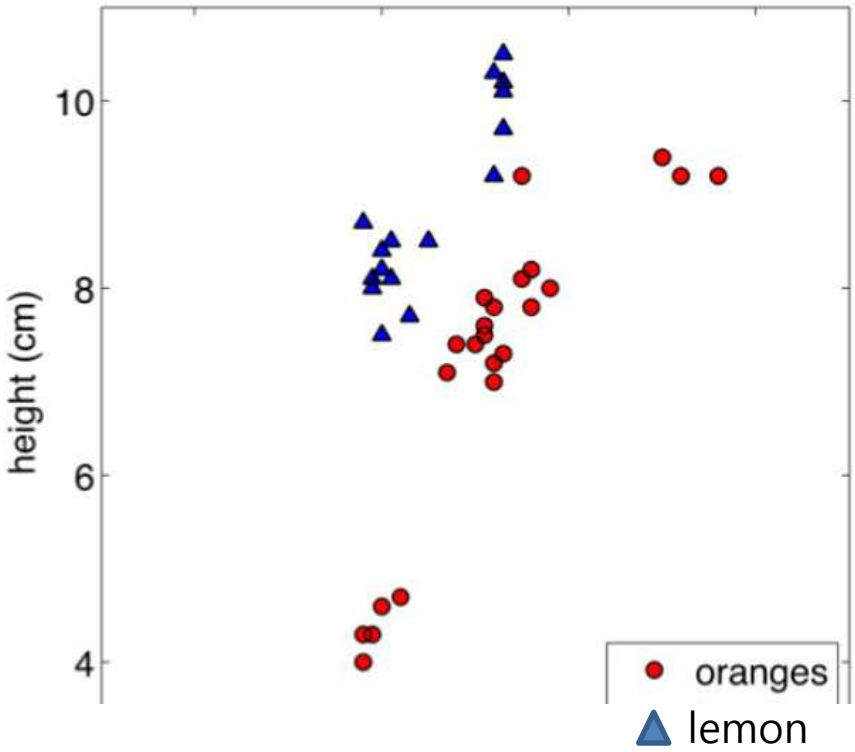
Decision Tree

Classification Problem

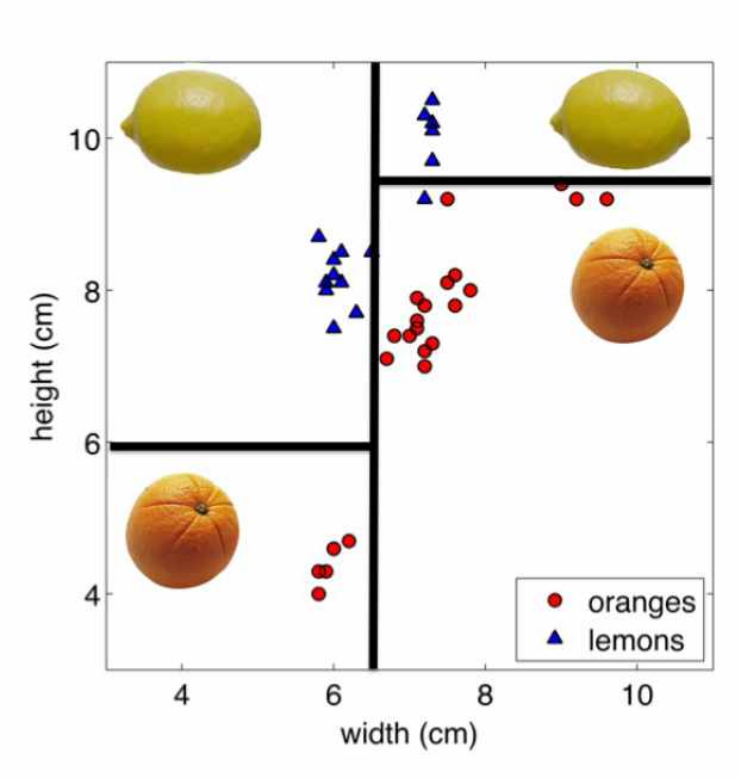
- Feature space



Classification Idea
- kNN
- SVM
- ?????

