앙상블 학습
Ensemble Learning

이건명

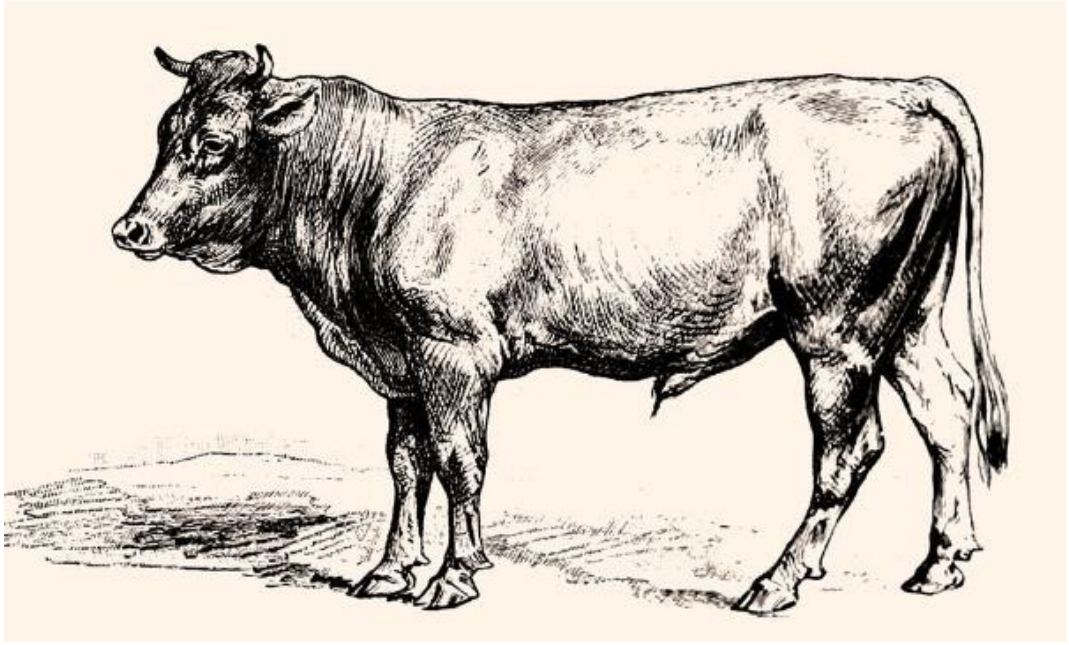
충북대학교 대학원 산업인공지능학과

학습 내용

- **앙상블 학습**의 전략에 대해서 알아본다.
- **배깅 기법**으로 결정트리를 이용한 방법과 Random Forest 알고리즘을 알아본다.
- **부스팅 기법**으로 AdaBoost, Gradient Boost, XGB 알고리즘을 알아본다.

The wisdom of crowds

- **British polymath Francis Galton, 1906**
  - Guess the weight of this ox at a livestock fair
  - Some 800 people submitted their guesse
  - weigh 1,198 pounds
  - took the average of all the guesses: 1,197 pounds



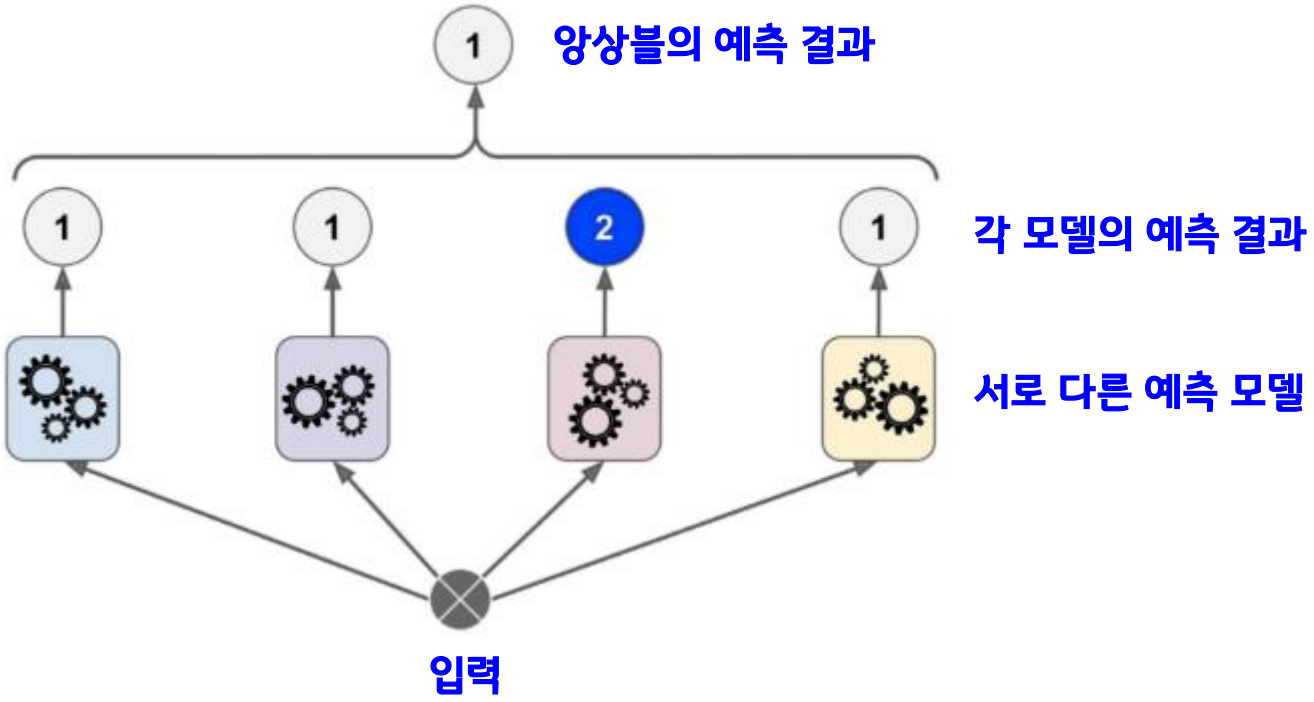
1. 앙상블 학습

❖ **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



Source: Hands-on Machine Learning with Scikit-Learn & TensorFlow, ch7

앙상블 학습

❖ 붓스트랩(bootstrap)

