המחלקה להנדסת תעשייה
הפקולטה להנדסה ע"ש איבי ואלדר פליישמן
אוניברסיטת תל אביב

מבוא ללמידת מכונה

Introduction to Machine Learning

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

תרגול 9

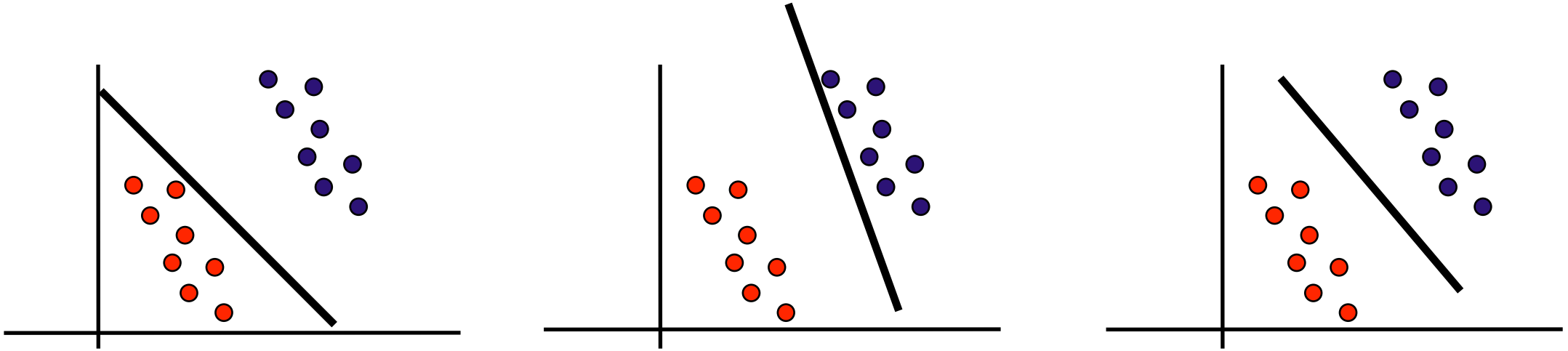
Kernel SVM & Decision Trees

אילן וסילבסקי

תשפ"ב 2022

SVM – Recap from lecture..

If there exists one solution (linear classifier that separate the training data) , there exist many.

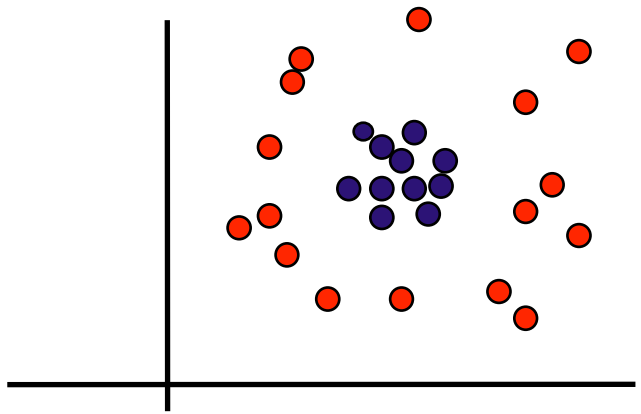


So , Which one should we choose?

Intuitively one that’s farther away from the points.

SVM-Kernels

But what can we do if the data behave like this?



data can’t be separated using a straight line…

We use **kernel functions** in this case that help transform the data into another dimension that has a clear dividing margin between the two classes. Kernel functions help transform non-linear spaces into linear spaces.

SVM-Kernels

How does Kernels works?

The kernel functions return the inner product between two points in a suitable feature space. Thus by defining a notion of similarity, with little computational cost even in very high-dimensional spaces.

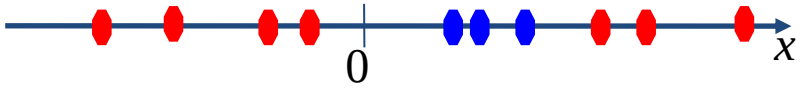
$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{x}')$$