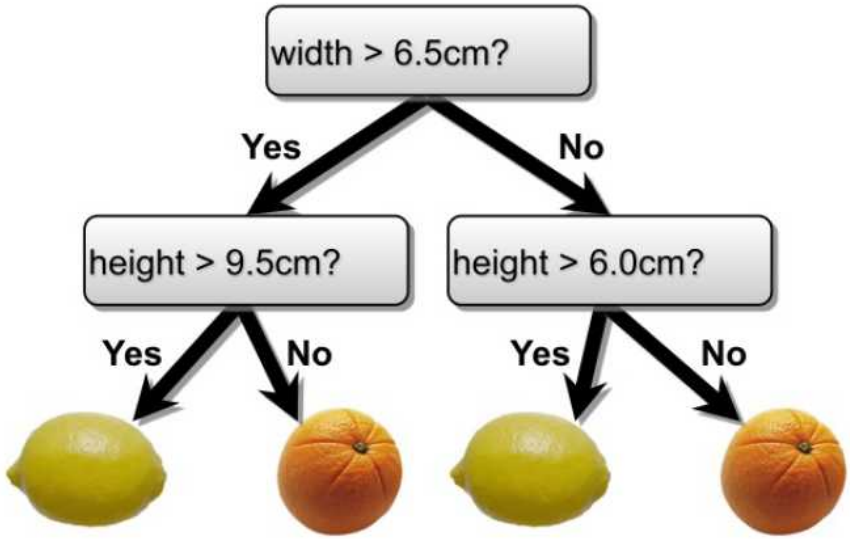
Classification Idea

- Axes-aligned decision boundaries



Classification Idea

- Decision Tree



Internal nodes test attributes

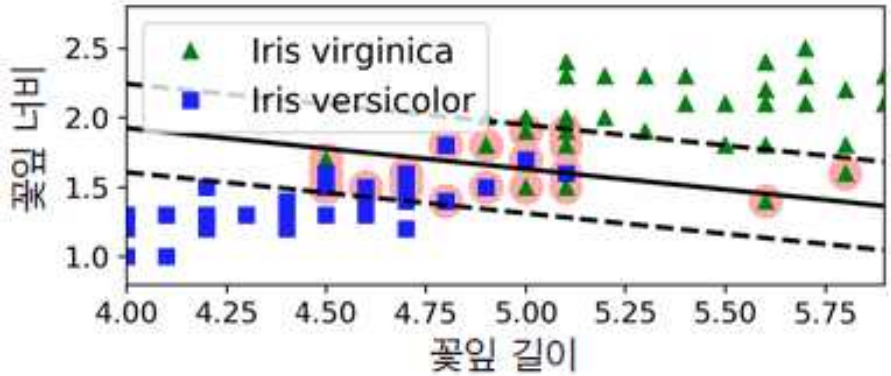
Branching is determined by attribute value

Leaf nodes are outputs (class assignments)

Training

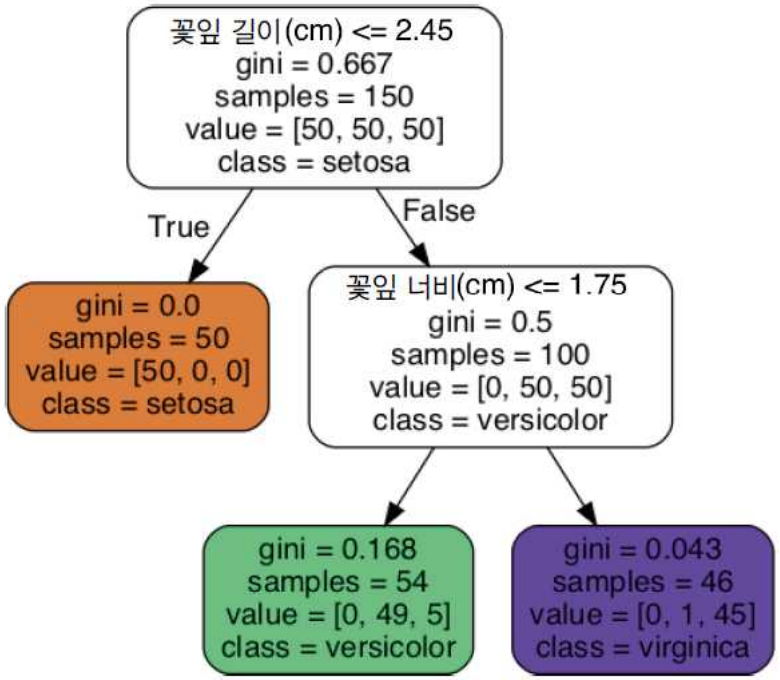
- IRIS (붓꽃) Data

- Feature space
  - 꽃잎 길이 (length)
  - 꽃잎 너비 (width)