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

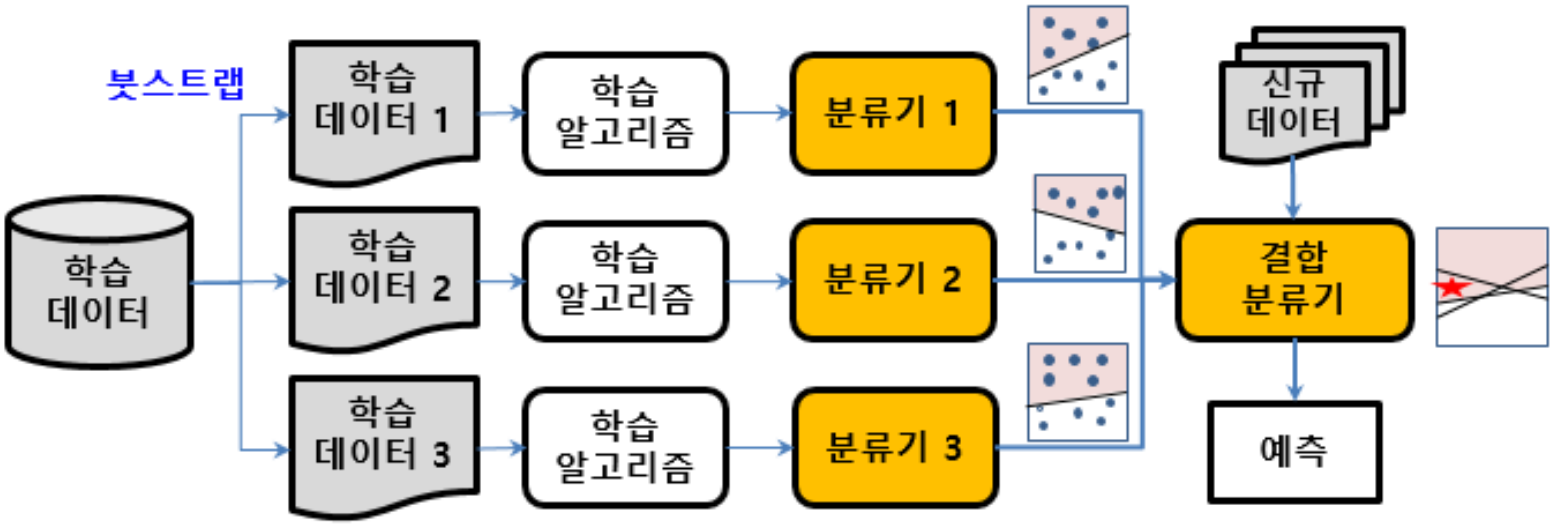
❖ 배깅(bagging, bootstrap aggregating)

❖ 부스팅(boosting)

2. 배깅 알고리즘

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

- **붓스트랩**을 통해 **여러 개**의 **학습 데이터 집합** 생성
- 각 학습 데이터 집합별로 **분류기** 또는 **회귀모델 생성**
- 최종판정
  - 분류기들의 **투표**나 **가중치 투표**
  - 회귀 모델들의 **평균**



❖ [실습] 배깅

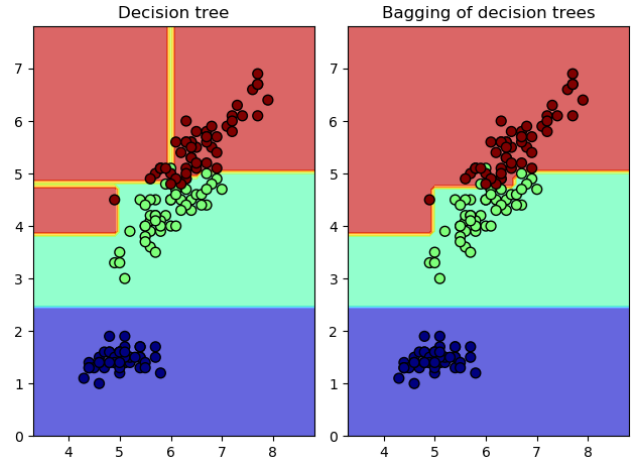
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
iris = load_iris( )
X, y = iris.data[:, [0, 2]], iris.target

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( )
```



https://scikit-learn.org/

배깅 알고리즘 : 랜덤 포리스트

❖ **랜덤 포리스트(random forest) 알고리즘**

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

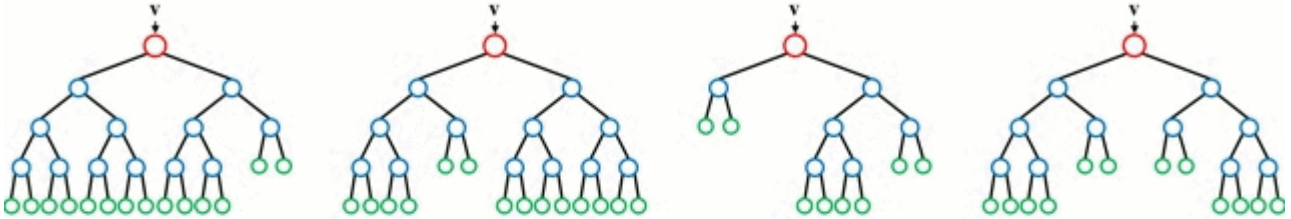


image source: Wikipedia.org

❖ [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)  # 데스트 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
```

