

앙상블 학습

Ensemble Learning

이건명

충북대학교 소프트웨어학과

학습 내용

- 앙상블 학습의 전략에 대해서 알아본다.
- 배깅 기법으로 결정트리를 이용한 방법과 Random Forest 알고리즘을 알아본다.
- 부스팅 기법으로 AdaBoost, Gradient Boost, XGB 알고리즘을 알아본다.

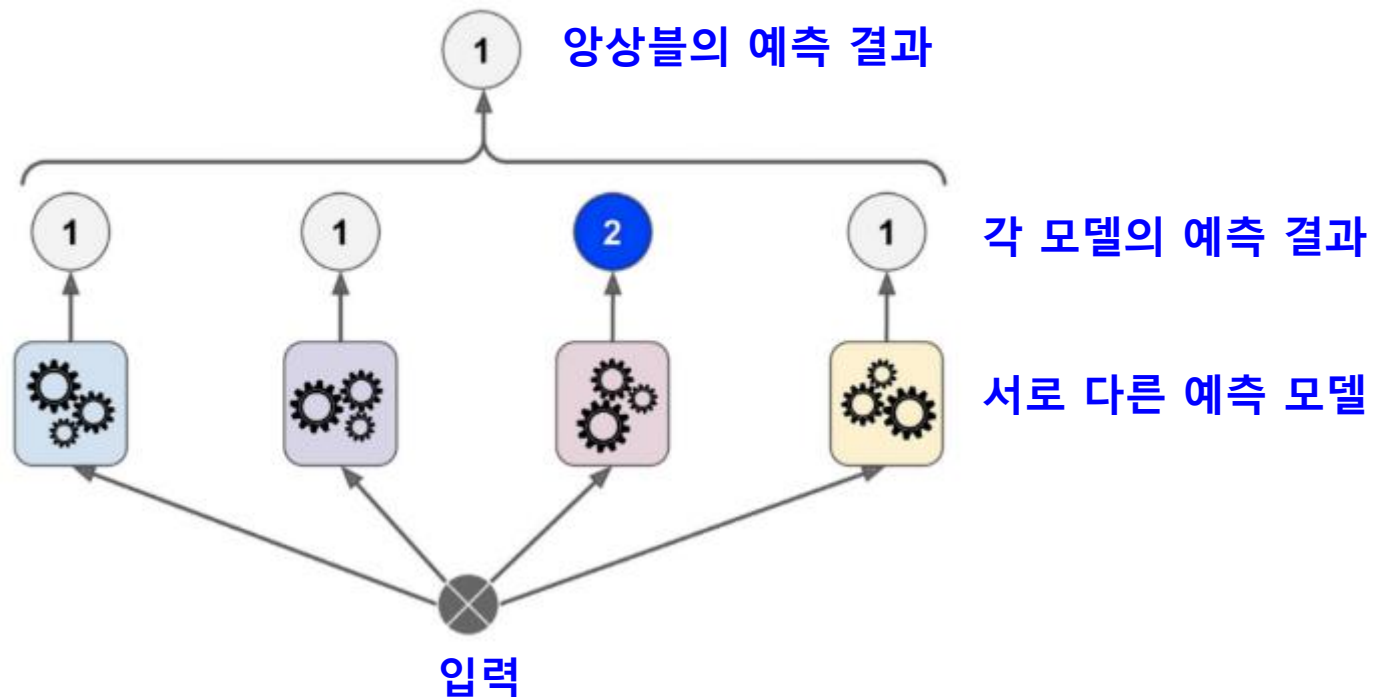
1. 앙상블 학습

❖ 대중의 지혜(wisdom of crowd)

- 무작위로 선택된 많은 사람의 답변을 모은 것이 전문가의 답보다 낫다

❖ 앙상블 학습 (ensemble learning)

- 일련의 예측 모델(분류 또는 회귀 모델)을 사용한 모델의 학습



앙상블 학습

❖ 붓스트랩(bootstrap)

- 주어진 학습 데이터 집합에서 **복원추출**(resampling with replacement) 하여 **다수의 학습 데이터 집합**을 만들어내는 기법