Training

- Decision Tree Model
  - 학습샘플을 사용하여 Model 생성 => Training algorithm



학습샘플: {(길이,너비,클래스)}, 3종류 x 50 개 =150 개

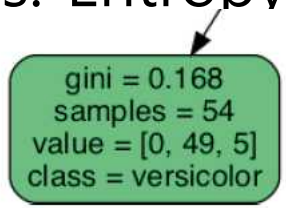
value = [setosa, versicolor, virginica]

max_depth = 2

Training

- Gini Impurity vs. Entropy

Node i

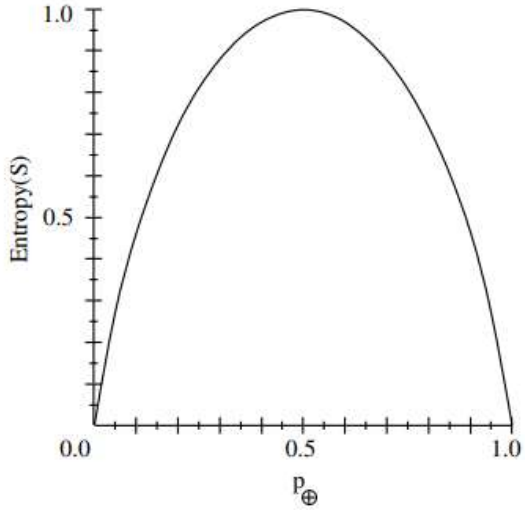


$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^{\ 2} \qquad \text{(Gini)}$$

$$H_i = -\sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log_2(p_{i,k}) \qquad \text{(entropy)}$$

$$G_i = 1 - \left(\frac{0}{54}\right)^2 - \left(\frac{49}{54}\right)^2 - \left(\frac{5}{54}\right)^2 = 0.168$$

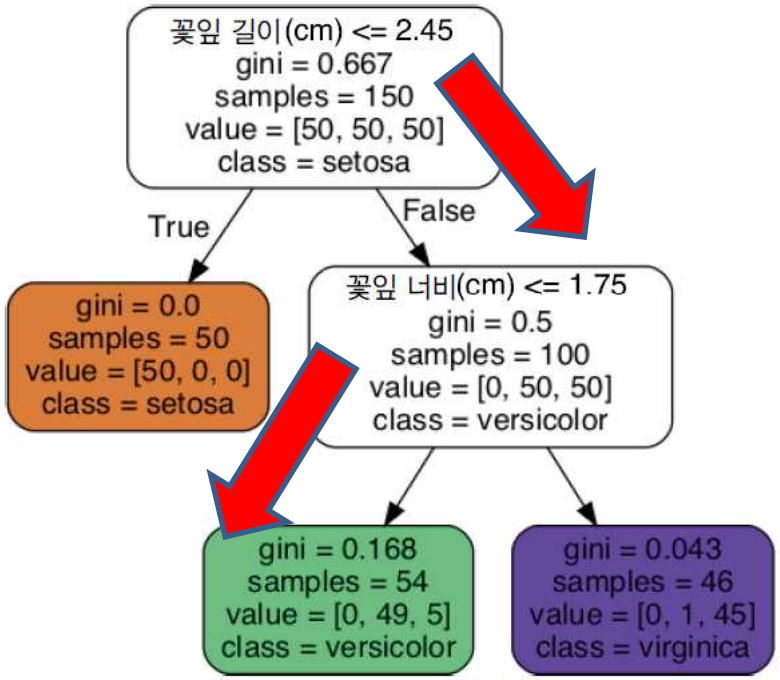
$$H_i = -\frac{49}{54}\log_2\left(\frac{49}{54}\right) - \frac{5}{54}\log_2\left(\frac{5}{54}\right) = 0.445$$

