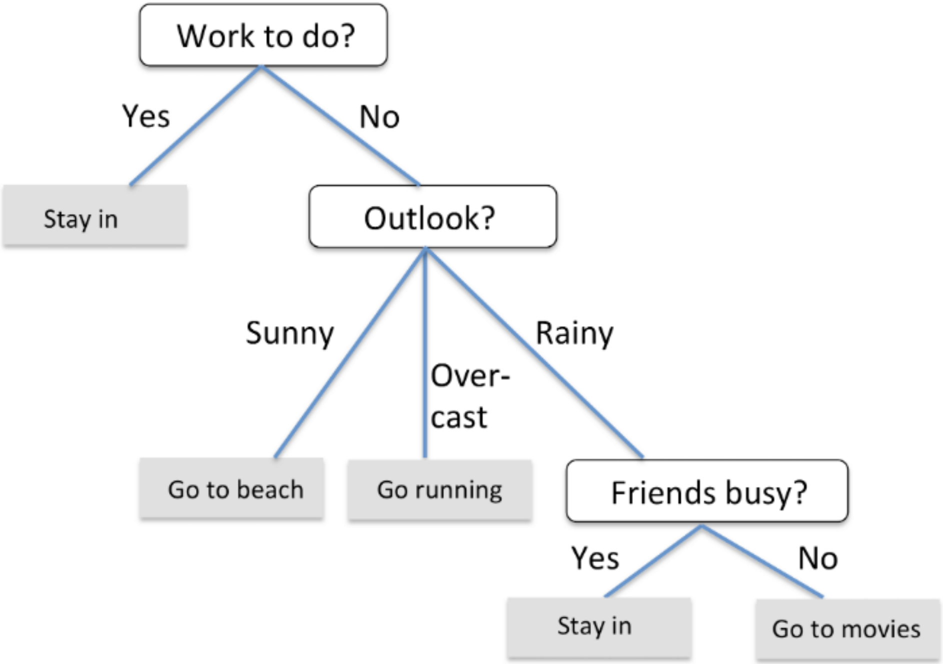
동국대학교
dongguk university

Decision Trees, Bayesian Classifier

Machine Learning

Decision Tree Learning

- What is the Decision Tree Learning?
  - In computer science, Decision tree learning uses a decision tree(as a predictive model) to go from observations about an item(represented in the branches) to conclusions about the item's target value(represented in the leaves)



Decision Tree Learning

- Maximizing information gain
  - Objective function

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$

f : features for classification

D_p: Parent node

D_j : jth child node

I : Impurity

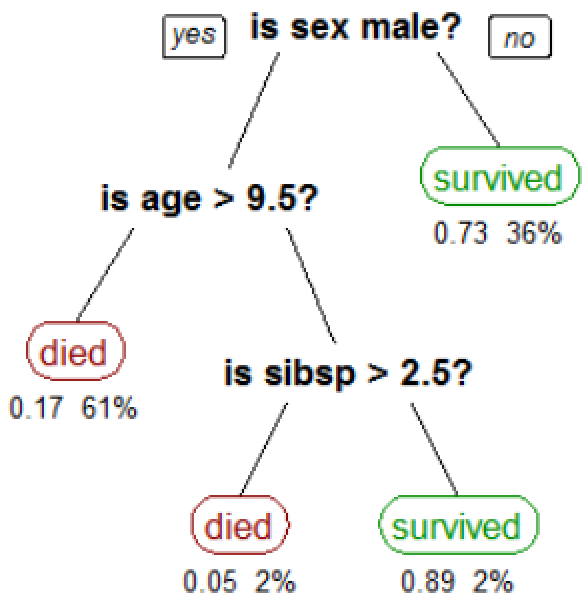
N_p: Number of samples in parent node

N_j : Number of samples in jth child node

Decision Tree Learning

- Maximizing information gain
  - Objective function for binary decision tree

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$



Most libraries (Including Scikit-learn) use Binary Decision Tree to reduce Search Space

Decision Tree Learning

- Maximizing information gain
  - Three Impurity methods
    - Entropy(I_H)