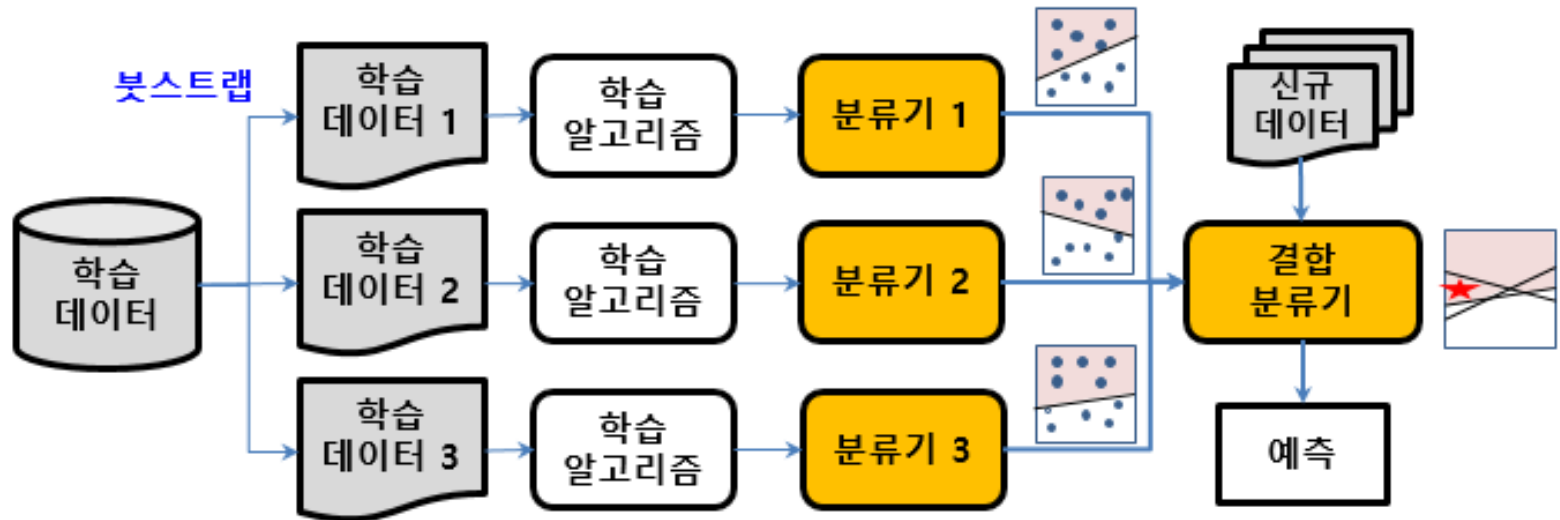
❖ 배깅(bagging, **bootstrap aggregating**)

❖ 부스팅(boosting)

2. 배깅 알고리즘

❖ 배깅(bagging, bootstrap aggregating)

- 붓스트랩을 통해 여러 개의 학습 데이터 집합 생성
- 각 학습 데이터 집합별로 분류기 또는 회귀모델 생성
- 최종판정
 - 분류기들의 투표나 가중치 투표
 - 회귀 모델들의 평균



❖ [실습] 배경

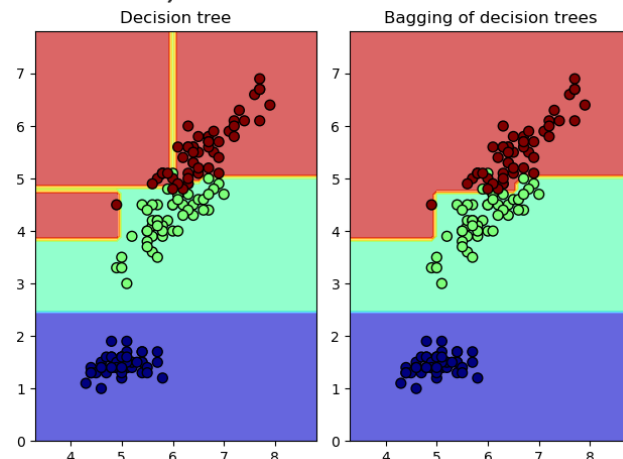
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier

iris = load_iris( )
X, y = iris.data[:, [0, 2]], iris.target

model1 = DecisionTreeClassifier(max_depth=10, random_state=0).fit(X, y)
model2 = BaggingClassifier(DecisionTreeClassifier(max_depth=4),
                           n_estimators=50, random_state=0).fit(X, y)

x_min, x_max = X[:, 0].min( ) - 1, X[:, 0].max( ) + 1
y_min, y_max = X[:, 1].min( ) - 1, X[:, 1].max( ) + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min, y_max, 0.1))

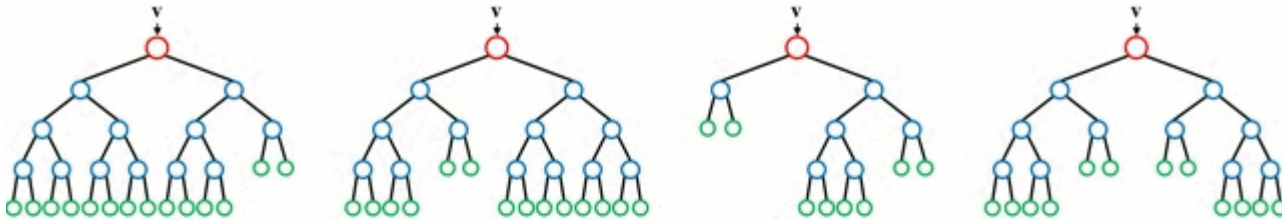
plt.subplot(121)
Z1 = model1.predict(np.c_[xx.ravel( ), yy.ravel( )]).reshape(xx.shape)
plt.contourf(xx, yy, Z1, alpha=0.6, cmap=mpl.cm.jet)
plt.scatter(X[:, 0], X[:, 1], c=y, alpha=1, s=50, cmap=mpl.cm.jet, edgecolors="k")
plt.title("Decision tree")
plt.subplot(122)
Z2 = model2.predict(np.c_[xx.ravel( ), yy.ravel( )]).reshape(xx.shape)
plt.contourf(xx, yy, Z2, alpha=0.6, cmap=mpl.cm.jet)
plt.scatter(X[:, 0], X[:, 1], c=y, alpha=1, s=50, cmap=mpl.cm.jet,
            edgecolors="k")
plt.title("Bagging of decision trees")
plt.tight_layout()
plt.show( )
```



배깅 알고리즘 : 랜덤 포리스트

❖ 랜덤 포리스트(random forest) 알고리즘

- 분류기로 결정트리를 사용하는 배깅 기법
- **Random (무작위) + Forest (숲)**
 - Random : 무작위로 선택한 속성중에서 분할 속성을 선택
 - Forest : 여러 결정트리로 구성



❖ [실습] Random Forest