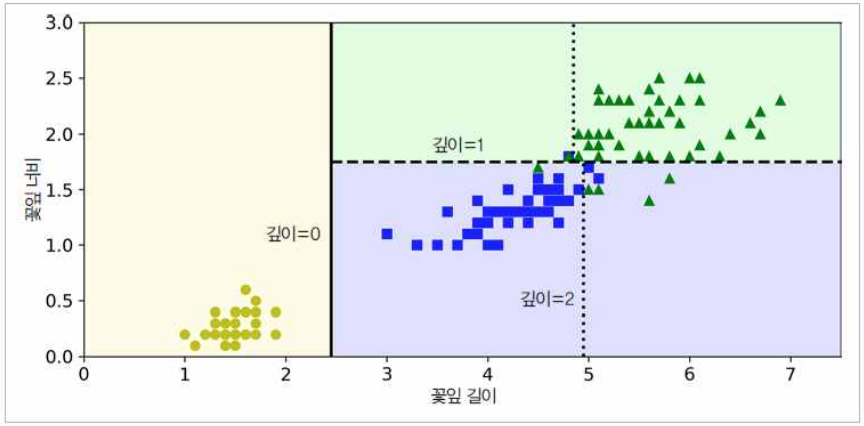


=> Gini impurity 또는 Entropy 를 최소화하는 Tree 모델 생성

Prediction

- IRIS Classification
  - (예) 길이=5, 너비=1.5





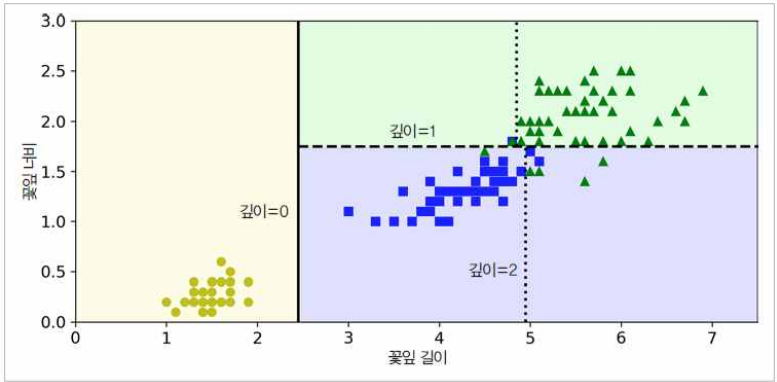
Decision Boundaries

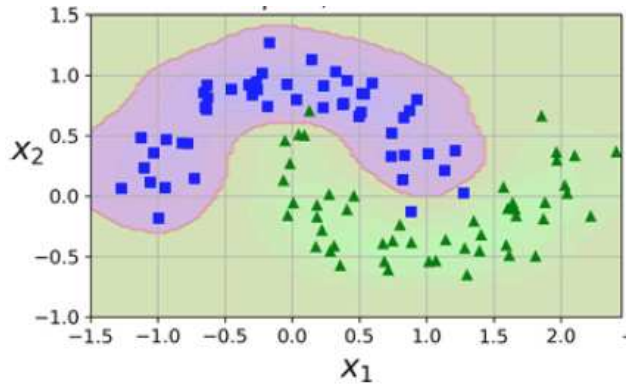
=> 확률값 출력: setosa 0% (0/54), versicolor 90.7% (49/54), virgibica 9.3% (5/54)

계산 복잡도: $\log_2 n$ (n= 노드 수)

Prediction

- Decision Boundaries
  - 결정방식이 직관적이고 이해하기 쉬움
  - 수동으로 작성 가능
  - White-box model (cf. black-box model: mlp, svm, ….)