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) log_2 p(i|t)$$

    - Gini impurity(I_G)

$$I_G(t) = \sum_{i=1}^{c} p(i|t)\big(1 - p(i|t)\big) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

Decision Tree Learning

- Maximizing information gain
  - Three Impurity methods
    - Entropy(I_H)

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) log_2 p(i|t)$$

$p(i|t)$: the proportion of the samples that belong to class c for particular node t

Ex) binary classification($c = 2$)

$p(i = 0|t) = 0(or\ 1)$
$\rightarrow I_H(t) = 0\ (\text{min})$

$p(i = 0|t) = \frac{1}{2}$
$\rightarrow I_H(t) = 1\ (\text{max})$

Decision Tree Learning

- Maximizing information gain
  - Three Impurity methods
    - Gini impurity(I_G)

$$I_G(t) = \sum_{i=1}^{c} p(i|t)\big(1 - p(i|t)\big) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

$p(i|t)$: the proportion of the samples that belong to class c for particular node t

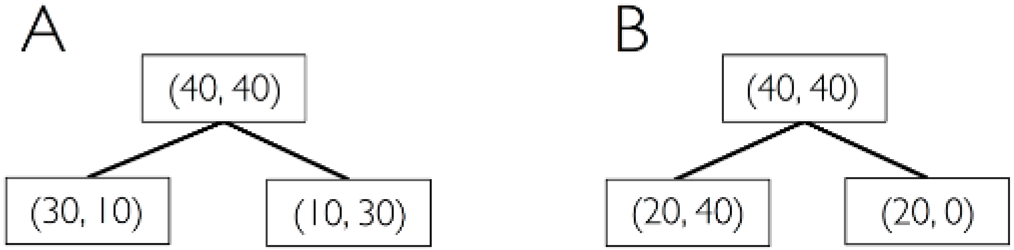
Ex) binary classification($c = 2$)

$p(i = 0|t) = 0$
$\rightarrow I_G(t) = 1 - 1^2 - 0^2 = 0(\mathrm{min})$

$p(i = 0|t) = \frac{1}{2}$
$\rightarrow I_G(t) = 1 - \sum_{i=1}^{c} 0.5^2 = 0.5(\mathrm{max})$

Decision Tree Learning

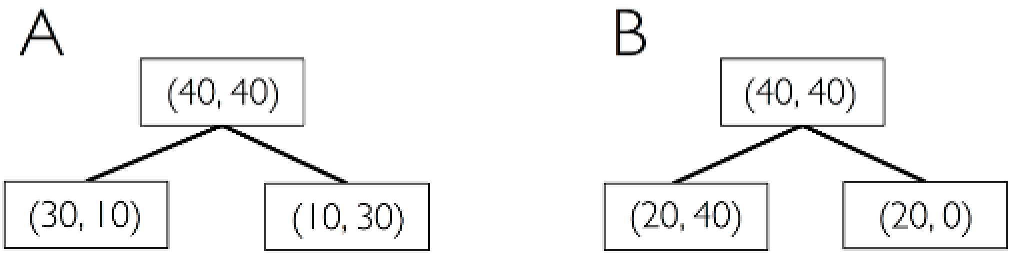
- Maximizing information gain
  - Three Impurity methods



- Entropy(I_H)
- Gini impurity(I_G)

Decision Tree Learning

- Maximizing information gain
  - Three Impurity methods

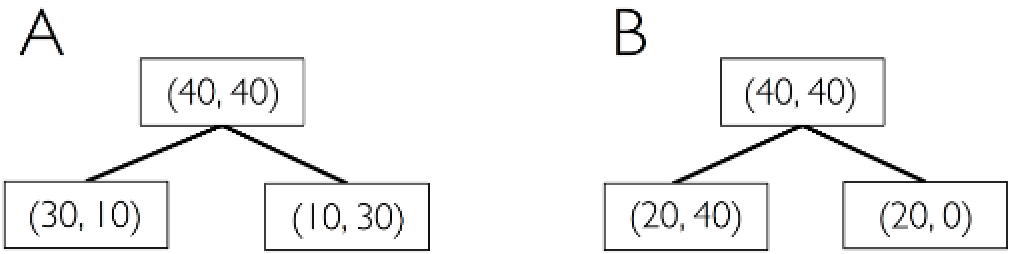


- Entropy(I_H)

