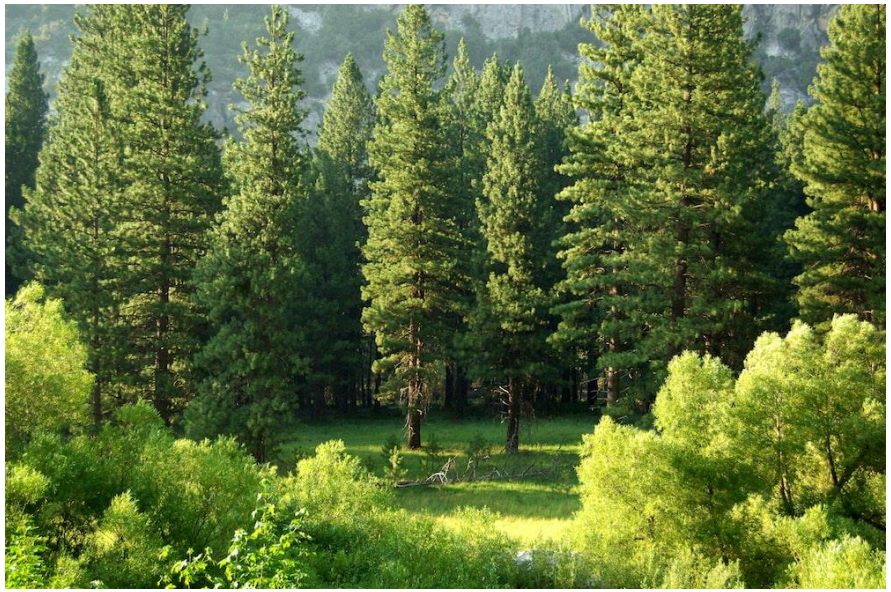


Predict 0



Predict 1

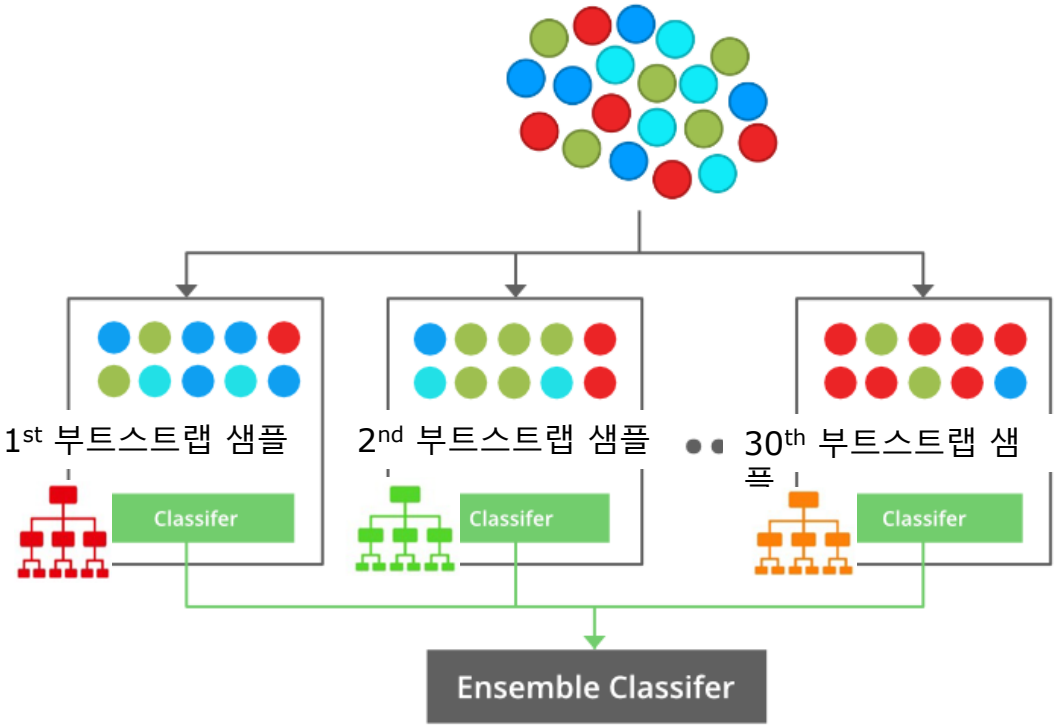
- 의사결정 나무(Decision Tree)
- 앙상블(Ensemble: Bagging, Random Forest)



