

# Lectures 5 & 6 Module 3. Decision Tree

## COMP 551 Applied machine learning

Yue Li  
Assistant Professor  
School of Computer Science  
McGill University

January 21 & 23, 2025

# Outline

Objectives

Decision Tree (DT)

Cost functions

DT training algorithm

Choice of tree depth

The issue of overfitting in DT

Feature importance

Summary

## Learning objectives

Understanding the following concepts

- DT prediction
- Cost functions
- DT training algorithm
- Overfitting in DT

# Outline

Objectives

Decision Tree (DT)

Cost functions

DT training algorithm

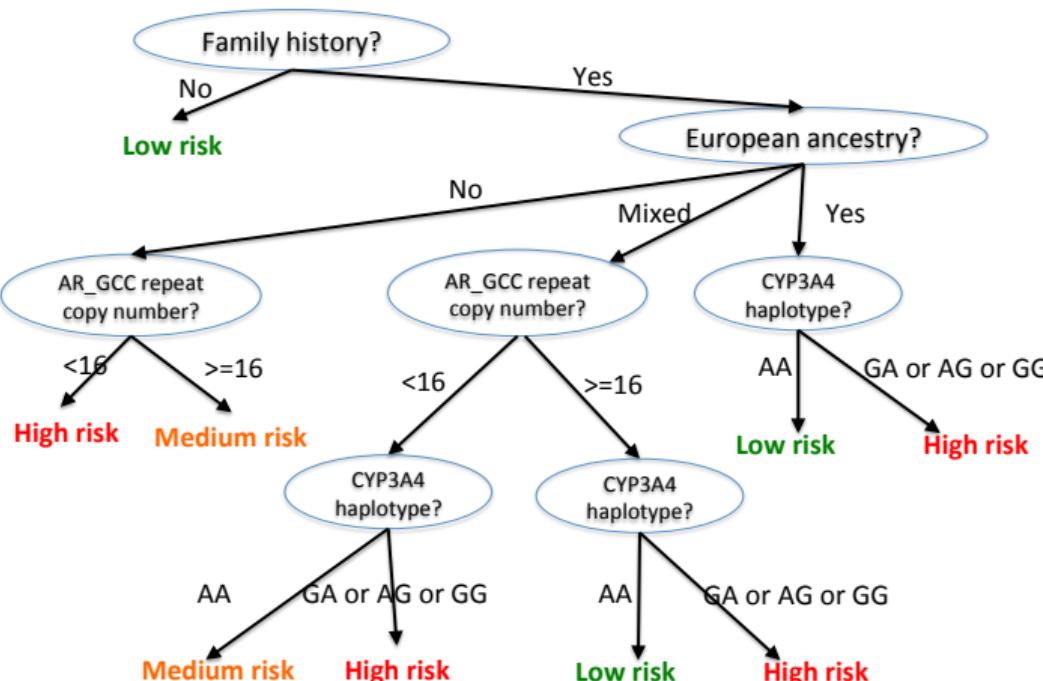
Choice of tree depth

The issue of overfitting in DT

Feature importance

Summary

# Decision Tree (DT) aka Classification And Regression Tree (CART)



A toy DT for diagnosing non-small cell lung cancer

## Pros:

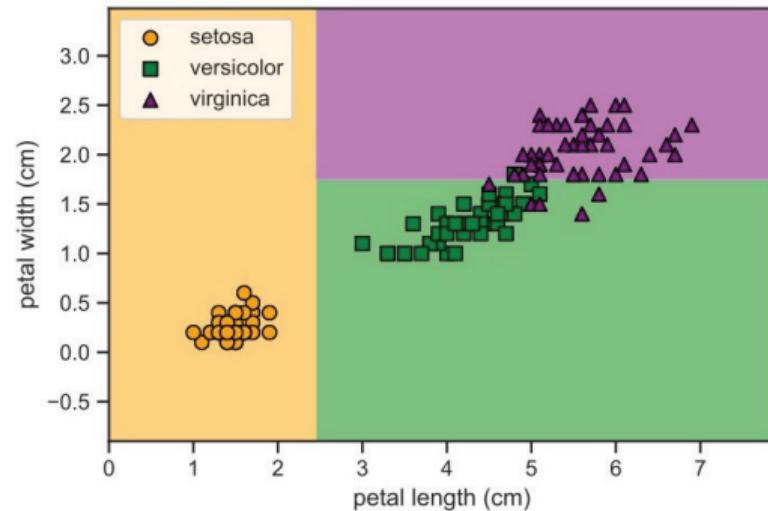
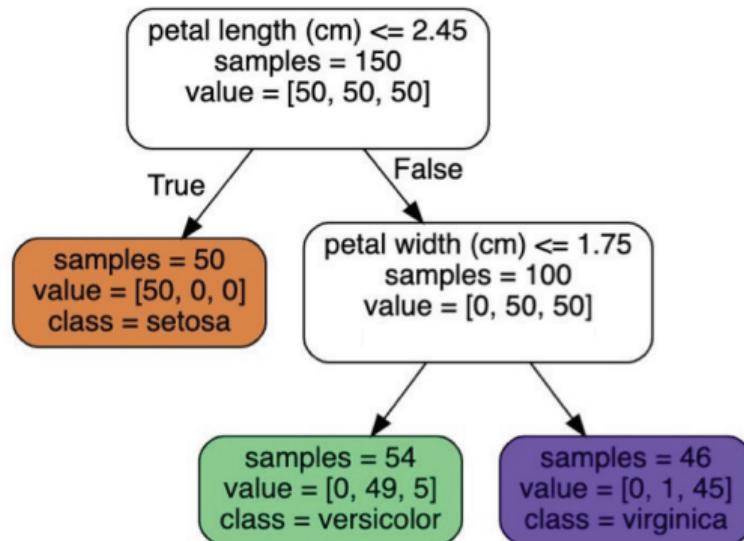
- Easy to interpret
- Handle mixed discrete and continuous inputs
- Insensitive to scaling
- Built-in variable selection
- Non-linear approach

## Cons:

- Produce coarse grained probabilities especially at low tree depth
- Prone to overfitting at high tree depth

## DT trained on Iris flower data for multi-class prediction

An example of classifying Iris flower using a classification tree (Murphy22 Figure 18.3):



# Notation review for general supervised learning problem

Our dataset  $\mathcal{D}$  consists of  $N$  pairs of input vector and target variable:

$$\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$$

## Target variable

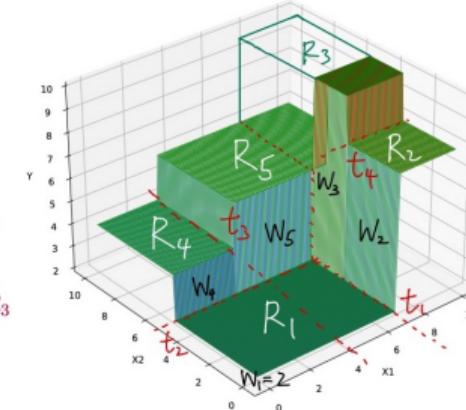
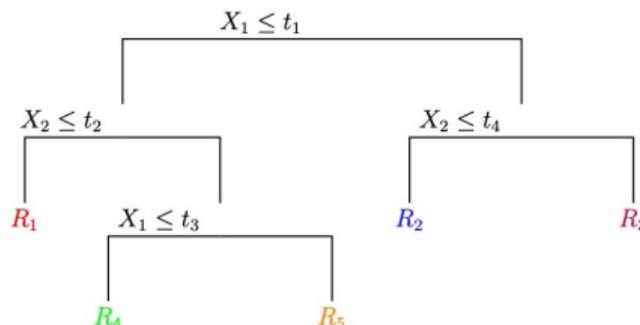
- Classification:  $y^{(n)} \in \{1, \dots, C\}$
- Regression:  $y^{(n)} \in \mathbb{R}$

