

의사결정 나무와 앙상블

- Tree based classification -



조상구



과목명 : 인공지능 분류 및 회귀모형

- 빅데이터를 바탕으로 분류(Classification) 및 회귀(Regression) 머신러닝 모형(모델) 이론 및 실습

* Python 프로그래밍, Scikit-learn, Keras

강의 내용	
1. 인공지능 시작하기	<ul style="list-style-type: none">인공지능 개요 및 머신러닝 End-to-End 구성 및 절차 이해데이터, 알고리즘, 모델의 이해, Scikit learn API 메커니즘 이해 및 실습모델의 분산과 편향, 손실함수와 최적화, 평가지표 및 선택Auto ML, Low code(Pycares) 소개
2. 인공지능 모델	<ul style="list-style-type: none">의사결정 나무와 앙상블(Decision Tree & Ensemble)최근접 이웃(K-Nearest Neighbors)로지스틱 회귀와 회귀(Logistic Regression & Regression)가우시안 나이브 베이즈(Gaussian Naïve Bayes)서포트 벡터 머신(Support Vector Machine)심층신경망(Deep learning , FNN, CNN, RNN with Keras)

* 강의 부교재는 <https://github.com/jakevdp/PythonDataScienceHandbook.pdf>

강의 내용

강의 내용

- 분류(Classification) 문제 해결을 위해 의사결정 나무, 배깅, 랜덤 포레스트 앙상블(Ensemble) 등 Tree based 알고리즘 이해
- 파이썬과 Scikit-learn API를 사용하여 실습
 - * 강의에서 사용한 데이터와 Python 코드는 github.com/ancestor9/KB/kb.ipynb를 참고