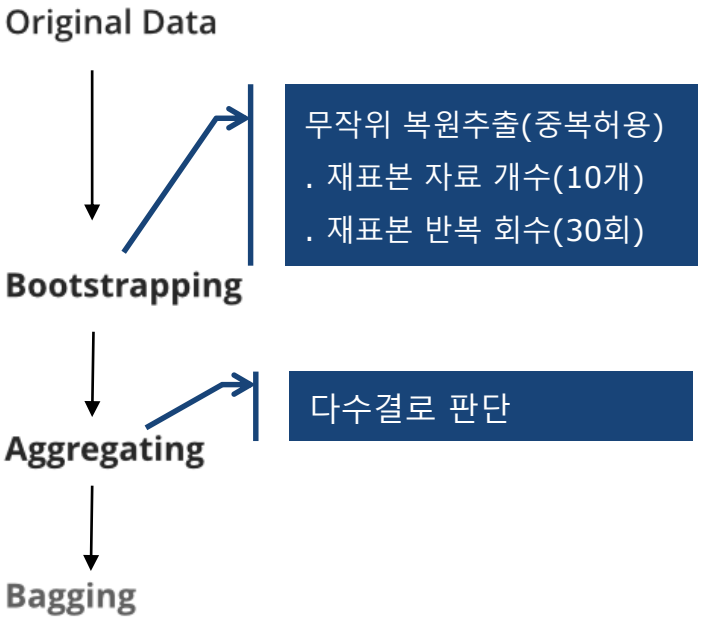
Bagging(Bootstrapping aggragation)

- Bagging은 Bootstrap(Resampling) tree들의 총합(aggregation)
- 원자료를 일정한 크기의 재샘플(복원추출 방법)을 여러 번 거친 부트스트래핑후 다수결로 최종 예측

※ Bootstrap 표본을 쓰므로 다양한 자료를 사용하여 예측 오류의 추정에서 예상외의 효과가 있음



원자료 자료 개수 (20개)



Bootstrapping

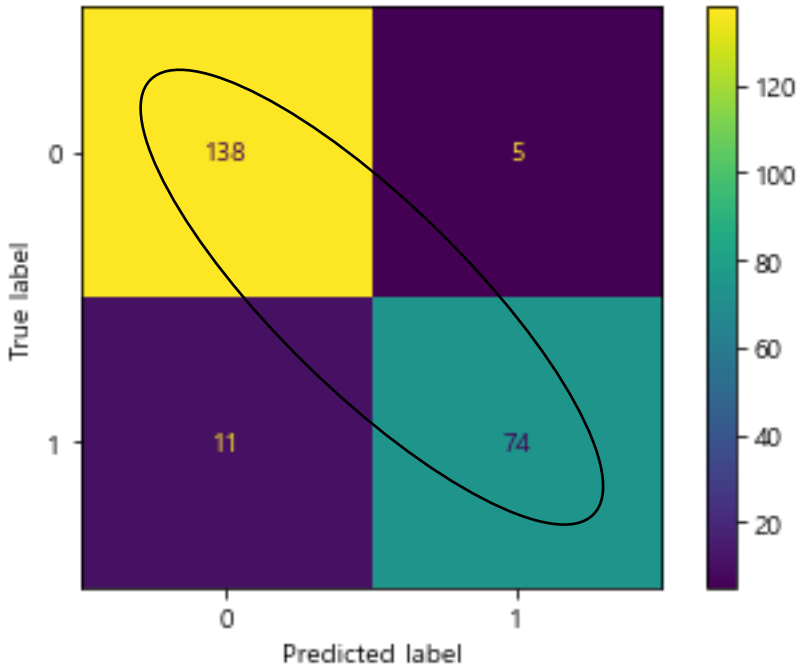
- 표본의 수가 적거나 추가 자료를 얻기 어려워 통계가설이 부재
- 재표본 추출(Resampling) = 부트스트래핑(Boostrapping)
- 표본 N개에서 k개를 복원 추출(중복 선택)하여 모수 추정치의 표준오차를 줄이게 된다.
  - N=100개인 표본에서 200회 재표본(재표본 수는 100개) 추출하는 부트스트래핑을 적용하면 관측치 수가 100개인 표본 200개를 생성