## One-hot-encoding

For computational convenience, we often represent the categorical variable using **one-hot encoding** by creating  $C_j$  binary input features from variable  $j$  (although this is not necessary for DT to work):

Data	Color	Variable	Red	Green	Blue
$x^{(1)}$	Red	$x^{(1)}$	1	0	0
$x^{(2)}$	Green	$x^{(2)}$	0	1	0
$x^{(3)}$	Blue	$x^{(3)}$	0	0	1

# A simple illustration of a regression tree



- Divide the training data into 5 regions  $R_k \in \{R_1, \dots, R_5\}$  by the thresholds  $t_1, \dots, t_4$ . For instance,  $R_1 = \{\mathbf{x}_1 \leq t_1, \mathbf{x}_2 \leq t_2\}$ .
- The height of a region (i.e., regression weight) is the *average* target values in region  $k$ :

$$w_k = \frac{\sum_{n=1}^N y^{(n)} \mathbb{I}[\mathbf{x}^{(n)} \in R_k]}{\sum_{n=1}^N \mathbb{I}[\mathbf{x}^{(n)} \in R_k]} = \frac{1}{N_k} \sum_{n \in S_k} y^{(n)} \quad (S_k \text{ is the set of data indices in Region } k)$$

- The predicted value of a new data point  $\mathbf{x}^{(*)}$  is set to the weight of the region it belongs:  
$$\hat{y}^{(*)} \equiv f(\mathbf{x}^{(*)}; \theta) = \sum_{k=1}^K w_k \mathbb{I}[\mathbf{x}^{(*)} \in R_k] \quad (\theta \text{ denotes the trained tree and its regression weights})$$

## Prediction per region in classification

Most frequent label in region  $k$  (i.e., the mode):

$$w_k = \arg \max_c \sum_{n \in S_k} \mathbb{I}[y^{(n)} = c]$$

$$\hat{y}^{(*)} = \sum_{k=1}^K w_k \mathbb{I}[\mathbf{x}^{(*)} \in R_k]$$

Class probability in region  $k$  (i.e., proportion of class  $c$  in region  $k$ ):

$$p(y = c | R_k) = \frac{1}{N_k} \sum_{n \in S_k} \mathbb{I}[y^{(n)} = c] \equiv \pi_k(c)$$

$$p(\hat{y}^{(*)} = c) = \sum_{k=1}^K \pi_k(c) \mathbb{I}[\mathbf{x}^{(*)} \in R_k]$$

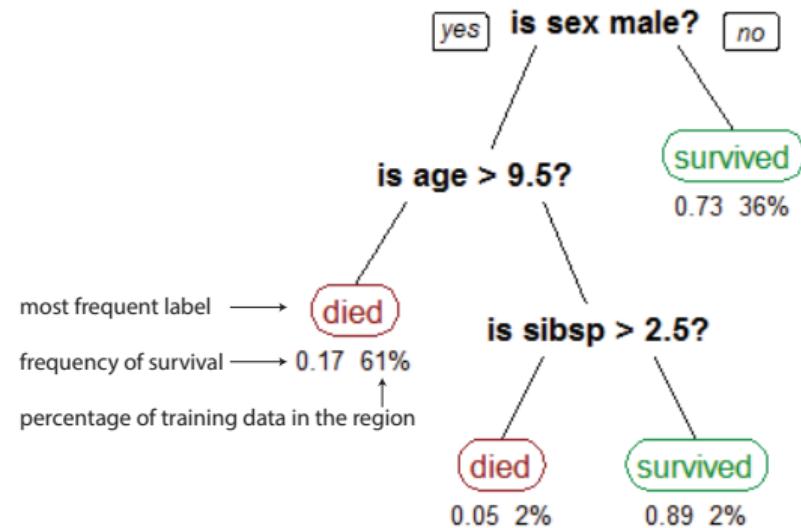


image source from [here](#)

Survived is the positive label and death is the negative label in the above DT trained on the Titanic dataset.

## Two main questions to be answered in this learning module

1. How to split data at each tree node?
    - How to evaluate quality of a split?
    - How to choose feature & threshold?
  2. How to decide when to stop the tree from splitting?
1. Evaluate each split by a **cost function**
  2. Choose feature and threshold that **minimize** the cost
  3. Use a validation set to choose tree depth

# Outline

Objectives

Decision Tree (DT)

**Cost functions**

DT training algorithm

Choice of tree depth

The issue of overfitting in DT

Feature importance

Summary

A common cost function for regression is *Mean Squared Error (MSE)*

Our prediction on the training data points in region  $k$  is their average target values:

$$w_k = \frac{1}{N_k} \sum_{n \in \mathcal{S}_k} y^{(n)}$$

Note: each training data point falls in exactly one of the regions.

Sum of squared error (SSE) and Mean squared error per region  $k$  are:

$$\text{SSE}(R_k, \mathcal{D}; w_k) = \sum_{n \in \mathcal{S}_k} (y^{(n)} - w_k)^2, \quad \text{MSE}(R_k, \mathcal{D}; w_k) = \frac{1}{N_k} \text{SSE}(R_k, \mathcal{D}; w_k)$$

SSE and MSE over all  $K$  regions (i.e., all training data points) are:

$$\text{SSE}(\mathcal{D}; \theta) = \sum_{k=1}^K \text{SSE}(R_k, \mathcal{D}; w_k) = \sum_{k=1}^K \sum_{n \in \mathcal{S}_k} (y^{(n)} - w_k)^2, \quad \text{MSE}(\mathcal{D}; \theta) = \frac{1}{N} \text{SSE}(\mathcal{D}; \theta)$$

## Measuring classification cost using *Misclassification Rate*

Most frequent label in region  $k$  (i.e., the mode):

$$w_k = \arg \max_c \sum_{n \in \mathcal{S}_k} \mathbb{I}[y^{(n)} = c]$$

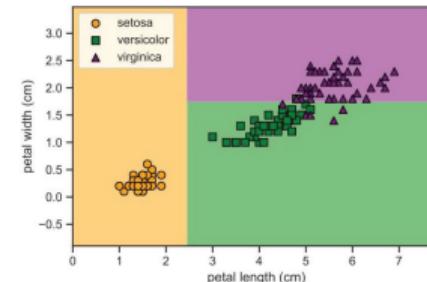
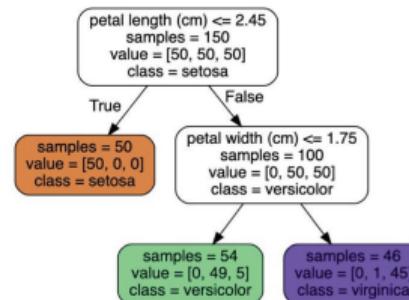
Cost per region  $k$ :

$$\text{cost}(R_k, \mathcal{D}; w_k) = \sum_{n \in \mathcal{S}_k} \mathbb{I}[y^{(n)} \neq w_k]$$

Total cost is the overall misclassification rate:

$$\text{total cost}(\mathcal{D}; \theta) = \sum_{k=1}^K \text{cost}(R_k, \mathcal{D}; w_k)$$

$$\text{misclassification rate} = \frac{1}{N} \text{cost}(\mathcal{D}; \theta)$$

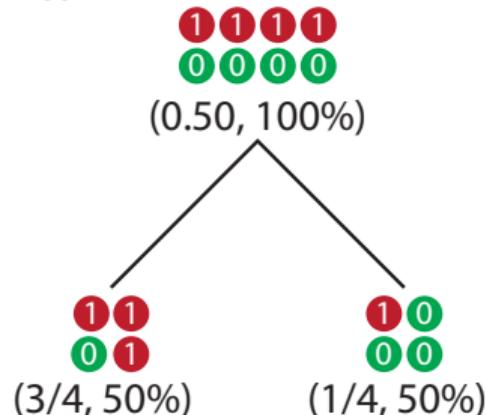


Pop quiz: what's training  
misclassification rate of the above DT?  
Answer:  $6/150 = 0.04$

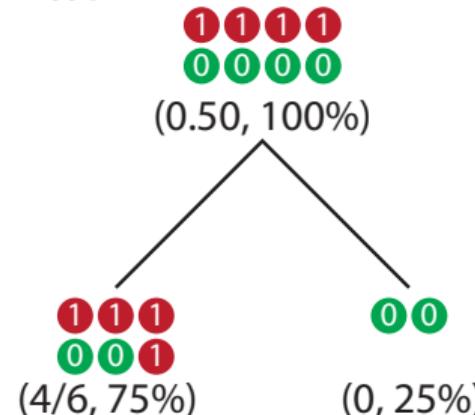
## Misclassification rate is not sensitive enough

The total misclassification rates are the same for these two trees but the classification cost at each leaf node is different:

Tree A



Tree B



$$cost(\mathcal{D}; \theta) = \frac{1}{N} \sum_{k=1}^K \sum_{n \in S_k} \mathbb{I}[y^{(n)} \neq w_k] = 2/8 = 0.25$$

Idea: we need a more sensitive measure of the **model uncertainties** in each region.