배깅 알고리즘

❖ 배깅에 의한 회귀

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

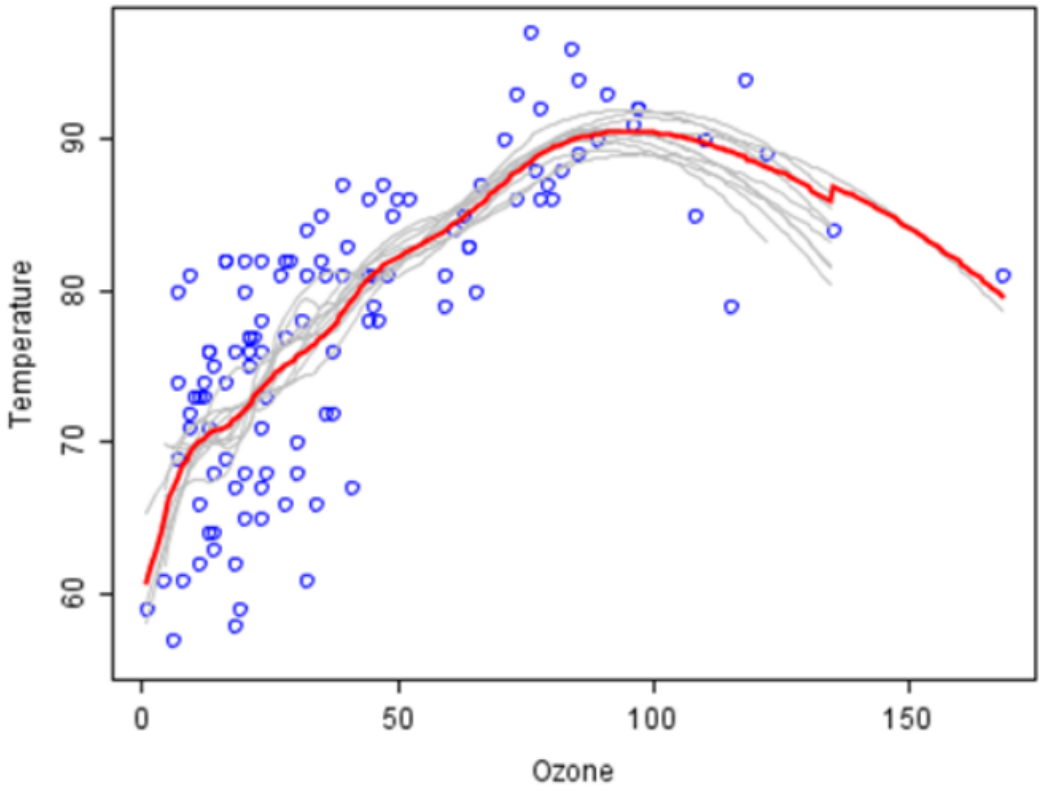


image source: Wikipedia.org

❖ [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))
```

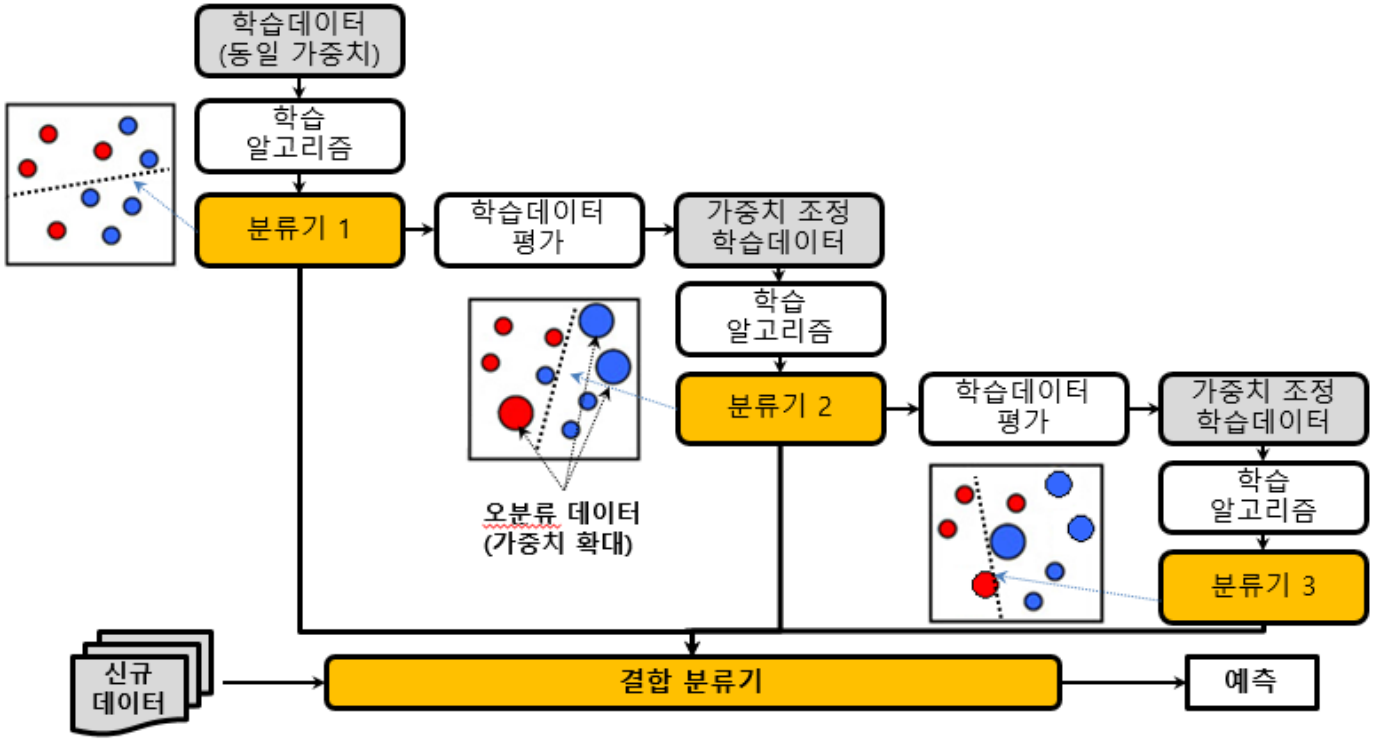
```
     CRIM    ZN  INDUS  CHAS    NOX  ...    TAX  PTRATIO       B  LSTAT  PRICE
0  0.00632  18.0   2.31   0.0  0.538  ...  296.0     15.3  396.90   4.98   24.0
1  0.02731   0.0   7.07   0.0  0.469  ...  242.0     17.8  396.90   9.14   21.6
2  0.02729   0.0   7.07   0.0  0.469  ...  242.0     17.8  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
```

https://scikit-learn.org/

3. 부스팅 알고리즘

