Computer Science, CNU

Lab 5: Decision Tree, Ensemble, SVM and Model Evaluation

Dongil Kim

Dept. of Computer Science and Engineering

Chungnam National University

문제

❖ Iris Data

- 가장 쉽게 사용하는 toy example
- 3가지 종류의 Iris 꽃의 특성에 대한 데이터

Data Preparation

In [44]:
```
## Import basic libraries to handle data
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [46]:
```
## Import data
iris_data = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv')
iris_data.head(5)
```

Out[46]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [47]:
```
## Explore data
np_iris_data = np.array(iris_data)
print(np_iris_data[0:5,:])
```

```
[[5.1 3.5 1.4 0.2 'setosa']
 [4.9 3.0 1.4 0.2 'setosa']
 [4.7 3.2 1.3 0.2 'setosa']
 [4.6 3.1 1.5 0.2 'setosa']
 [5.0 3.6 1.4 0.2 'setosa']]
```

In [51]:
```
datax = np_iris_data[:,0:4]
datay = np_iris_data[:,-1]
print(datax[0:5,:])
print(datay[0:10])
```

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.0 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.0 3.6 1.4 0.2]]
['setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa' 'setosa'
 'setosa' 'setosa']
```

Data 준비
Iris dataset을
Down받아서 썼습니다.

저는 numpy로 하는게 편해서
Pandas는 특별한 상황이 아니면
그냥 numpy로 일괄변환해서
사용합니다.

역시 습관인데
X와 y를 따로 분리해놓고
사용합니다.
편하신대로 하면 됩니다.

Data Preprocessing

```
In [62]: from sklearn.model_selection import train_test_split
         trnx, tstx, trny, tsty = train_test_split(datax, datay, test_size=0.2)
         print(trnx.shape, tstx.shape, trny.shape, tsty.shape)

(120, 4) (30, 4) (120,) (30,)
```

다들 아시는
Data partition 관련된 부분

```
In [63]: from sklearn.preprocessing import MinMaxScaler
         scaler = MinMaxScaler()
         scaler.fit(trnx)
         trnx_scale = scaler.transform(trnx)
         tstx_scale = scaler.transform(tstx)
         print(np.min(trnx_scale[:,0]), np.max(trnx_scale[:,0]))
         print(np.min(tstx_scale[:,0]), np.max(tstx_scale[:,0]))

0.0 1.0
0.08333333333333326 0.9444444444444442
```

Min-Max Scaler를 통해서
데이터 정규화를 시켜주는 부분입니다.

다른 sklearn과 비슷하게
빈 object 를 만든 후에
그 object 에 데이터를 넣어서 fit하는 형태입니다.

Training data로 fit을 하고
그 결과로 test data에 적용하는게 보다 현실적입니다.

Decision Tree

Decision Tree

In [64]:
```
from sklearn import tree
tree_model = tree.DecisionTreeClassifier(max_depth=4, min_samples_split=3)
tree_model.fit(X=trnx, y=trny)
```

Out[64]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=4,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=3,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [65]:
```
tree_pred = tree_model.predict(X=tstx)
```

In [66]:
```
from sklearn.tree import export_graphviz
export_graphviz(tree_model, out_file='tree.dot')
```

위 dot 파일을 그대로 http://www.webgraphviz.com/ 에 긁어붙이면 시각화 가능

Decision Tree 학습 부분.
DT의 option들은 개별설정 가능합니다.