$$I_H(D_P) = -(0.5 \cdot log_2(0.5) + 0.5 \cdot log_2(0.5)) = 1$$
$$A: I_H\left(D_{left}\right) = -(0.75 \cdot log_2(0.75) + 0.25 \cdot log_2(0.25)) = 0.81$$
$$A: I_H\left(D_{right}\right) = -(0.25 \cdot log_2(0.25) + 0.75 \cdot log_2(0.75)) = 0.81$$
$$A: IG_H = 1 - \frac{4}{8} \cdot 0.81 - \frac{4}{8} \cdot 0.81 = 0.19$$

Decision Tree Learning

- Maximizing information gain
  - Three Impurity methods



- Entropy(I_H)

$$B: I_H\left(D_{left}\right) = -\left(\frac{2}{6} \cdot log_2\left(\frac{2}{6}\right) + \frac{4}{6} \cdot log_2\left(\frac{4}{6}\right)\right) = 0.92$$

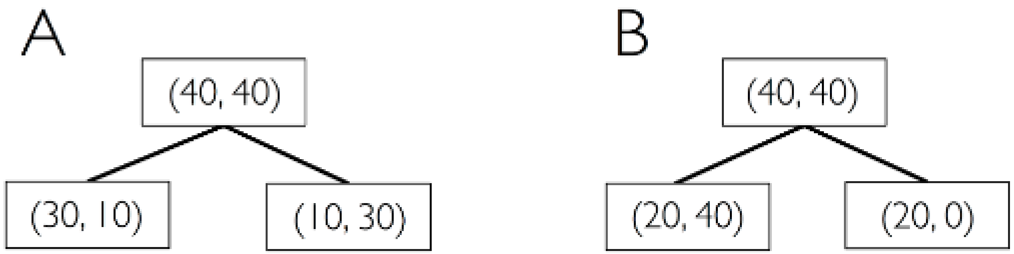
$$B: I_H\left(D_{right}\right) = 0$$

$$B: IG_H = 1 - \frac{6}{8} \cdot 0.92 - \frac{2}{8} \cdot 0 = 0.31$$

Using Entropy(I_H), Scenario B will be selected!($0.19 < 0.31$)

Decision Tree Learning

- Maximizing information gain
  - Three Impurity methods

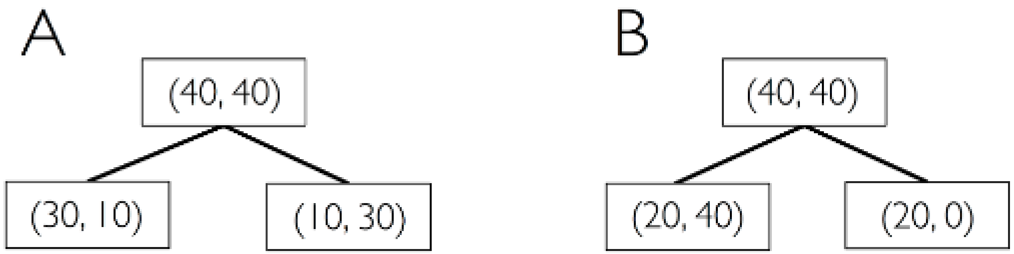


- Gini impurity(I_G)

$$I_G(D_P) = 1 - (0.5^2 + 0.5^2) = 0.5$$
$$A: I_G\left(D_{left}\right) = 1 - (0.75^2 + 0.25^2) = 0.375$$
$$A: I_G\left(D_{right}\right) = 1 - (0.25^2 + 0.75^2) = 0.375$$
$$A: IG_G = 0.5 - \frac{4}{8}0.375 - \frac{4}{8}0.375 = 0.125$$