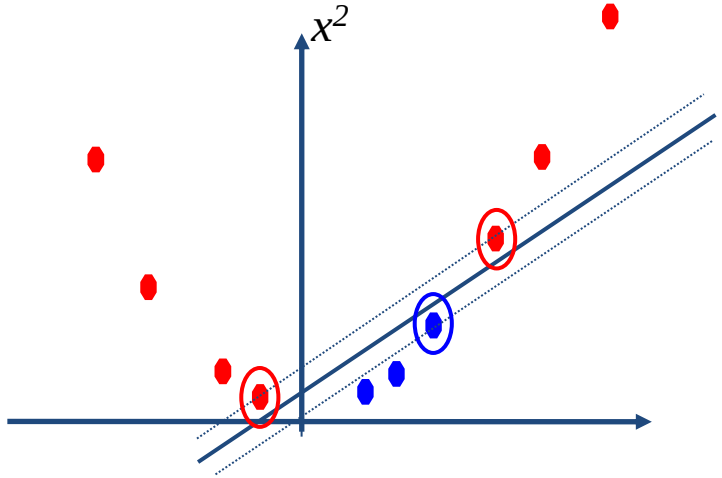
kernel trick- perform inner product to a higher dimensional space without have the actual points in a higher dimensional space In this way, we get our solution from a higher dimensional space without even visiting it (little computational cost).

Popular kernels are: Polynomial Kernel, Gaussian Kernel, Radial Basis Function (RBF)…

SVM - Kernels

mapping inputs into high-dimensional feature spaces.



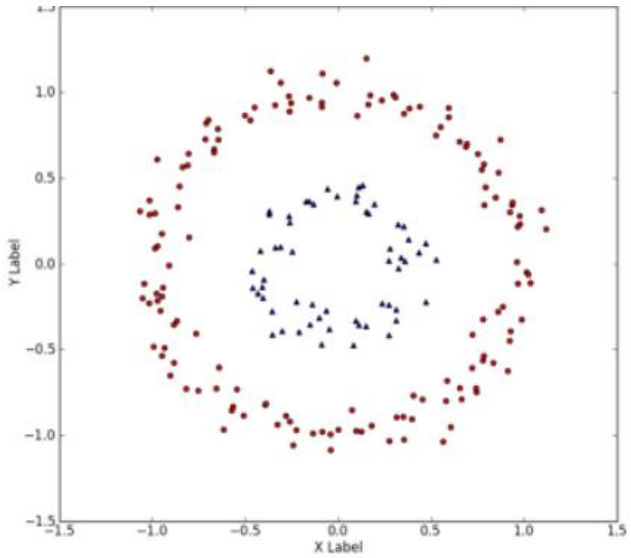


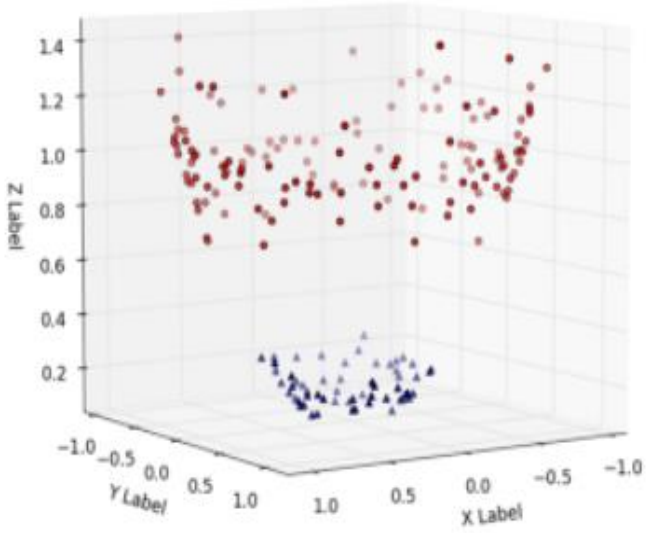
$$\Phi(x) = (x_1, x_1^2)$$

Kernels take low dimensional input space and convert them into high dimensional input space. It converts non-separable classes into the separable one, and enable to find a linear hyperplane.

SVM - Kernels

mapping inputs into high-dimensional feature spaces.





Kernels take low dimensional input space and convert them into high dimensional input space. It converts non-separable classes into the separable one, and enable to find a linear hyperplane.

