zero-base /

Life Changing Education

Chapter 18. Boosting Algorithm

앙상블 기법

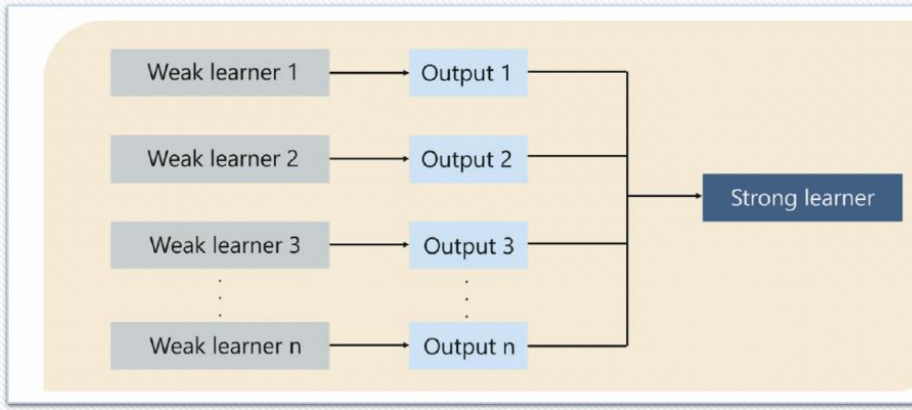
zero-base /

앙상블 기법

- 앙상블은 전통적으로 Voting, Bagging, Boosting, 스태깅 등으로 나눔
- 보팅과 배깅은 여러개의 분류기가 투표를 통해 최종 예측 결과를 결정하는 방식
- 보팅과 배깅의 차이점은 보팅은 각각 다른 분류기, 배깅은 같은 분류기를 사용
- 대표적인 배깅 방식이 랜덤 포레스트