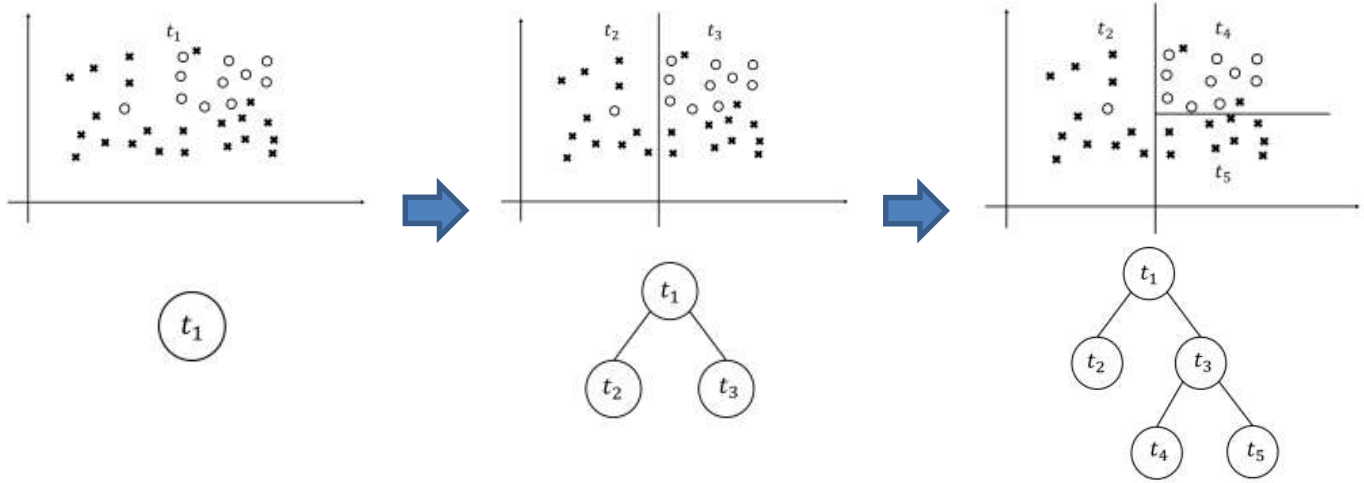


Training Algorithm

- NP complete problem
  - Greedy approach
    - Start from an empty decision tree
    - Split on next best attribute
    - Recurse

* Confusion = entropy or gini impurity

```
Decision-Tree(examples)
if no confusion in the examples or cannot branch anymore then
   build and return a leaf node with the associated final decision
else
   find a branch such that the total confusion is smallest, store the branch in the root of the tree
   separate examples to two subsets, one for the left-child and one for the right-child
   set the left-subtree to be DecisionTree(example subset for the left-child)
   set the right-subtree to be DecisionTree(example subset for the right-child)
   return the tree
end if
```



Training Algorithm

- 종류

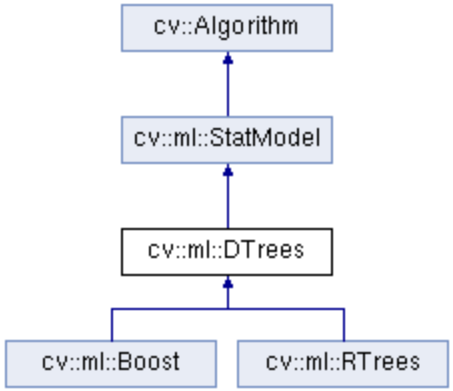
- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- C5.0 (successor of ID4)
- CART (Classification And Regression Tree)
- CHAID (CHi-squared Automatic Interaction Detector) : 이 알고리즘은 분류 트리를 계산할 때 다단계 분할을 수행한다.
- MARS (Multivariate adaptive regression splines) : 더 많은 수치 데이터를 처리하기 위해 결정 트리를 사용한다.
- 조건부 추론 트리 (Conditional Inference Trees) : 과적합을 피하기 위해 여러 테스트에 대해 보정 분할 기준으로 비 - 파라미터 테스트를 사용하는 통계 기반의 방법이다. 이 방법은 편견 예측 선택 결과와 가지 치기가 필요하지 않다.

Demonstration

https://www.snaplogic.com/machine-learning-showcase/the-decision-tree

OpenCV

- DTrees Class



  - Binary decision trees
  - CART algorithm

- Member functions

```
int     getCVFolds() const;                 // get num cross validation folds
int     getMaxCategories() const;           // get max number of categories
int     getMaxDepth() const;                // get max tree depth
int     getMinSampleCount() const;          // get min sample count
Mat     getPriors() const;                  // get priors for categories
float   getRegressionAccuracy() const;      // get required regression acc.
bool    getTruncatePrunedTree() const;      // get to truncate pruned trees
bool    getUse1SERule() const;              // get for use 1SE rule in pruning
bool    getUseSurrogates() const;           // get to use surrogates

void    setCVFolds( int val );              // set num cross validation folds
void    setMaxCategories( int val );        // set max number of categories
void    setMaxDepth( int val );             // set max tree depth
void    setMinSampleCount( int val );       // set min sample count
void    setPriors( const cv::Mat &val );    // set priors for categories
void    setRegressionAccuracy( float val ); // set required regression acc.
void    setTruncatePrunedTree( bool val );  // set to truncate pruned trees
void    setUse1SERule( bool val );          // set for use 1SE rule in pruning
void    setUseSurrogates( bool val );       // set to use surrogates
```

OpenCV

- Mushroom Dataset
  - Attributes: cap-shape, cap-color 등 22개
  - Classification: poisonous / edible (p/e)
  - 8124 개 sample