SVM-Kernels

Key ideas to remember:

- We can find the coefficients using only inner products (dual)

$$\max_{\alpha} \quad \sum_i \alpha_i - 0.5 \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{K}(\mathbf{x}_i, \boldsymbol{x_j})$$
$$s.t. \quad \alpha_i \geq 0, \sum_i \alpha_i y_i = 0$$

- We never compute the hyperplane w explicitly.

$$\mathbf{w} = \sum_i \alpha_i \, y_i \phi(\mathbf{x}_i)$$

- We classify by calculating dot products with support vectors

$$\mathbf{w} \cdot \phi(\mathbf{x}) = \sum_i \alpha_i y_i \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$$

SVM in Scikit-Learn

```
from sklearn.svm import SVC

clf = SVC  ( C=1.0,     # regularization parameter (Must be strictly positive). The higher it gets, Regularization strength reduces – prioritize making few misclassification (low generalization).

                 kernel='rbf',              # Kernel type, {linear', 'poly', 'rbf', 'sigmoid', 'precomputed'},

                degree=3,                   # Degree of the polynomial kernel function ('poly').

                gamma=0.01,     # Kernel coefficient for 'rbf', 'poly' and 'sigmoid'. {'scale', 'auto'} or float. The higher it gets, Regularization strength reduces - prioritize making few misclassification (low generalization).
                )
```

SVM Pros and Cons

- **Pros**:
  - It has the ability to handle large feature spaces.
  - SVM's are very good when we have no idea about our data.
  - The kernel trick is real strength of SVM. With an appropriate kernel function, we can solve any complex problem
- **Cons**:
  - It is sensitive to noise.
  - Difficult to understand and interpret the final model, variable weights and individual impact.
  - The SVM hyper parameters are Cost -C and gamma. It is not that easy to fine-tune these hyper-parameters. It is hard to visualize their impact

