Lecture 13 - Feb 21

Decision Tree Classifiers

Reading

Data Mining and Machine Learning
19 Decision Tree Classifier

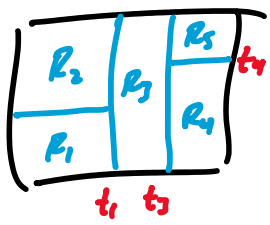
Upcoming Deadlines

Project 1 Proposal (coming soon)

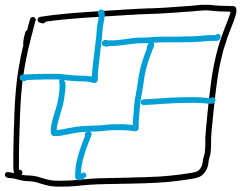
Tree-based methods partition feature space into a set of rectangles and fit a simple model in each

Major methods: CART + ID3 C4.5 C5 } Ross Quinlan — J48

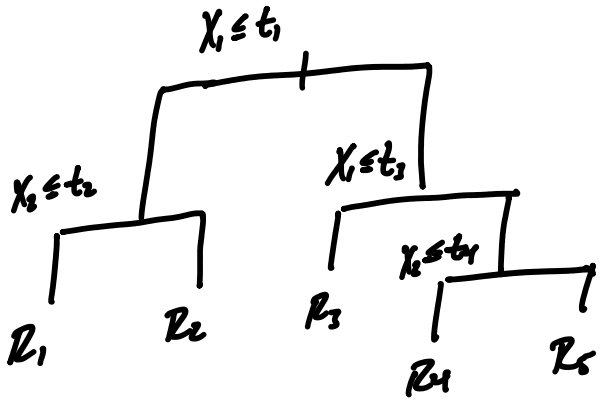
This



Not this



We restrict to binary partitions X_2 X_1



Recursive binary tree is very interpretable and works best to represent the partitions in higher dimensions

(popular in medicine)

$X \in \mathbb{R}^{n \times d}$ $\quad x_i \in \mathbb{R}^d$ $\quad y_i \in \{c_1, \ldots, c_k\}$

Decision tree predicts $\hat{y}_i$ for each x_i

$\mathcal{R}$ - data space where points lie

Decision trees split $\mathcal{R}$ with axis-parallel hyperplanes until partitions are "pure" w.r.t. classes. The tree nodes represent splits + leaves are resulting regions

19.1

Axis-parallel hyperplanes $h(x): w^T x + b = 0$

- w: weight vector in $\mathbb{R}^d$ normal to the hyperplane
- b: offset of hyperplane to origin

$\downarrow$

w restricted to $\{e_1, \ldots, e_d\}$

$\Rightarrow h(x): e_j^T x + b = 0$

$x_j + b = 0$

Cut Points

Split Points

A hyperplane specifies a **decision** or **split point** since it splits $\mathcal{R}$ into 2 half-spaces: $h(x) \leq 0$ vs $h(x) > 0$

$x_j + b \leq 0$ $\quad$ $x_j + b > 0$

$x_j \leq -b$ $\quad$ $x_j > -b$

Generic split: $X_j \leq v$ ($v = -b$) $\rightarrow$ splits $\mathcal{R} = \mathcal{R}_Y + \mathcal{R}_N$