## Entropy: a measure of uncertainty (the lower the better)

Entropy is the expected amount of information in observing a random variable  $y$ :

$$H(y) = - \sum_{c=1}^C p(y=c) \log_2 p(y=c)$$

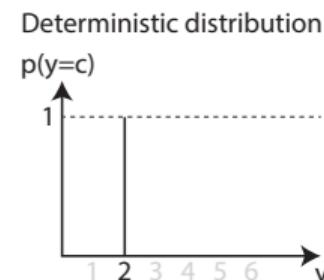
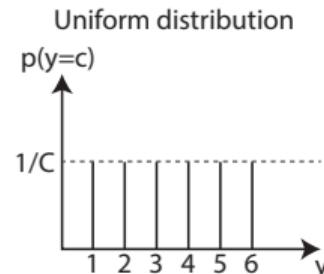
The *less certain* the *higher* the entropy. For example,

- Uniform distribution where  $p(y=c) = \frac{1}{C} \forall c$  has the highest entropy:

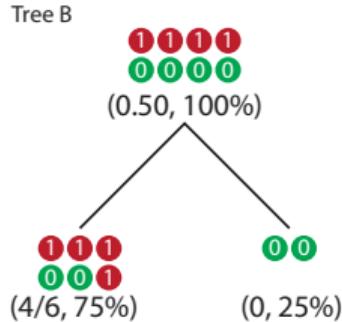
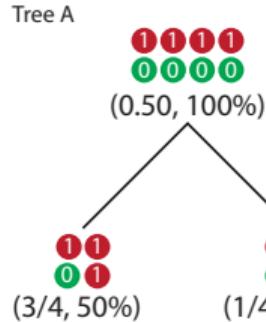
$$H(y) = - \sum_{c=1}^C \frac{1}{C} \log_2 \left( \frac{1}{C} \right) = \log_2(C)$$

- When  $p(y=c) = 1$  and  $p(y=c') = 0 \forall c' \neq c$  the entropy is the lowest:

$$H(y) = - \log_2(1) = 0$$



# Entropy for classification cost



$$\begin{aligned} H(y; R) &= - \sum_c p(y = c; R) \log_2 p(y = c; R) \\ &= -p(y = 0; R) \log_2 p(y = 0; R) \\ &\quad - p(y = 1; R) \log_2 p(y = 1; R) \end{aligned}$$

Tree A

$$H(y; R_{left}) = -\frac{1}{4} \log_2 \left(\frac{1}{4}\right) - \frac{3}{4} \log_2 \left(\frac{3}{4}\right) \approx 0.81$$

$$H(y; R_{right}) = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} \approx 0.81$$

$$\text{Mean Cost} = \frac{4}{8} H(y; R_{left}) + \frac{4}{8} H(y; R_{right}) = 0.81$$

Tree B

$$H(y; R_{left}) = -\frac{2}{6} \log_2 \left(\frac{2}{6}\right) - \frac{4}{6} \log_2 \left(\frac{4}{6}\right) \approx 0.92$$

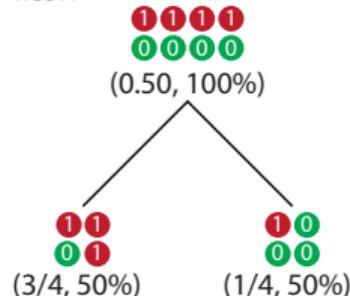
$$H(y; R_{right}) = -1 \log_2 1 - 0 \approx 0$$

$$\text{Mean Cost} = \frac{6}{8} H(y; R_{left}) + \frac{2}{8} H(y; R_{right}) = \mathbf{0.69}$$

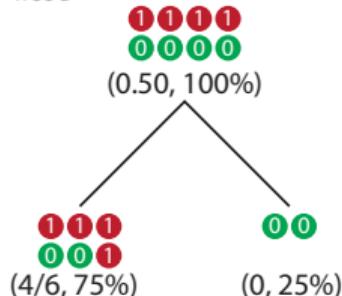
Tree B has lower cost than Tree A.

# Entropy for classification cost

Tree A



Tree B



$$\begin{aligned} H(y; R) &= - \sum_c p(y = c; R) \log_2 p(y = c; R) \\ &= -p(y = 0; R) \log_2 p(y = 0; R) \\ &\quad - p(y = 1; R) \log_2 p(y = 1; R) \end{aligned}$$

Tree A

$$H(y; R_{left}) = -\frac{1}{4} \log_2 \left(\frac{1}{4}\right) - \frac{3}{4} \log_2 \left(\frac{3}{4}\right) \approx 0.81$$

$$H(y; R_{right}) = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} \approx 0.81$$

$$\text{Mean Cost} = \frac{4}{8} H(y; R_{left}) + \frac{4}{8} H(y; R_{right}) = 0.81$$

Tree B

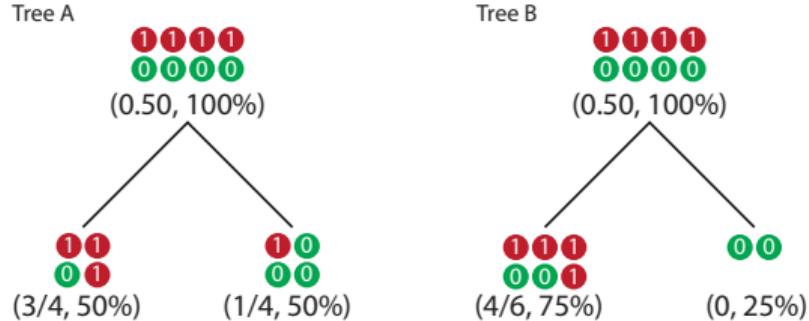
$$H(y; R_{left}) = -\frac{2}{6} \log_2 \left(\frac{2}{6}\right) - \frac{4}{6} \log_2 \left(\frac{4}{6}\right) \approx 0.92$$

$$H(y; R_{right}) = -1 \log_2 1 - 0 \approx 0$$

$$\text{Mean Cost} = \frac{6}{8} H(y; R_{left}) + \frac{2}{8} H(y; R_{right}) = 0.69$$

Tree B has lower cost than Tree A.