zero-base /

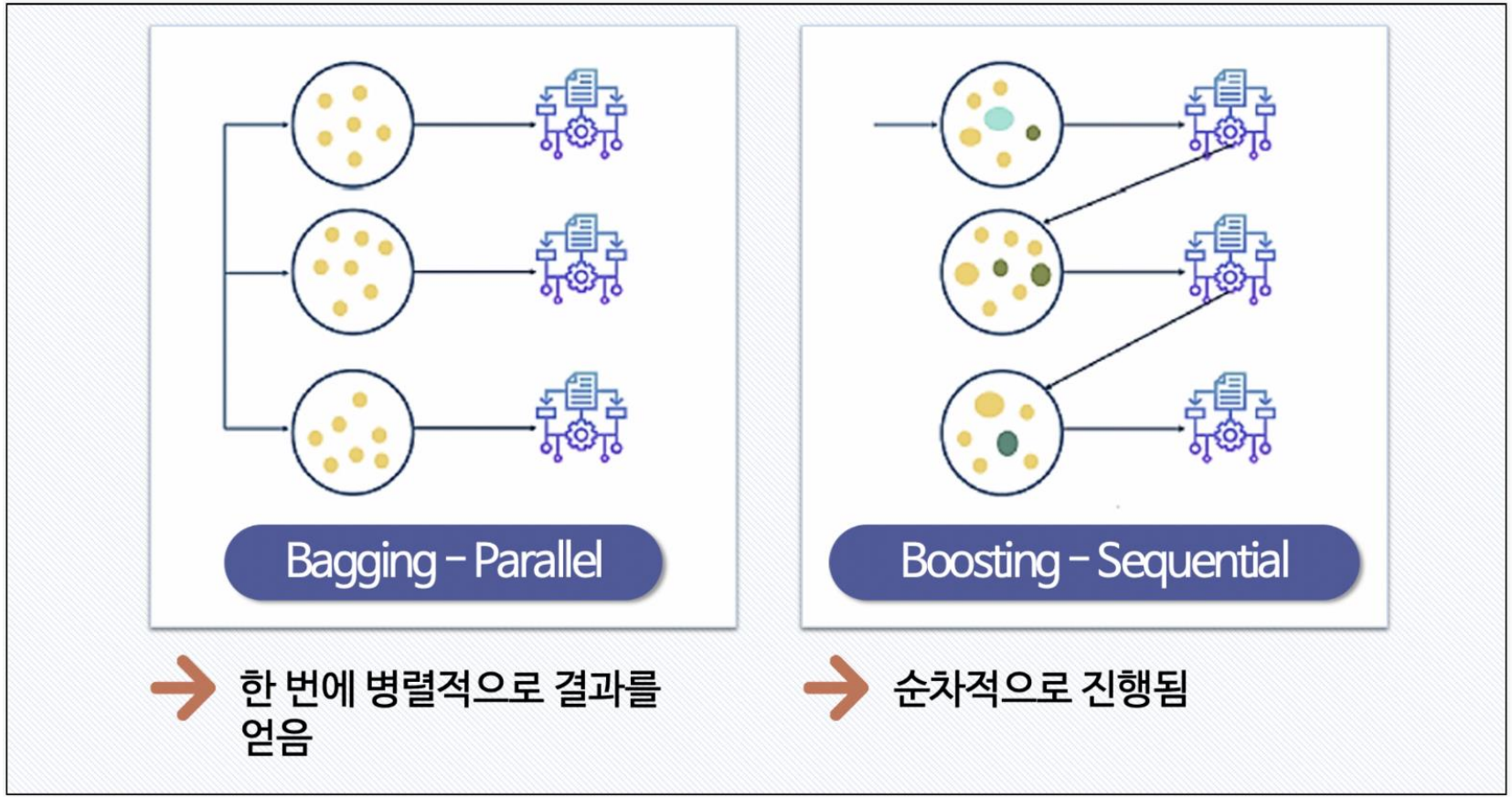
Boosting 의 개요



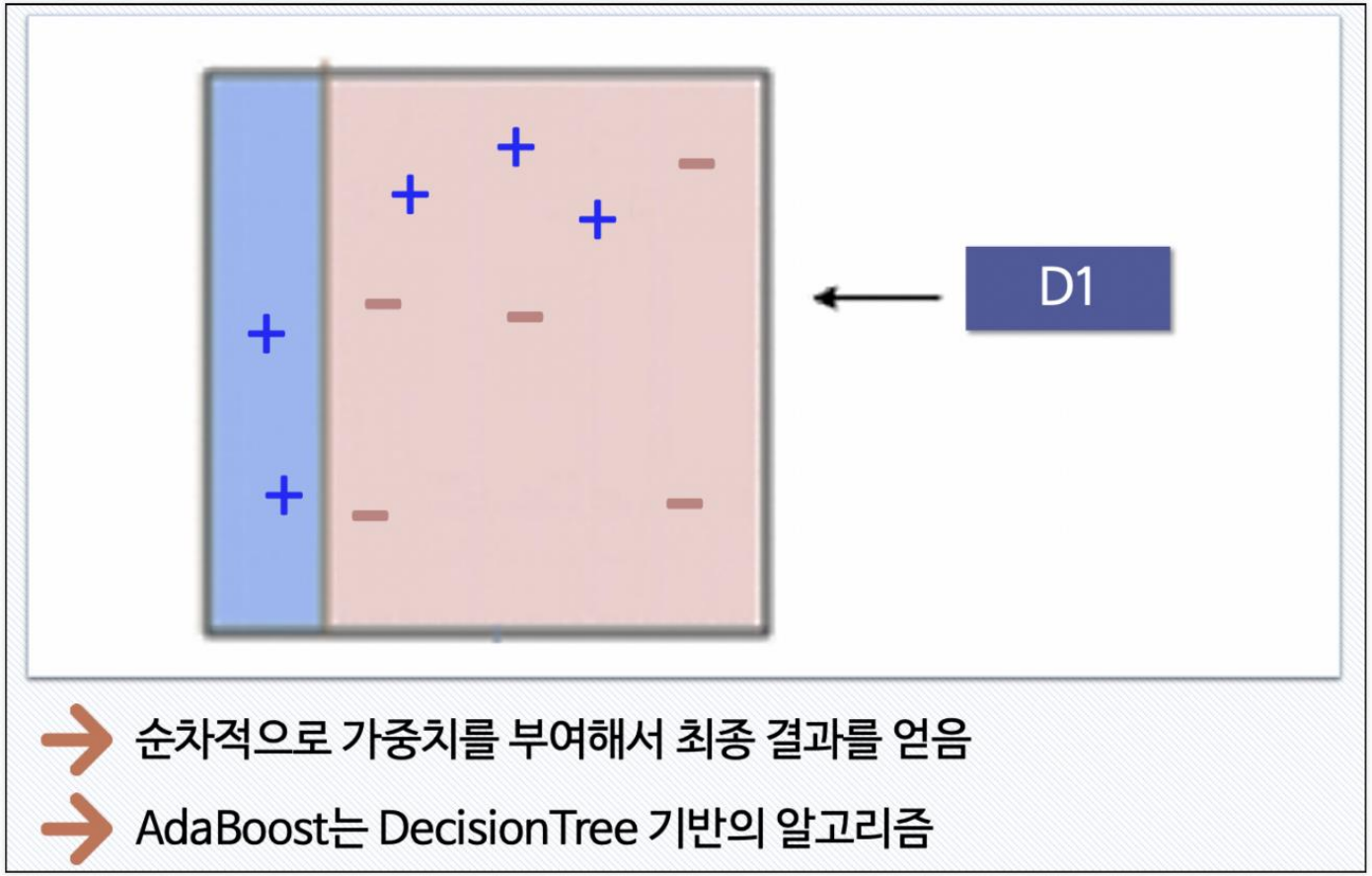
- 여러 개의 (약한)분류기가 순차적으로 학습을 하면서, 앞에서 학습한 분류기가 예측이 틀린 데이터에 대해 다음 분류기가 가중치를 인가해서 학습을 이어 진행하는 방식
- 예측 성능이 뛰어나서 앙상블 학습을 주도하고 있음
  - 그래디언트부스트, XGBoost(eXtra Gradient Boost), LightGBM(Light Gradient Boost) 등이 있음