```
import pandas as pd
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
```

```
iris = datasets.load_iris()
print('Class names :', iris.target_names)
print('target : [0:setosa, 1:versicolor, 2:virginica]')
print('No. of Data :', len(iris.data))
print('Feature names :', iris.feature_names)
```

```
data = pd.DataFrame( {
    'sepal length': iris.data[:, 0], 'sepal width': iris.data[:, 1], 'petal length': iris.data[:, 2],
    'petal width': iris.data[:, 3], 'species': iris.target }
)
print(data.head()) # 일부 데이터 출력
```

```
x = data[['sepal length', 'sepal width', 'petal length', 'petal width']] # 입력
y = data[ ' species ' ] # 출력
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3) # 테스트 데이터 30%
print("No. of training data: ", len(x_train))
print("No. of test data:", len(y_test))
```

```
forest = RandomForestClassifier(n_estimators=100) # 모델 생성
forest.fit(x_train, y_train)
```

```
y_pred = forest.predict(x_test) # 추론(예측)
print('Accuracy :', metrics.accuracy_score(y_test, y_pred))
```

```
Class names : ['setosa' 'versicolor' 'virginica']
target : [0:setosa, 1:versicolor, 2:virginica]
No. of Data : 150
Feature names : ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
    sepal length  sepal width  petal length  petal
width  species
0         5.1         3.5         1.4         0.2      0
1         4.9         3.0         1.4         0.2      0
2         4.7         3.2         1.3         0.2      0
3         4.6         3.1         1.5         0.2      0
4         5.0         3.6         1.4         0.2      0
No. of training data: 105
No. of test data: 45
Accuracy : 1.0
```

학습데이터

X_train

y_train

테스트데이터

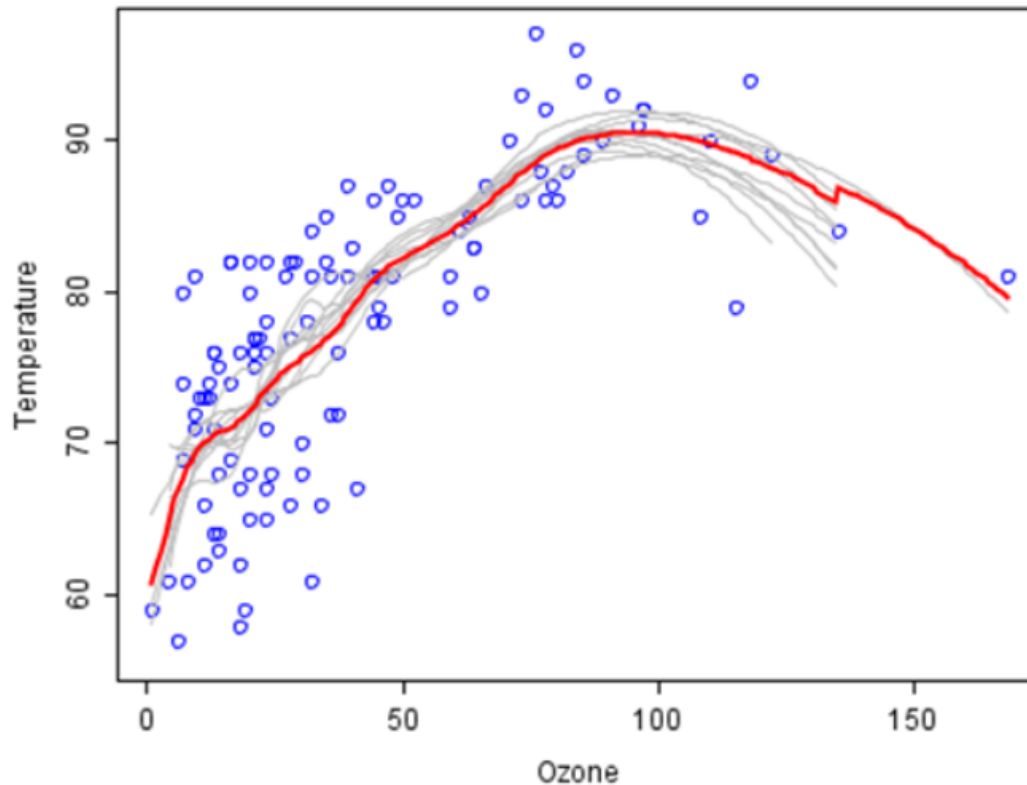
X_test

y_test

배깅 알고리즘

❖ 배깅에 의한 회귀

- 붓스트랩을 통해 다수의 학습데이터 집합 생성
- 각 학습데이터 별로 회귀모델 생성
- 이들 회귀모델의 평균값으로 최종 회귀 생성
- 예.



❖ [실습] Bagging Regression

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.ensemble import BaggingRegressor
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt

boston = load_boston()
data = pd.DataFrame(boston.data)
data.columns = boston.feature_names
data['PRICE'] = boston.target
print(data.head())
X, y = data.iloc[:, :-1], data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
bag = BaggingRegressor(base_estimator = DecisionTreeRegressor( ), n_estimators = 10,
                        max_features=1.0, bootstrap_features=False, random_state=0)
bag.fit(X_train, y_train)
preds = bag.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, preds))
print("RMSE: %f" % (rmse))
```