❖ **부스팅(boosting)**

- k개의 예측 모델을 **순차적**으로 만들어 가는 **앙상블 모델 생성**
- **오차**에 따라 **학습 데이터**에 **가중치** 또는 **값**을 **변경**해가면서 예측 모델 생성



- **AdaBoost**
- **Gradient Boosting**
- **XGB**

AdaBoost - 분류

❖ AdaBoost

- N개의 학습 데이터 d_i에 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 분류기들 만을 사용
- **학습**
  - 학습 오류값이 0.5미만인 분류기가 학습되는 경우
  - **분류기 신뢰도** : α
    - $\alpha = 0.5 \ln(\frac{1-\epsilon}{\epsilon})$
  - **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

https://scikit-learn.org/

AdaBoost – 회귀

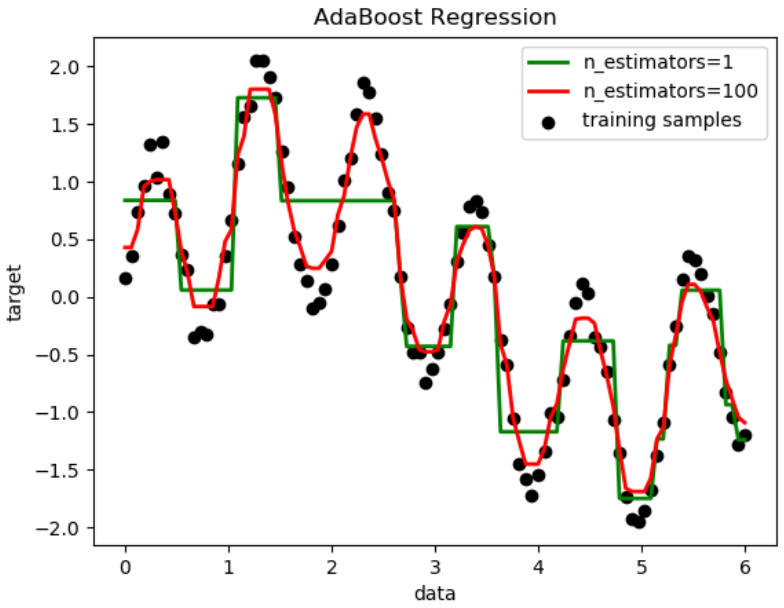
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor

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( )
```