zero-base /

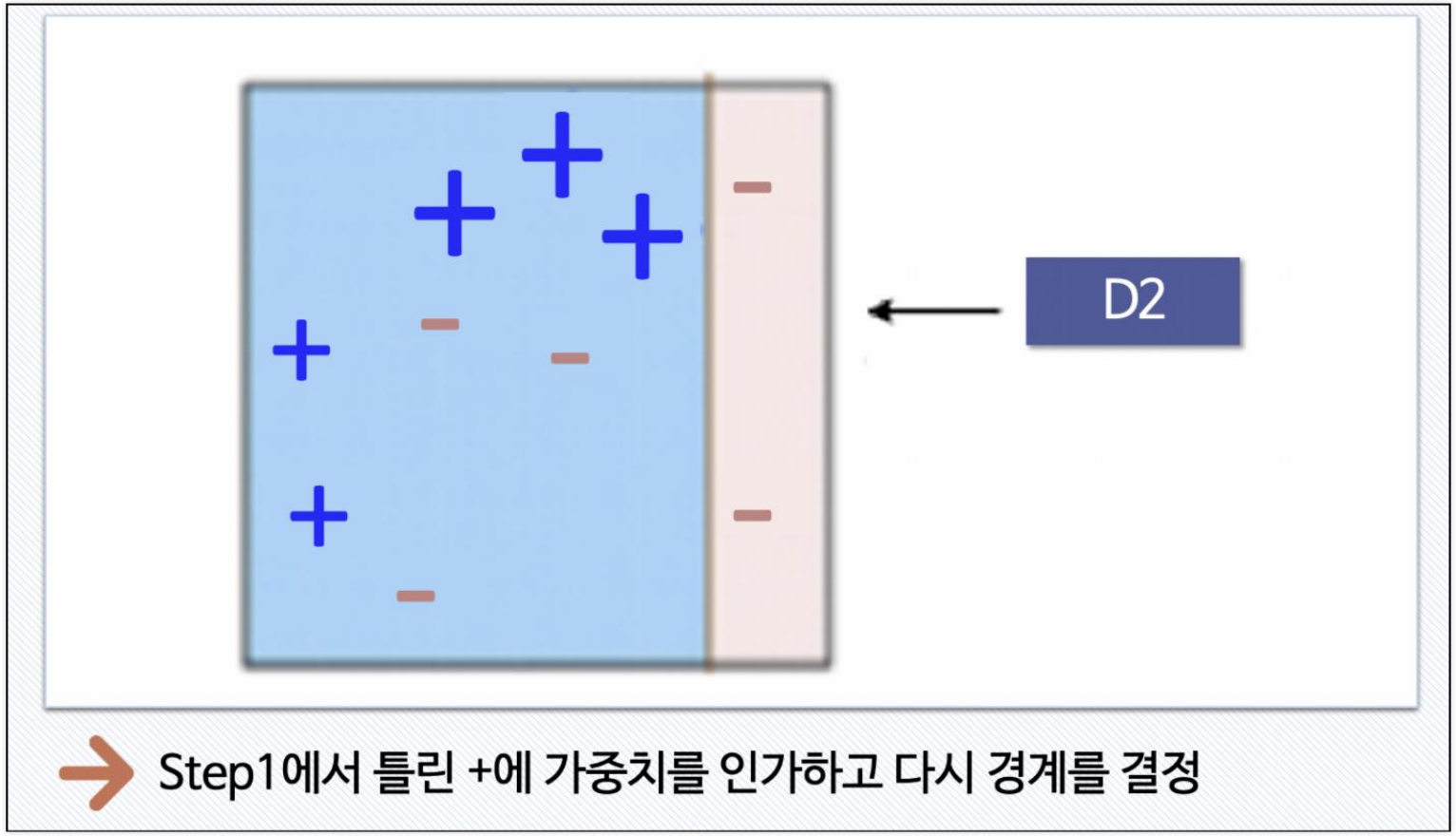
배깅과 부스팅의 차이



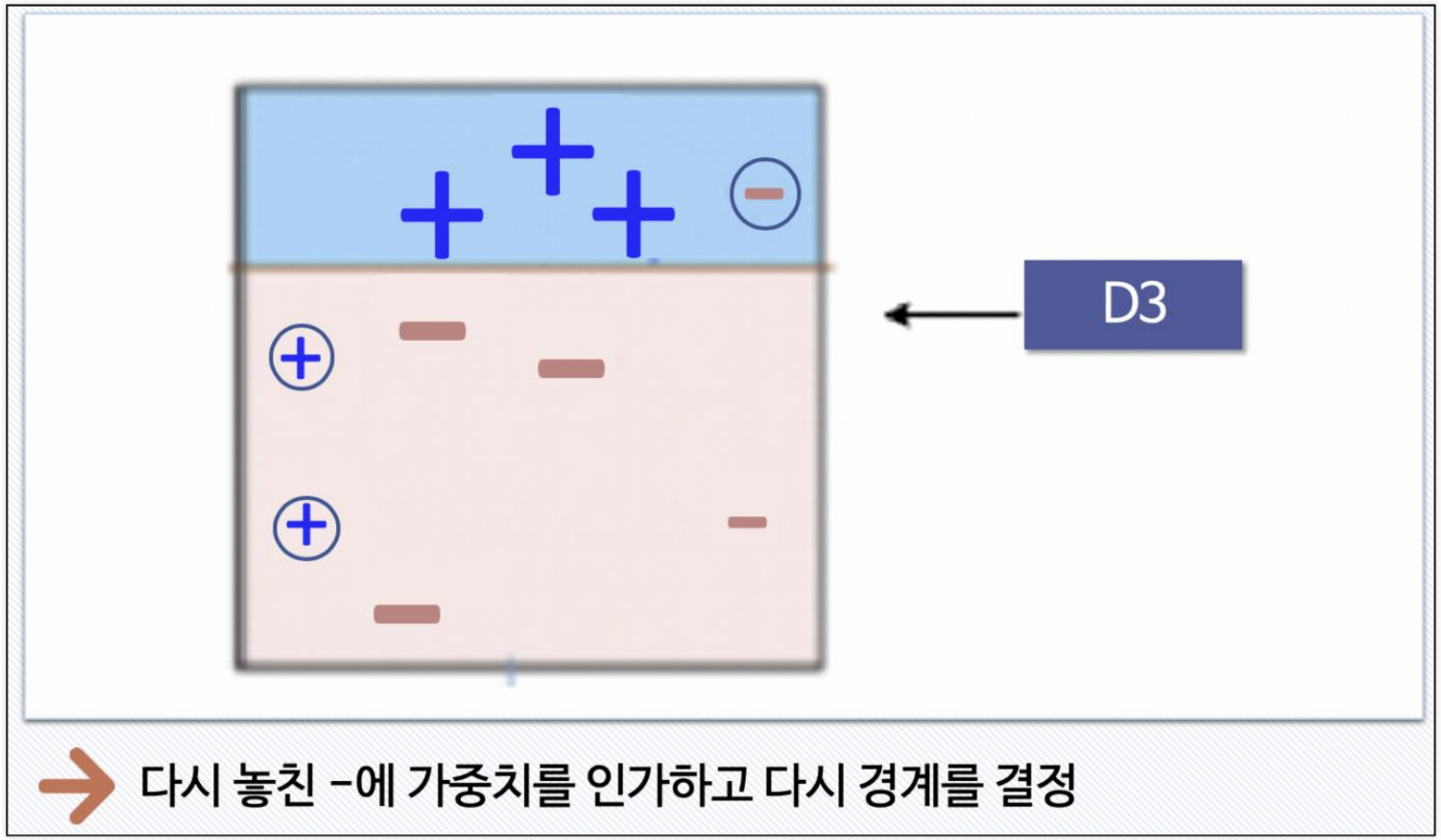
Adaboost - STEP1)