# Entropy for classification cost



Tree A

$$H(y; R_{left}) = -\frac{1}{4} \log_2 \left(\frac{1}{4}\right) - \frac{3}{4} \log_2 \left(\frac{3}{4}\right) \approx 0.81$$

$$H(y; R_{right}) = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} \approx 0.81$$

$$\text{Mean Cost} = \frac{4}{8} H(y; R_{left}) + \frac{4}{8} H(y; R_{right}) = 0.81$$

$$\begin{aligned} H(y; R) &= - \sum_c p(y = c; R) \log_2 p(y = c; R) \\ &= -p(y = 0; R) \log_2 p(y = 0; R) \\ &\quad - p(y = 1; R) \log_2 p(y = 1; R) \end{aligned}$$

Tree B

$$H(y; R_{left}) = -\frac{2}{6} \log_2 \left(\frac{2}{6}\right) - \frac{4}{6} \log_2 \left(\frac{4}{6}\right) \approx 0.92$$

$$H(y; R_{right}) = -1 \log_2 1 - 0 \approx 0$$

$$\text{Mean Cost} = \frac{6}{8} H(y; R_{left}) + \frac{2}{8} H(y; R_{right}) = \mathbf{0.69}$$

Tree B has lower cost than Tree A.

Resume here in Lec 6 (Jan 24)

## Gini Index (GI): the default cost in DT (the lower the better)

$$p(y = c; R_k) = \frac{1}{N_k} \sum_{n \in S_k} \mathbb{I}(y^{(n)} = c) \equiv \pi_k(c)$$

$$GI(R_k) = \sum_{c=1}^C \pi_k(c)(1 - \pi_k(c)) = \sum_{c=1}^C \pi_k(c) - \sum_{c=1}^C \pi_k(c)^2 = 1 - \sum_{c=1}^C \pi_k(c)^2$$

$$GI = \frac{1}{N} \sum_{k=1}^K N_k GI(R_k)$$

- In theory,  $\pi_k(c)(1 - \pi_k(c))$  is the *variance* of the *Bernoulli distribution* for the Boolean indicator  $\mathbb{I}[y = c]$  in region  $k$  with rate  $\pi_k(c)$  for each class  $c$ :

$$y \sim (\pi_k(c))^{\mathbb{I}[y=c]} (1 - \pi_k(c))^{\mathbb{I}[y \neq c]}$$

- Therefore,  $\sum_c \pi_k(c)(1 - \pi_k(c))$  measures *the sum of variances* over all classes.
- Therefore, minimizing GI is equivalent to minimizing the expected variance.
- For a “pure” region, where all data points in the region belong to a single class  $c$ ,  $p(y = c; R_k) = 1$  and  $p(y = c'; R_k) = 0$  for  $c' \neq c$ ,  $GI(R_k) = 0$ .

## Gini Index (GI): the default cost in DT (the lower the better)

$$p(y = c; R_k) = \frac{1}{N_k} \sum_{n \in S_k} \mathbb{I}(y^{(n)} = c) \equiv \pi_k(c)$$

$$GI(R_k) = \sum_{c=1}^C \pi_k(c)(1 - \pi_k(c)) = \sum_{c=1}^C \pi_k(c) - \sum_{c=1}^C \pi_k(c)^2 = 1 - \sum_{c=1}^C \pi_k(c)^2$$

$$GI = \frac{1}{N} \sum_{k=1}^K N_k GI(R_k)$$

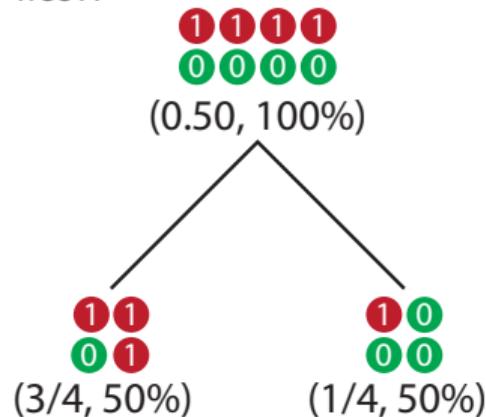
- In theory,  $\pi_k(c)(1 - \pi_k(c))$  is the *variance* of the **Bernoulli distribution** for the Boolean indicator  $\mathbb{I}[y = c]$  in region  $k$  with rate  $\pi_k(c)$  for each class  $c$ :

$$y \sim (\pi_k(c))^{\mathbb{I}[y=c]} (1 - \pi_k(c))^{\mathbb{I}[y \neq c]}$$

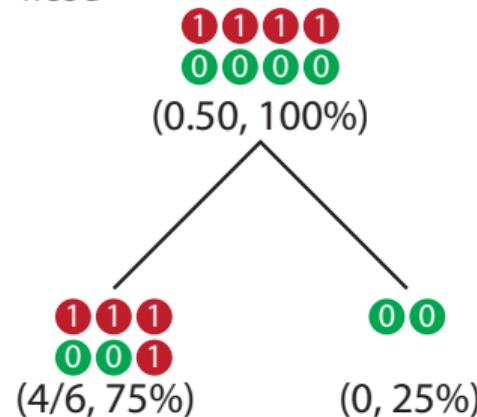
- Therefore,  $\sum_c \pi_k(c)(1 - \pi_k(c))$  measures *the sum of variances* over all classes.
- Therefore, minimizing GI is equivalent to minimizing the expected variance.
- For a “pure” region, where all data points in the region belong to a single class  $c$ ,  $p(y = c; R_k) = 1$  and  $p(y = c'; R_k) = 0$  for  $c' \neq c$ ,  $GI(R_k) = 0$ .

## Compute GI for the two example trees

Tree A



Tree B



$$GI(y; R_{A,\text{left}}) = 1 - (0.25^2 + 0.75^2) = 0.375$$

$$GI(y; R_{A,\text{right}}) = 1 - (0.75^2 + 0.25^2) = 0.375$$

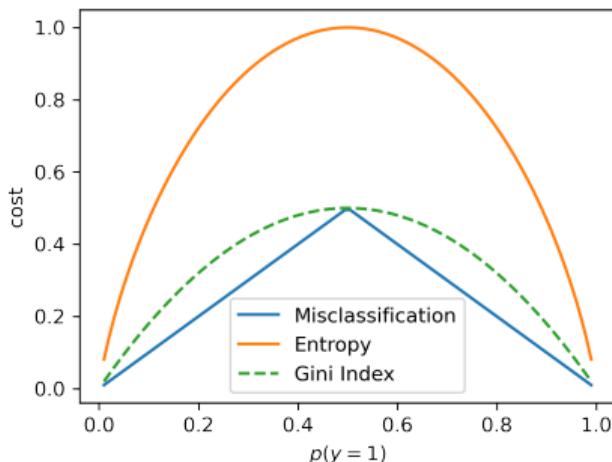
$$\begin{aligned} GI(y; \text{Tree}A) &= \frac{4}{8} GI(y; R_{\text{left}}) + \frac{4}{8} GI(y; R_{\text{right}}) \\ &= 0.375 \end{aligned}$$

$$GI(y; R_{B,\text{left}}) = 1 - ((\frac{2}{6})^2 + (\frac{4}{6})^2) = 0.44$$

$$GI(y; R_{B,\text{right}}) = 1 - (1 + 0) = 0$$

$$GI(y; \text{Tree}B) = \frac{6}{8} GI(y; R_{\text{left}}) + 0 = \mathbf{0.33}$$

# Analysis of the 3 *training* cost functions in binary classification (Colab)



Two-class cost as the function  $p(y = 1) \equiv p$ :

$$\text{misclas. rate} = 1 - \max(p, 1 - p) = \min(p, 1 - p)$$

$$\text{entropy} = -p \log_2(p) - (1 - p) \log_2(1 - p)$$

$$\text{gini index} = p(1 - p) + (1 - p)p = 2p(1 - p)$$

Comments:

- All 3 cost functions quantify the “impurity” of a tree node.
- Training a DT involves minimizing one of the cost functions.
- Entropy and Gini Index behave similarly and are usually better choice than misclassification rate as a training cost function because they take into account *the prediction uncertainties* by using prediction probabilities in each leaf node.