Decision Tree Learning

- Maximizing information gain
  - Three Impurity methods



  - Gini impurity(I_G)

$$B: I_G\left(D_{left}\right) = 1 - ((2/6)^2 + (4/6)^2) = \frac{4}{9}$$

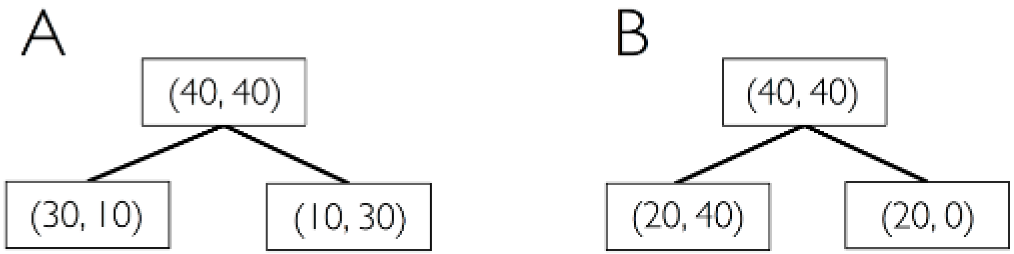
$$B: I_G\left(D_{right}\right) = 1 - (1^2 + 0^2) = 0$$

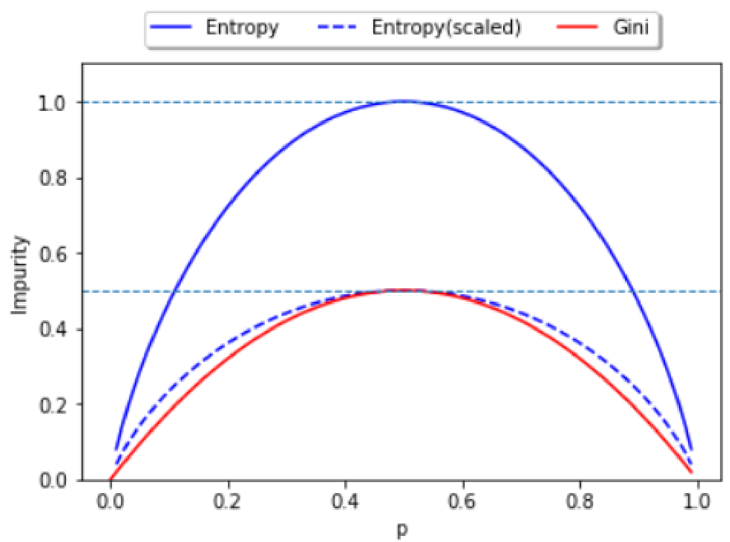
$$B: IG_G = 0.5 - \frac{6}{8} \cdot \frac{4}{9} - \frac{2}{8} \cdot 0 = \frac{1}{6}$$

Using Gini impurity(I_G), Scenario B will be selected!($0.125 < \frac{1}{6}$)

Decision Tree Learning

- Maximizing information gain
  - Three Impurity methods





Decision Tree Learning with Scikit-learn

- Load Iris Dataset

```
from sklearn import datasets
import numpy as np
iris = datasets.load_iris()
# use features 2 and 3 only
X = iris.data[:, [2, 3]]
y = iris.target
print('Class labels:', np.unique(y))
```

```
Class labels: [0 1 2]
```

- Splitting data into 70% training data & 30% test data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=1, stratify=y)

print('Labels counts in y:', np.bincount(y))
print('Labels counts in y_train:', np.bincount(y_train))
print('Labels counts in y_test:', np.bincount(y_test))
```