	CRIM	ZN	INDUS	CHAS	NOX	...	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	...	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	...	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	...	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	...	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	...	222.0	18.7	396.90	5.33	36.2

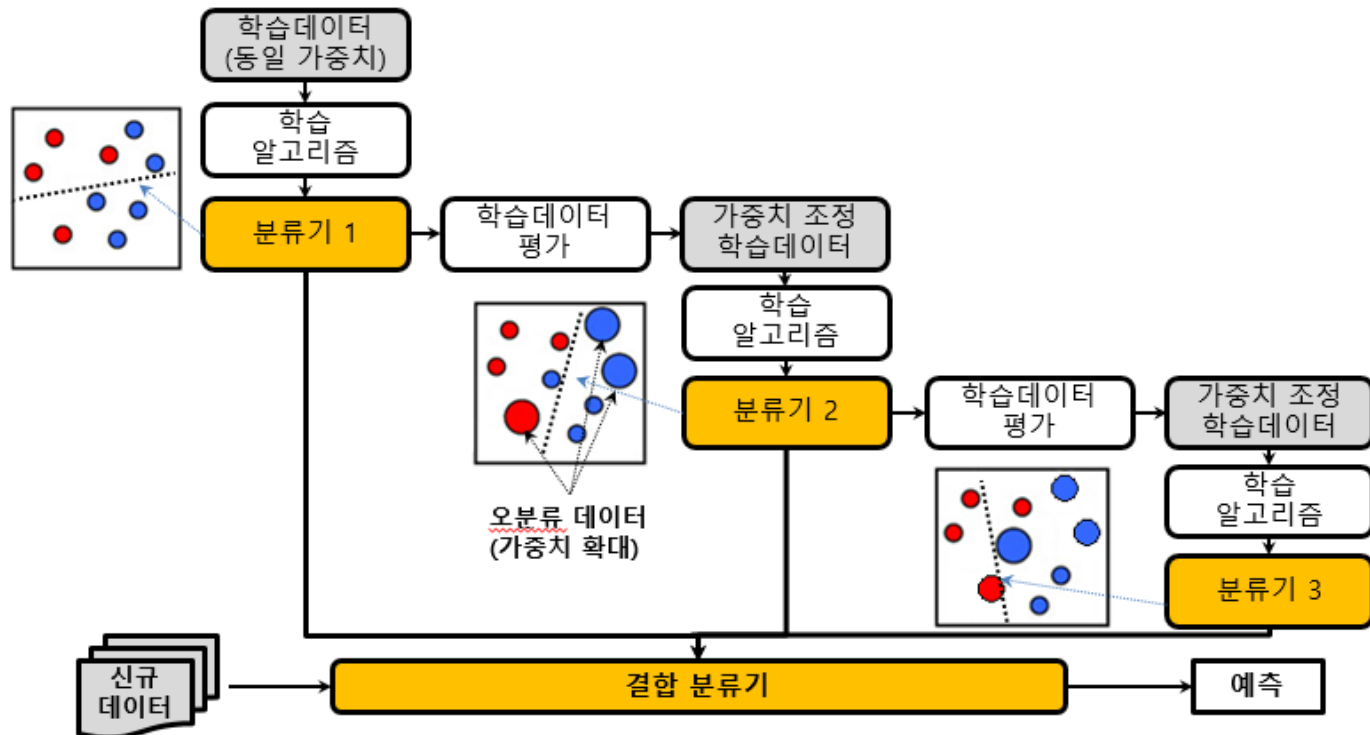
[5 rows x 14 columns]

RMSE: 4.555474

3. 부스팅 알고리즘

❖ 부스팅(boosting)

- k개의 예측 모델을 순차적으로 만들어 가는 앙상블 모델 생성
- 오차에 따라 학습 데이터에 가중치 또는 값을 변경해가면서 예측 모델 생성



- AdaBoost
- Gradient Boosting
- XGB

AdaBoost - 분류

❖ AdaBoost

- N 개의 학습 데이터 d_i 에 대한 초기 가중치 w_i
 - $w_i = \frac{1}{N'}$ 가중치의 합 : 1
- 학습 오류값 ϵ
 - 잘못 분류한 학습데이터의 가중치의 합으로 표현
 - 학습 오류값이 0.5미만인 분류기들만을 사용
- 학습
 - 학습 오류값이 0.5미만인 분류기가 학습되는 경우
 - 분류기 신뢰도 : α
 - $\alpha = 0.5 \ln\left(\frac{1-\epsilon}{\epsilon}\right)$
 - 잘못 판정한 학습 데이터의 가중치는 증대
 - $w_i \leftarrow w_i e^{\alpha}$
 - 제대로 판정한 학습 데이터의 가중치는 축소
 - $w_i \leftarrow w_i e^{-\alpha}$
 - 가중치의 합이 1이 되도록 정규화