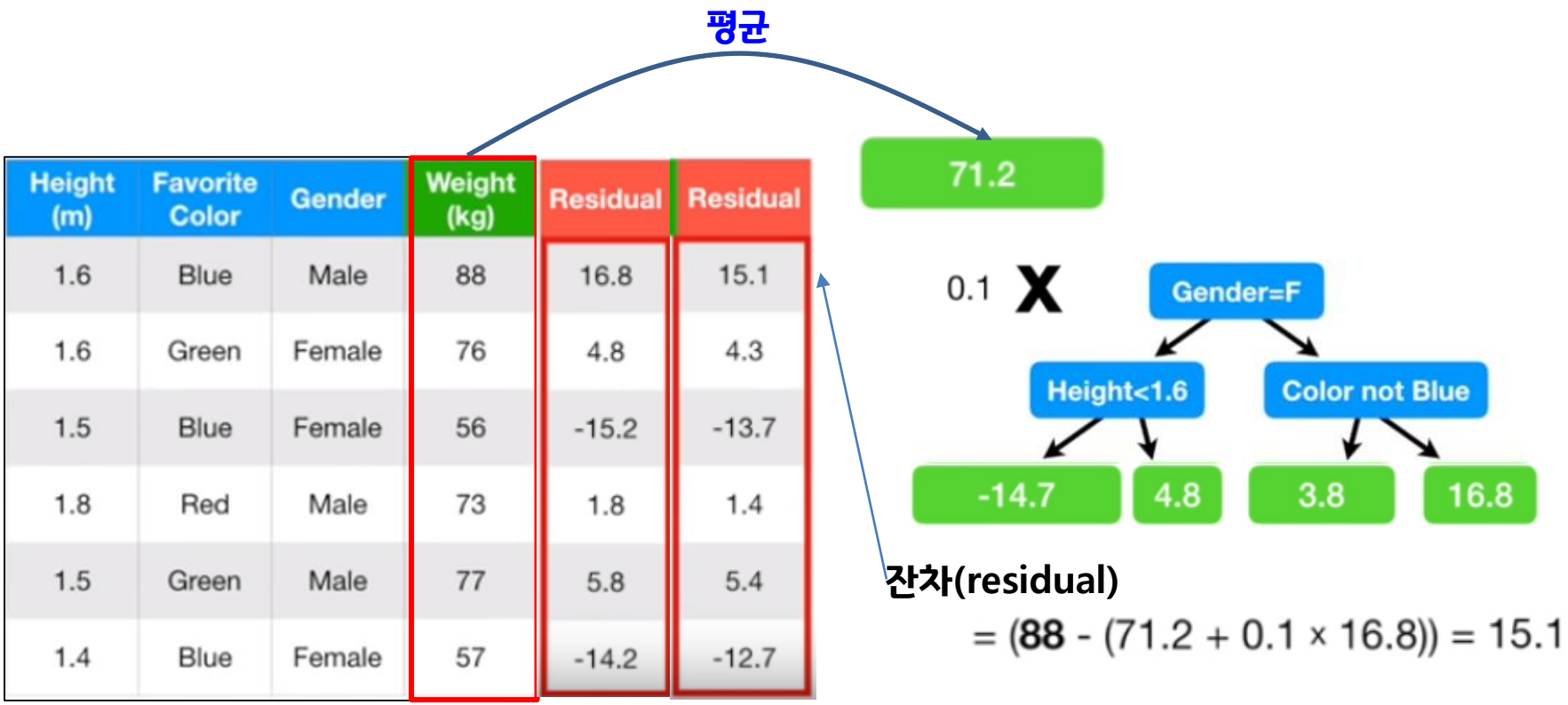


https://scikit-learn.org/

Gradient Boosting - 회귀

❖ **그레디언트 부스팅(Gradient Boosting) 알고리즘**

- 앙상블에 이전까지의 오차를 보정하도록 결정트리 모델을 순차적으로 추가

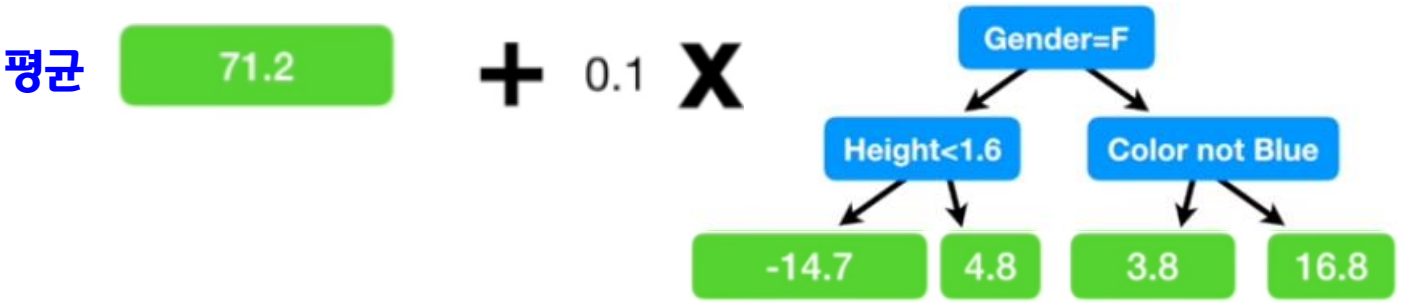


Height (m)	Favorite Color	Gender	Weight (kg)	Residual	Residual
1.6	Blue	Male	88	16.8	15.1
1.6	Green	Female	76	4.8	4.3
1.5	Blue	Female	56	-15.2	-13.7
1.8	Red	Male	73	1.8	1.4
1.5	Green	Male	77	5.8	5.4
1.4	Blue	Female	57	-14.2	-12.7

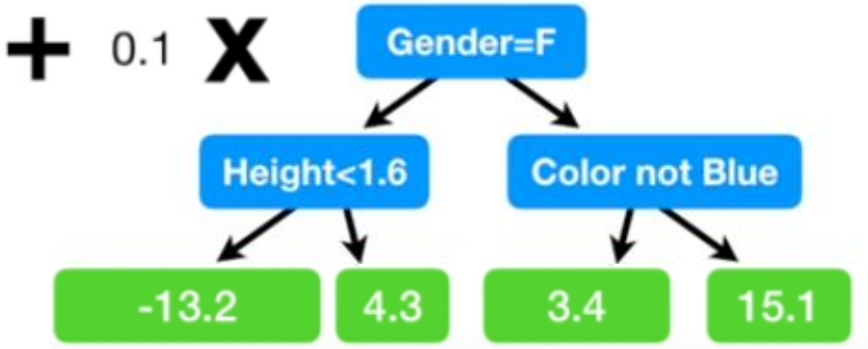
https://www.youtube.com/watch?v=3CC4N4z3GJc&t=532s

Gradient Boosting – cont.