Data Partition

$$D_Y \subset \mathcal{R}_Y = \{x^T \mid x \in D, x_j \leq v\}$$

$$D_N \in \mathcal{R}_N = \{x^T \mid x \in D, x_j > v\}$$

Purity

$$\text{purity}(D_j) = \max_i \left\{ \frac{n_{ji}}{n_j} \right\}$$

where $|D_j| = n_j$ and n_{ji} is the number of points in D_j in class c_i

fraction of points in D_j with the majority

with the majority label

Categorical Attributes

If X_j is categorical, $V \subseteq range(X_j)$

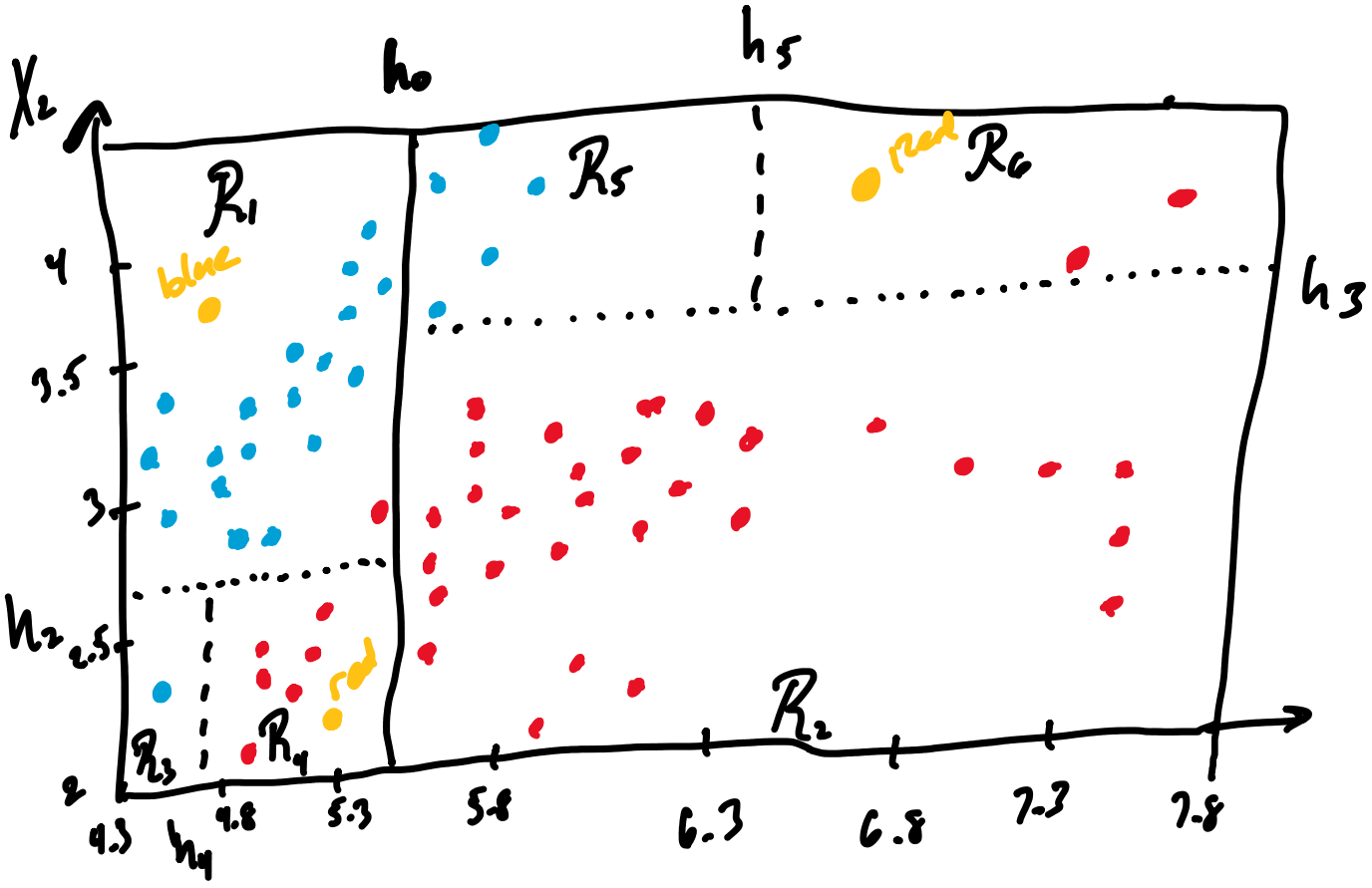
$R_Y \rightarrow$ points x with $x_j \in V$