AdaBoost - 분류

❖ [실습] AdaBoost

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn import metrics

iris = datasets.load_iris()
X = iris.data    # 입력
y = iris.target  # 출력

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

abc = AdaBoostClassifier(n_estimators=50, learning_rate=1) # 모델 생성
model = abc.fit(X_train, y_train)

y_pred = model.predict(X_test) # 추론(예측)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

Accuracy: 0.9333333333333333

AdaBoost – 회귀

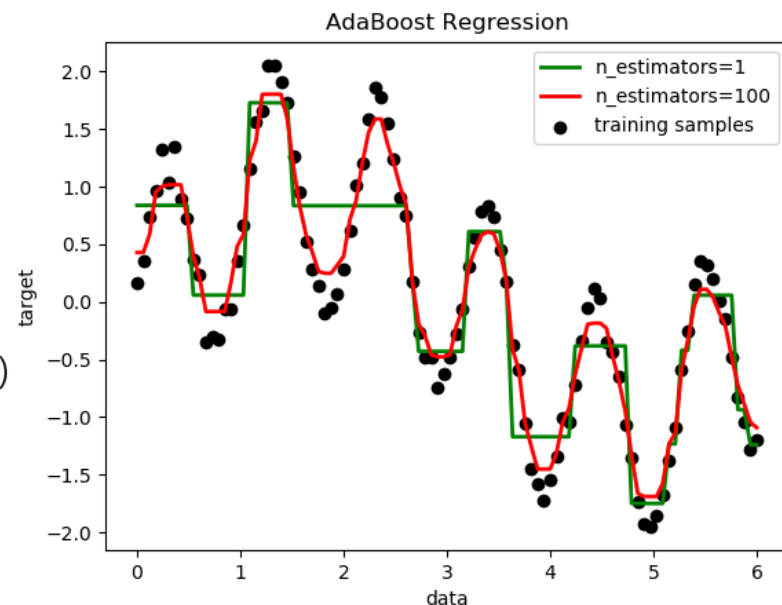
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor
```

```
rng = np.random.RandomState(1)
X = np.linspace(0, 6, 100)[:, np.newaxis]
y = np.sin(X).ravel() + np.sin(6 * X).ravel( ) + rng.normal(0, 0.1, X.shape[0])
```

```
regr_1 = DecisionTreeRegressor(max_depth=4)
regr_2 = AdaBoostRegressor(DecisionTreeRegressor(max_depth=4), n_estimators=100,
random_state=rng)
```

```
regr_1.fit(X, y)
regr_2.fit(X, y)
y_1 = regr_1.predict(X)
y_2 = regr_2.predict(X)
```