```
Labels counts in y: [50 50 50]
Labels counts in y_train: [35 35 35]
Labels counts in y_test: [15 15 15]
```

Decision Tree Learning with Scikit-learn

- Entropy and Gini Impurity

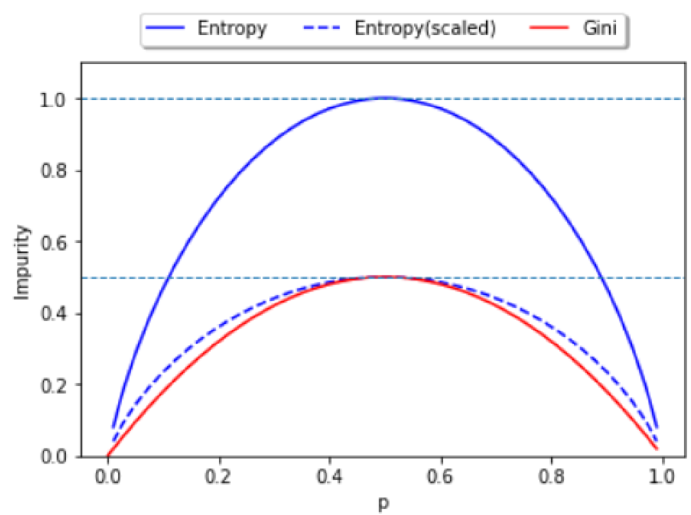
```
def gini(p):
    return p * (1 - p) + (1 - p) * (1 - (1 - p))

def entropy(p):
    return - p * np.log2(p) - (1 - p) * np.log2((1 - p))
```

$$I_G(t) = \sum_{i=1}^{c} p(i|t)\big(1 - p(i|t)\big)$$

$$= 1 - \sum_{i=1}^{c} p(i|t)^2$$

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) log_2 p(i|t)$$



Decision Tree Learning with Scikit-learn

- Learning Decision Tree

```
from sklearn.tree import DecisionTreeClassifier

tree = DecisionTreeClassifier(criterion='gini', random_state=1)
tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                       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, presort=False,
                       random_state=1, splitter='best')
```

Decision Tree Learning with Scikit-learn

- Model evaluation

```
X_test[10:15]
```

```
array([[1.5, 0.4],
       [4.9, 1.8],
       [1.4, 0.2],
       [3.3, 1. ],
       [1.4, 0.2]])
```

```
y_test[10:15]
```

```
array([0, 2, 0, 1, 0])
```

```
# predict class of X_test[10] ~ X_test[14]
tree.predict(X_test[10:15])
```

```
array([0, 2, 0, 1, 0])
```

```
# Compute train accuracy
acc = tree.score(X_train, y_train)
print("Train Accuracy : ", acc)
```

```
Train Accuracy : 0.9904761904761905
```

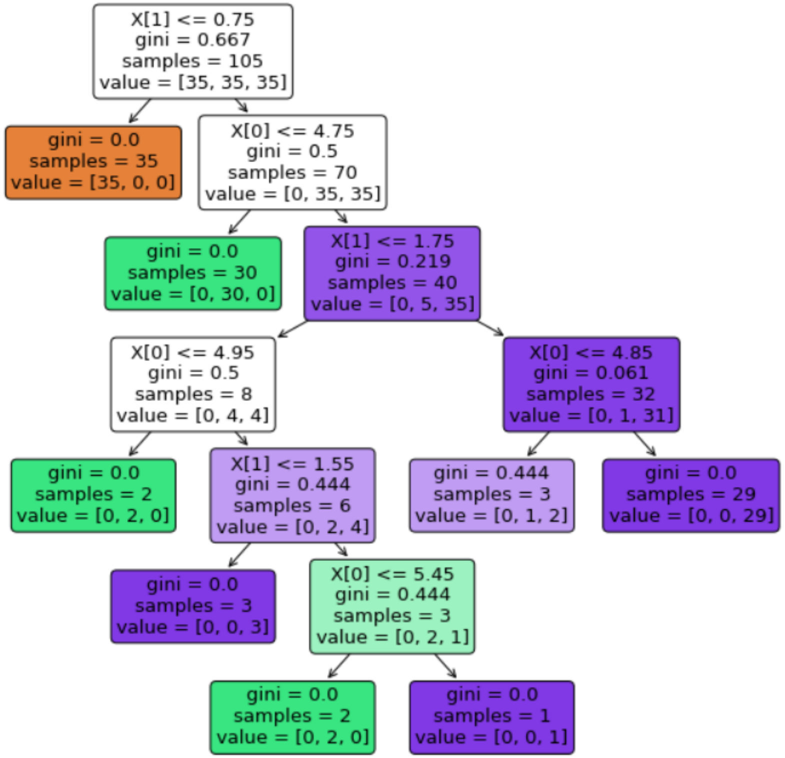
```
# Compute test accuracy
acc = tree.score(X_test, y_test)
print("Test Accuracy : ", acc)
```