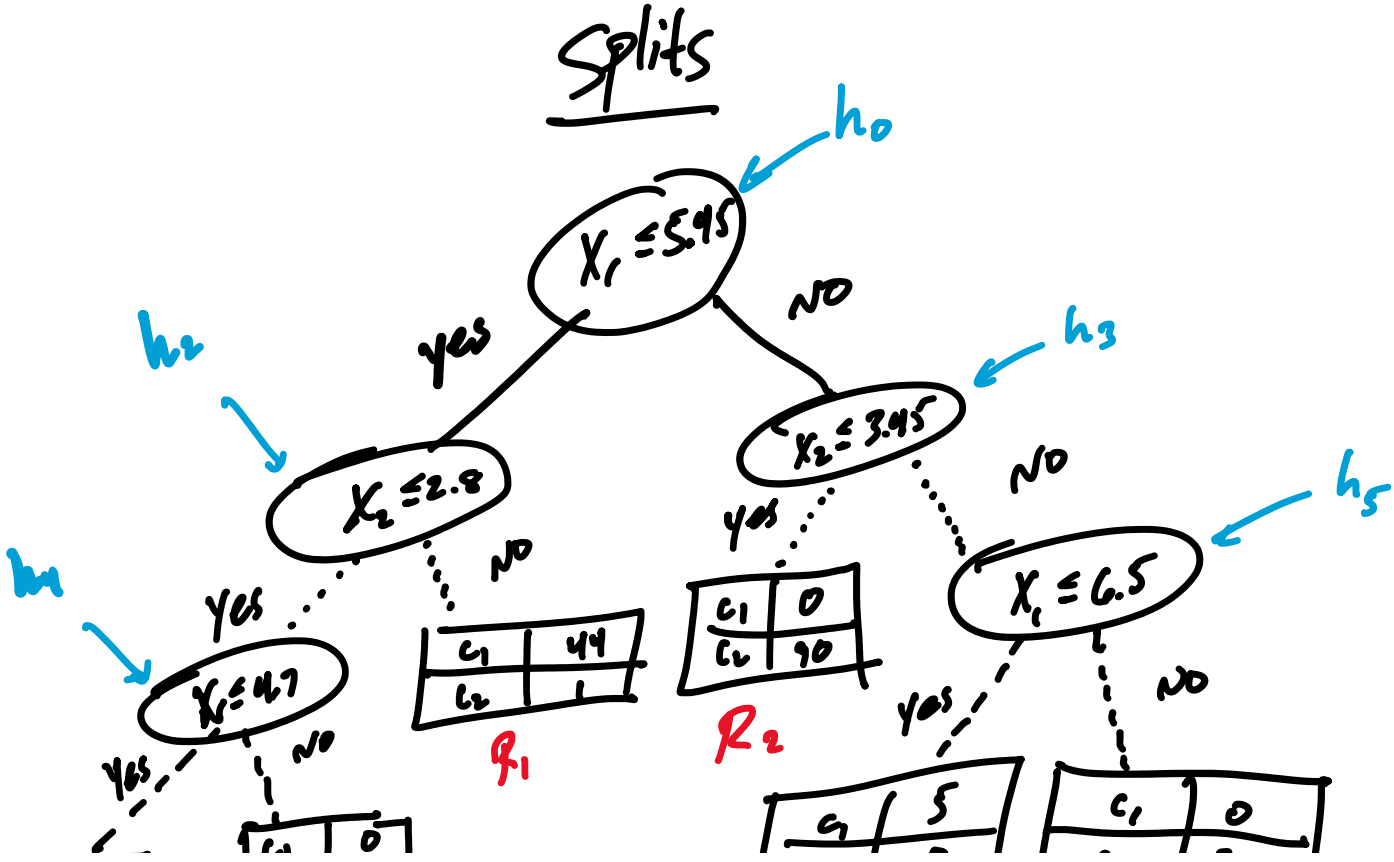
$R_N \rightarrow$ points x with $x_j \in V$