# Outline

Objectives

Decision Tree (DT)

Cost functions

**DT training algorithm**

Choice of tree depth

The issue of overfitting in DT

Feature importance

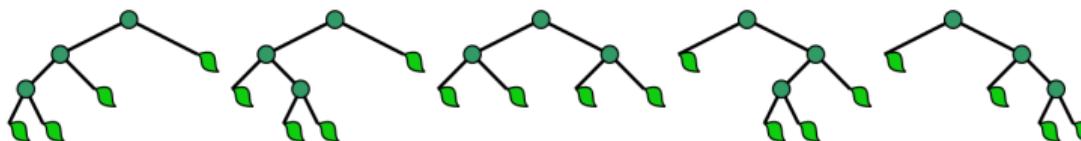
Summary

## Combinatorial search space of all possible trees

- **Objective:** find the optimal DT that minimizes the training cost.
- The number of all possible topologies of *full* binary trees with  $T$  tests (i.e.,  $T$  non-leaf nodes) follows the [Catalan sequence](#), where the  $T^{th}$  Catalan number is

$$C_T = \frac{1}{T+1} \binom{2T}{T} = \frac{(2T)!}{(T+1)(2T-T)!T!} = \frac{(2T)!}{(T+1)!T!}$$

For 3 tests, for instance, there are 5 full binary trees to choose from:



- The number of trees grows quite fast as we increase the number of tests. Starting with  $T = 0$ , the first 12 numbers in the Catalan sequence are: 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786.
- It turns out finding the optimal decision tree is an **NP-hard** combinatorial optimization problem (NP stands for Nondeterministic Polynomial).

## Peudocode for greedy heuristic search at each tree node (See Colab)

---

### Algorithm 1 greedy\_test(node)

---

```
1: min_cost = cost(node.y) // (better than min_cost =  $\infty$ )
2: best_feature_index = best_feature_thres = NULL
3: for each feature  $d \in \{1, \dots, D\}$  do
4:   for each threshold  $t \in \mathcal{T}_d$  for feature  $d$  do // Tests for all unique values of feature  $d$ 
5:     cost_left = cost(node.y[node.X[:, d]  $\leq t$ ])
6:     cost_right = cost(node.y[node.X[:, d] > t])
7:     cost_d =  $\frac{N_{left}}{N} \text{cost\_left} + \frac{N_{right}}{N} \text{cost\_right}$ 
8:     if cost_d < min_cost then
9:       min_cost = cost_d;
10:      best_feature_index = d;
11:      best_feature_thres = threshold
12:    end if
13:  end for
14: end for
15: return min_cost, best_feature_index, best_feature_thres
```

## Recursively find the best feature and threshold at every node (Colab)

---

### Algorithm 2 \_fit\_tree(node, max\_depth, min\_leaf\_instance)

```
1: if node.depth ≤ max_depth and node.X.shape[0] > min_leaf_instance and  
   cost(node.y[node.data_indices]) > 0 then  
2:   split_cost, split_feat, split_value = greedy_test(node)  
3:   node.split_feature = split_feat  
4:   node.split_value = split_value  
5:   test_logicals = node.X[node.data_indices, split_feat] <= split_value  
6:   left_node = Node(data_indices=node.data_indices[test_logical], parent=node)  
7:   right_node = Node(data_indices=node.data_indices[!test_logical], parent=node)  
8:   _fit_tree(left_node, max_depth, min_leaf_instance)  
9:   _fit_tree(right_node, max_depth, min_leaf_instance)  
10:  node.left = left_node  
11:  node.right = right_node  
12: end if
```

---

### Algorithm 3 fit(tree, max\_depth, min\_leaf\_instance=1)

---

```
1: _fit_tree(tree.root, max_depth, min_leaf_instance)
```

---

# Outline

Objectives

Decision Tree (DT)

Cost functions

DT training algorithm

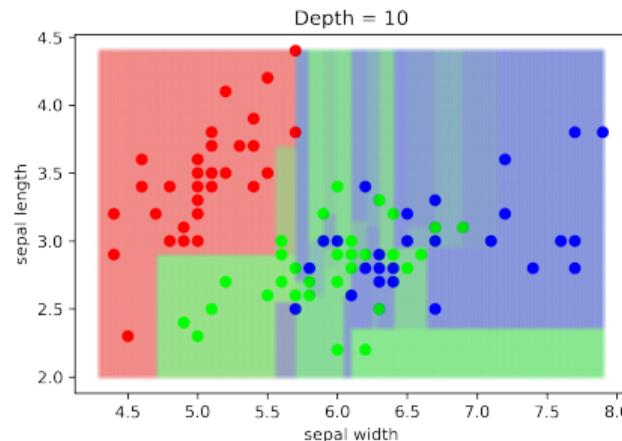
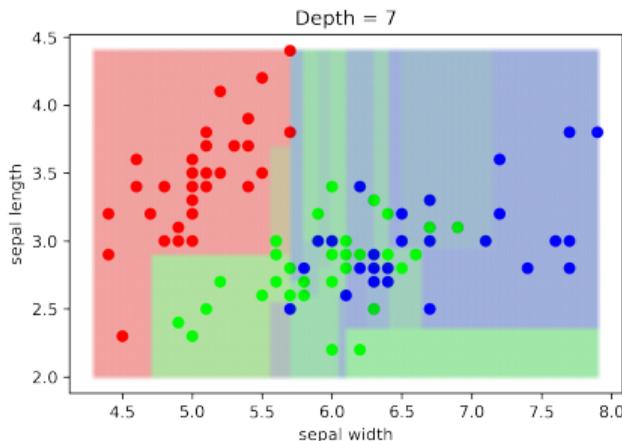
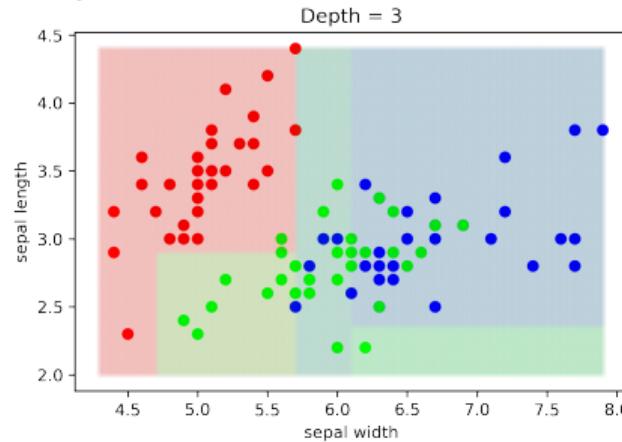
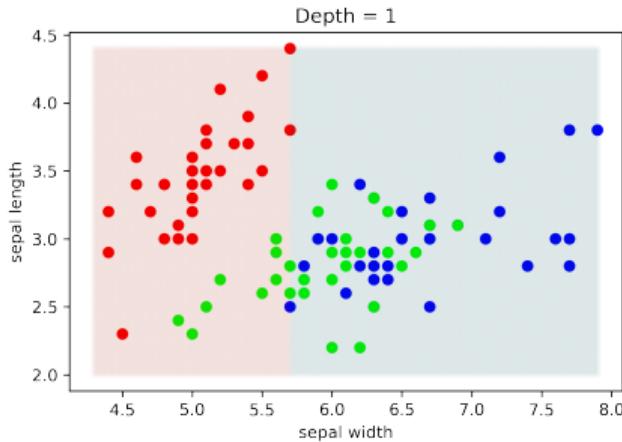
Choice of tree depth