```
plt.figure()
plt.scatter(X, y, c="k", label="training samples")
plt.plot(X, y_1, c="g", label="n_estimators=1", linewidth=2)
plt.plot(X, y_2, c="r", label="n_estimators=100", linewidth=2)
plt.xlabel("data")
plt.ylabel("target")
plt.title(" AdaBoost Regression")
plt.legend( )
plt.show( )
```



Gradient Boosting - 회귀

❖ 그레디언트 부스팅(Gradient Boosting) 알고리즘

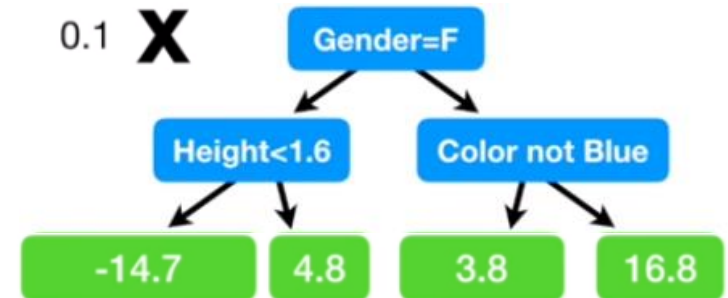
- 앙상블에 이전까지의 오차를 보정하도록 결정트리 모델을 순차적으로 추가

평균

Height (m)	Favorite Color	Gender	Weight (kg)	Residual	Residual
1.6	Blue	Male	88	16.8	15.1
1.6	Green	Female	76	4.8	4.3
1.5	Blue	Female	56	-15.2	-13.7
1.8	Red	Male	73	1.8	1.4
1.5	Green	Male	77	5.8	5.4
1.4	Blue	Female	57	-14.2	-12.7

71.2

0.1 X



잔차(residual)

$$= (88 - (71.2 + 0.1 \times 16.8)) = 15.1$$

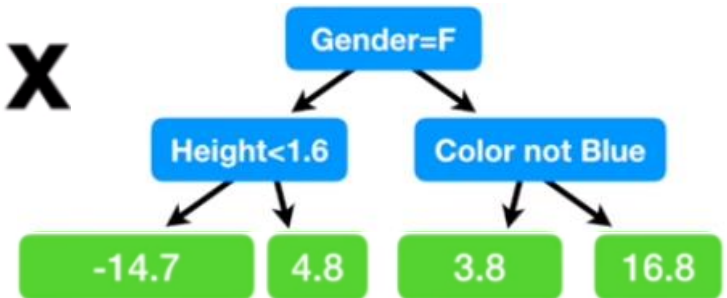
Gradient Boosting – cont.

❖ 그레디언트 부스팅(Gradient Boosting) 알고리즘

평균

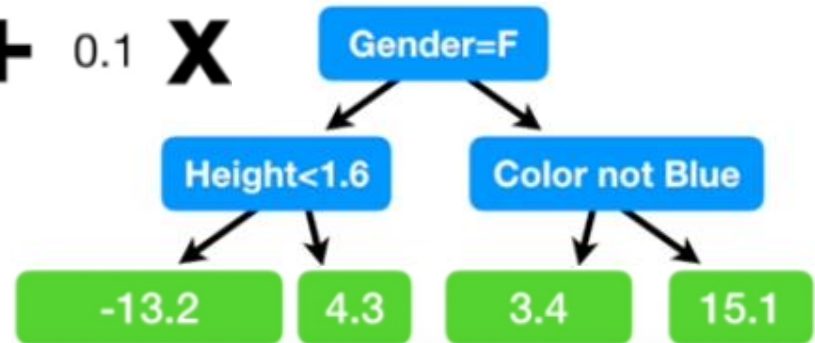
71.2

+ 0.1 X



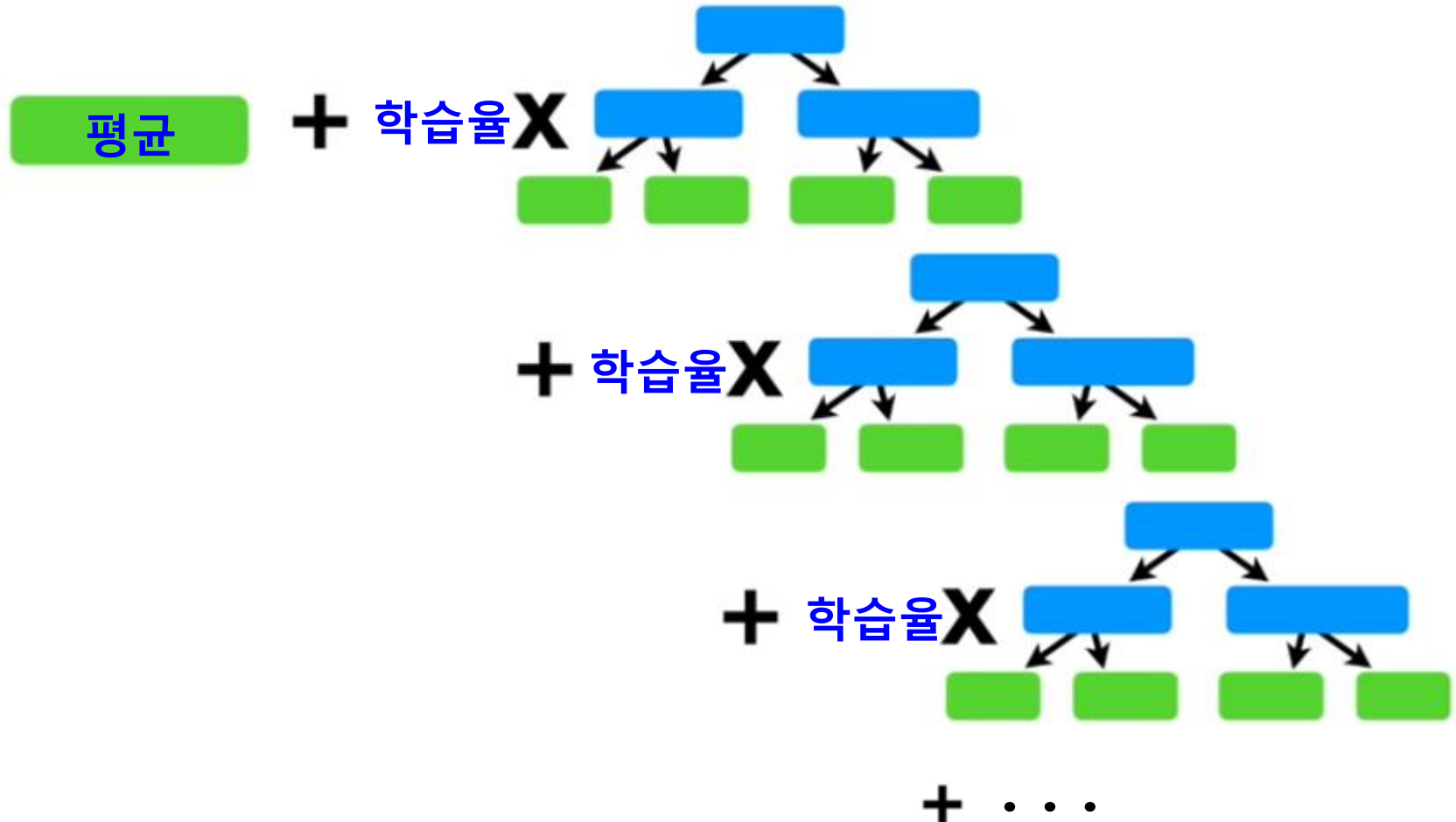
Height (m)	Favorite Color	Gender	Residual
1.6	Blue	Male	15.1
1.6	Green	Female	4.3
1.5	Blue	Female	-13.7
1.8	Red	Male	1.4
1.5	Green	Male	5.4
1.4	Blue	Female	-12.7

+ 0.1 X



Gradient Boosting – cont.

❖ 그레디언트 부스팅(Gradient Boosting) 알고리즘



❖ [실습] Gradient Boosting 기반 회귀