❖ 그레디언트 부스팅(Gradient Boosting) 알고리즘



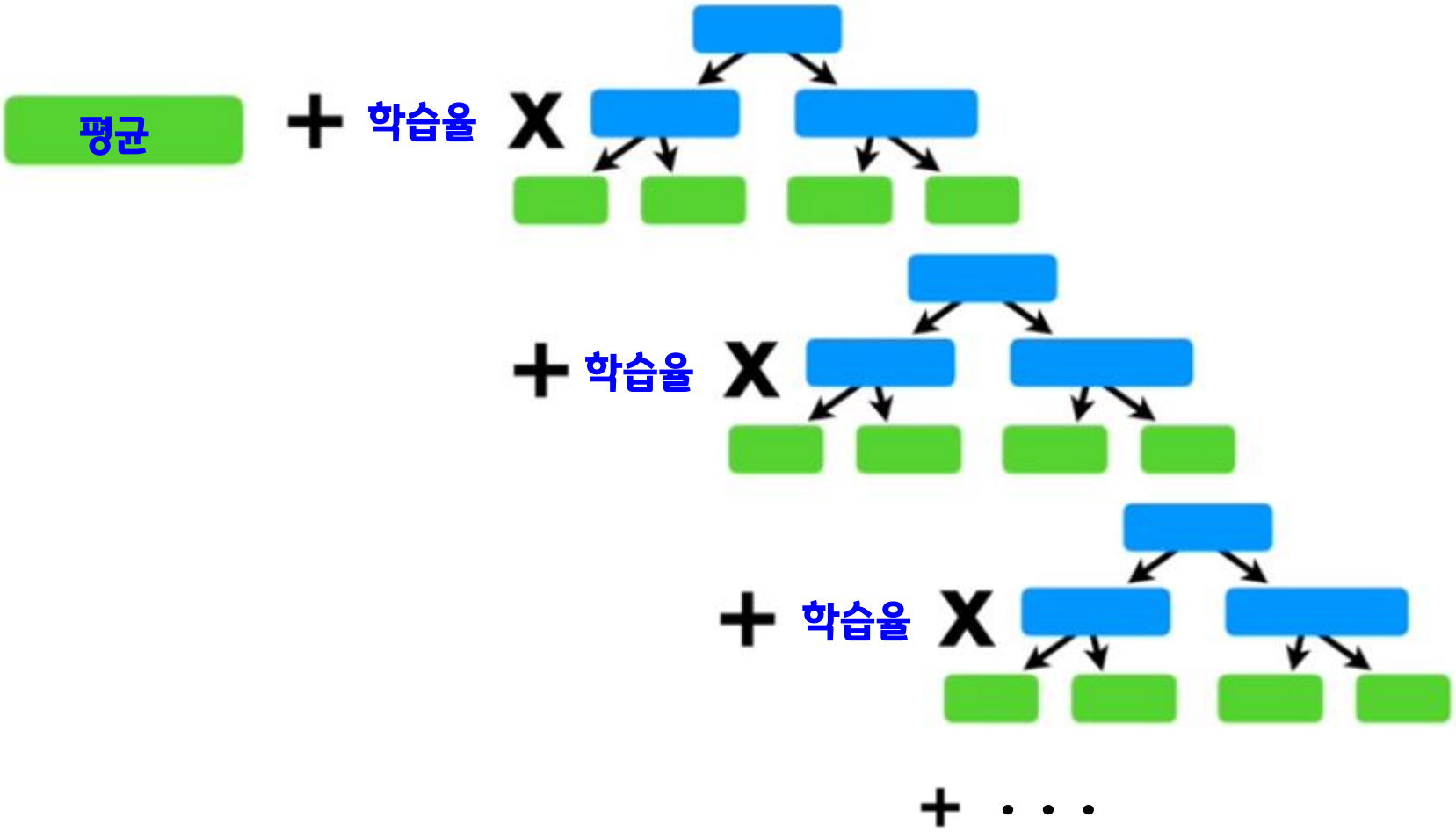
Height (m)	Favorite Color	Gender	Residual
1.6	Blue	Male	15.1
1.6	Green	Female	4.3
1.5	Blue	Female	-13.7
1.8	Red	Male	1.4
1.5	Green	Male	5.4
1.4	Blue	Female	-12.7



https://www.youtube.com/watch?v=3CC4N4z3GJc&t=532s

Gradient Boosting – cont.

❖ 그레디언트 부스팅(Gradient Boosting) 알고리즘



https://www.youtube.com/watch?v=3CC4N4z3GJc&t=532s

❖ [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)
```

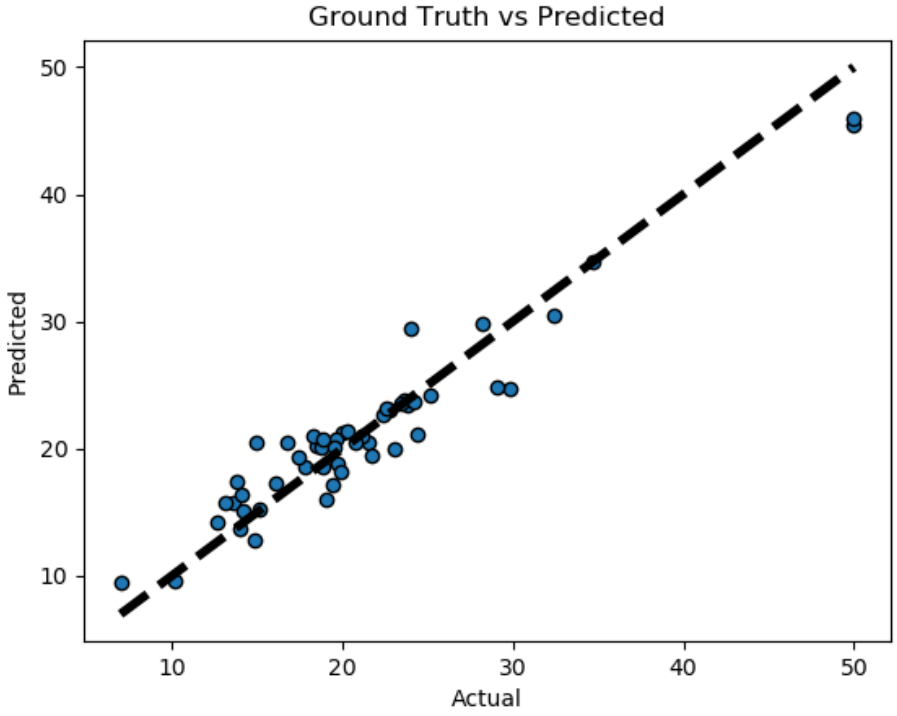
```
(506, 13) (506,)
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO' 'B' 'LSTAT']
      CRIM    ZN  INDUS  CHAS    NOX  ...    TAX  PTRATIO       B  LSTAT  MEDV
0  0.00632  18.0   2.31   0.0  0.538  ...  296.0     15.3  396.90   4.98  24.0
1  0.02731   0.0   7.07   0.0  0.469  ...  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()
```

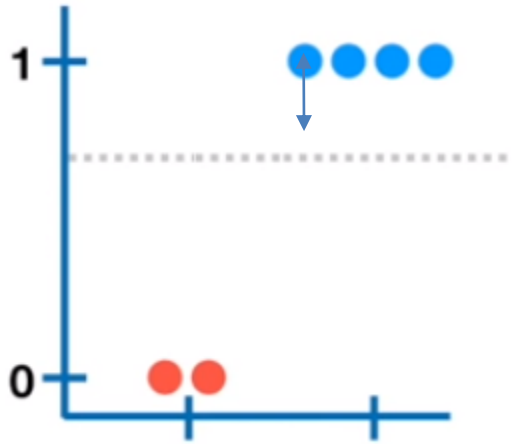
R2 sq: 0.9800347273281851
Mean squared error: 5.43
Test Variance score: 0.91



Gradient Boosting - 분류

❖ 그레디언트 부스팅 알고리즘 – 분류

- 이진 분류 문제
  - 부류 {A, B} ➔ {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
```