The issue of overfitting in DT

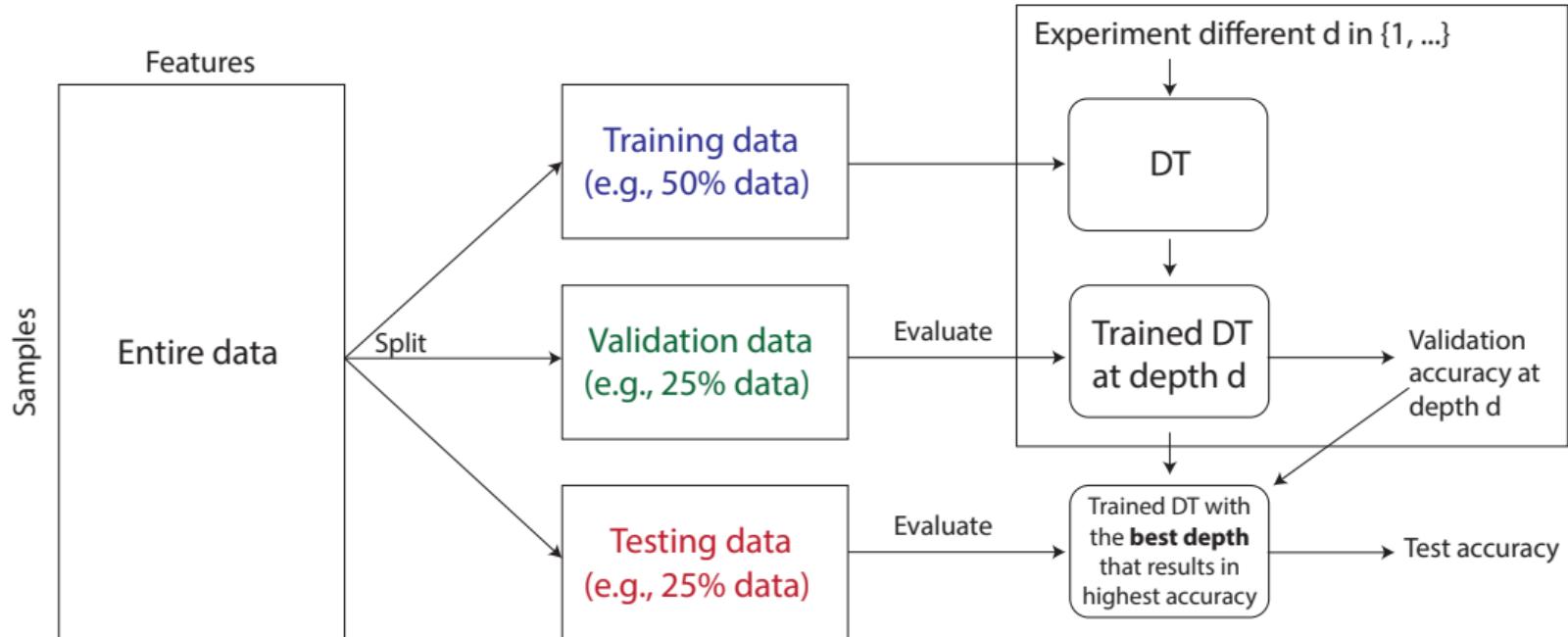
Feature importance

Summary

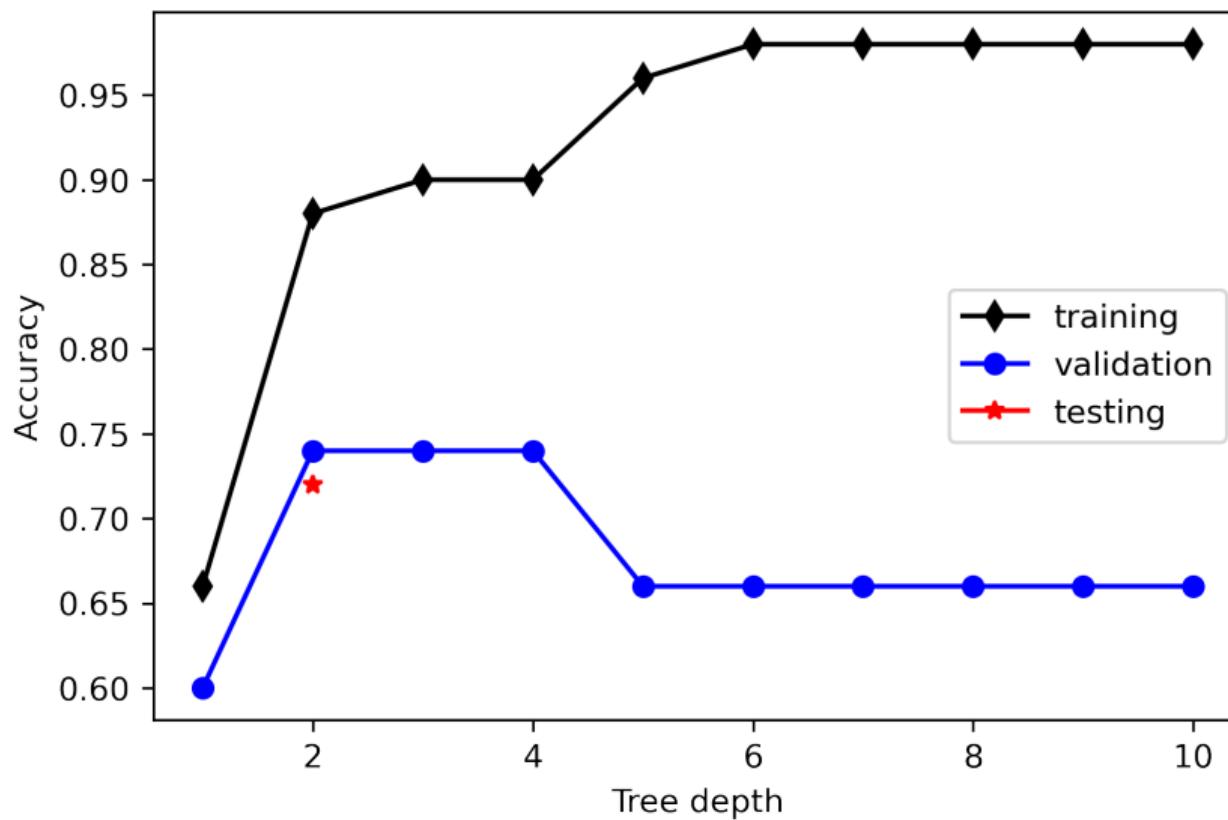
# How to choose maximum tree depth (the hyperparameter for DT)?



# Choose tree depth using a validation set (same as choosing K in KNN)



## Choose optimal tree depth using a validation set (Colab)



# Outline

Objectives

Decision Tree (DT)

Cost functions

DT training algorithm

Choice of tree depth

The issue of overfitting in DT

Feature importance

Summary

## Decision tree using Scikit-Learn

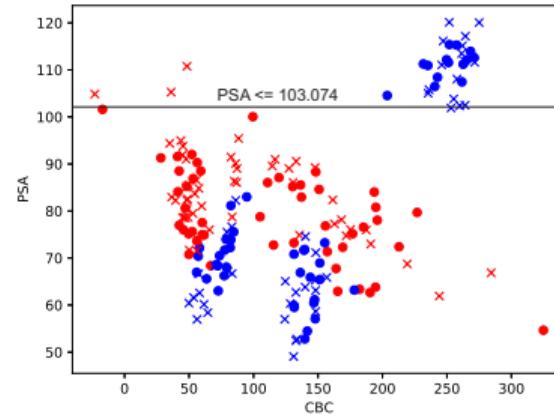
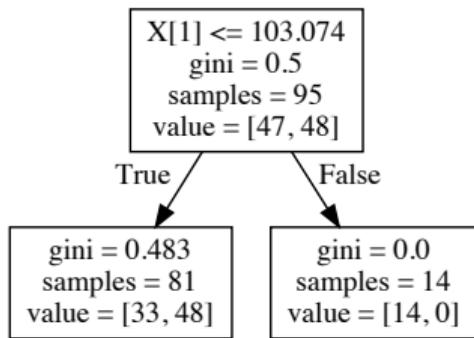
---

```
1 from sklearn import model_selection,tree
2 import graphviz
3 for depth in range(1,6):
4     dt = tree.DecisionTreeClassifier(max_depth=depth)
5     dt.fit(X_train, y_train)
6     p_train = dt.predict(X_train)
7     p_test = dt.predict(X_test)
8     #plot tree
9     dot_data = tree.export_graphviz(dt, out_file=None)
10    graph = graphviz.Source(dot_data)
11    graph.render("prostate_tree_depth_"+str(depth))
```