Data diversity ~ bootstrap aggregating, cross validation

Bagging 오차 행렬

- Bagging으로 예측하면 정확도가 92.98%로 Decision Tree 모델보다 약간 높아짐

오차 행렬(Confusion Matrix)



정확도(Accuracy)= (138+74)/228 = 92.98%

Python script

```
from sklearn import ensemble

dt = DecisionTreeClassifier()

Bag = ensemble.BaggingClassifier
        (dt,
         n_estimators = 30,
         max_samples =0.8)

model.fit(X_train, y_train)

model.predict(X_test)
```

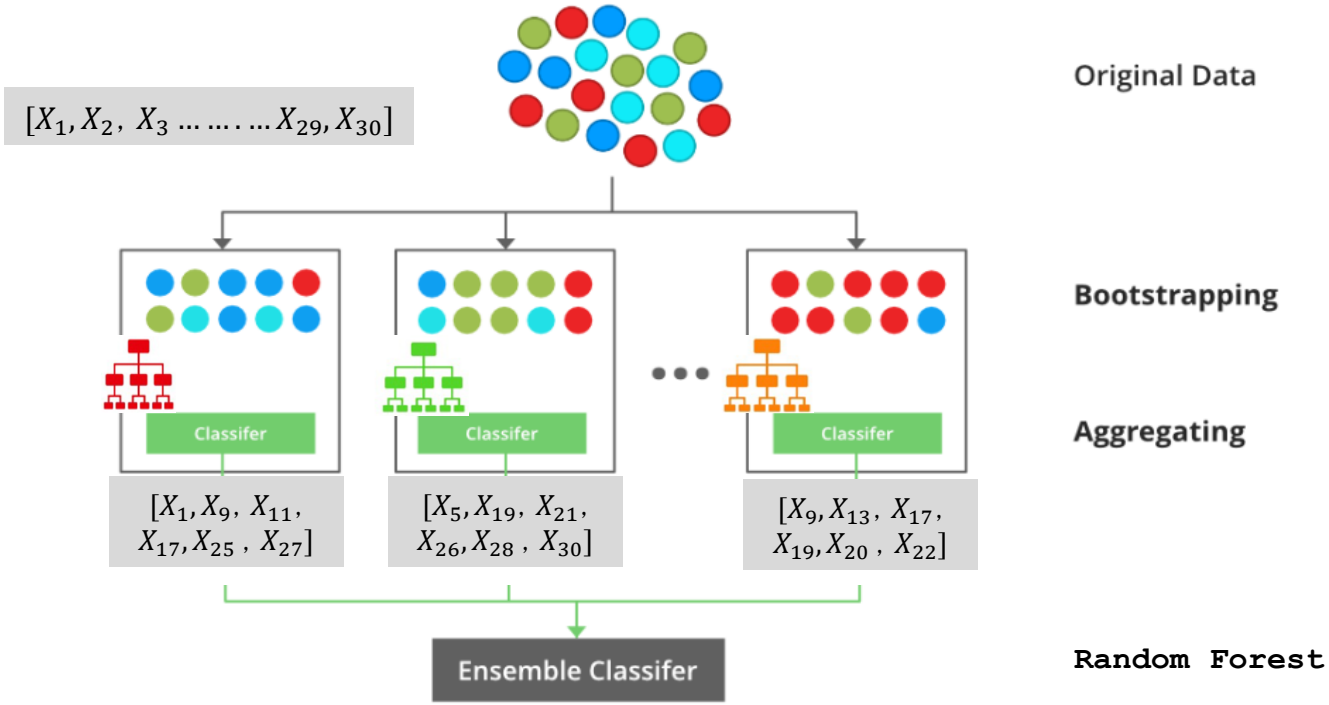
. 의사결정나무 30개 구성 (n_estimators=30)

