Chapter 10: Demo BoostClassifier

In [1]:
```
# from google.colab import drive
# drive.mount("/content/gdrive", force_remount=True)
```

In [2]:
```
# %cd '/content/gdrive/My Drive/LDS6_MachineLearning/practice/Chapter10_Boosting/
```

In [3]:
```
from sklearn.ensemble import AdaBoostClassifier
import pandas as pd
```

In [4]:
```
# Load data
iris = pd.read_excel("Iris.xls")
iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepallength    150 non-null float64
sepalwidth     150 non-null float64
petallength    150 non-null float64
petalwidth     150 non-null float64
iris           150 non-null object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [5]:
```
X = iris[["sepallength","sepalwidth","petallength","petalwidth"]]
X.head()
```

Out[5]:

	sepallength	sepalwidth	petallength	petalwidth
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

In [6]:
```
# 5 first samples
y = iris["iris"]
y.head()
```

Out[6]:
```
0    Iris-setosa
1    Iris-setosa
2    Iris-setosa
3    Iris-setosa
4    Iris-setosa
Name: iris, dtype: object
```

AdaBoostClassifier

In [7]:
```
# Create adaboost-decision tree classifer object
# n_estimators: It controls the number of weak learners.
# learning_rate:Controls the contribution of weak learners in the final combinat
# There is a trade-off between learning_rate and n_estimators.
# base_estimators: It helps to specify different ML algorithm.
from sklearn.tree import DecisionTreeClassifier
ml = DecisionTreeClassifier() # neu mac dinh thi ko can ghi
clf = AdaBoostClassifier(n_estimators=50,
                         base_estimator=ml,
                         learning_rate=1)
```

In [8]:
```
# Train model
model = clf.fit(X, y)
```

In [9]:
```
model
```

Out[9]:
```
AdaBoostClassifier(algorithm='SAMME.R',
                   base_estimator=DecisionTreeClassifier(class_weight=None,
                                                         criterion='gini',
                                                         max_depth=None,
                                                         max_features=None,
                                                         max_leaf_nodes=None,
                                                         min_impurity_decrease=
0.0,
                                                         min_impurity_split=Non
e,
                                                         min_samples_leaf=1,
                                                         min_samples_split=2,
                                                         min_weight_fraction_le
af=0.0,
                                                         presort=False,
                                                         random_state=None,
                                                         splitter='best'),
                   learning_rate=1, n_estimators=50, random_state=None)
```

In [10]:
```
X_test = [[6.4, 3.2, 4.5, 1.5], [5.9, 3. , 5.1, 1.8]]
y_pred = clf.predict(X_test)
y_pred
```

Out[10]:
```
array(['Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [11]:
```
# Evaluate a score by cross-validation
from sklearn.model_selection import cross_val_score
scores = cross_val_score(clf, X, y)
scores
```

```
c:\program files\python36\lib\site-packages\sklearn\model_selection\_split.py:1
978: FutureWarning: The default value of cv will change from 3 to 5 in version
0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
```

Out[11]:
```
array([0.98039216, 0.92156863, 1.        ])
```

In [12]:
```
clf_1 = ml.fit(X,y)
```

In [13]:
```
y_pred = clf_1.predict(X_test)
y_pred
```

Out[13]:
```
array(['Iris-versicolor', 'Iris-virginica'], dtype=object)
```

In [14]:
```
scores1 = cross_val_score(clf_1, X, y, cv=5)
scores1
```

Out[14]:
```
array([0.96666667, 0.96666667, 0.9       , 0.96666667, 1.        ])
```

XGBoostClassifier

In [15]:
```
import xgboost as xgb
```

In [16]:
```
# https://xgboost.readthedocs.io/en/latest/parameter.html
```

In [17]:
```
xgb_model = xgb.XGBClassifier(random_state=42)
xgb_model.fit(X, y)
```

Out[17]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='multi:softprob', random_state=42,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

In [18]:
```
scores2 = cross_val_score(xgb_model, X, y, cv=5)
scores2
```

Out[18]:
```
array([0.96666667, 0.96666667, 0.93333333, 0.9       , 1.        ])
```

In [19]:
```
# predict new sample
["sepallength","sepalwidth","petallength","petalwidth"]

X_new = pd.DataFrame({'sepallength': [6.4,5.9],
                      'sepalwidth':[3.2,3],
                      'petallength':[4.5, 5.1],
                      'petalwidth':[1.5,1.8]
})

y_pred = xgb_model.predict(X_new)
y_pred
```

Out[19]:
```
array(['Iris-versicolor', 'Iris-virginica'], dtype=object)
```