Adaboost - STEP2)

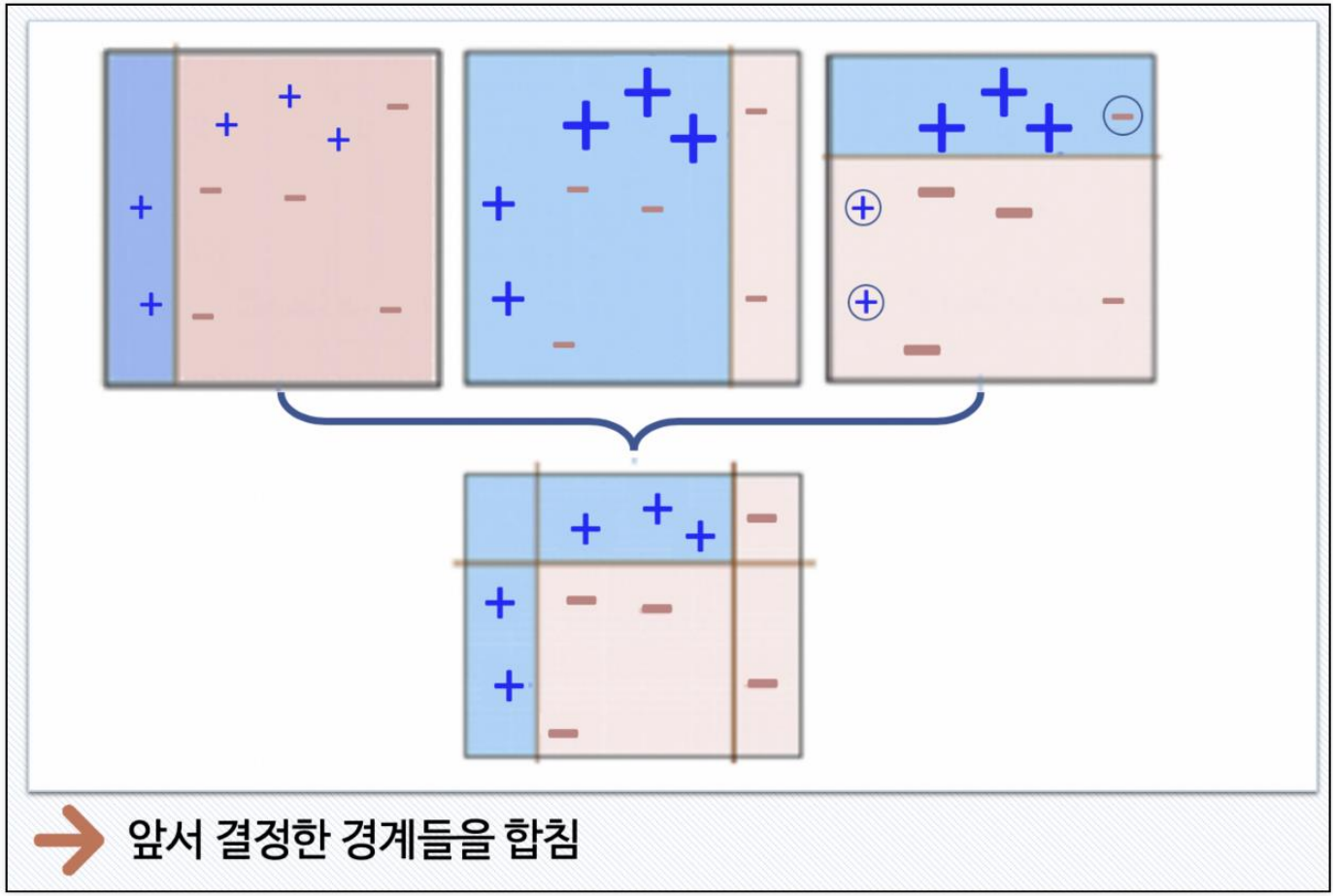


Adaboost - STEP3)