. 340개의 훈련데이터에서 272개(80%)를 복원추출(bootstrapping)을 30회 반복하여 재표본 구성

. 의사결정나무 30개 총합 다수결(aggregation)로 진단 결과 예측

Random Forest

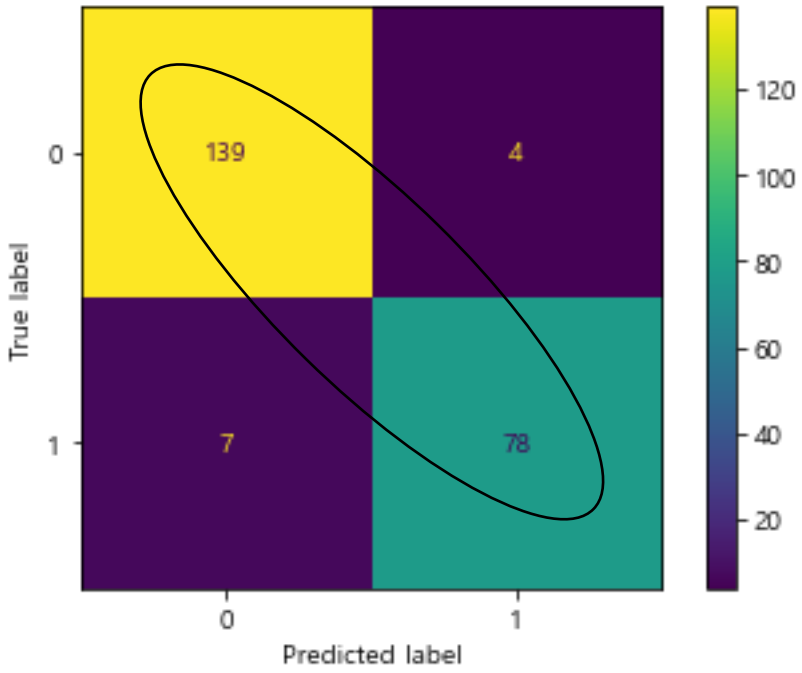
- Bagging은 부분적인 중복성이 있는 Decision Tree를 사용할 수 있어 모델 성능을 저하시킬 수 있는데, 이런 문제점을 개선한 것이 랜덤 포레스트(Random Forest)
- Random Forest 핵심 특징
  1) 원 표본에서 중복을 허용하여 같은 크기의 재표본을 추출하여(Bootstrapping) 훈련자료로 사용하고
  2) 각 노드 분리에서 전체 p개의 변수 중에서 임의로(random) 선택된 m개의 변수를 비복원 추출하여 예측



Random Forest 오차 행렬

- 30개의 의사결정 나무로 구성된 Random Forest로 예측하면 정확도가 95.18로 크게 개선됨
- 모든 데이터에 대해 Random Forest가 항상 성능이 좋은 것이 아니라 데이터의 특성에 따라 차이가 있음

오차 행렬(Confusion Matrix)



정확도(Accuracy)= (139+78)/228 = 95.18%

Python script

```
Rf =
ensemble.RandomForestClassifier(
        n_estimators=30,
        max_features=4,
        )
```

. 의사결정나무 30개 구성 (n_estimators=30)

. 340개의 훈련데이터에서 340개(100%)를 복원추출을 30회 반복하여 재표본 구성

. 특성변수 4개를 무작위 선정하여 예측

```
model.fit(X_train, y_train)
```

```
model.predict(X_test)
```