Decision Tree

WebGraphviz is Graphviz in the Browser

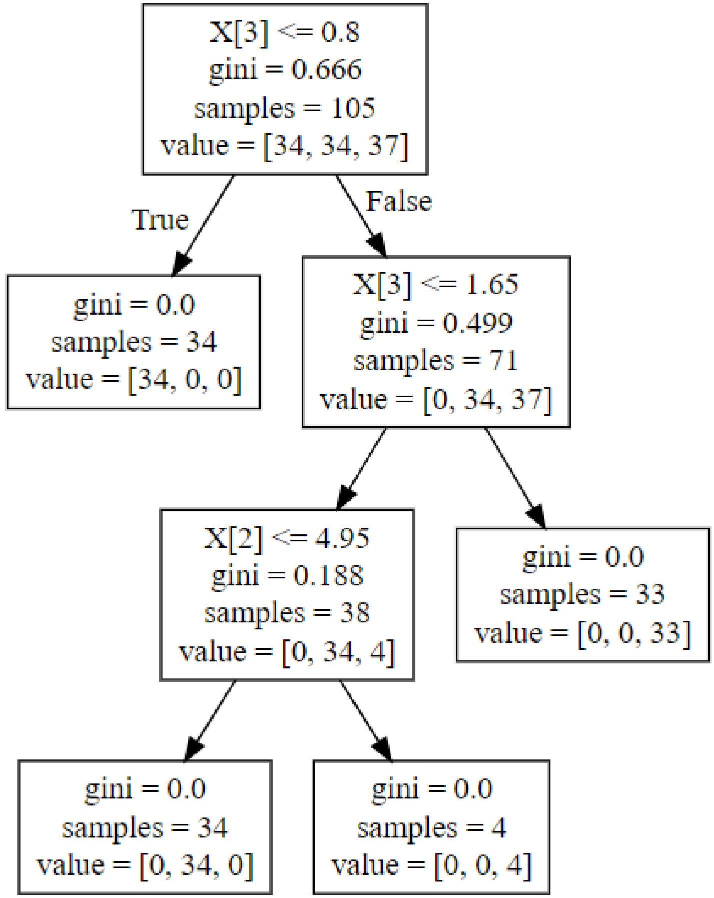
Enter your graphviz data into the Text Area:

(Your Graphviz data is private and never harvested)

Sample 1 | Sample 2 | Sample 3 | Sample 4 | Sample 5

```
digraph Tree {
node [shape=box] ;
0 [label="X[3] <= 0.8₩ngini = 0.666₩nsamples = 105₩nvalue = [34, 34, 37]"] ;
1 [label="gini = 0.0₩nsamples = 34₩nvalue = [34, 0, 0]"] ;
0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"] ;
2 [label="X[3] <= 1.65₩ngini = 0.499₩nsamples = 71₩nvalue = [0, 34, 37]"] ;
0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"] ;
3 [label="X[2] <= 4.95₩ngini = 0.188₩nsamples = 38₩nvalue = [0, 34, 4]"] ;
2 -> 3 ;
4 [label="gini = 0.0₩nsamples = 34₩nvalue = [0, 34, 0]"] ;
3 -> 4 ;
5 [label="gini = 0.0₩nsamples = 4₩nvalue = [0, 0, 4]"] ;
3 -> 5 ;
6 [label="gini = 0.0₩nsamples = 33₩nvalue = [0, 0, 33]"] ;
2 -> 6 ;
}
```

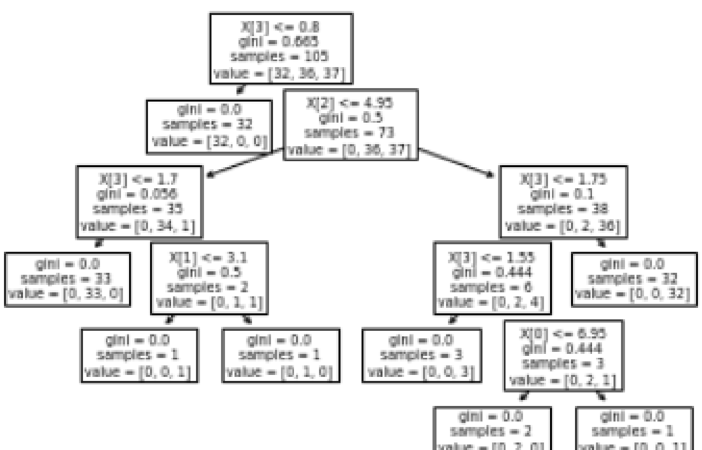
Generate Graph!