과제 제출

- 주어진 데이터로 분류 및 회귀생성을 위한 앙상블 모델을 적용하여 과제 제출
 - * 과제 수행 코드는 Python script 파일로 제출

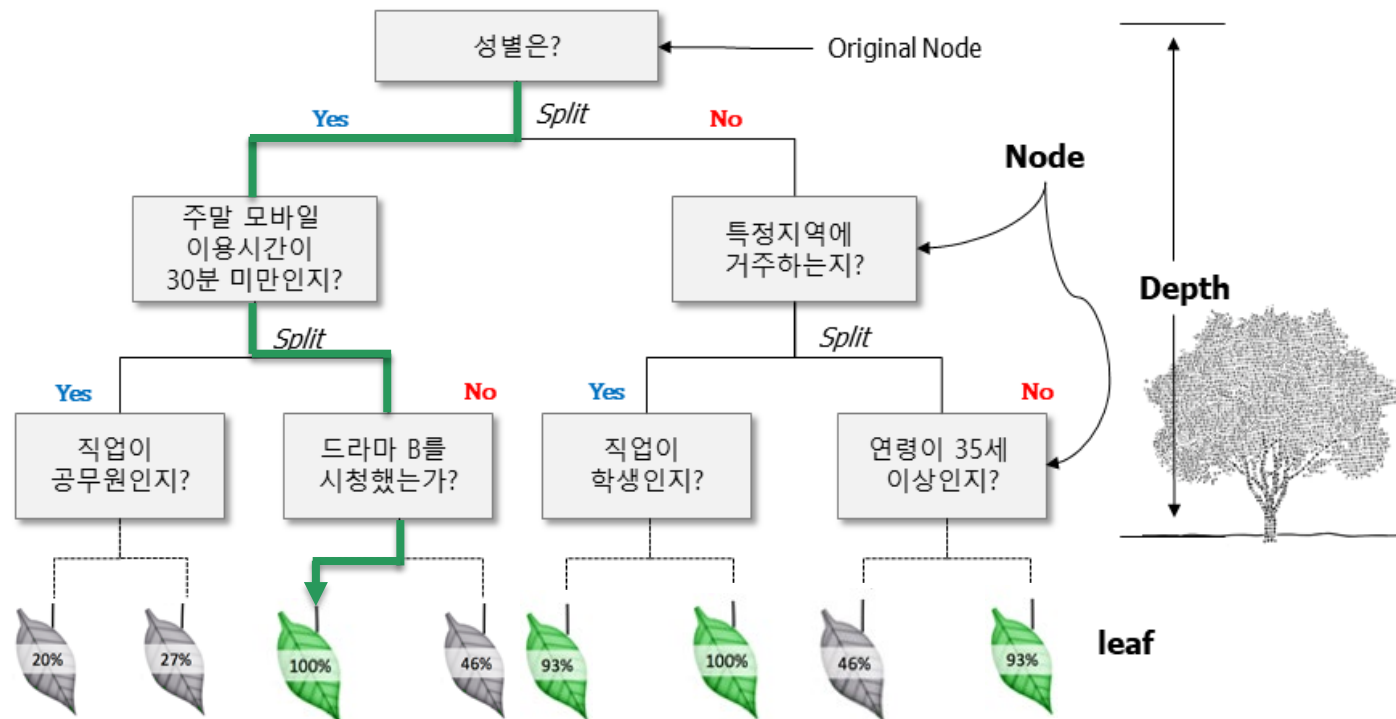
목차

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



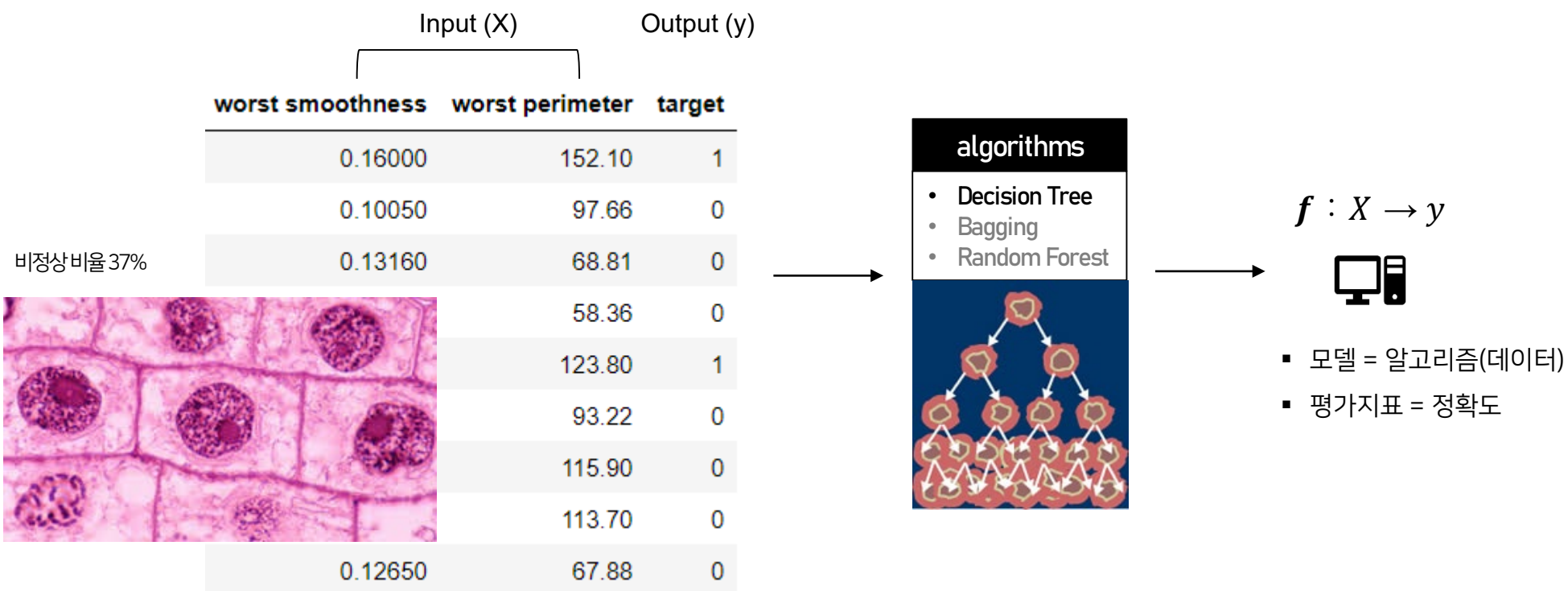
Decision Tree

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

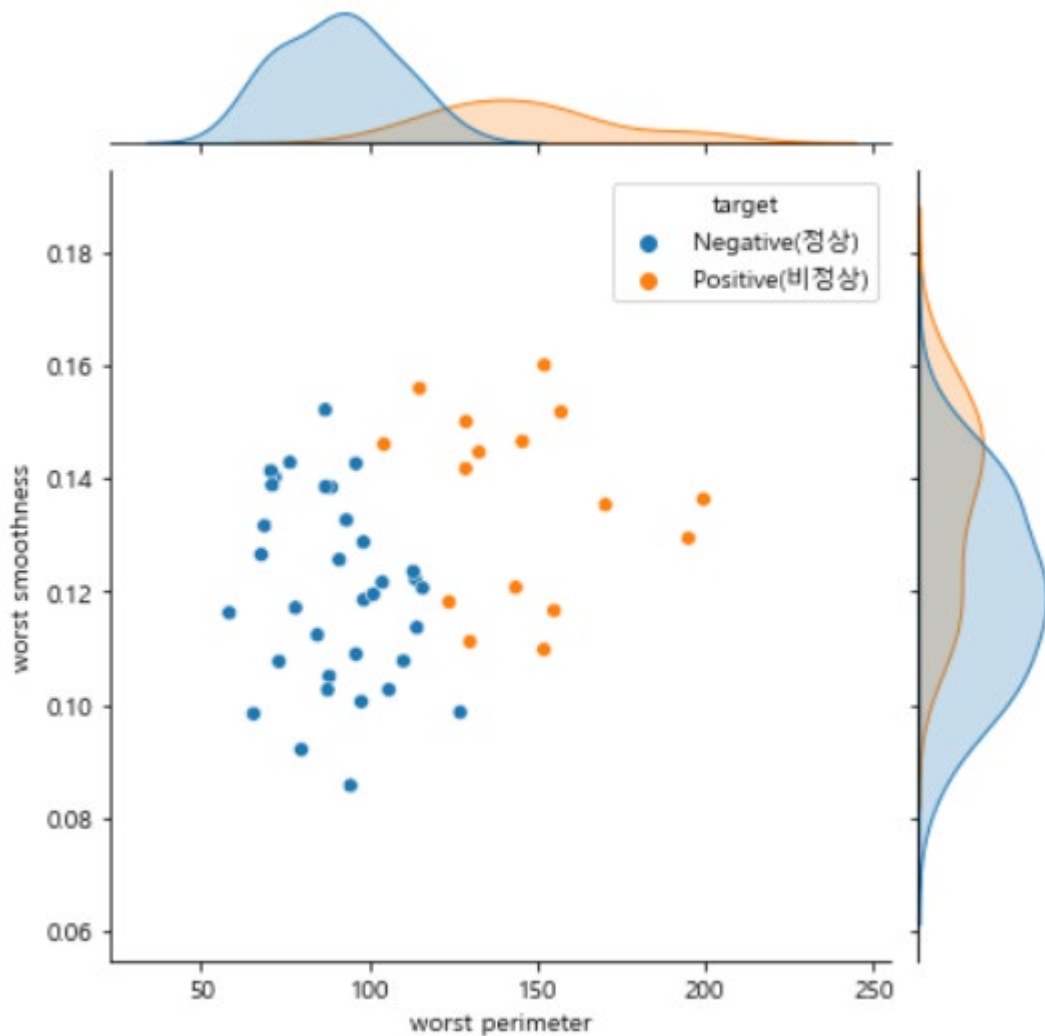


문제 및 해결 방안

- 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



분류를 가장 잘할 수 있는 방법



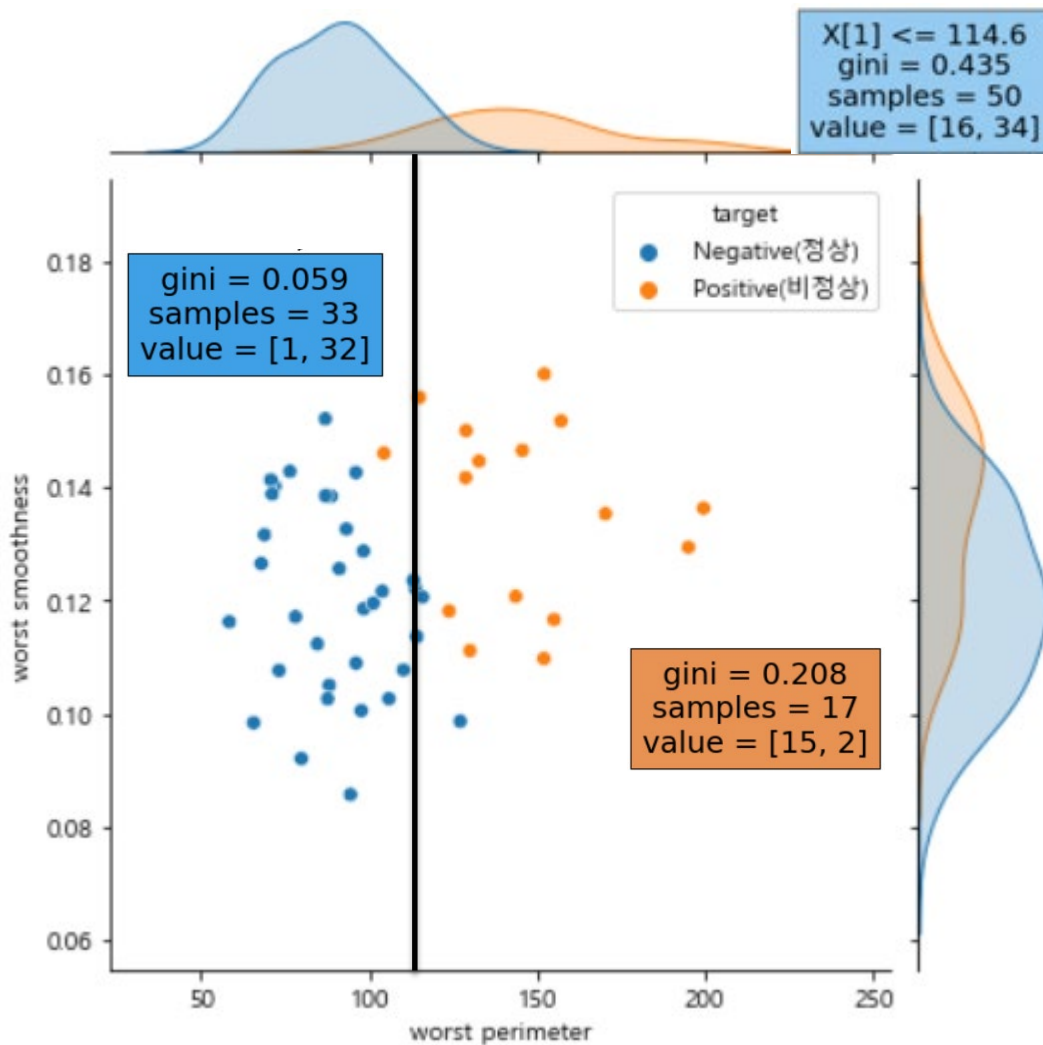
- 특성변수가 없을 경우에는 항상 정상이라고 예측하면 정확도가 68%

[Level 0 비중]