https://scikit-learn.org/

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
```

https://scikit-learn.org/

LightGBM

❖ LightGBM

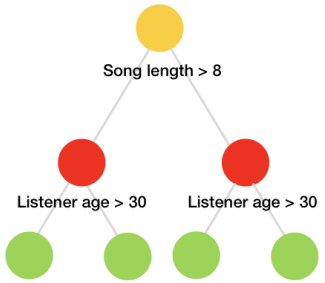
- Microsoft에서 개발한 오프소스
- XGBoost 대비하여 **메모리 사용량** 및 **처리속도**를 **개선**한 방법
- 효율적인 대규모 데이터 처리
  - 히스토그램 기반의 접근 방식
  - **히스토그램** 사용한 **spilt 위치** 결정
- 단말(leaf) 노드 중심 분할(leaf-wise split)
  - 대부분의 부스팅 알고리즘들은 깊이에 기반하여 트리를 균등하게 성장(너비 우선 분할)
  - LightGBM은 단말 중심 분할 사용 – 정확도 개선 가능성(과적합 위험)
- 빠른 학습과 예측 시간
  - GPU를 활용한 학습 지원
- 범주형 특징의 자동 변환과 최적 분할
  - 추가적인 인코딩 없이 범주형 데이터를 쉽게 처리
- 결손값(missing value) 자동 처리
- 분류(Classification), 회귀(Regression), 순위(Ranking) 문제 등 다양한 학습 문제에 적용 가능

CatBoost

Yandex CatBoost

❖ CatBoost (Categorical Boosting)

- 러시아 **Yandex**사에게 개발한 오프소스 라이브러리
- **Gradient boosting**의 기법을 따르는 **트리** 기반의 **앙상블 모델**
- **범주형 속성**을 수치속성으로 표현하여 사용
  - 별도 전처리 불필요
- 결손값(missing value) 자동 처리
- **텍스트 속성** 처리 지원
- 시계열(time series) 데이터 지원
  - 시간에 따라 변화하는 데이터에 대한 예측 가능
- **대칭형 트리**(oblivious tree)로 구성된 앙상블 모델
  - 균형 트리(balanced trees) 사용
- 대규모 데이터 처리 지원
- GPU와 멀티 CPU 환경 모두 지원
- **회귀, 분류, 다부류 분류, 순서결정**(ranking) 지원

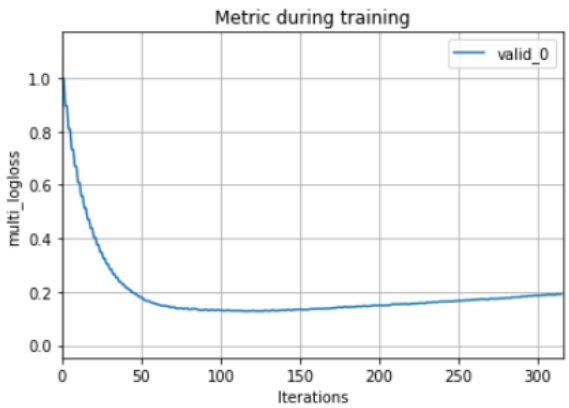


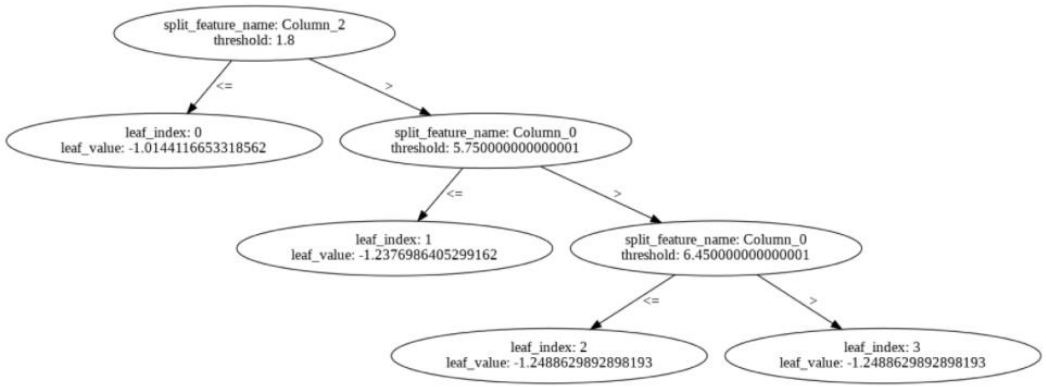
```
!pip install lightgbm
from lightgbm import LGBMClassifier, LGBMRegressor
from lightgbm import plot_importance, plot_metric, plot_tree
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate

iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.2, random_state=123)
lgbmc = LGBMClassifier(n_estimators=400)
evals = [(X_test, y_test)]
lgbmc.fit(X_train, y_train, early_stopping_rounds=100, eval_metric='logloss', eval_set=evals, verbose=True)
preds = lgbmc.predict(X_test)

cross_val = cross_validate(
    estimator=lgbmc,
    X=iris.data, y=iris.target,
    cv=5
)
print('avg fit time: {} (+/- {})'.format(cross_val['fit_time'].mean(), cross_val['fit_time'].std()))
print('avg score time: {} (+/- {})'.format(cross_val['score_time'].mean(), cross_val['score_time'].std()))
print('avg test score: {} (+/- {})'.format(cross_val['test_score'].mean(), cross_val['test_score'].std()))

plot_metric(lgbmc)
plot_importance(lgbmc, figsize=(10,12))
plot_tree(lgbmc,figsize=(28,24))
```





Quiz

❖ 랜덤 포리스트는 어떤 앙상블 학습 기법에 속하나요?

① 부스팅
② 배깅
③ 스태킹
④ 교차 검증

❖ AdaBoost에서 잘못 분류된 데이터에 대한 가중치 업데이트 방식은?

① 가중치 감소
② 가중치 유지
③ 가중치 증가
④ 가중치 초기화

❖ 랜덤 포리스트의 특성 선택에 대한 설명으로 옳지 않은 것은?

① 무작위로 특성을 선택한다.
② 선택된 특성 중 최적의 분할을 찾는다.
③ 모든 특성을 고려하여 최적의 분할을 찾는다.
④ 각 결정 트리는 서로 다른 특성의 부분집합을 사용한다.

Quiz

❖ **배깅에 대한 설명으로 옳지 않는 것을 선택하시오.**

① 주어진 학습데이터 집합에서 복원추출 방법으로 다수의 학습 데이터 집합을 만들어서 사용한다.

② 만들어진 각 학습 데이터 집합별로 분류기 또는 회귀 모델을 생성한다.

③ 복수의 예측 모델을 순차적으로 만들어가게 된다.

④ Random Forest 알고리즘은 배깅 기법의 하나이다.

❖ **Gradient Boosting 방법에 대한 설명으로 옳지 않은 것을 선택하시오.**

① Gradient Boosting 방법은 분류와 회귀 문제에 적용될 수 있다.

② Gradient Boosting을 회귀 문제에 적용할 때는 트리의 단말노드에 log(odds) 값이 부여된다.

③ Gradient Boosting은 결정트리를 순차적으로 생성하여 앙상블을 구성한다.

④ Gradient Boosting은 직전 단계까지의 모델에 의한 예측 오류를 다음 결정트리가 축소하도록 학습한다.

Quiz

❖ 배깅의 주요 특징으로 올바르지 않은 것은?

① 복원추출을 사용한다.
② 각 모델은 독립적으로 학습된다.
③ 최종 결정은 투표나 평균 등으로 이루어진다.
④ 순차적 모델 학습을 진행한다.

❖ 부스팅 기법에서 잘못된 설명은?

① 순차적 모델 학습
② 오차를 줄이는 방향으로 가중치 업데이트
③ 각 모델은 독립적으로 학습
④ AdaBoost와 Gradient Boost가 있다.

❖ AdaBoost에서 데이터들에 대한 초기 가중치의 합은?

① 0.5
② 1
③ 2
④ 가중치는 초기에 설정되지 않는다.

Quiz

❖ **앙상블 기법 중 배깅의 대표적인 예는?**

① AdaBoost
② Random Forest
③ Gradient Boosting
④ SVM

❖ **부스팅 알고리즘의 기본 아이디어는?**

① 모든 모델을 동시에 학습시킨다.
② 한 번의 학습으로 최적의 모델을 찾는다.
③ 이전 모델의 오류를 보정하면서 순차적으로 모델을 학습시킨다.
④ 특정 데이터를 선택하여 학습시킨다.

❖ **앙상블 모델에서 사용되는 개별 모델들은 어떠한 특징을 가지는 것이 바람직한가?**

① 서로 독립적이다.
② 서로 동일한 특성을 가진다.
③ 서로의 오류를 반복한다.
④ 서로의 성능이 동일하다.

Quiz

❖ CatBoost가 다루기 어려운 데이터 유형은?

① 텍스트 데이터
② 범주형 데이터
③ 결손값을 포함한 데이터
④ 이미지 데이터

❖ 앙상블 학습에서 다양성을 확보하는 이유는?

① 모델 복잡도를 증가시키기 위해
② 모든 데이터 포인트를 정확히 예측하기 위해
③ 일반화 성능을 향상시키기 위해
④ 학습 시간을 단축시키기 위해

❖ Gradient Boosting에서 '그레디언트'는 무엇을 의미하는가?

① 학습률
② 손실 함수의 기울기
③ 모델의 복잡도
④ 데이터의 분산