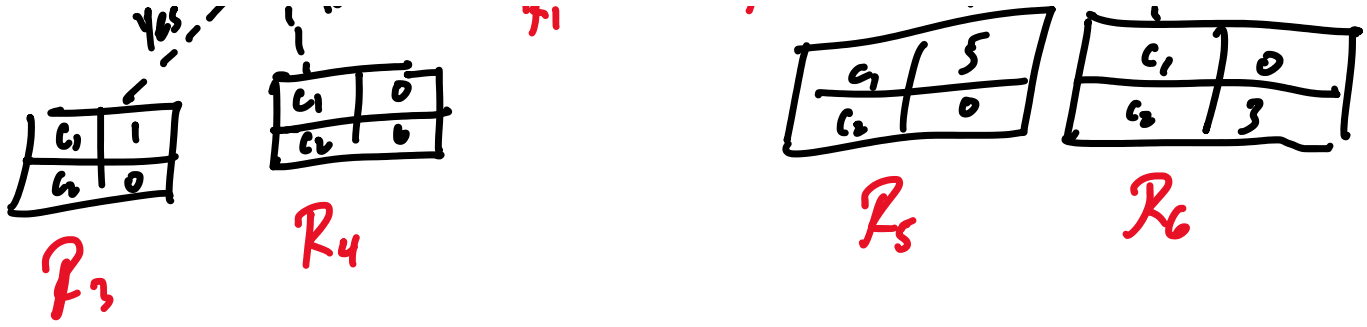
Decision Rules

DTs are easy to interpret → splits = decision rules

How It Works


X_2
h_0
h_5
R_1
R_5
R_6
blue
red
h_3
4
3.5
3
h_2
2.5
2
R_3
R_4
R_2
h_4
4.3
4.8
5.3
5.8
6.3
6.8
7.3
7.8


Splits
h_0
$X_1 \leq 5.45$
yes
NO
h_2
h_3
$X_2 \leq 2.8$
$X_2 \leq 3.45$
yes
NO
h_4
h_5
$X_1 \leq 4.7$
$X_1 \leq 6.5$
R_1
R_2



R3 R4 R5 R6

(CART)

Gini index is another measure of purity

$$G(D) = 1 - \sum_{i=1}^{k} P(c_i|D)^2$$

$P(c_j|D) = 1$ for some c_j

$P(c_i|D) = \frac{1}{k}$

high index = high disorder $\left(0 \leq G(D) \leq \frac{k-1}{k}\right)$

Weighted Gini index for split

$$G(D_Y, D_N) = \frac{n_Y}{n} G(D_Y) + \frac{n_N}{n} G(D_N)$$

Low = less disorder = good

Classification And Regression Trees (CART) measure can also measure quality of splits

"goodness"

$$CART(D_Y, D_N) = 2 \cdot \frac{n_Y}{n} \cdot \frac{n_N}{n} \sum_{i=1}^{k} \left| P(c_i|D_Y) - P(c_i|D_N) \right|$$

high = good

↳ try to maximize difference between class PMF for the partitions

Hyperparameters: η - leaf size

π - leaf purity threshold

The "best split" points are chosen

Split Point Evaluation

How do we objectively score split point candidates

Entropy measures the disorder in a system. A partition has lower entropy if it is purer

$$H(D) = -\sum_{i=1}^{k} \mathbb{P}(c_i|D) \log_2 \mathbb{P}(c_i|D) \qquad \text{(entropy)}$$