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

- 세포핵 외벽 '평탄도'와 '길이' 2개의 특성변수가 주어진다면, 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%)	

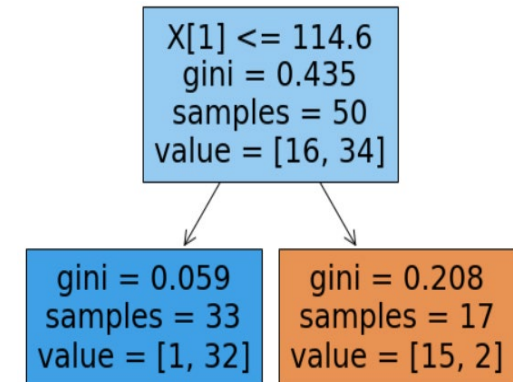
- 컴퓨터가 이해할 수 있도록 지니지수(Gini Index)를 계산하여 노드분리를 자동으로 하게 함

지니 지수(Gini Index)

- 지니지수는 정보의 불확실성이라는 개념을 계량화한 지표
- 의사결정나무는 모든 변수를 대상으로 지니지수(Impurity, Uncertainty)가 낮은 순으로 데이터분류(Node Splitting)를 순차적으로 수행하여 불확실성을 감소하면서 예측

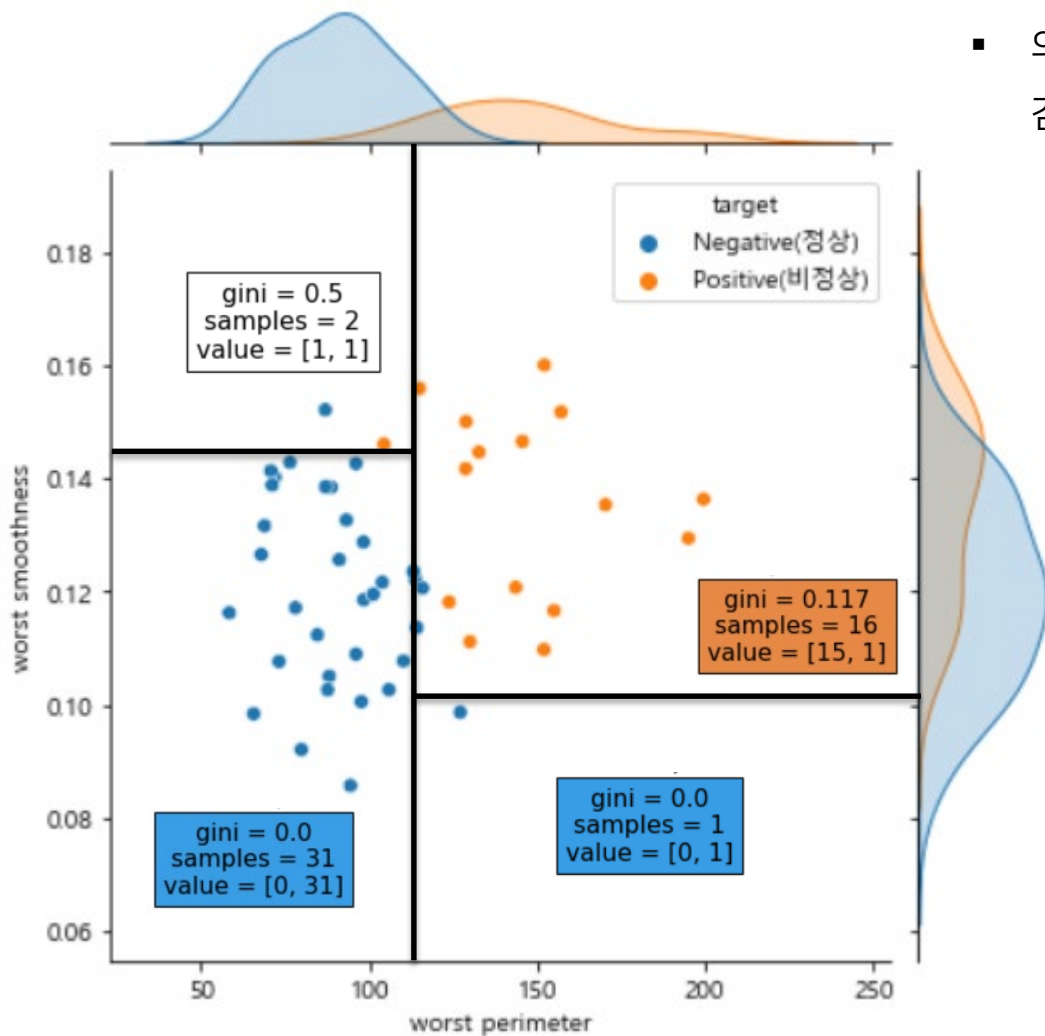
$$\text{지니지수 (Gini Index)} = \sum_{C=1}^2 P_C * (1 - P_C) = 2P_1 * (1 - P_1)$$