zero-base /

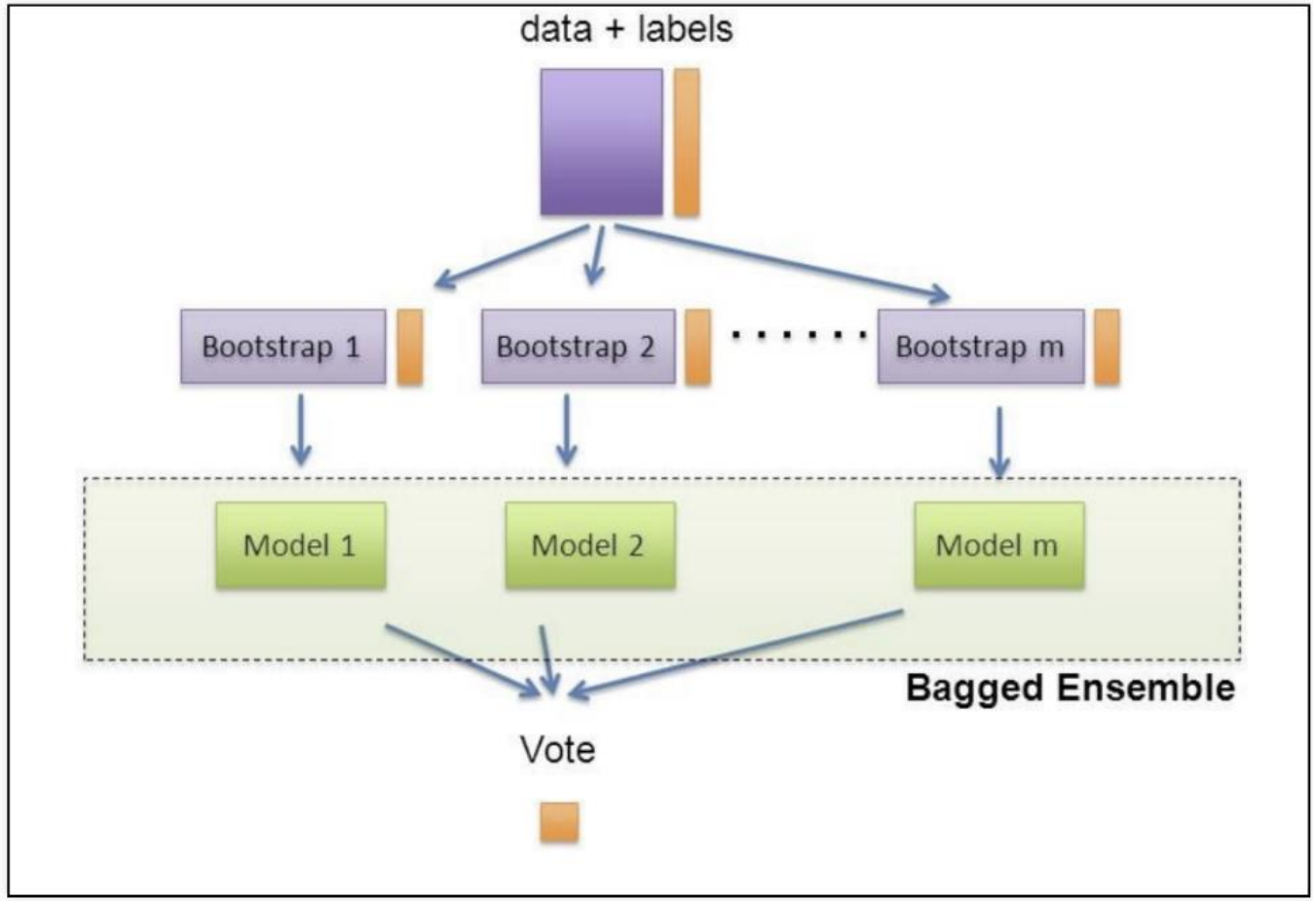
Adaboost - STEP4)



부스팅 기법

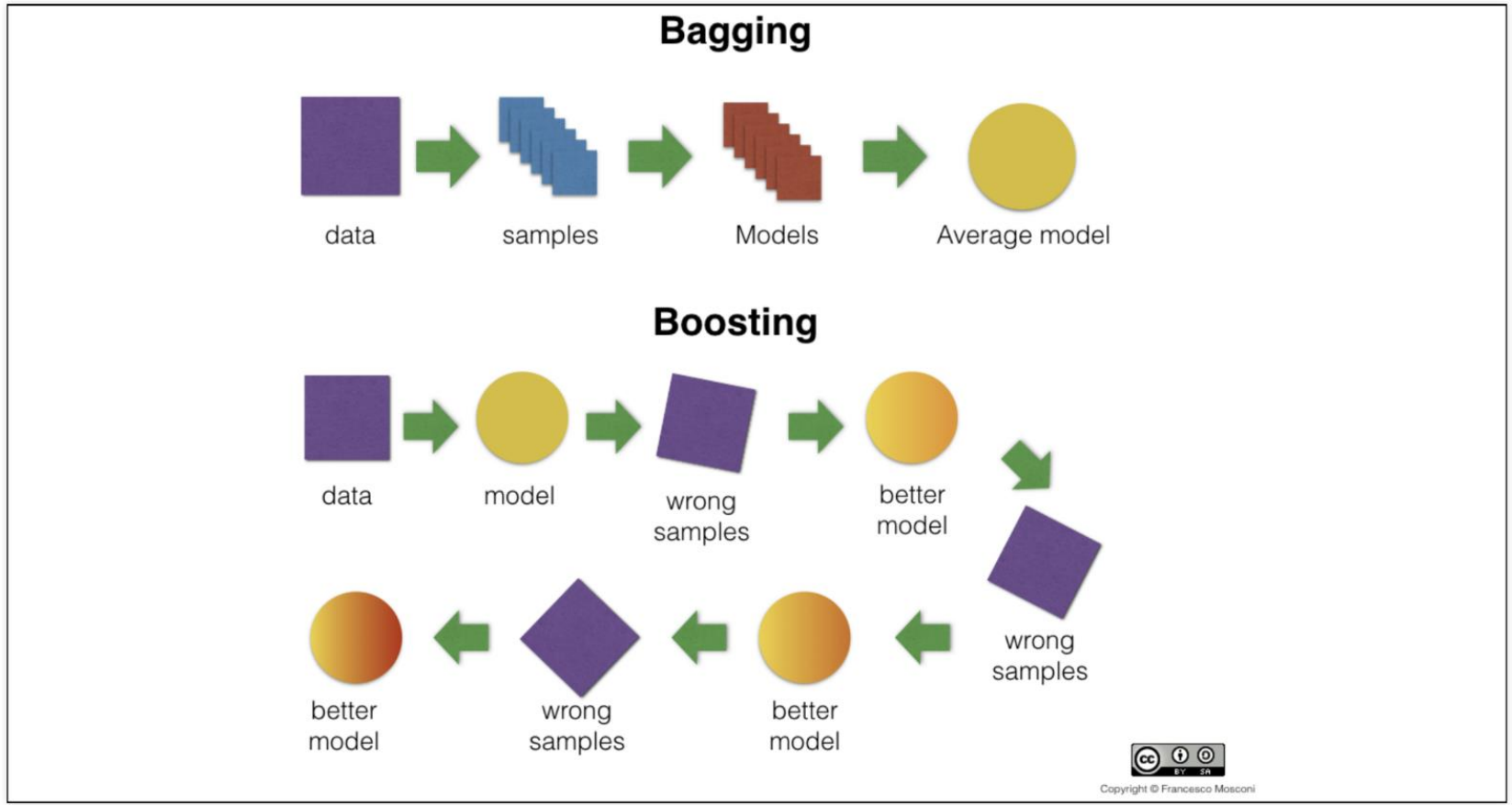
GBM Gradient Boosting Machine	AdaBoost 기법과 비슷하지만, 가중치를 업데이트할 때 경사하강법(Gradient Descent)을 사용
XGBoost (eXtra Gradient Boost)	GBM에서 PC의 파워를 효율적으로 사용하기 위한 다양한 기법에 채택되어 빠른 속도와 효율을 가짐
LightGBM	XGBoost보다 빠른 속도를 가짐