$\mathbb{P}(c_i|D)$: prob. of c_i in D

if near 1 (pure), this is zero

maximized at $\log_2 k$ if k

Split entropy: $H(D_Y, D_N) = \frac{n_Y}{n} H(D_Y) + \frac{n_N}{n} H(D_N)$

Information gain from a split: ← (ID3, C4.5, C5.0)

$$Gain(D, D_Y, D_N) = H(D) - H(D_Y, D_N)$$

high = good

If this is high, entropy is reduced more

Gini index is another measure of purity ← (CART)

$$G(D) = 1 - \sum_{i=1}^{k} P(c_i|D)^2$$

$P(c_j|D) = 1$ for some c_j ← $P(c_i|D) = \frac{1}{k}$

$$\left(0 \le G(D) \le \frac{k-1}{k}\right)$$

high index = high disorder

Weighted Gini index for split

$$G(D_Y, D_N) = \frac{n_Y}{n} G(D_Y) + \frac{n_N}{n} G(D_N)$$

Low = less disorder = good

Classification And Regression Trees (CART) measure can also measure quality of splits ← "goodness"

$$CART(D_Y, D_N) = 2 \cdot \frac{n_Y}{n} \cdot \frac{n_N}{n} \sum_{i=1}^{k} \left| P(c_i|D_Y) - P(c_i|D_N) \right|$$

high = good

↳ try to maximize difference between class PMF for the partitions

need to compute PMFs $\hat{P}(c_i|D)$ for each possible split points, which is expensive, so we can incrementally compute them instead

Numerical Attributes

$\exists$ v possible split points... one shortcut is to only take midpoints between successive X_j coordinates of points

midpoints: $\{v_1, \ldots, v_m\}$, $m \leq n-1$

$$\hat{P}(c_i|D_Y) = \hat{P}(c_i|X \leq v)$$

$$\hat{P}(c_i|D_N) = \hat{P}(c_i|X > v)$$

Bayes + LTP

$$= \frac{\hat{P}(X \leq v | c_i)\hat{P}(c_i)}{\hat{P}(X \leq v)} = \frac{\hat{P}(X \leq v | c_i)\hat{P}(c_i)}{\sum_{j=1}^{k} \hat{P}(X \leq v | c_j)\hat{P}(c_j)}$$

$\hat{P}(c_i) = \frac{n_i}{n}$

$$\hat{P}(X \le v \mid c_i) = \frac{\hat{P}(X \le v \text{ and } c_i)}{\hat{P}(c_i)} = \frac{\frac{1}{n}\sum_{j=1}^{n} \mathbb{1}\{x_j \le v \text{ and } y_j = c_i\}}{\frac{n_i}{n}} = \frac{N_{vi}}{n_i}$$

$$\hat{P}(c_i \mid D_N) = \hat{P}(c_i \mid X \le v) = \frac{\frac{N_{vi}}{n_i} \cdot \frac{n_i}{n}}{\sum_{j=1}^{k} \frac{N_{vj}}{n_j} \cdot \frac{n_j}{n}} = \boxed{\frac{N_{vi}}{\sum_{j=1}^{k} N_{vj}}}$$

$$\hat{P}(X > v \mid c_i) = 1 - \hat{P}(X \le v \mid c_i) = 1 - \frac{N_{vi}}{n_i} = \frac{n_i - N_{vi}}{n_i}$$

$$\Rightarrow \hat{P}(c_i \mid D_N) = \frac{\frac{n_i - N_{vi}}{n_i} \cdot \frac{n_i}{n}}{\sum_{j=1}^{K} \frac{n_j - N_{vj}}{n_j} \cdot \frac{n_j}{n}} = \boxed{\frac{n_i - N_{vi}}{\sum_{j=1}^{k} n_j - N_{vj}}}$$

```
DecisionTree(X, y, n, π)
    n = |X|
    n_i = |X_i|
    purity(X) = max_i n_i / n
    if n ≤ n or purity(X) ≥ π
        c* = argmax_{c_i} n_i / n
        create leaf node, label with class c*
        return
    (split point*, score) = (∅, 0)
    for i in range(d)
        if X_j is numeric:
            (v, score) = eval-numeric-attribute(X, X_i)   ← find best cut in X_i
            if score > score*: (split point*, score*) = (X_j ≤ v, score)
        if X_j is categorical:
            (V, score) = eval-cat-attribute(X, X_i)
            if score > score*: (split point*, score*) = (X_j ∈ V, score)
```