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)
	정상	32/33(97%)		
> 114.6	비정상	15/17(88%)	0.208 = 2(88% x 12%)	+ 0.208*(17/50)
	정상	2/17(12%)		

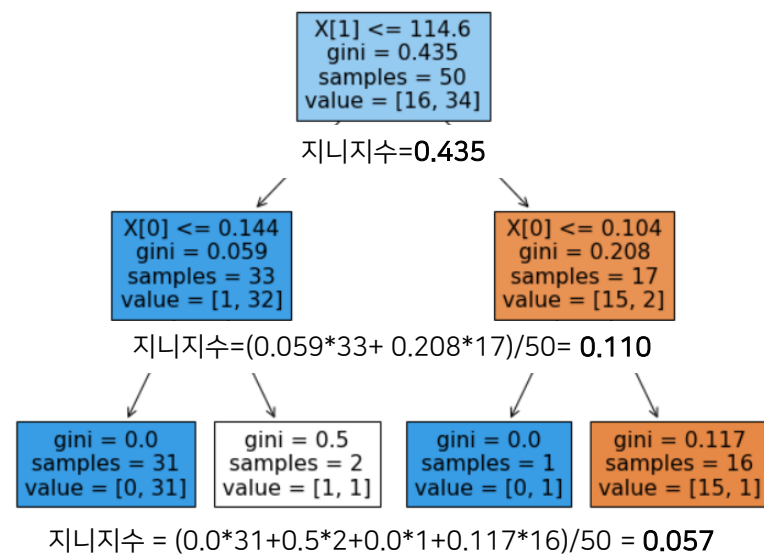


* 각 노드의 데이터 개수를 가중치 평균

2단계 노드분리 변수와 분리 값의 결정



- 의사결정 나무 depth가 '2'에서는 depth 1보다 지니계수가 감소하는 방향으로 데이터를 분할(노드분리와 분리값 결정)