Bagging = Bootstrap AGGregatING



zero-base /

Bagging과 Boosting의 차이



다시 wine

zero-base /

와인~~~ 또 봐요

zero-base /

데이터 읽고 맛에 대한 컬럼을 만들고~~

```
import pandas as pd

wine_url = 'https://raw.githubusercontent.com/PinkWink/ML_tutorial/'+\
                                                    'master/dataset/wine.csv'
wine = pd.read_csv(wine_url, index_col=0)
wine.head()
```

```
wine['taste'] = [1. if grade>5 else 0. for grade in wine['quality']]

X = wine.drop(['taste', 'quality'], axis=1)
y = wine['taste']
```

zero-base /

이번에는 pipeline이 아니라 직접 StandardScaler를 적용

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_sc = sc.fit_transform(X)
```

zero-base /

Scaler 적용 후에 데이터 나누기

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_sc, y, test_size=0.2,
                                                    random_state = 13)
```

- 아직 혼돈이 오면 안되는데, 이 상태에서 cross-validation을 한다면 X_train만 대상이 된다

zero-base /

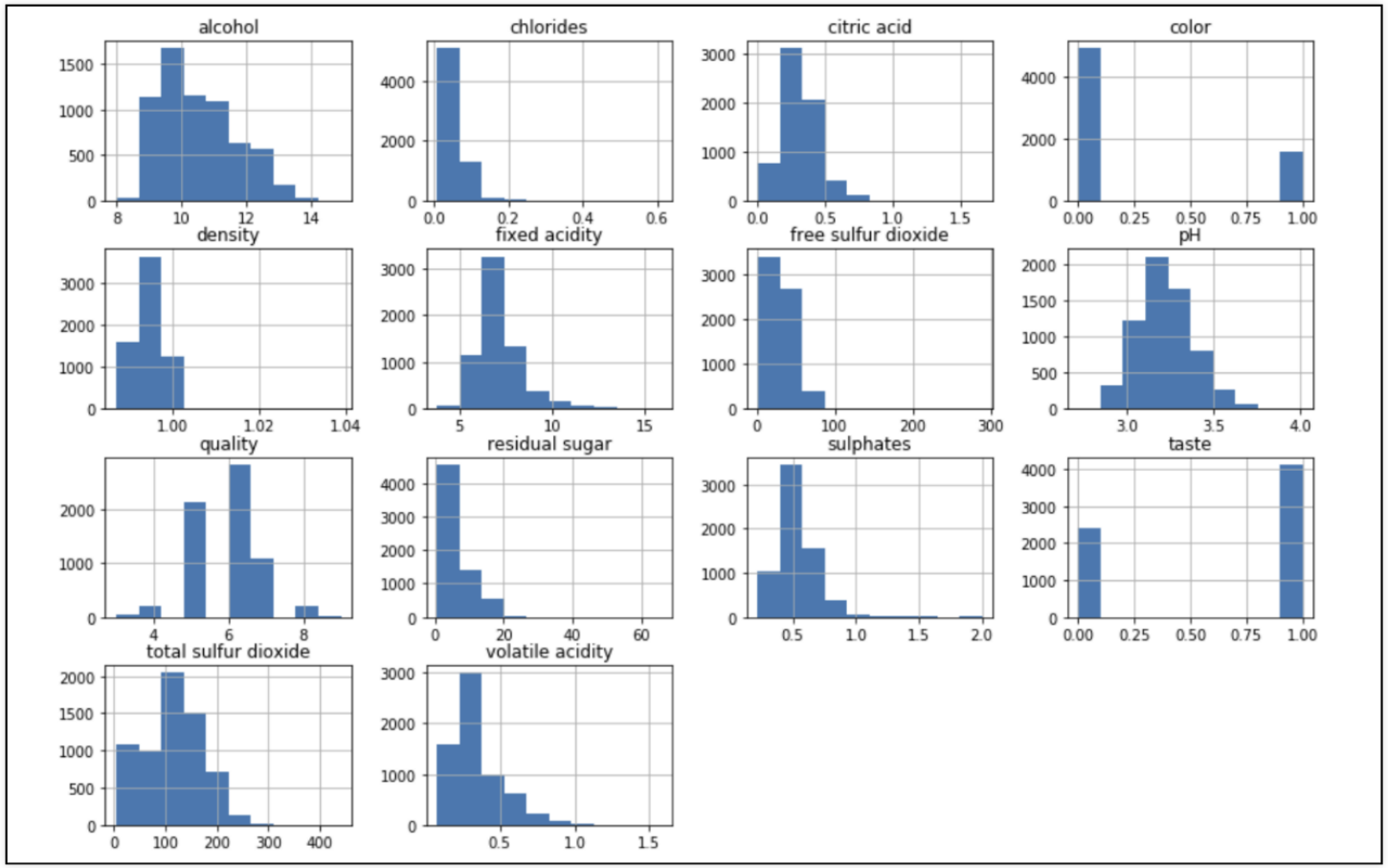
모든 컬럼의 히스토그램 조사

```
import matplotlib.pyplot as plt
%matplotlib inline

wine.hist(bins=10,figsize=(15, 10))
plt.show()
```