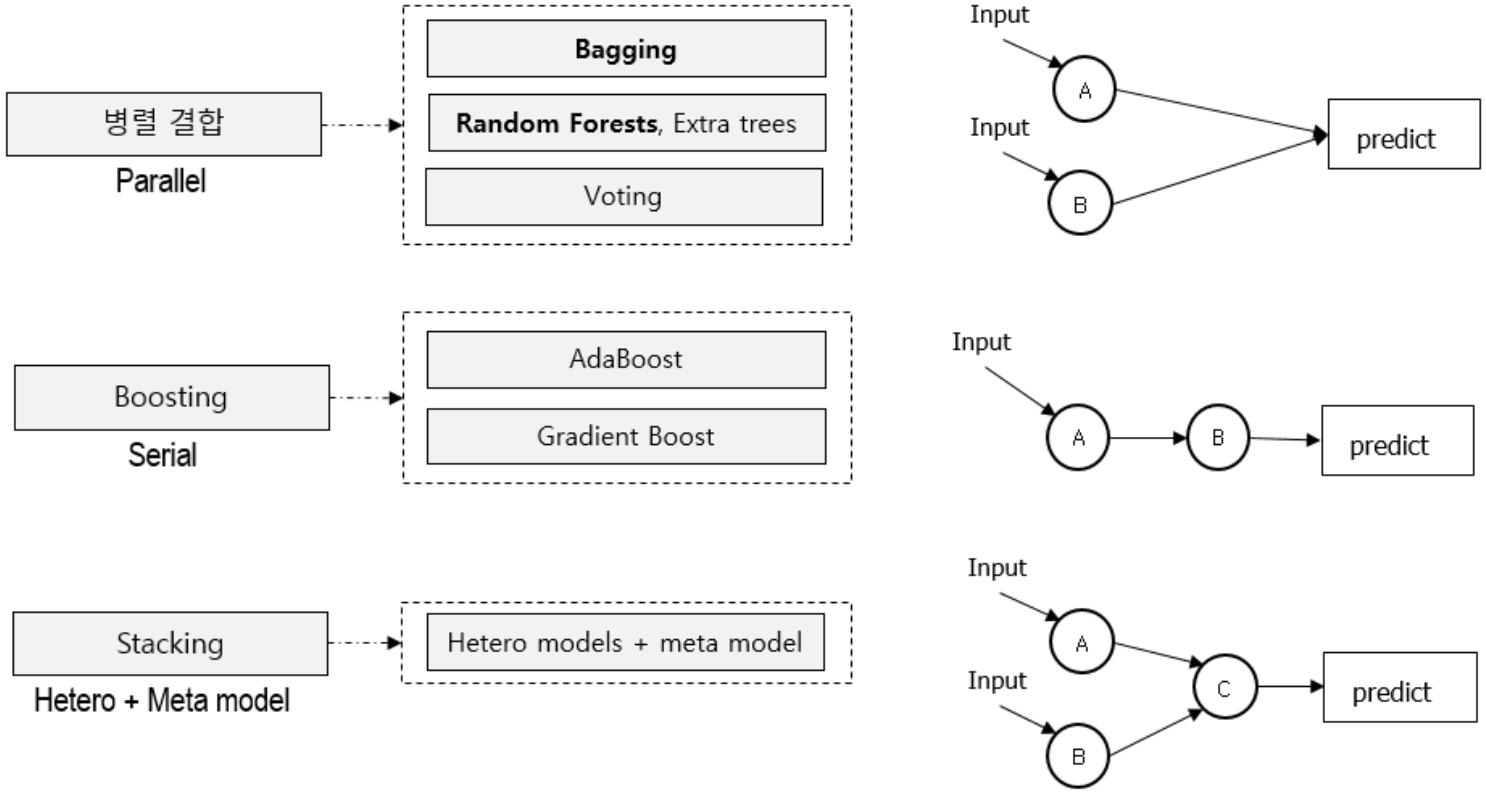
. 의사결정나무 30개 총합 다수결(aggregation)로 진단 결과 예측