예
측

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

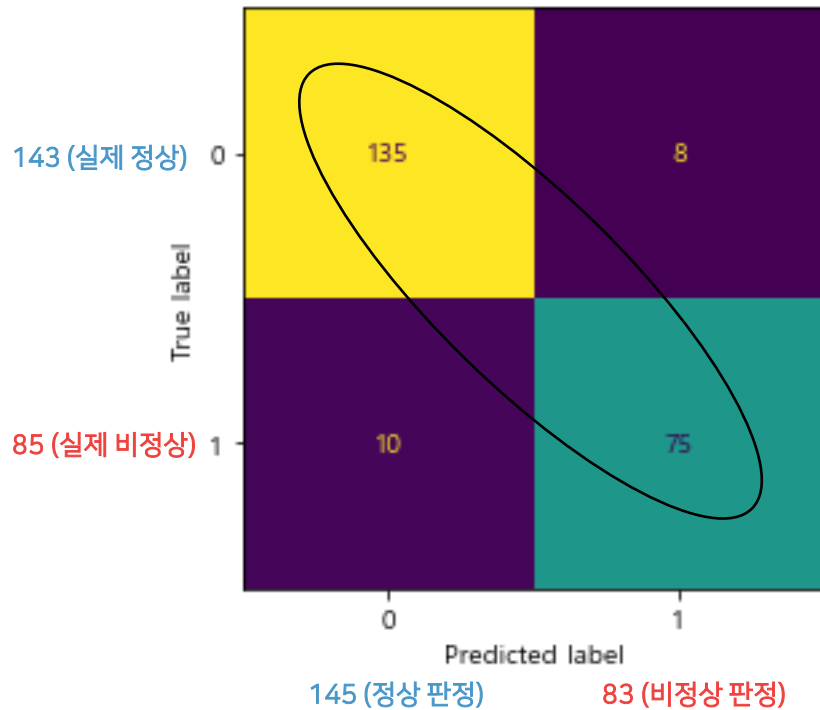
Input (X)													Output (y)
	worst smoothness	worst symmetry	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	...	worst fractal dimension	target
0	0.16220	0.4601	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	0.11890	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 학습하여 228개(test)를 대상으로 실제 정상·비정상의 자료를 판정하면(135개, 75개) 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)
```

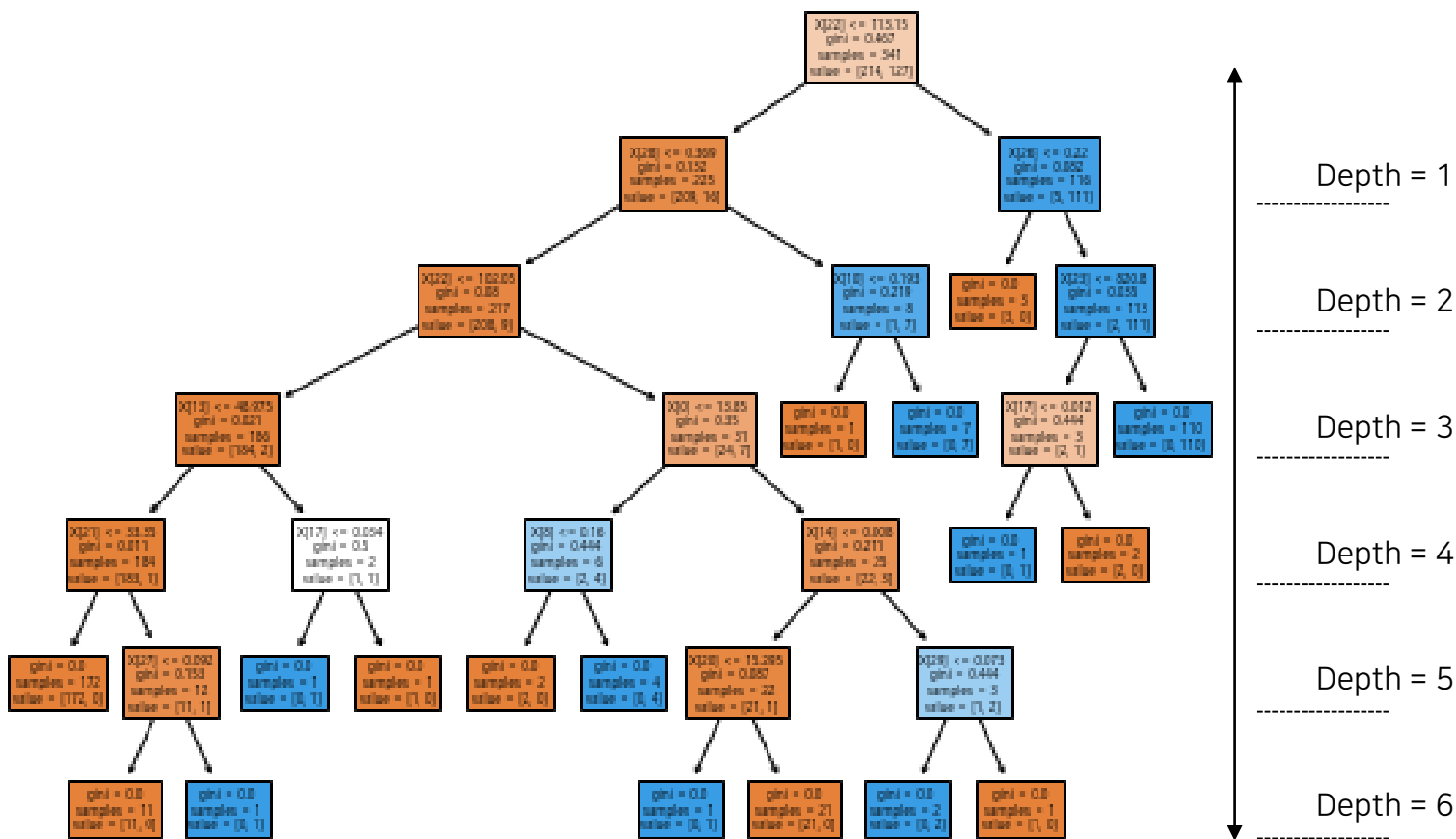
4. 학습