zero-base /

잘 분포되어 있는 컬럼이 좋을 때가 많다



zero-base /

혹시나 하고, quality 별 다른 특성이 어떤지 확인해보자

```
colum_names = ['fixed acidity', 'volatile acidity', 'citric acid',
               'residual sugar', 'chlorides', 'free sulfur dioxide',
               'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol']
df_pivot_table = wine.pivot_table(colum_names, ['quality'], aggfunc='median')
print(df_pivot_table)
```

zero-base /

사실 눈으로 그게 보일리가~~~^^

quality	alcohol	chlorides	citric acid	density	fixed acidity
3	10.15	0.0550	0.33	0.995900	7.45
4	10.00	0.0505	0.26	0.994995	7.00
5	9.60	0.0530	0.30	0.996100	7.10
6	10.50	0.0460	0.31	0.994700	6.90
7	11.40	0.0390	0.32	0.992400	6.90
8	12.00	0.0370	0.32	0.991890	6.80
9	12.50	0.0310	0.36	0.990300	7.10

quality	free sulfur dioxide	pH	residual sugar	sulphates
3	17.0	3.245	3.15	0.505
4	15.0	3.220	2.20	0.485
5	27.0	3.190	3.00	0.500
6	29.0	3.210	3.10	0.510
7	30.0	3.220	2.80	0.520
8	34.0	3.230	4.10	0.480
9	28.0	3.280	2.20	0.460

quality	total sulfur dioxide	volatile acidity
3	102.5	0.415
4	102.0	0.380
5	127.0	0.330
6	117.0	0.270
7	114.0	0.270
8	118.0	0.280
9	119.0	0.270

zero-base /

quality에 대한 나머지 특정들의 상관관계는?

```
corr_matrix = wine.corr()
print(corr_matrix["quality"].sort_values(ascending=False))
```

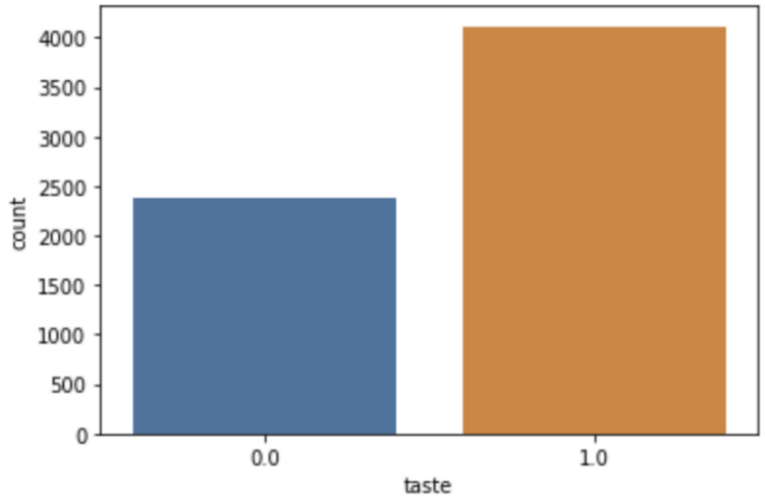
```
quality                1.000000
taste                  0.814484
alcohol                0.444319
citric acid            0.085532
free sulfur dioxide    0.055463
sulphates              0.038485
pH                     0.019506
residual sugar        -0.036980
total sulfur dioxide  -0.041385
fixed acidity         -0.076743
color                 -0.119323
chlorides             -0.200666
volatile acidity      -0.265699
density               -0.305858
Name: quality, dtype: float64
```

zero-base /

taste 컬럼의 분포는

```
import seaborn as sns

sns.countplot(wine['taste'])
plt.show()
```