if score > score*: (split pt*, score*) = (X_j ∈ v, score)

partition X into X_Y and X_N using split pt* and call recursively

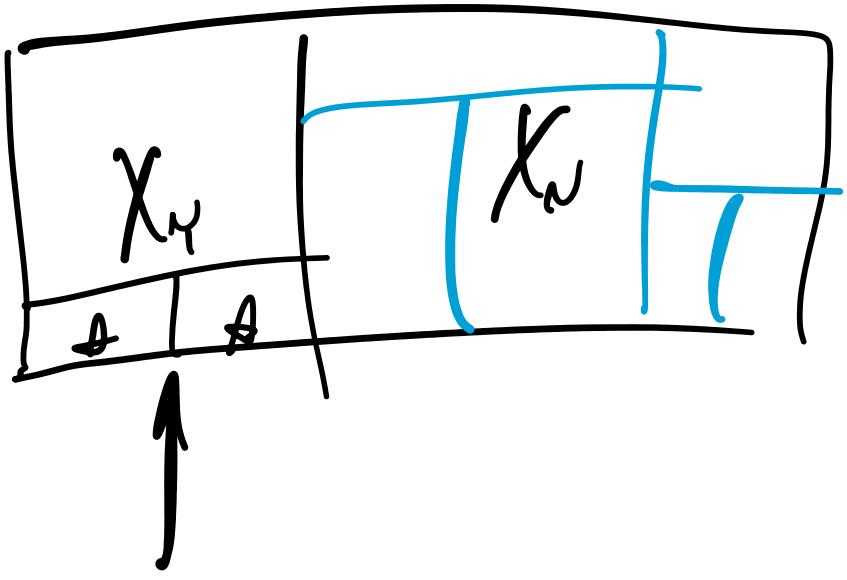
$X_Y = \{x^T \mid x \in X$ satisfies split pt$^*\}$

$X_N = \{x^T \mid x \in X$ does not ... $\}$

create internal node at split pt* with child nodes X_Y, X_N

DecisionTree(X_Y, y_Y, n, π)

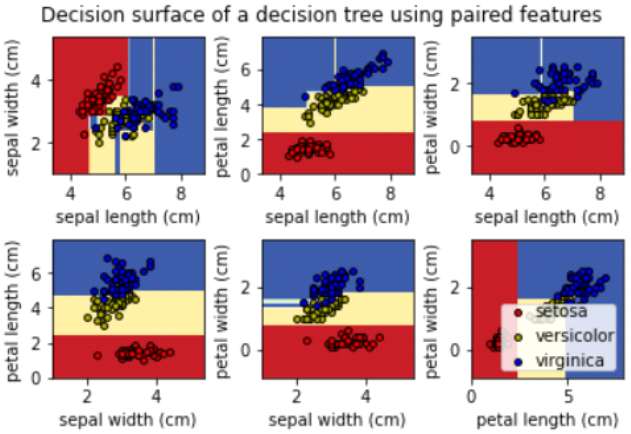
DecisionTree(X_N, y_N, n, π)