```
Test Accuracy : 0.9777777777777777
```

Decision Tree Learning with Scikit-learn

- Visualizing the model

```
from sklearn.tree import plot_tree #scikit-learn >= 22.0
# plot the tree
plot_tree(tree, filled=True, rounded=True)
```

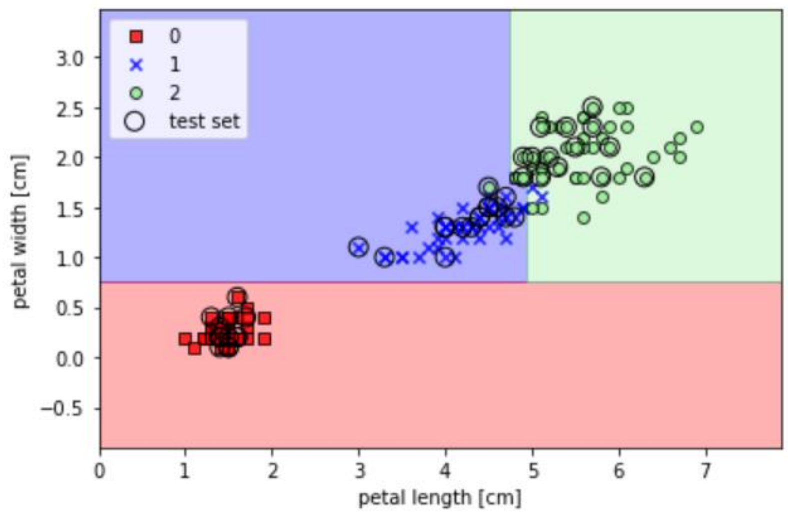


Decision Tree Learning with Scikit-learn

- Plotting the decision boundary

```
X_combined = np.vstack((X_train, X_test))
y_combined = np.hstack((y_train, y_test))

plot_decision_regions(X_combined, y_combined, classifier=tree, test_idx=range(105, 150))
plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



Decision Tree Learning with Scikit-learn

- Building Decision Tree with max depth

```
from sklearn.tree import DecisionTreeClassifier
# learn decision tree of depth 2
tree = DecisionTreeClassifier(criterion='gini', max_depth=2, random_state=1)
tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=2,
                       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, presort=False,
                       random_state=1, splitter='best')
```

Decision Tree Learning with Scikit-learn

- Model evaluation

```
# Compute train accuracy
acc = tree.score(X_train, y_train)
print("Train Accuracy : ", acc)
```

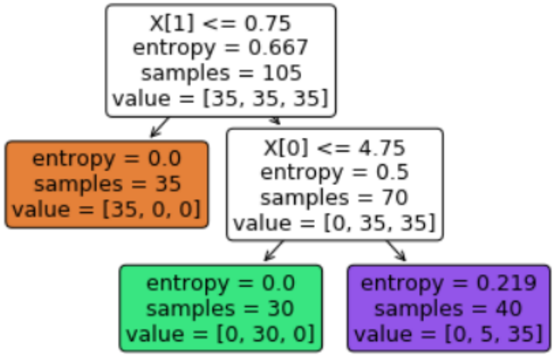
```
Train Accuracy : 0.9523809523809523
```

```
# Compute test accuracy
acc = tree.score(X_test, y_test)
print("Test Accuracy : ", acc)
```

```
Test Accuracy : 0.9555555555555556
```

- Visualizing the model

```
from sklearn.tree import plot_tree #scikit-learn >= 22.0
# plot the tree
plot_tree(tree, filled=True, rounded=True)
```