```
model.predict(X_test, y_test)
```

5. 예측

모델의 과적합(Overfitting)

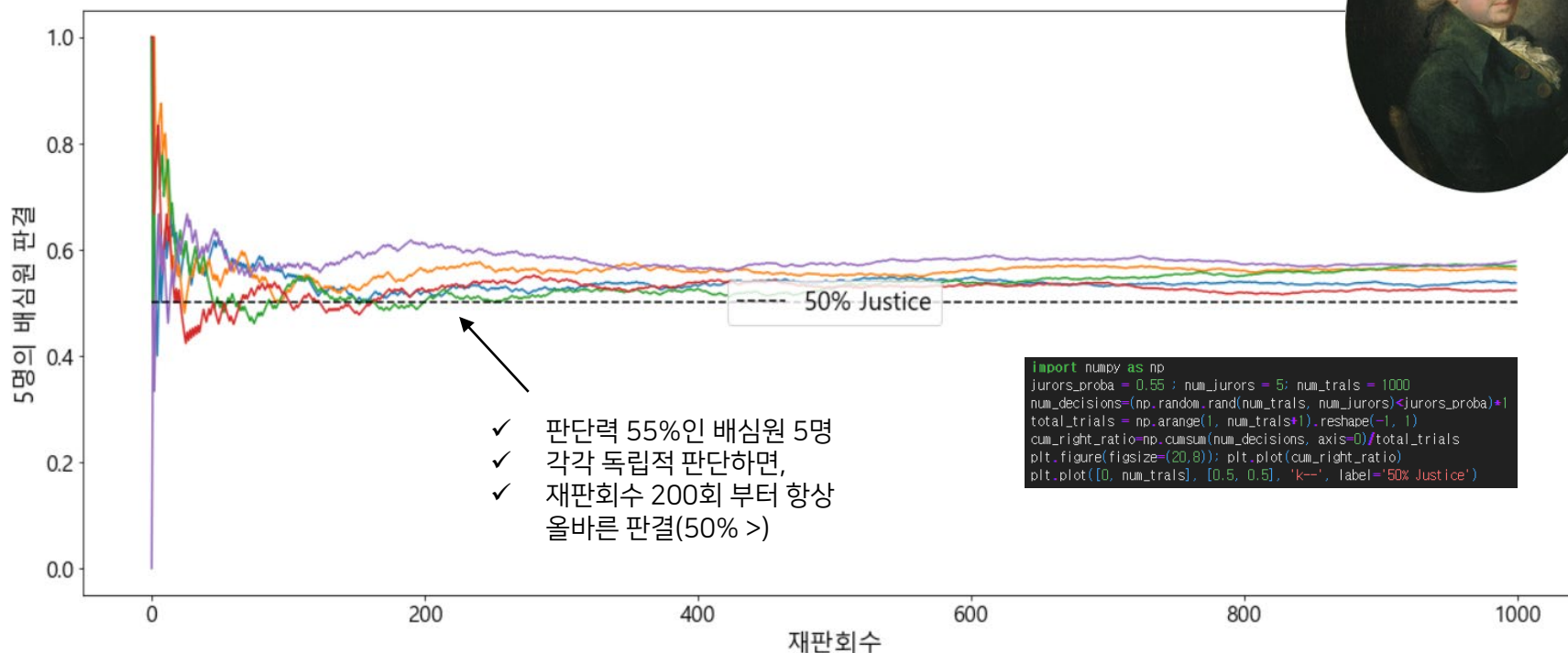
- 의사결정 나무 특성상 자료의 레이블이 완전히 분류될 때까지 노드 분리(node splitting)를 하면서 훈련하기 때문에 과적합(Overfitting)이 발생하기 쉬움 → Depth를 줄여 일반화 모델 필요



콩도르세의 배심원 정리

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

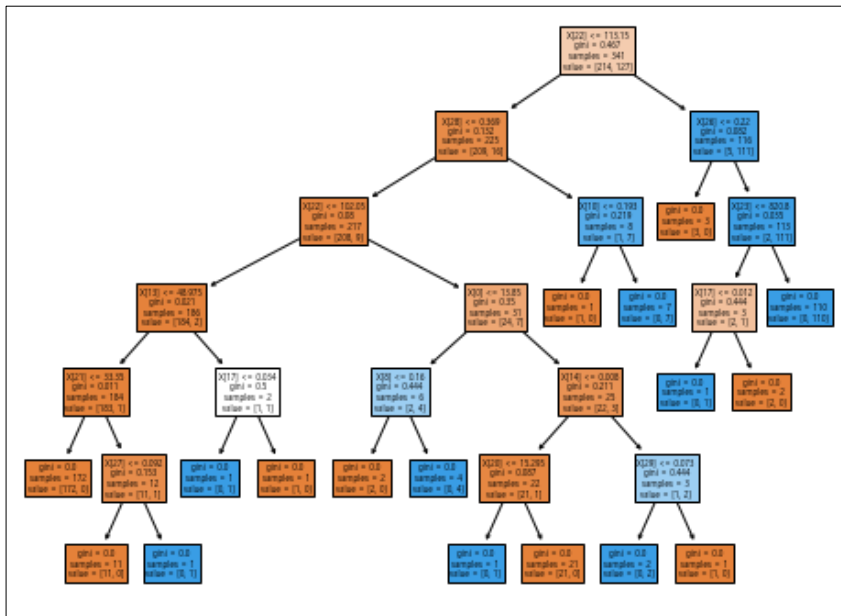
Nicolas de Condorcet's Jury Theorem
(20 Sep 1792 – 8 Jul 1794)



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)

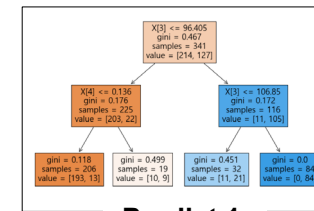
Prediction : 0



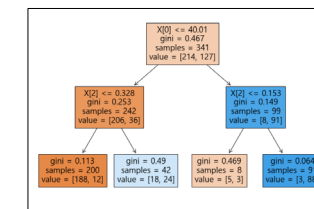
Prediction : 1

1 : 0
Three : Two

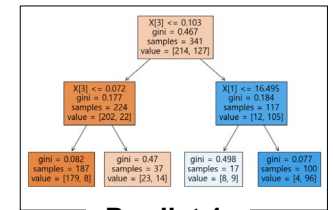
VS



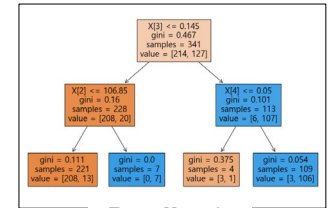
Predict 1



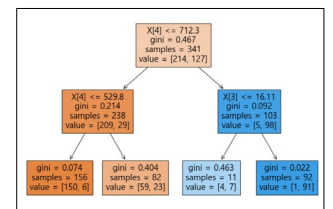
Predict 0



Predict 1



Predict 0



Predict 1

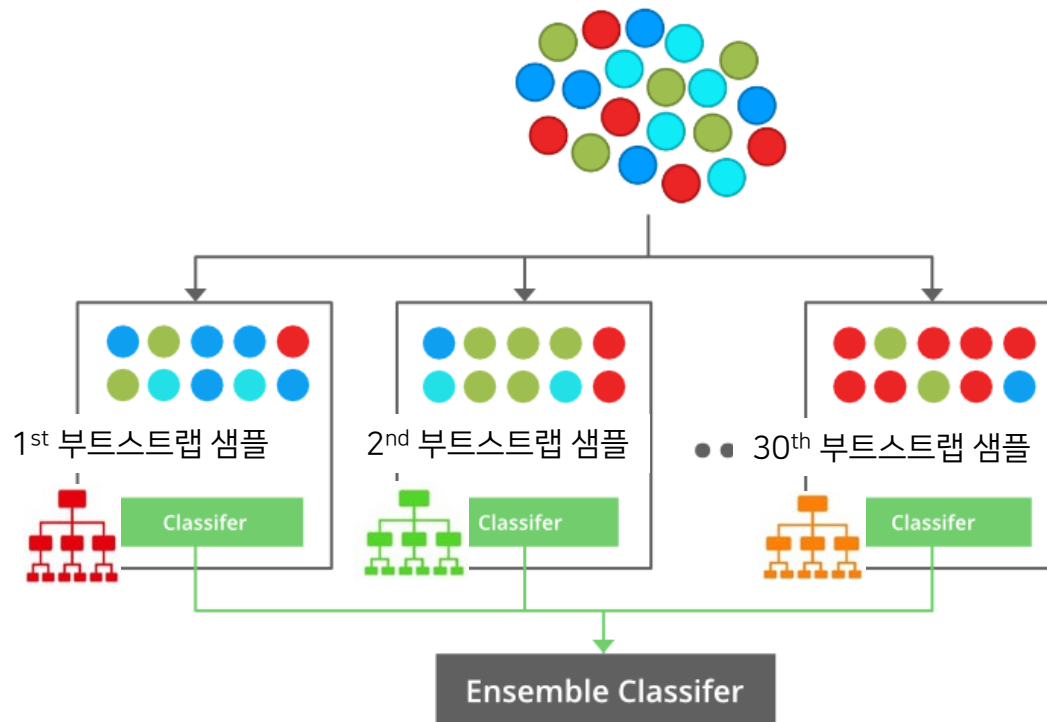
- 의사결정 나무(Decision Tree)
- 앙상블(Ensemble: Bagging, Random Forest)



Bagging(Bootstrapping aggregation)

- 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)

Original Data

Bootstrapping

Aggregating

Bagging

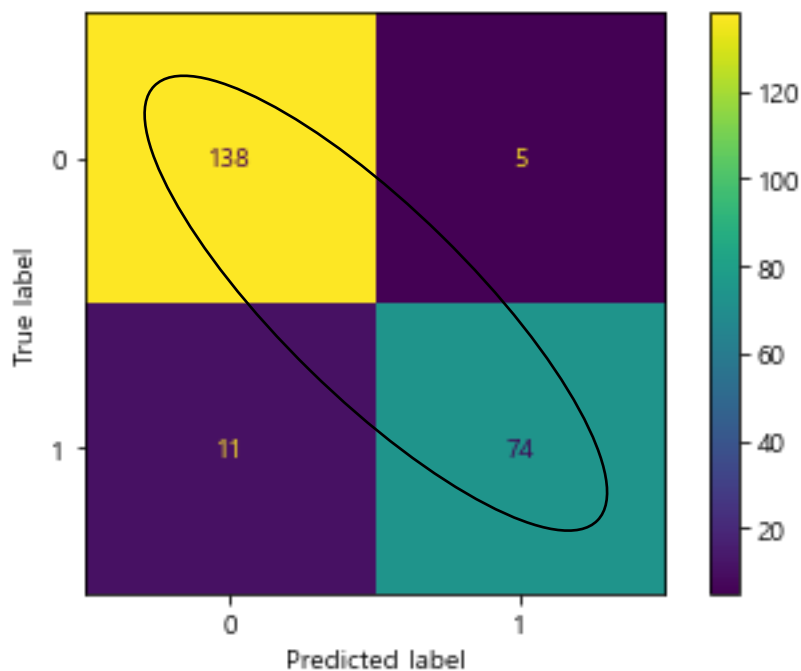
무작위 복원추출(중복허용)
· 재표본 자료 개수(10개)
· 재표본 반복 회수(30회)

다수결로 판단

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)
```