Decision Tree

In [20]:
```python
# after sklearn v.0.21
from sklearn.tree import plot_tree
plot_tree(tree_model)
```

Out[20]:
```
[Text(133.92000000000002, 199.32, 'X[3] <= 0.8₩ngini = 0.665₩nsamples = 105₩nvalue = [32, 36, 37]'),
 Text(100.44000000000001, 163.07999999999998, 'gini = 0.0₩nsamples = 32₩nvalue = [32, 0, 0]'),
 Text(167.40000000000003, 163.07999999999998, 'X[2] <= 4.95₩ngini = 0.5₩nsamples = 73₩nvalue = [0, 36, 37]'),
 Text(66.96000000000001, 126.83999999999999, 'X[3] <= 1.7₩ngini = 0.056₩nsamples = 35₩nvalue = [0, 34, 1]'),
 Text(33.480000000000004, 90.6, 'gini = 0.0₩nsamples = 33₩nvalue = [0, 33, 0]'),
 Text(100.44000000000001, 90.6, 'X[1] <= 3.1₩ngini = 0.5₩nsamples = 2₩nvalue = [0, 1, 1]'),
 Text(66.96000000000001, 54.359999999999985, 'gini = 0.0₩nsamples = 1₩nvalue = [0, 0, 1]'),
 Text(133.92000000000002, 54.359999999999985, 'gini = 0.0₩nsamples = 1₩nvalue = [0, 1, 0]'),
 Text(267.84000000000003, 126.83999999999999, 'X[3] <= 1.75₩ngini = 0.1₩nsamples = 38₩nvalue = [0, 2, 36]'),
 Text(234.36, 90.6, 'X[3] <= 1.55₩ngini = 0.444₩nsamples = 6₩nvalue = [0, 2, 4]'),
 Text(200.88000000000002, 54.359999999999985, 'gini = 0.0₩nsamples = 3₩nvalue = [0, 0, 3]'),
 Text(267.84000000000003, 54.359999999999985, 'X[0] <= 6.95₩ngini = 0.444₩nsamples = 3₩nvalue = [0, 2, 1]'),
 Text(234.36, 18.119999999999976, 'gini = 0.0₩nsamples = 2₩nvalue = [0, 2, 0]'),
 Text(301.32000000000005, 18.119999999999976, 'gini = 0.0₩nsamples = 1₩nvalue = [0, 0, 1]'),
 Text(301.32000000000005, 90.6, 'gini = 0.0₩nsamples = 32₩nvalue = [0, 0, 32]')]
```



Sklean 최신 버전(0.21 이상)은
바로 plot 됩니다.

Ensemble

Random Forest

Ensemble

In [39]:
```
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(max_depth=15, n_estimators=100, random_state=0)
rf_model.fit(X=trnx, y=trny)
```

Out[39]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=15, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

In [40]:
```
rf_pred = rf_model.predict(X=tstx)
```

In [41]:
```
from sklearn import metrics
print(metrics.accuracy_score(tsty, rf_pred))
```

```
0.9333333333333333
```

Ensemble

Gradient Boost

In [42]:
```
from sklearn.ensemble import GradientBoostingClassifier
gbm_model = GradientBoostingClassifier(max_depth=3, n_estimators=30, random_state=0)
gbm_model.fit(X=trnx, y=trny)
```

Out[42]:
```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=30,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=0, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

In [43]:
```
gbm_pred = gbm_model.predict(X=tstx)
```

In [44]:
```
from sklearn import metrics
print(metrics.accuracy_score(tsty, gbm_pred))
```

```
0.9555555555555556
```

SVM

SVM

In [75]:
```
from sklearn.svm import SVC
svc_model = SVC(C=100, kernel='rbf', degree=3, gamma=0.1,
                coef0=0.0, shrinking=True, probability=True, tol=0.001, cache_size=200,
                class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', random_state=None)
svc_model.fit(X=trnx, y=trny)
```

Out[75]:
```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.1, kernel='rbf',
    max_iter=-1, probability=True, random_state=None, shrinking=True, tol=0.001,
    verbose=False)
```

In [76]:
```
svc_pred = svc_model.predict(X=tstx)
```

SVM 학습 부분.
SVM의 option들은 개별설정 가능합니다.

SVR의 경우 sklearn.svm.SVR을 사용하시면 됩니다.
1-SVM은
sklearn.svm.OneClassSVM 입니다.

SVM

SVR