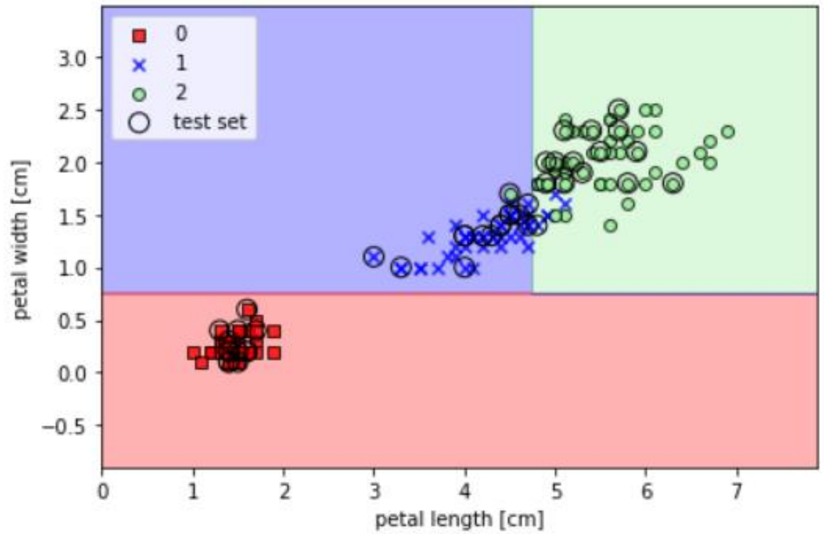


Decision Tree Learning with Scikit-learn

- Plotting the decision boundary

```
X_combined = np.vstack((X_train, X_test))
y_combined = np.hstack((y_train, y_test))

plot_decision_regions(X_combined, y_combined, classifier=tree, test_idx=range(105, 150))
plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



Naïve Bayesian Classifiers

- What is the Naïve Bayesian Classifiers?
  - In machine learning, Naïve Bayesian Classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (Naïve) independence assumptions between the features
  - Probabilistic model
    - Naïve Bayes is a conditional probability model : given a problem instance to be classified, represented by vector $x = (x_1, x_2, \dots, x_n)$ representing some n features(independent), it assigns to this instance probabilities

$$p(C_k | x_1, \dots, x_n)$$

for each of K possible outcomes of classes C_k

    - Bayesian probability terminology

$$posterior = \frac{prior \times likelihood}{evidence} \qquad p(C_k|x) = \frac{p(C_k)p(x|C_k)}{p(x)}$$

Naïve Bayesian Classifiers

- Constructing a classifier from the probability model
  - Bayesian Reasoning

$$p(x_i|x_{i+1},\dots,x_n,C_k) = p(x_i|C_k)$$

$$p(C_k|x_1,\dots,x_n) = \frac{1}{Z}p(C_k)\prod_{i=1}^{n}p(x_i|C_k)$$

$$where\ Z = p(x) = \sum_k p(C_k)p(x|C_k)$$

Thus, the joint model can be expressed as

$$p(C_k|x_1,\dots,x_n) \propto p(C_k,x_1,\dots,x_n)$$

$$= p(C_k)p(x_1|C_k)p(x_2|C_k)p(x_3|C_k)\dots = p(C_k)\prod_{i=1}^{n}p(x_i|C_k)$$

  - Prediction

$$\hat{y} = \underset{k\in\{1,\dots,K\}}{\operatorname{argmax}}\ p(C_k)\prod_{i=1}^{n}p(x_i|C_k)$$

Naïve Bayesian Classifiers Using Scikit-learn

- Load Iris Dataset

```
from sklearn import datasets
import numpy as np
iris = datasets.load_iris()
# use features 2 and 3 only
X = iris.data[:, [2, 3]]
y = iris.target
print('Class labels:', np.unique(y))
```

```
Class labels: [0 1 2]
```

- Splitting data into 70% training data & 30% test data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=1, stratify=y)

print('Labels counts in y:', np.bincount(y))
print('Labels counts in y_train:', np.bincount(y_train))
print('Labels counts in y_test:', np.bincount(y_test))
```

```
Labels counts in y: [50 50 50]
Labels counts in y_train: [35 35 35]
Labels counts in y_test: [15 15 15]
```

Naïve Bayesian Classifiers Using Scikit-learn

- Learning Gaussian Naïve Bayes Classifier

```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

- Model evaluation - (1)