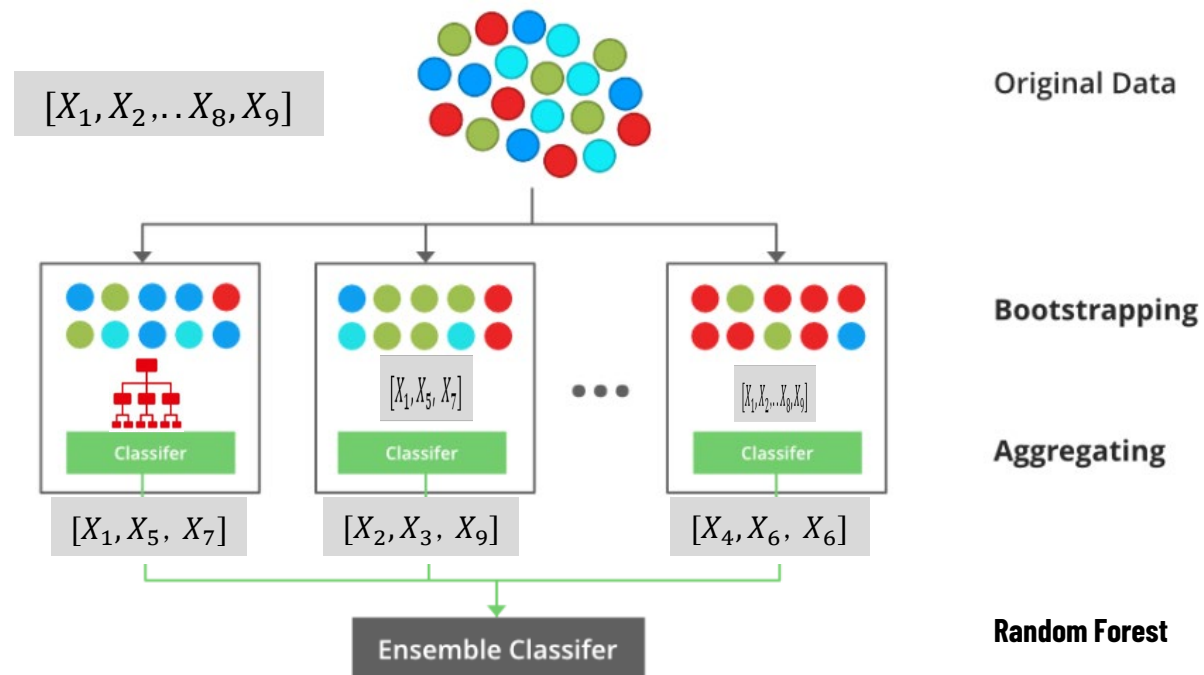
. 의사결정나무 30개 구성
(n_estimators=30)

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

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

Random Forest

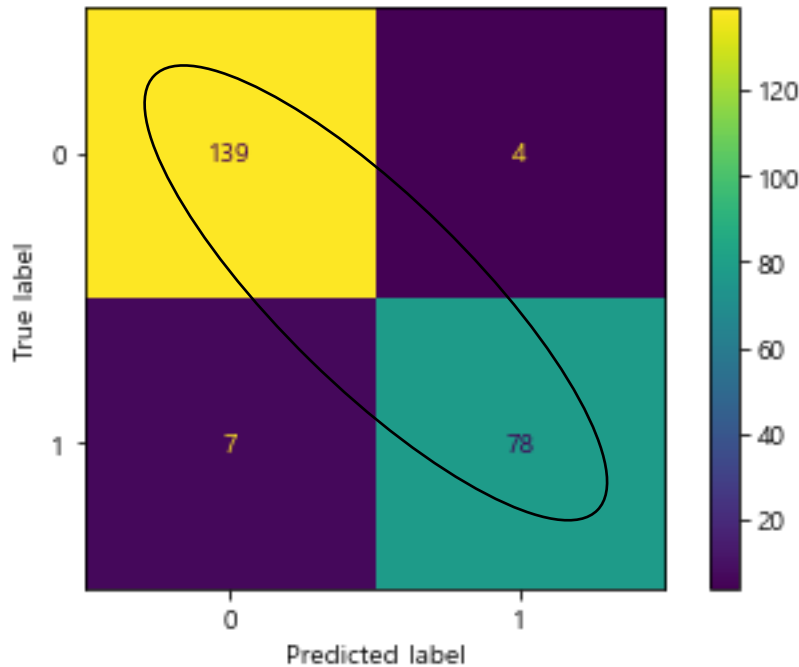
- Bagging은 부분적인 중복성이 있는 Decision Tree를 사용할 수 있어 모델 성능을 저하시킬 수 있는데, 이런 문제점을 개선한 것이 랜덤 포레스트(Random Forest)
- Random Forest 핵심 특징
 - 1) 원 표본에서 중복을 허용하여 같은 크기의 재표본을 추출하여(Bootstrapping) 훈련자료로 사용하고
 - 2) 각 노드 분리에서 전체 p 개의 변수 중에서 임의로(random) 선택된 m 개의 변수를 비복원 추출하여 예측