1. cap-shape: bell=b,conical=c,convex=x,flat=f, knobbed=k,sunken=s
2. cap-surface: fibrous=f,grooves=g,scaly=y,smooth=s
3. cap-color: brown=n,buff=b,cinnamon=c,gray=g,green=r, pink=p,purple=u,red=e,white=w,yellow=y
4. bruises?: bruises=t,no=f
5. odor: almond=a,anise=l,creosote=c,fishy=y,foul=f, musty=m,none=n,pungent=p,spicy=s
6. gill-attachment: attached=a,descending=d,free=f,notched=n
7. gill-spacing: close=c,crowded=w,distant=d
8. gill-size: broad=b,narrow=n
9. gill-color: black=k,brown=n,buff=b,chocolate=h,gray=g, green=r,orange=o,pink=p,purple=u,red=e, white=w,yellow=y
10. stalk-shape: enlarging=e,tapering=t
11. stalk-root: bulbous=b,club=c,cup=u,equal=e, rhizomorphs=z,rooted=r,missing=?
12. stalk-surface-above-ring: fibrous=f,scaly=y,silky=k,smooth=s
13. stalk-surface-below-ring: fibrous=f,scaly=y,silky=k,smooth=s
14. stalk-color-above-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
15. stalk-color-below-ring: brown=n,buff=b,cinnamon=c,gray=g,orange=o, pink=p,red=e,white=w,yellow=y
16. veil-type: partial=p,universal=u
17. veil-color: brown=n,orange=o,white=w,yellow=y
18. ring-number: none=n,one=o,two=t
19. ring-type: cobwebby=c,evanescent=e,flaring=f,large=l, none=n,pendant=p,sheathing=s,zone=z
20. spore-print-color: black=k,brown=n,buff=b,chocolate=h,green=r, orange=o,purple=u,white=w,yellow=y
21. population: abundant=a,clustered=c,numerous=n, scattered=s,several=v,solitary=y
22. habitat: grasses=g,leaves=l,meadows=m,paths=p, urban=u,waste=w,woods=d