```
X_test[10:15]
```

```
array([[1.5, 0.4],
       [4.9, 1.8],
       [1.4, 0.2],
       [3.3, 1. ],
       [1.4, 0.2]])
```

```
y_test[10:15]
```

```
array([0, 2, 0, 1, 0])
```

```
# predict class of X_test[10] ~ X_test[14]
gnb.predict(X_test[10:15])
```

```
array([0, 2, 0, 1, 0])
```

Naïve Bayesian Classifiers Using Scikit-learn

- Model evaluation –(2)

```
# Compute train accuracy
acc = gnb.score(X_train, y_train)
print("Train Accuracy : ", acc)
```

```
Train Accuracy : 0.9523809523809523
```

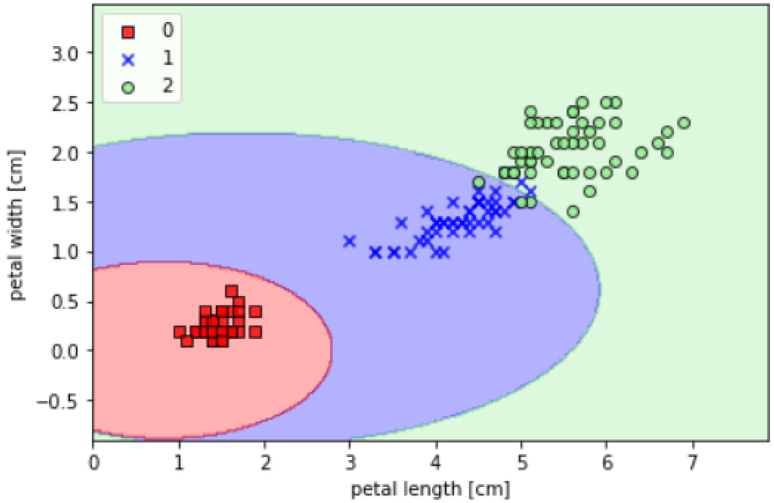
```
# Compute test accuracy
acc = gnb.score(X_test, y_test)
print("Test Accuracy : ", acc)
```

```
Test Accuracy : 0.9777777777777777
```

Naïve Bayesian Classifiers Using Scikit-learn

- Plotting the decision boundary

```
from sklearn.naive_bayes import GaussianNB
X_combined = np.vstack((X_train, X_test))
y_combined = np.hstack((y_train, y_test))
plot_decision_regions(X_combined, y_combined,
classifier=gnb)
plt.xlabel('petal length [cm]')
plt.ylabel('petal width [cm]')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```



Submit

- To make sure if you have completed this practice,
  Submit your practice file(Week06_givencode.ipynb) to e-class.
- **Deadline : tomorrow 11:59pm**
- Modify your ipynb file name as “Week06_StudentNum_Name.ipynb”
  Ex) **Week06_2020123456_홍길동.ipynb**
- You can upload this file without taking the quiz, but homework will be provided like a quiz every three weeks, so it is recommended to take the quiz as well.

Quiz 1 : Decision Tree

- Learn decision tree model for Heart Diseases classification using heart_disease.csv dataset
  - Dataset information : https://archive.ics.uci.edu/ml/datasets/heart+Disease
  - 297 samples, all 13 features
  - Label : The feature 'num' refers to the presence of heart disease in the patient(from 0 to 4). We will convert values 2, 3, 4 to 1(to do binary classification)
  - Find the smallest tree among the trees that show highest test accuracy
  - Visualize the tree, and compute the accuracies
  - Predict the class of following data

    [[52, 2, 5, 135, 250, 0, 3, 180, 1, 0, 1, 0, 2]]

Quiz 2 : Naïve Bayesian Classifier

- Predicting presence of Heart Diseases using Bayesian Classifier
  - Dataset information : https://archive.ics.uci.edu/ml/datasets/heart+Disease
  - Use the same training and test dataset of Quiz 1
  - Compute the accuracies of the classifier
  - Predict the class of following data with probability

    [[52, 2, 5, 135, 250, 0, 3, 180, 1, 0, 1, 0, 2]]