생각해 볼 문제

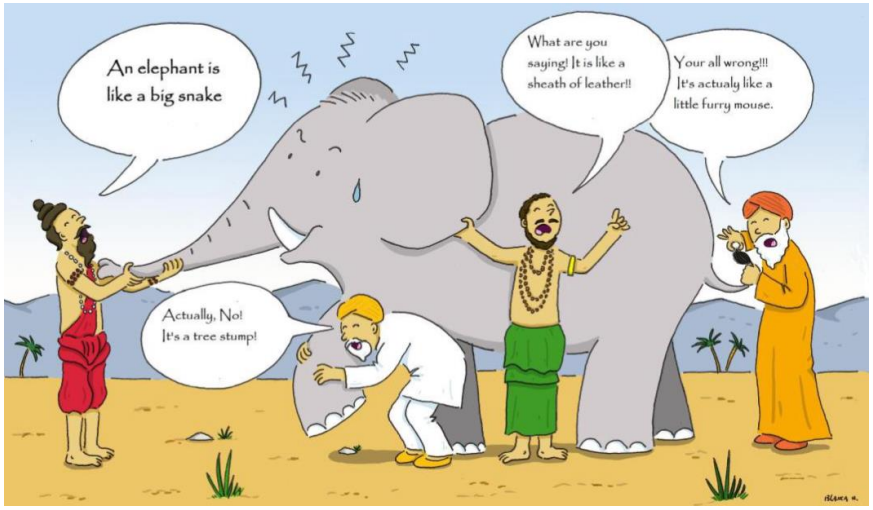
- 5명이 1팀을 이루어 장학 퀴즈에 참가하게 되는 경우 어떤 학생들로 구성하여 나가야 우승할 확률이 가장 높을까?

앙상블(Ensemble) 종류

- 병렬 결합 방법은 병렬적(Parallel) 형태 여러 모델(A, B)의 다수결로 예측
- 직렬 결합 ,부스팅(Boosting) 방법은 선행 모델(A)의 예측 오류를 후행 모델(B)이 이어 받아 학습 · 예측
- 스태킹(Stackinig) 방법은 여러 이종 모델(A, B)의 예측 결과를 입력변수로 메타 모델(C)이 학습 · 예측



Better predictions with bagging, boosting & stacking



https://becominghuman.ai/ensemble-learning-bagging-and-boosting-d20f38be9b1e

→ **Bagging**

→ **Boosting**

Why Tree?

Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?

Manuel Fernández-Delgado MANUEL.FERNANDEZ.DELGADO@USC.ES
Eva Cernadas EVA.CERNADAS@USC.ES
Senén Barro SENEN.BARRO@USC.ES
CITIUS: Centro de Investigación en Tecnoloxías da Información da USC
University of Santiago de Compostela
Campus Vida, 15872, Santiago de Compostela, Spain

Dinani Amorim DINANIAMORIM@GMAIL.COM
Departamento de Tecnologia e Ciências Sociais- DTCS
Universidade do Estado da Bahia
Av. Edgard Chastinet S/N - São Geraldo - Juazeiro-BA, CEP: 48.305-680, Brasil

Editor: Russ Greiner

Abstract

We evaluate **179 classifiers** arising from **17 families** (discriminant analysis, Bayesian, neural networks, support vector machines, decision trees, rule-based classifiers, boosting, bagging, stacking, random forests and other ensembles, generalized linear models, nearest-neighbors, partial least squares and principal component regression, logistic and multinomial regression, multiple adaptive regression splines and other methods), implemented in Weka, R (with and without the caret package), C and Matlab, including all the relevant classifiers available today. We use **121 data sets**, which represent **the whole UCI** data base (excluding the large-scale problems) and other own real problems, in order to achieve significant conclusions about the classifier behavior, not dependent on the data set collection. **The classifiers most likely to be the bests are the random forest** (RF) versions, the best of which (implemented in R and accessed via caret) achieves 94.1% of the maximum accuracy overcoming 90% in the 84.3% of the data sets. However, the difference is not statistically significant with the second best, the SVM with Gaussian kernel