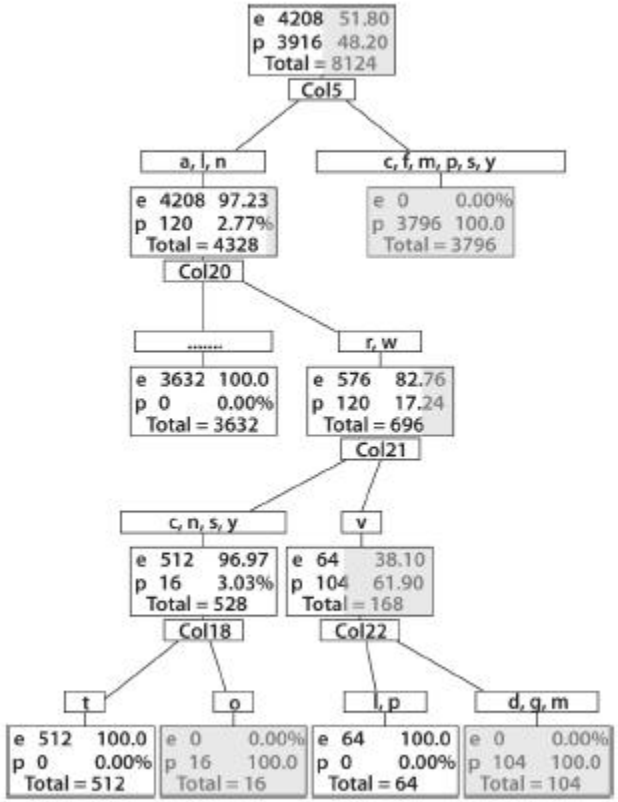
OpenCV

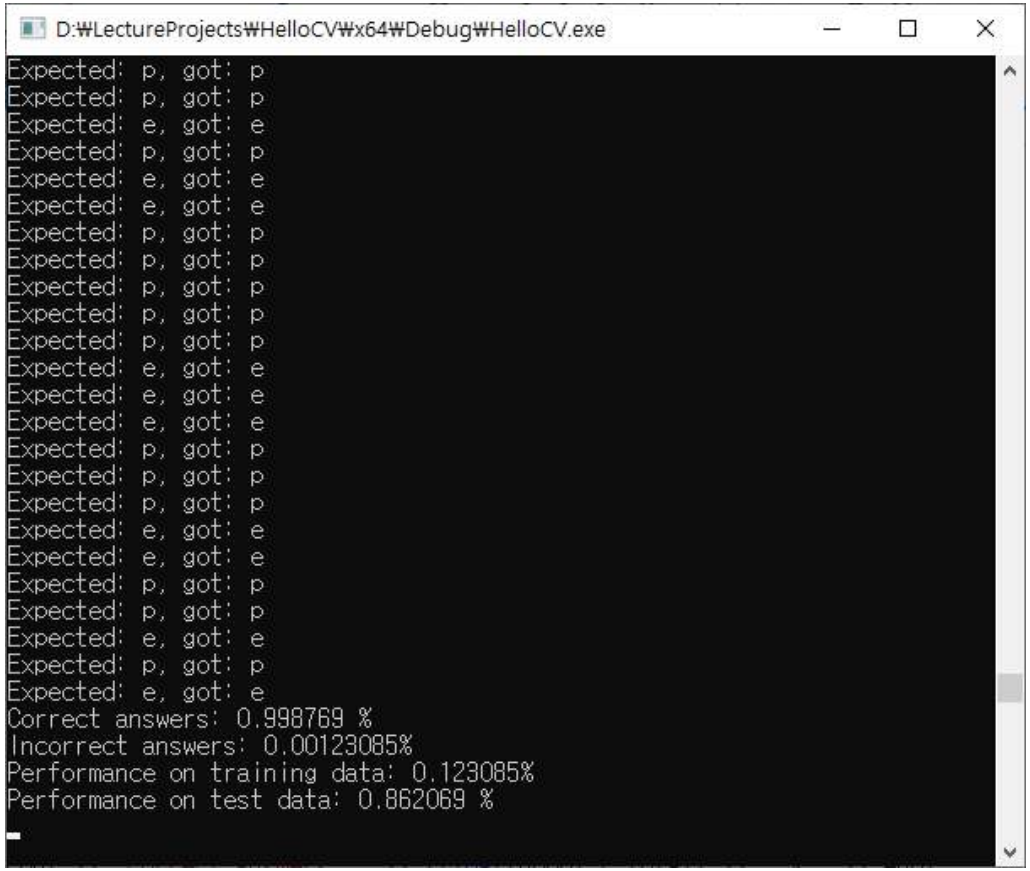
```
#include <opencv2/opencv.hpp>
#include <stdio.h>
#include <iostream>

using namespace std;
using namespace cv;

int main(int argc, char *argv[]) {
    const char *csv_file_name = "agaricus-lepiota.data";

    // Read in the CSV file that we were given.
    cv::Ptr<cv::ml::TrainData> data_set =
        cv::ml::TrainData::loadFromCSV(csv_file_name, // Input file name
        0, // Header lines (ignore this many)
        0, // Responses are (start) at thie column
        1, // Inputs start at this column
        "cat[0-22]" // All 23 columns are categorical
        );

    int n_samples = data_set->getNSamples();
    if (n_samples == 0) {
        cerr << "Could not read file: " << csv_file_name << endl;
        exit(-1);
    }
    else {
        cout << "Read " << n_samples << " samples from " << csv_file_name << endl;
    }
```

```
// Split the data, so that 90% is train data
data_set->setTrainTestSplitRatio(0.90, false);
int n_train_samples = data_set->getNTrainSamples();
int n_test_samples = data_set->getNTestSamples();
cout << "Found " << n_train_samples << " Train Samples, and "
<< n_test_samples << " Test Samples" << endl;

// Create a DTrees classifier.
cv::Ptr<cv::ml::RTrees> dtree = cv::ml::RTrees::create();

// Set parameters
dtree->setMaxDepth(8);
dtree->setMinSampleCount(10);
dtree->setRegressionAccuracy(0.01f);
dtree->setUseSurrogates(false /* true */);
dtree->setMaxCategories(15);
dtree->setCVFolds(0 /*10*/); // nonzero causes core dump
dtree->setUse1SERule(true);
dtree->setTruncatePrunedTree(true);
dtree->setPriors(cv::Mat()); // ignore priors for now...
// Now train the model
// we are only using the "train" part of the data set
dtree->train(data_set);

// Calculate the error on both the training data, as well as the test data
cv::Mat results;
float train_performance = dtree->calcError(data_set,
                                           false, // use train data
                                           results // cv::noArray());
std::vector<cv::String> names;
data_set->getNames(names);
Mat flags = data_set->getVarSymbolFlags();
```

```
// Compute some statistics on our own:
{
    cv::Mat expected_responses = data_set->getResponses();
    int good = 0, bad = 0, total = 0;
    for (int i = 0; i < data_set->getNTrainSamples(); ++i) {
        float received = results.at<float>(i, 0);
        float expected = expected_responses.at<float>(i, 0);
        cv::String r_str = names[(int)received];
        cv::String e_str = names[(int)expected];
        cout << "Expected: " << e_str << ", got: " << r_str << endl;
        if (received == expected)
           good++;
        else
           bad++;
        total++;
    }
    cout << "Correct answers: " << (float(good) / total) << " % " << endl;
    cout << "Incorrect answers: " << (float(bad) / total) << "%"
    << endl;
    }
    float test_performance = dtree->calcError(data_set,
                                              true, // use test data
                                              results // cv::noArray());
    cout << "Performance on training data: " << train_performance << "%" << endl;
    cout << "Performance on test data: " << test_performance << " % " << endl;
    return 0;
}
```