zero-base /

또 다른 시도, 다양한 모델을 한 번에 테스트해보자

```
from sklearn.ensemble import (AdaBoostClassifier, GradientBoostingClassifier,
                              RandomForestClassifier)
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression

models = []
models.append(('RandomForestClassifier', RandomForestClassifier()))
models.append(('DecisionTreeClassifier', DecisionTreeClassifier()))
models.append(('AdaBoostClassifier', AdaBoostClassifier()))
models.append(('GradientBoostingClassifier', GradientBoostingClassifier()))
models.append(('LogisticRegression', LogisticRegression()))
```

zero-base /

결과를 저장하기 위한 작업

```
from sklearn.model_selection import KFold, cross_val_score, KFold

results = []
names = []

for name, model in models:
    kfold = KFold(n_splits=5, random_state=13, shuffle=True)
    cv_results = cross_val_score(model, X_train, y_train,
                                 cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)

    print(name, cv_results.mean(), cv_results.std())
```

```
RandomForestClassifier 0.8200825497889982 0.017566331518921702
DecisionTreeClassifier 0.7567800029614274 0.010054702536658161
AdaBoostClassifier 0.7533103205745169 0.02644765901536818
GradientBoostingClassifier 0.7663959428444511 0.021596556352125432
LogisticRegression 0.74273191678389 0.015548839626296565
```

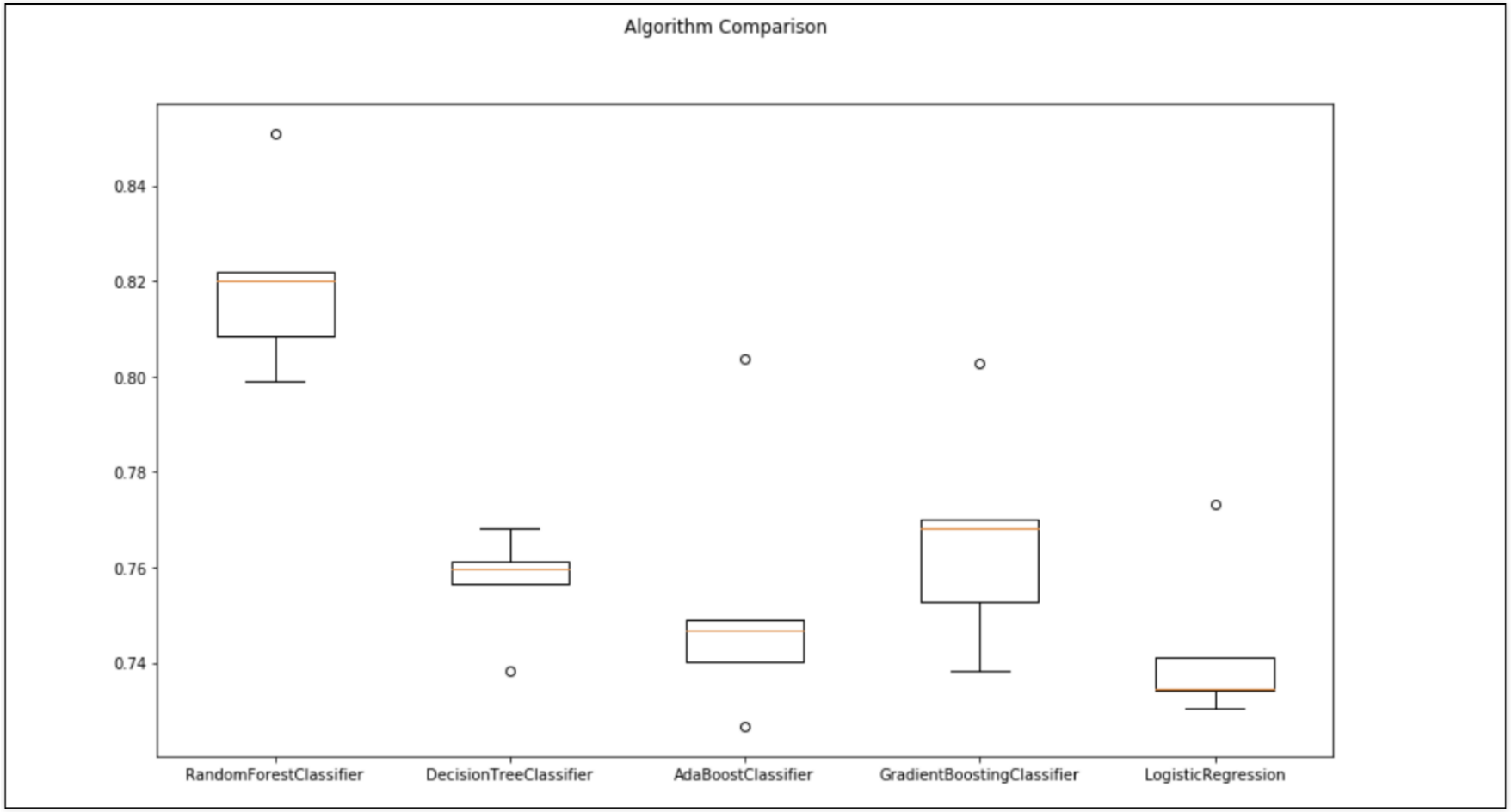
zero-base /

cross-validation 결과를 일목요연하게 확인하기

```
fig = plt.figure(figsize=(14, 8))
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

다시 wine

일단 지금은 RandomForest가 유리해 보인다. - 방법할까 -



zero-base /

테스트 데이터에 대한 평가 결과

```
from sklearn.metrics import accuracy_score

for name, model in models:
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    print(name, accuracy_score(y_test, pred))
```

```
RandomForestClassifier 0.8369230769230769
DecisionTreeClassifier 0.79
AdaBoostClassifier 0.7553846153846154
GradientBoostingClassifier 0.7884615384615384
LogisticRegression 0.7469230769230769
```