---

Note: Line 9-11 requires installing graphviz: pip install graphviz

## Decision tree (max\_depth = 1)



Training data:

	PN	PP
AN	14	33
AP	0	48

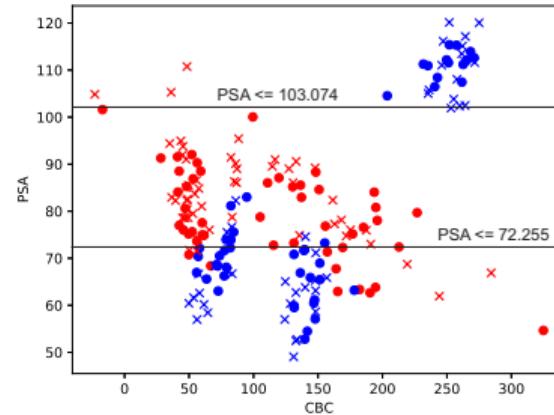
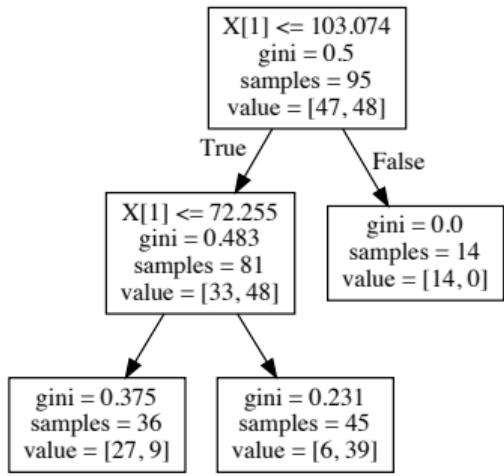
$$\begin{aligned} \text{TPR} &= 48/(48+0) = 1.0 \\ \text{FPR} &= 33/(33+14) = 0.7 \end{aligned}$$

Test data:

	PN	PP
AN	13	30
AP	3	49

$$\begin{aligned} \text{TPR} &= 49/(49+3) = 0.94 \\ \text{FPR} &= 30/(30+13) = 0.7 \end{aligned}$$

## Decision tree (max\_depth = 2)



Training data:

	PN	PP
AN	41	6
AP	9	39

$$TPR = 39/(39+9) = 0.81$$

$$FPR = 6/(6+41) = 0.13$$

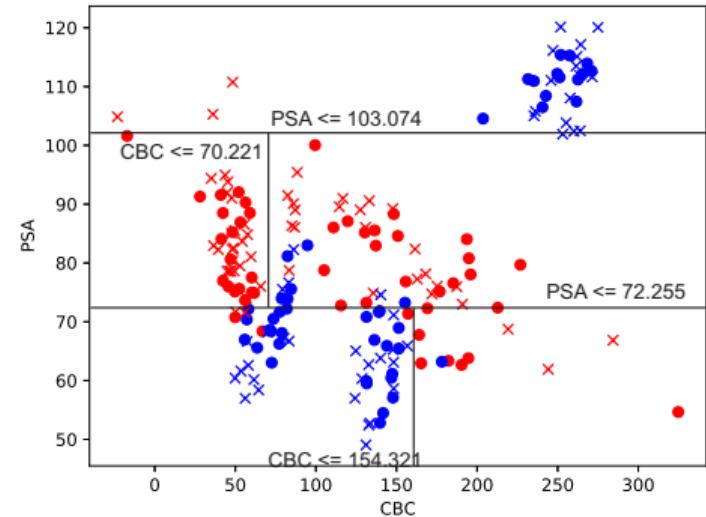
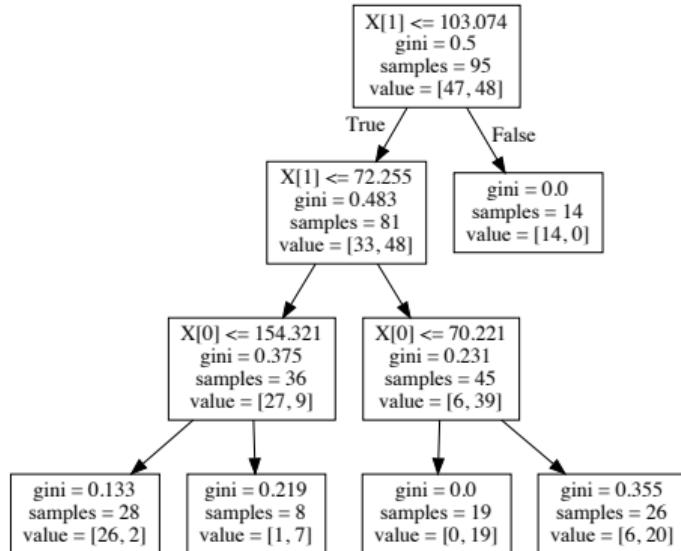
Test data:

	PN	PP
AN	36	7
AP	8	44

$$TPR = 44/(44+8) = 0.85$$

$$FPR = 7/(7+36) = 0.16$$

# Decision tree (max\_depth = 3)



	PN	PP
Training data:		
AN	40	7
AP	2	46

$$TPR = 46/(46+2) = 0.96$$

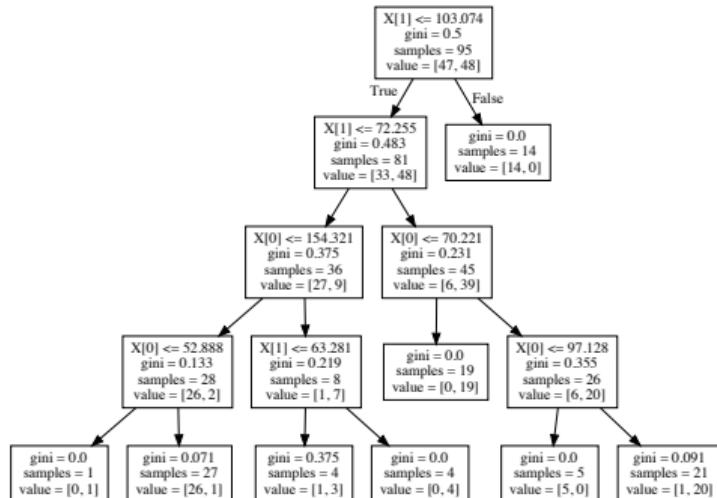
$$FPR = 7/(7+40) = 0.15$$

	PN	PP
Test data:		
AN	35	8
AP	5	47

$$TPR = 47/(47+5) = 0.9$$

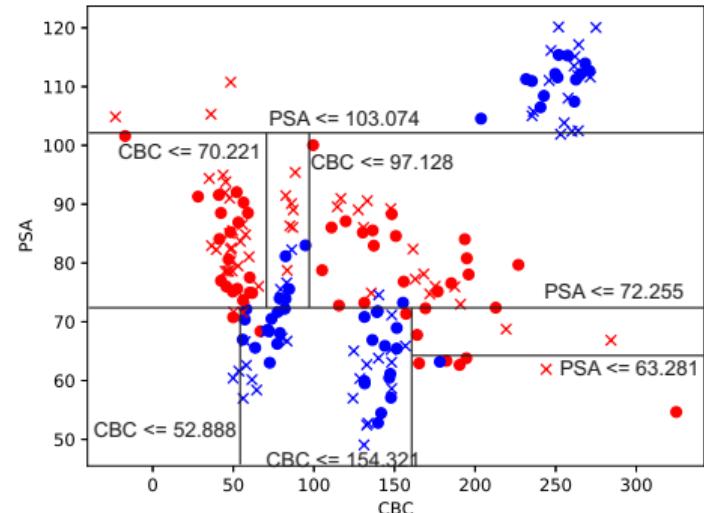
$$FPR = 8/(8+35) = 0.19$$

# Decision tree (max\_depth = 4) - overfitting occurs



	PN	PP
Training data:	AN 45	2
	AP 1	47

$$\begin{aligned} \text{TPR} &= 47/(47+1) = 0.98 \\ \text{FPR} &= 2/(2+45) = 0.04 \end{aligned}$$