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(No Free Lunch)

오차 행렬(Confusion Matrix)



정확도(Accuracy) = $(139+78)/228 = 95.18\%$

Python script

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

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

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

. 의사결정나무 30개 구성 (n_estimators=30)

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

. 특성변수 4개를 무작위 선정하여 예측

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

Python실습 : Tree based classification & regression

Search or jump to...

Pull requestsIssuesCodeSpacesMarketplaceExplore

ancestor9 / KBPrivate

<> CodeIssuesPull requestsActionsProjectsSecurityInsightsSettings

mainKB / kb.ipynbGo to file...

ancestor9Add files via uploadLatest commit Sc33bb last weekHistory

Ak1 contributor

1.57 MB

Tree based classification 이해 및 실습

In [1]:

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
warnings.filterwarnings(action='ignore')
```

A. 머신러닝 학습 프로세스

1. 목적 정의

정확도(Accuracy)를 평가자료로 예측 모형 선정

2. 자료 정제 분석

자료 확보

In [2]:

```
from sklearn.datasets import load_breast_cancer
data = load_breast_cancer()
# print(data.DESCR)
```

3. 특성변수

부적합에 영향을 미치는 변수 선정

'worst smoothness', 'worst perimeter' 2개의 특성변수로 'target'을 예측

In [3]:

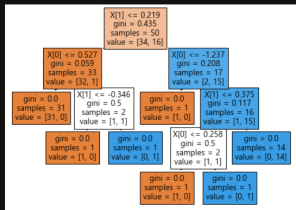
```
X, y = load_breast_cancer(return_X_y=True, as_frame=True)
df = X.merge(y, left_index=True, right_index=True)
df.target.value_counts()
```

Level 4 - Overfitting

In [11]:

```
x = data.drop('target', axis=1); y = data['target']
dt = DecisionTreeClassifier(random_state=1)
dt.fit(x, y)
plt.figure(figsize=(8, 6))
ax = plot_tree(dt, filled=True)
```

Out[11]: DecisionTreeClassifier(random_state=1)
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org
Out[11]: <Figure size 576x432 with 0 Axes>



Nicolas de Condorcet's Jury Theorem

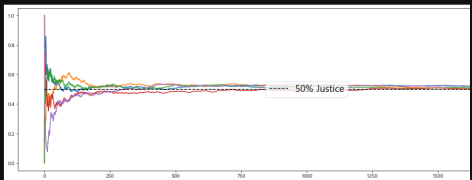
In [12]:

```
import numpy as np
np.random.seed(1234); n_maj_juries = 5; n_maj_trials = 2000
n_mj_decision=np.random.randn(n_maj_trials, n_maj_juries).probab=1
total_trials=n_mj_decision*n_maj_trials*(condorcet=1)
count_right_or_no_count=n_mj_decisions_axis=1/total_trials
plt.figure(figsize=(10, 6)); plt.plot(count_right_or_no_count)
plt.plot([0, n_maj_trials], [0.5, 0.5], 'r--'); label="50% Justice"
plt.legend(fontsize=10, loc="center")
```

Out[12]: <Figure size 1440x432 with 0 Axes>

Out[12]: <matplotlib.axes._lines.Line2D at 0x26367875d>
<matplotlib.axes._lines.Line2D at 0x26367880ad>
<matplotlib.axes._lines.Line2D at 0x263678873d>
<matplotlib.axes._lines.Line2D at 0x2636788c0d>
<matplotlib.axes._lines.Line2D at 0x26367893bd>

Out[12]: <matplotlib.axes._lines.Line2D at 0x263678973d>
Out[12]: <matplotlib.legend.Legend at 0x26367894fd>

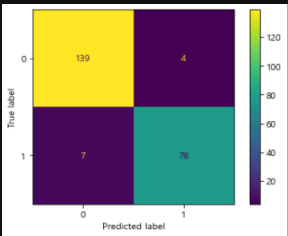


3. Random forest 모형

In [19]:

```
np.random.seed(4321)
rf = ensemble.RandomForestClassifier(random_state=11)
y_prediet = rf.fit(X_train, y_train).predict(X_test)
plot_confusion_matrix(rf, X_test, y_test); plt.show()
print(classification_report(y_test, y_prediet))
```

Out [19]: <sklearn.metrics._plot_confusion_matrix.ConfusionMatrixDisplay at 0x2636a701910>



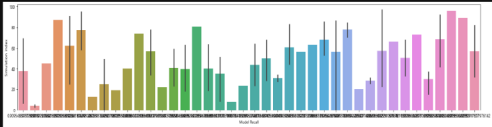
	precision	recall	f1-score	support
0	0.95	0.97	0.96	143
1	0.95	0.92	0.93	65
accuracy			0.95	228
macro avg	0.95	0.94	0.95	228
weighted avg	0.95	0.95	0.95	228

In [17]:

```
data = pd.Series(real[[1,1,1,real_label]],
                names=['index', 'Simulation index', '0 - Node Recall'])
print("시뮬레이션 평균 값출력 : ", data["Node Recall"].mean())
```

fig, ax = plt.subplots()
fig.set_size_inches(8, 4)
ax = sns.barplot(x="Node Recall", y="Simulation index", data=data)

시뮬레이션 평균 값출력 : 0.93625919627238





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
- 스택킹(Stacking) 방법은 여러 이종 모델(A, B)의 예측 결과를 입력변수로 메타 모델(C)이 학습 · 예측