```
import numpy as np
import pandas as pd
from sklearn import datasets
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn import ensemble
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import cross_val_predict

boston = datasets.load_boston() # Boston 집값 데이터, 13개 속성, 마지막 중간값 정보
print(boston.data.shape, boston.target.shape)
print(boston.feature_names)

data = pd.DataFrame(boston.data, columns=boston.feature_names)
data = pd.concat([data, pd.Series(boston.target, name='MEDV')], axis=1)
print(data.head( ))

X = data.iloc[:, :-1]
y = data.iloc[:, -1]
x_training_set, x_test_set, y_training_set, y_test_set = train_test_split(X, y, test_size=0.10,
                                     random_state=42, shuffle=True)
```

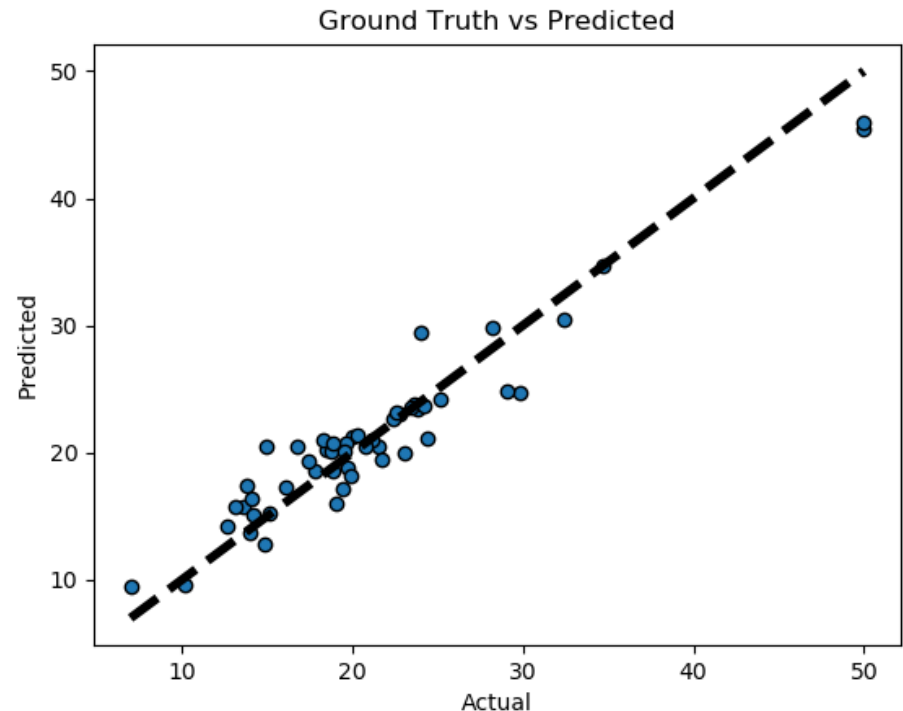
```
(506, 13) (506,)
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO' 'B' 'LSTAT']
      CRIM  ZN  INDUS  CHAS  NOX  ...  TAX  PTRATIO    B  LSTAT  MEDV
0  0.00632 18.0   2.31   0.0  0.538  ... 296.0    15.3 396.90  4.98  24.0
1  0.02731  0.0   7.07   0.0  0.469  ... 242.0    17.8 396.90  9.14  21.6
2  0.02729  0.0   7.07   0.0  0.469  ... 242.0    17.8 392.83  4.03  34.7
3  0.03237  0.0   2.18   0.0  0.458  ... 222.0    18.7 394.63  2.94  33.4
4  0.06905  0.0   2.18   0.0  0.458  ... 222.0    18.7 396.90  5.33  36.2
[5 rows x 14 columns]
```

```
params = {'n_estimators': 500, 'max_depth': 4, 'min_samples_split': 2, 'learning_rate': 0.01, 'loss': 'ls'}
model = ensemble.GradientBoostingRegressor(**params)
model.fit(x_training_set, y_training_set)
model_score = model.score(x_training_set, y_training_set)
print('R2 sq: ', model_score)
```

```
y_predicted = model.predict(x_test_set)
print("Mean squared error: %.2f" % mean_squared_error(y_test_set, y_predicted))
print('Test Variance score: %.2f' % r2_score(y_test_set, y_predicted))
```