- Result

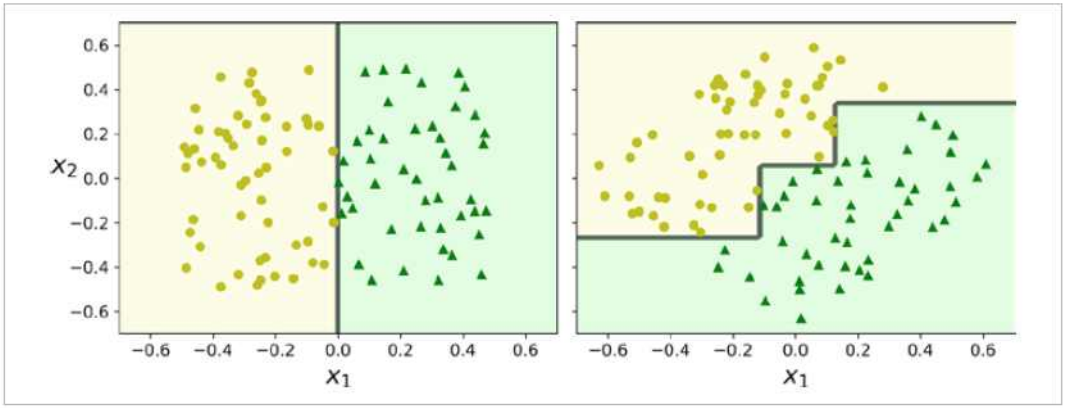

e 4208 51.80
p 3916 48.20
Total = 8124
Col5
a, l, n
c, f, m, p, s, y
e 4208 97.23
p 120 2.77%
Total = 4328
e 0 0.00%
p 3796 100.0
Total = 3796
Col20
.....
r, w
e 3632 100.0
p 0 0.00%
Total = 3632
e 576 82.76
p 120 17.24
Total = 696
Col21
c, n, s, y
v
e 512 96.97
p 16 3.03%
Total = 528
e 64 38.10
p 104 61.90
Total = 168
Col18
Col22
t
o
l, p
d, g, m
e 512 100.0
p 0 0.00%
Total = 512
e 0 0.00%
p 16 100.0
Total = 16
e 64 100.0
p 0 0.00%
Total = 64
e 0 0.00%
p 104 100.0
Total = 104


D:₩LectureProjects₩HelloCV₩x64₩Debug₩HelloCV.exe
Expected: p, got: p
Expected: p, got: p
Expected: e, got: e
Expected: p, got: p
Expected: e, got: e
Expected: e, got: e
Expected: p, got: p
Expected: p, got: p
Expected: p, got: p
Expected: p, got: p
Expected: p, got: p
Expected: p, got: p
Expected: e, got: e
Expected: e, got: e
Expected: e, got: e
Expected: p, got: p
Expected: p, got: p
Expected: p, got: p
Expected: e, got: e
Expected: e, got: e
Expected: p, got: p
Expected: p, got: p
Expected: e, got: e
Expected: p, got: p
Expected: e, got: e
Correct answers: 0.998769 %
Incorrect answers: 0.00123085%
Performance on training data: 0.123085%
Performance on test data: 0.862069 %

Summary

- 장점
  - 이해하고 해석하기 쉬움
  - 여러 용도로 사용 (Classification & Regression)
  - 계산복잡도 대비 성능 우수

- 단점
  - 계단 모양의 결정 경계 (decision boundaries) => 회전에 민감
  - Training set 에 민감 => 과적합 (Overfitting) 문제



(training set 의 회전에 민감)