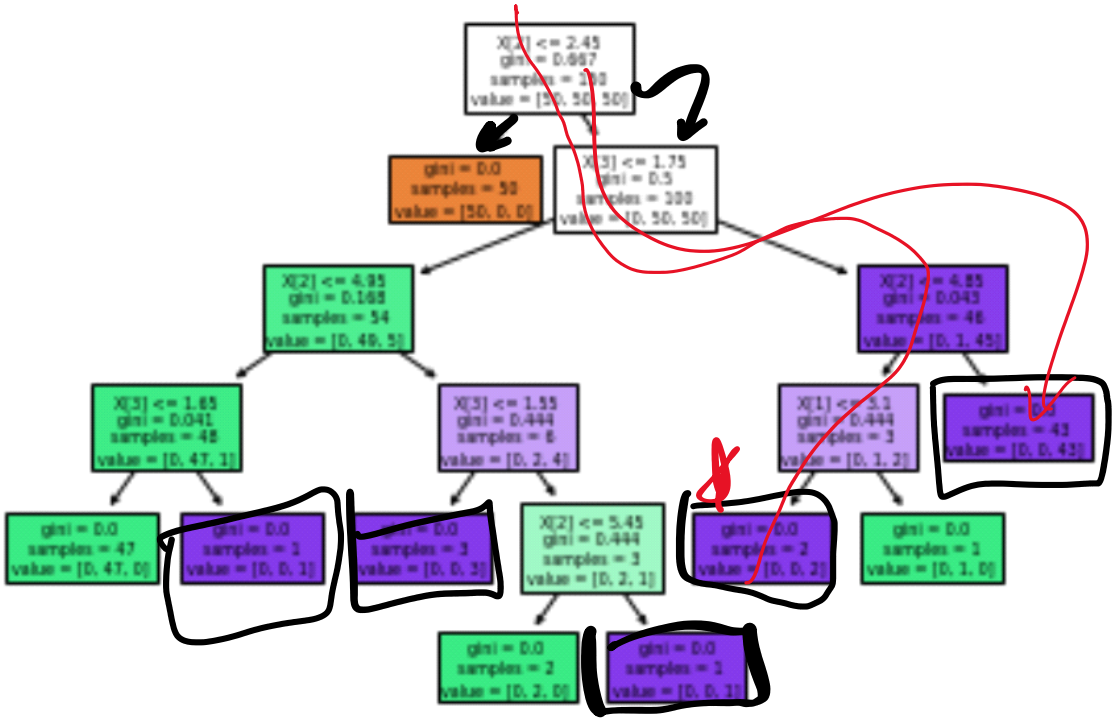


Iris Example

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree

# hyperparameters
n_classes = 3
plot_colors = "ryb"
plot_step = 0.02

# Load the iris dataset
iris = load_iris()

# run the model for each pair of features
for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]):
    # take the two corresponding features
    X = iris.data[:, pair]
    y = iris.target

    # fit the classifier
    clf = DecisionTreeClassifier().fit(X, y)

    # plot the decision boundary
    plt.subplot(2, 3, pairidx + 1)

    # predict the testing data on a tight mesh of points in space and color-code them
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step), np.arange(y_min, y_max, plot_step))
```

```
    plt.tight_layout(h_pad=0.5, w_pad=0.5, pad=2.5)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    cs = plt.contourf(xx, yy, Z, cmap=plt.cm.RdYlBu)

    plt.xlabel(iris.feature_names[pair[0]])
    plt.ylabel(iris.feature_names[pair[1]])

    # plot the training points
    for i, color in zip(range(n_classes), plot_colors):
        idx = np.where(y == i)
        plt.scatter(X[idx, 0], X[idx, 1], c=color, label=iris.target_names[i], cmap=plt.cm.RdYlBu, edgecolor='black', s=15)

plt.suptitle("Decision surface of a decision tree using paired features")
plt.legend(loc='lower right', borderpad=0, handletextpad=0)
plt.axis("tight")

plt.figure()
clf = DecisionTreeClassifier().fit(iris.data, iris.target)
plot_tree(clf, filled=True)
plt.show()
```


Decision surface of a decision tree using paired features
sepal width (cm)
petal length (cm)
petal width (cm)
sepal length (cm)
setosa
versicolor
virginica


gini = 0.0
samples = 50
value = [50, 0, 0]
X[3] <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
X[2] <= 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
gini = 0.0
samples = 43
value = [0, 0, 43]
gini = 0.0
samples = 47
value = [0, 47, 0]
gini = 0.0
samples = 2
value = [0, 0, 2]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 2
value = [0, 2, 0]
gini = 0.0
samples = 1
value = [0, 0, 1]