	PN	PP
Test data:	AN 37	6
	AP 11	41

$$\begin{aligned} \text{TPR} &= 41/(41+11) = 0.79 \\ \text{FPR} &= 6/(6+37) = 0.14 \end{aligned}$$

## Decision tree (max\_depth = 5 & 6) - more overfitting

Tree depth = 5

Training data:

	PN	PP
AN	46	1
AP	1	47

$$\text{TPR} = 47/(47+1) = 0.98$$

$$\text{FPR} = 1/(1+46) = 0.02$$

Test data:

	PN	PP
AN	37	6
AP	11	41

$$\text{TPR} = 41/(41+11) = 0.79$$

$$\text{FPR} = 6/(6+37) = 0.14$$

Tree depth = 6

Training data:

	PN	PP
AN	47	0
AP	0	48

$$\text{TPR} = 48/(48+0) = 1.0$$

$$\text{FPR} = 0/(0+47) = 0.0$$

Test data:

	PN	PP
AN	37	6
AP	11	41

$$\text{TPR} = 41/(41+11) = 0.79$$

$$\text{FPR} = 6/(6+37) = 0.14$$

# Outline

Objectives

Decision Tree (DT)

Cost functions

DT training algorithm

Choice of tree depth

The issue of overfitting in DT

Feature importance

Summary

## Feature importance

- Reduction of the cost (e.g., Gini index) due to the split at node  $j \in \{1, \dots, J\}$ :

$$\Delta g_j = \frac{N_j}{N} \times g_j - \frac{N_j^{(\text{left})}}{N} g_j^{(\text{left})} - \frac{N_j^{(\text{right})}}{N} g_j^{(\text{right})}$$

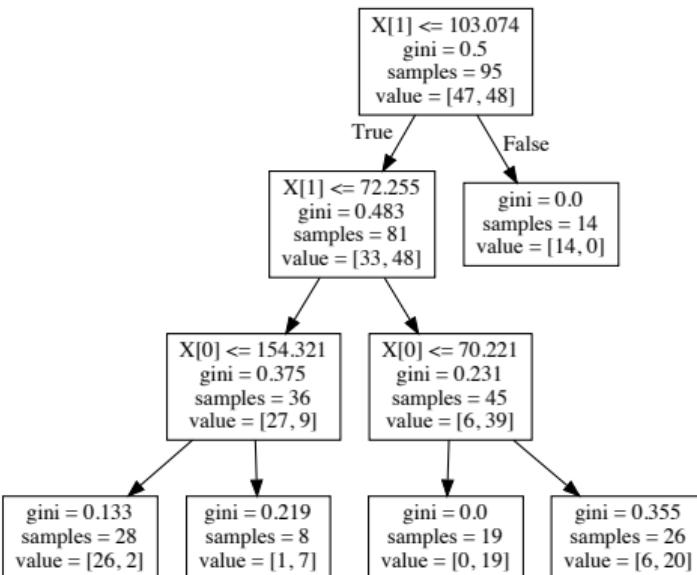
where  $\frac{N_j}{N}$  is the fraction of samples in node  $j$  and  $g_j$  is the cost at node  $j$ ;  $\frac{N_j^{(\text{left})}}{N}$  and  $\frac{N_j^{(\text{right})}}{N}$  are fractions of samples in the left and right child of node  $j$ ;  $g_j^{(\text{left})}$  and  $g_j^{(\text{right})}$  are cost in the left and right child of node  $j$ .

- The feature importance of feature  $d \in \{1, \dots, D\}$  is the total reduction over all the  $J$  non-leaf nodes, where feature  $d$  is used:

$$r_d = \sum_{j=1}^J \Delta g_j \mathbb{I}[v_j = d]$$

where  $\mathbb{I}[v_j = d]$  indicates feature  $d$  is used in node  $j$ . The formula for computing feature importance as one of the optional tasks in Assignment 1 has been updated.

## Example: feature importance of $x_0$ in DT at depth 3



- Gini reduction at node 2 (level 3 left node):

$$\Delta g_2 = \frac{36}{95}0.375 - \frac{28}{95}0.133 - \frac{8}{95}0.219 = 0.0844$$

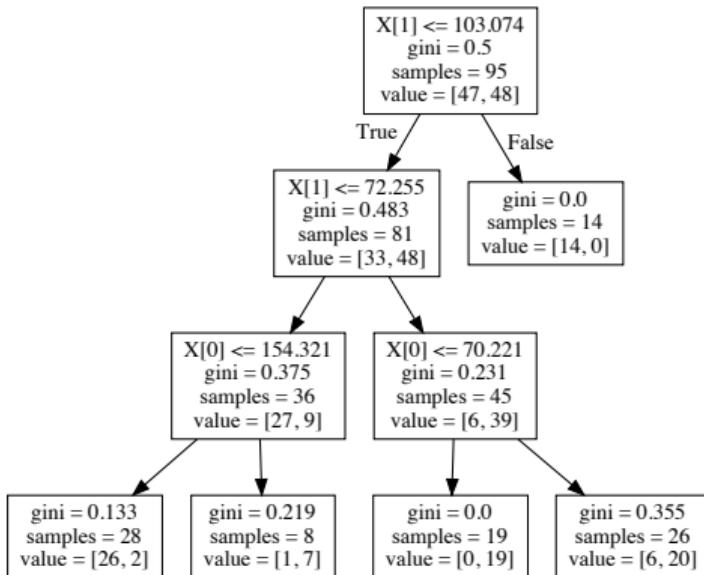
- Gini reduction at node 3 (level 3 right node):

$$\Delta g_3 = \frac{45}{95}0.231 - \frac{19}{95}0 - \frac{26}{95}0.355 = 0.0123$$

- Feature importance of  $x_0$ :

$$r_0 = \Delta g_2 + \Delta g_3 = 0.0844 + 0.0123 = \mathbf{0.0967}$$

## Example: feature importance of $x_1$ in DT at depth 3



- Gini reduction at root due to  $x_1$ :

$$\Delta g_{root} = \frac{95}{95}0.5 - \frac{81}{95}0.483 - \frac{14}{95}0 = 0.0881$$

- Gini reduction at node 1 (i.e., left child of root) also due to  $x_1$ :

$$\Delta g_1 = \frac{81}{95}0.483 - \frac{36}{95}0.375 - \frac{45}{95}0.231 = 0.160$$

- Feature importance of  $x_1$ :

$$r_1 = \Delta g_{root} + \Delta g_1 = 0.0881 + 0.160 = \mathbf{0.2481}$$

Therefore,  $x_1$  has more than 2.5 times higher feature importance score than  $x_0$  ( $0.2481/0.0967 = 2.566$ ).

# Outline

Objectives

Decision Tree (DT)

Cost functions

DT training algorithm

Choice of tree depth

The issue of overfitting in DT

Feature importance

Summary

## Summary of DT

- Costs: Misclassification rate, Entropy, Gini index
- DT is a greedy and recursive algorithm that finds the best feature and the best threshold at each tree node to minimize the loss (e.g., Gini index) within the node.
- Compared to KNN, robust to scaling and noise, fast predictions, more interpretable
- Overfitting occurs when the model gets too complicated (i.e., high tree depth in the case of DT) and fits extremely well on the training data but generalizes poorly to the test data.
- Feature importance score derived from a trained DT can be used to rank features.
- DT is the building block of common ensemble learning methods such as Random Forest and XGBoost (Module 6)