Rank	Acc.	κ	Classifier	Rank	Acc.	κ	Classifier
32.9	82.0	63.5	parRF_t (RF)	67.3	77.7	55.6	pda_t (DA)
33.1	**82.3**	**63.6**	rf_t (RF)	67.6	78.7	55.2	elm_m (NNET)
36.8	81.8	62.2	svm_C (SVM)	67.6	77.8	54.2	SimpleLogistic_w (LMR)
38.0	81.2	60.1	svmPoly_t (SVM)	69.2	78.3	57.4	MAB_J48_w (BST)
39.4	81.9	62.5	rforest_R (RF)	69.8	78.8	56.7	BG_REPTree_w (BAG)
39.6	82.0	62.0	elm_kernel_m (NNET)	69.8	78.1	55.4	SMO_w (SVM)
40.3	81.4	61.1	svmRadialCost_t (SVM)	70.6	78.3	58.0	MLP_w (NNET)
42.5	81.0	60.0	svmRadial_t (SVM)	71.0	78.8	58.23	BG_RandomTree_w (BAG)
42.9	80.6	61.0	C5.0_t (BST)	71.0	77.1	55.1	mlm_R (GLM)
44.1	79.4	60.5	avNNet_t (NNET)	71.0	77.8	56.2	BG_J48_w (BAG)
45.5	79.5	61.0	nnet_t (NNET)	72.0	75.7	52.6	rbf_t (NNET)
47.0	78.7	59.4	pcaNNet_t (NNET)	72.1	77.1	54.8	fda_R (DA)
47.1	80.8	53.0	BG_LibSVM_w (BAG)	72.4	77.0	54.7	lda_R (DA)
47.3	80.3	62.0	mlp_t (NNET)	72.4	79.1	55.6	svmlight_C (NNET)
47.6	80.6	60.0	RotationForest_w (RF)	72.6	78.4	57.9	AdaBoostM1_J48_w (BST)
50.1	80.9	61.6	RRF_t (RF)	72.7	78.4	56.2	BG_IBk_w (BAG)
51.6	80.7	61.4	RRFglobal_t (RF)	72.9	77.1	54.6	ldaBag_R (BAG)
52.5	80.6	58.0	MAB_LibSVM_w (BST)	73.2	78.3	56.2	BG_LWL_w (BAG)
52.6	79.9	56.9	LibSVM_w (SVM)	73.7	77.9	56.0	MAB_REPTree_w (BST)
57.6	79.1	59.3	adaboost_R (BST)	74.0	77.4	52.6	RandomSubSpace_w (DT)
58.5	79.7	57.2	pnn_m (NNET)	74.4	76.9	54.2	lda2_t (DA)
58.9	78.5	54.7	cforest_t (RF)	74.6	74.1	51.8	svmBag_R (BAG)
59.9	79.7	42.6	dkp_C (NNET)	74.6	77.5	55.2	LibLINEAR_w (SVM)
60.4	80.1	55.8	gausssprRadial_R (OM)	75.9	77.2	55.6	rbfDDA_t (NNET)
60.5	80.0	57.4	RandomForest_w (RF)	76.5	76.9	53.8	sda_t (DA)
62.1	78.7	56.0	svmLinear_t (SVM)	76.6	78.1	56.5	END_w (OEN)
62.5	78.4	57.5	fda_t (DA)	76.6	77.3	54.8	LogitBoost_w (BST)
62.6	78.6	56.0	knn_t (NN)	76.6	78.2	57.3	MAB_RandomTree_w (BST)
62.8	78.5	58.1	mlp_C (NNET)	77.1	78.4	54.0	BG_RandomForest_w (BAG)
63.0	79.9	59.4	RandomCommittee_w (OEN)	78.5	76.5	53.7	Logistic_w (LMR)
63.4	78.7	58.4	Decorate_w (OEN)	78.7	76.6	50.5	ctreeBag_R (BAG)
63.6	76.9	56.0	mlpWeightDecay_t (NNET)	79.0	76.8	53.5	BG_Logistic_w (BAG)
63.8	78.7	56.7	rda_R (DA)	79.1	77.4	53.0	lvq_t (NNET)
64.0	79.0	58.6	MAB_MLP_w (BST)	79.1	74.4	50.7	pls_t (PLSR)
64.1	79.9	56.9	MAB_RandomForest_w (BST)	79.8	76.9	54.7	hdda_R (DA)

No Free Lunch

- 모든 데이터에 대해 항상 우월한 알고리즘은 존재하지 않고 상황에 따라 경험적 혹은 정량적으로 결정된다.
- 특정 알고리즘이 우수한 것은 해당 데이터의 패턴을 특정 알고리즘이 fit를 잘 한 것이다.
- 특정 알고리즘을 최적화하여 사용하는 것보다는 몇 개의 알고리즘을 적당하게 Ensemble 하는 것이 바람직하다.
- Ensemble almost always work better than the best of breeds, the single best model.


END
OPTIMIST
RESPONDER
BEENUTZ