```
# Fit regression model
svr_rbf = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
svr_lin = SVR(kernel='linear', C=100, gamma='auto')
svr_poly = SVR(kernel='poly', C=100, gamma='auto', degree=3, epsilon=.1,
               coef0=1)
```

https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html#sphx-glr-auto-examples-svm-plot-svm-regression-py

1-SVM

```
# fit the model
clf = svm.OneClassSVM(nu=0.1, kernel="rbf", gamma=0.1)
clf.fit(X_train)
y_pred_train = clf.predict(X_train)
y_pred_test = clf.predict(X_test)
```

https://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html#sphx-glr-auto-examples-svm-plot-oneclass-py

Performance Evaluation

Model Evaluation

In [77]:
```
from sklearn.metrics import accuracy_score
acc = np.array([accuracy_score(tsty, tree_pred),accuracy_score(tsty, svc_pred)])
print(acc)
```
```
[0.93333333 0.96666667]
```

In [78]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(tsty, tree_pred)
```
Out[78]:
```
array([[ 7,  0,  0],
       [ 0, 10,  1],
       [ 0,  1, 11]], dtype=int64)
```

In [79]:
```
confusion_matrix(tsty, svc_pred)
```
Out[79]:
```
array([[ 7,  0,  0],
       [ 0, 10,  1],
       [ 0,  0, 12]], dtype=int64)
```

간단하게 accuracy 와 confusion matrix만 써봤습니다.

Performance Evaluation

Classification metrics

See the Classification metrics section of the user guide for further details.

metrics.accuracy_score (y_true, y_pred[, …])	Accuracy classification score.
metrics.auc (x, y[, reorder])	Compute Area Under the Curve (AUC) using the trapezoidal rule
metrics.average_precision_score (y_true, y_score)	Compute average precision (AP) from prediction scores
metrics.balanced_accuracy_score (y_true, y_pred)	Compute the balanced accuracy
metrics.brier_score_loss (y_true, y_prob[, …])	Compute the Brier score.
metrics.classification_report (y_true, y_pred)	Build a text report showing the main classification metrics
metrics.cohen_kappa_score (y1, y2[, labels, …])	Cohen's kappa: a statistic that measures inter-annotator agreement.
metrics.confusion_matrix (y_true, y_pred[, …])	Compute confusion matrix to evaluate the accuracy of a classification
metrics.f1_score (y_true, y_pred[, labels, …])	Compute the F1 score, also known as balanced F-score or F-measure
metrics.fbeta_score (y_true, y_pred, beta[, …])	Compute the F-beta score
metrics.hamming_loss (y_true, y_pred[, …])	Compute the average Hamming loss.
metrics.hinge_loss (y_true, pred_decision[, …])	Average hinge loss (non-regularized)
metrics.jaccard_score (y_true, y_pred[, …])	Jaccard similarity coefficient score
metrics.log_loss (y_true, y_pred[, eps, …])	Log loss, aka logistic loss or cross-entropy loss.
metrics.matthews_corrcoef (y_true, y_pred[, …])	Compute the Matthews correlation coefficient (MCC)
metrics.multilabel_confusion_matrix (y_true, …)	Compute a confusion matrix for each class or sample
metrics.precision_recall_curve (y_true, …)	Compute precision-recall pairs for different probability thresholds
metrics.precision_recall_fscore_support (…)	Compute precision, recall, F-measure and support for each class
metrics.precision_score (y_true, y_pred[, …])	Compute the precision
metrics.recall_score (y_true, y_pred[, …])	Compute the recall
metrics.roc_auc_score (y_true, y_score[, …])	Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores.
metrics.roc_curve (y_true, y_score[, …])	Compute Receiver operating characteristic (ROC)
metrics.zero_one_loss (y_true, y_pred[, …])	Zero-one classification loss.

Sklearn API를 참고하시면 다양한 metric 사용 가능합니다.

Q&A