```
fig, ax = plt.subplots()
ax.scatter(y_test_set, y_predicted, edgecolors=(0, 0, 0))
ax.plot([y_test_set.min(), y_test_set.max()], [y_test_set.min(), y_test_set.max()], 'k--', lw=4)
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title("Ground Truth vs Predicted")
plt.show()
```

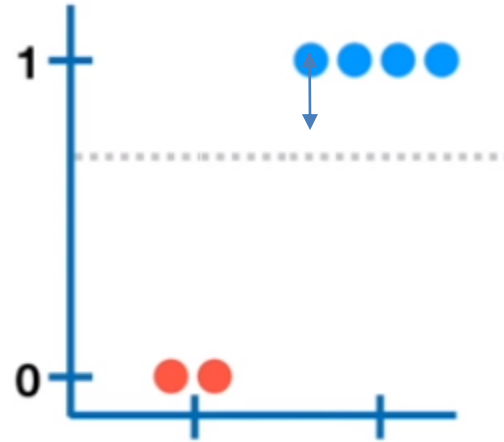
R2 sq: 0.9800347273281851
Mean squared error: 5.43
Test Variance score: 0.91



Gradient Boosting - 분류

❖ 그레디언트 부스팅 알고리즘 - 분류

- 이진 분류 문제
 - 부류 {A, B} \rightarrow {0, 1}



- 다부류 분류 문제 적용 가능

❖ [실습] Gradient Boosting 기반 분류

```
from sklearn.datasets import make_hastie_10_2
from sklearn.ensemble import GradientBoostingClassifier
import matplotlib.pyplot as plt

X, y = make_hastie_10_2(random_state=0)
X_train, X_test = X[:2000], X[2000:]
y_train, y_test = y[:2000], y[2000:]
print(X.shape, y.shape)
print(X[0:5,:])
print(y[0:5])

clf = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1,
                                max_depth=1, random_state=0)
clf.fit(X_train, y_train)
print("Accuracy score (training): {0:.3f}".format(clf.score(X_train, y_train)))
print("Accuracy score (testing): {0:.3f}".format(clf.score(X_test, y_test)))
```

```
(12000, 10) (12000,)
[[ 1.76405235  0.40015721  0.97873798  2.2408932  1.86755799 -0.97727788
   0.95008842 -0.15135721 -0.10321885  0.4105985 ]
 [ 0.14404357  1.45427351  0.76103773  0.12167502  0.44386323  0.33367433
   1.49407907 -0.20515826  0.3130677  -0.85409574]
 [-2.55298982  0.6536186  0.8644362  -0.74216502  2.26975462 -1.45436567
   0.04575852 -0.18718385  1.53277921  1.46935877]
 [ 0.15494743  0.37816252 -0.88778575 -1.98079647 -0.34791215  0.15634897
   1.23029068  1.20237985 -0.38732682 -0.30230275]
 [-1.04855297 -1.42001794 -1.70627019  1.9507754  -0.50965218 -0.4380743
  -1.25279536  0.77749036 -1.61389785 -0.21274028]]
[ 1. -1.  1. -1.  1.]
Accuracy score (training): 0.879
Accuracy score (testing): 0.819
```

XGBoost

❖ XGBoost (Extreme Gradient Boosting)

- Gradient Boost 기법의 개선
- 분류와 회귀 문제 지원
- 결손값(missing value)이 있는 데이터 지원
- 대규모 데이터 처리를 위해 분산환경에서도 실행 가능
- 모델 복잡도 및 과적합 조정을 위한 하이퍼파라미터 포함
- Kaggle의 대부분의 문제에서 상위 성능의 모델에서 사용
- 다양한 클라우드 서비스에서 제공

❖ [실습] XGBoosting 기반 회귀

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
import xgboost as xgb # https://www.lfd.uci.edu/~gohlke/pythonlibs/#xgboost

boston = load_boston()
data = pd.DataFrame(boston.data)
data.columns = boston.feature_names
data['PRICE'] = boston.target
print(data.head())
X, y = data.iloc[:, :-1], data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)
xg_reg = xgb.XGBRegressor(objective='reg:squarederror', colsample_bytree=0.3,
                           learning_rate=0.1, max_depth=5, alpha=10, n_estimators=10)
xg_reg.fit(X_train, y_train)
preds = xg_reg.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, preds))
print("RMSE: %f" % (rmse))
```

	CRIM	ZN	INDUS	CHAS	NOX	...	TAX	PTRATIO	B	LSTAT	PRICE
0	0.00632	18.0	2.31	0.0	0.538	...	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	...	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	...	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	...	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	...	222.0	18.7	396.90	5.33	36.2

[5 rows x 14 columns]
RMSE: 10.423243

Quiz

1. 붓스트랩은 데이터 집합에서 복원추출하여 다수의 학습 데이터 집합을 만들어내는 기법이다. (O,X)
2. 배깅은 여러 분류기를 순차적으로 만드는 앙상블 기법이다. (O,X)
3. 부스팅은 붓스트랩을 통해 여러 개의 학습 데이터 집합 생성하는 기법이다. (O,X)
4. 랜덤포리스트의 트리를 생성하는 과정에서 분할속성은 무작위로 선택된 것들 중에서 찾는다. (O,X)
5. Gradient Boosting에서는 직전까지 생성한 결정트리들을 결합하여 계산한 값과 목표값을 차이를 줄이기 위해 결정트리를 생성한다. (O,X)
6. XGB 알고리즘은 결손값이 있는 데이터에도 적용될 수 있다. (O,X)