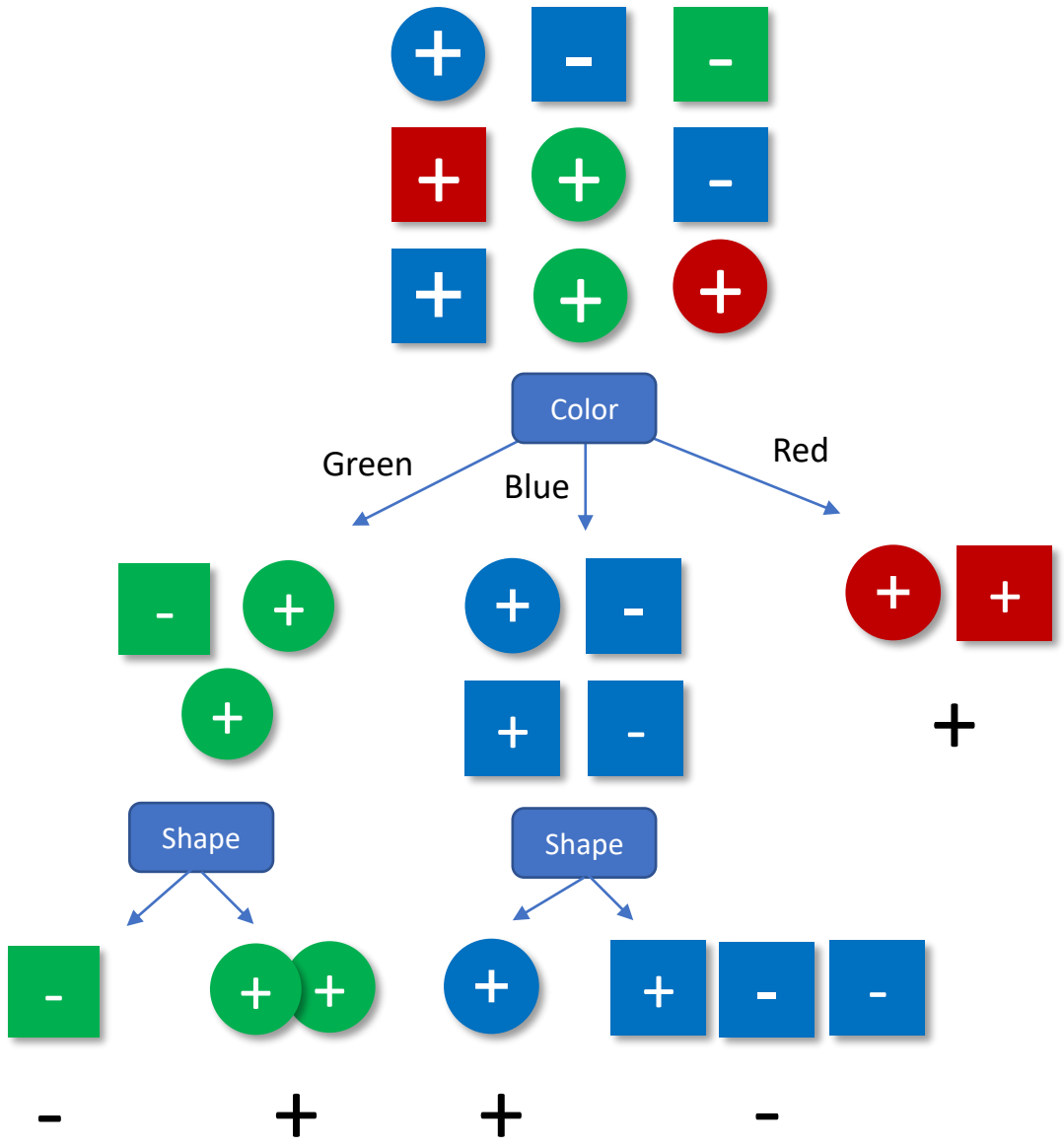
Decision Trees

Recap - Decision Trees

Definition

- A **Decision Tree** is a predictor, $h(x) = \hat{y}$, that predicts the label associated with an instance x by traveling from a **root node** of a tree to a **leaf**.

- At each node on the root-to-leaf path, the successor child is chosen on the basis of a splitting of the input space.

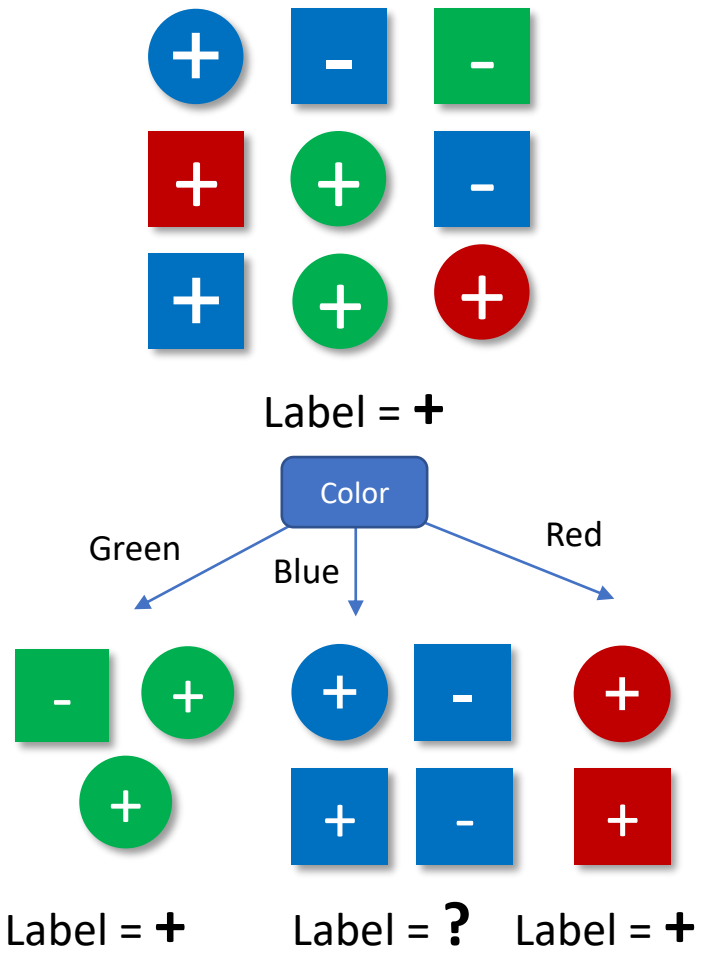


Recap - Decision Trees

The Learning Algorithm

A general framework for growing a decision tree is as follows:

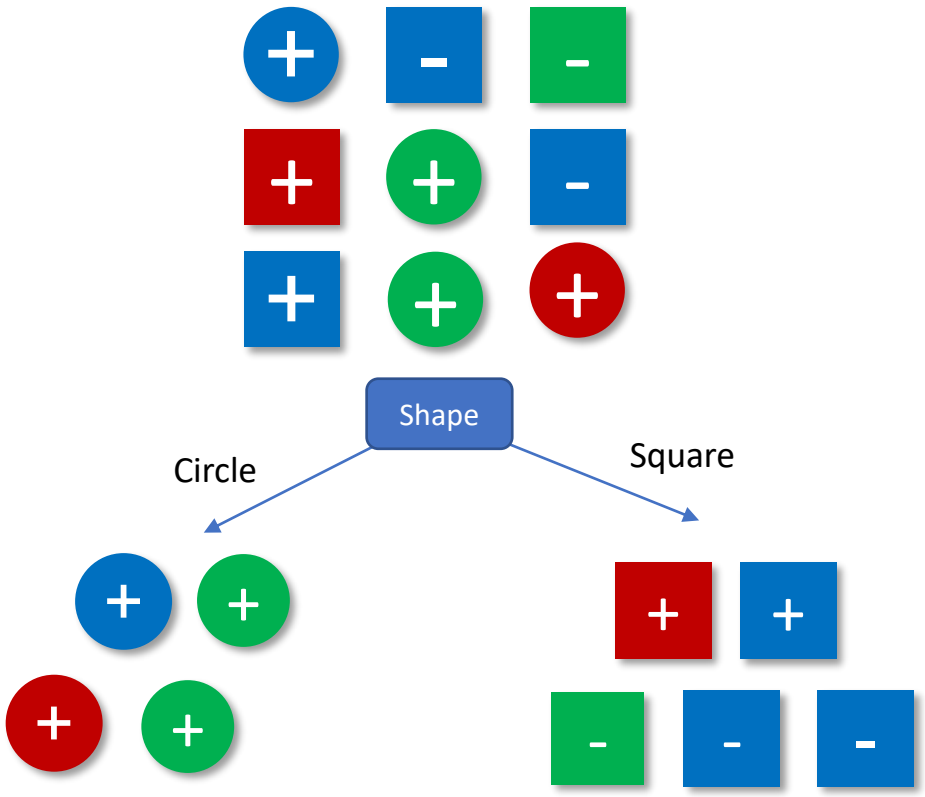
- We start with a tree with a single leaf (the root) and assign this leaf a label according to a majority vote among all labels over the training set.
- We now perform a series of iterations:
  - On each iteration, we examine the effect of splitting a single leaf. We define some “purity” measure that quantifies the improvement due to this split.
  - Then, among all possible splits, we either choose the one that maximizes the impurity reduction and perform it, or choose not to split the leaf at all.



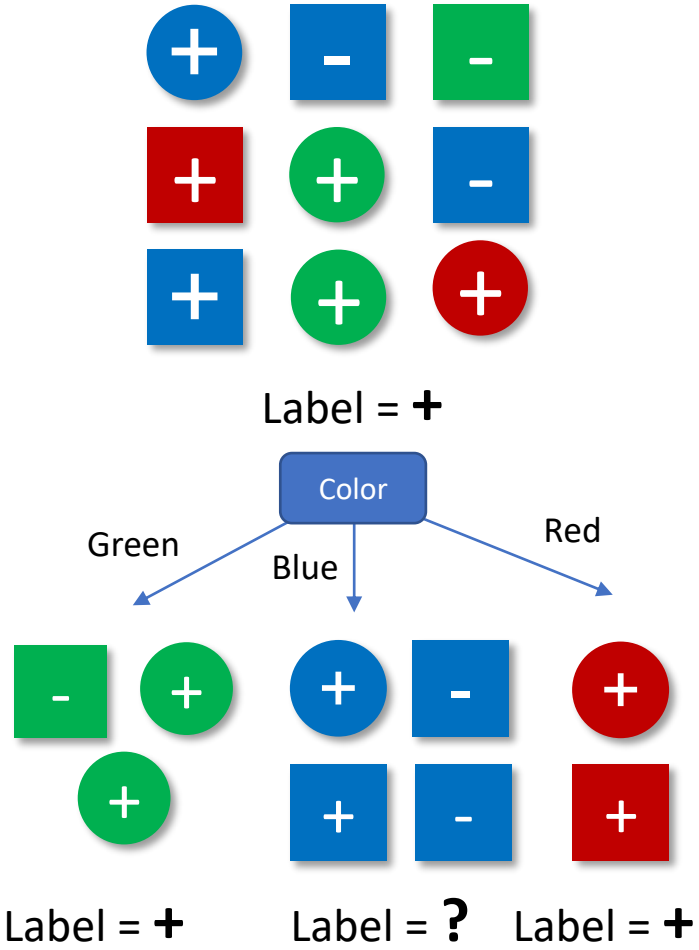
Recap - Decision Trees

The Learning Algorithm

Split Option 1:



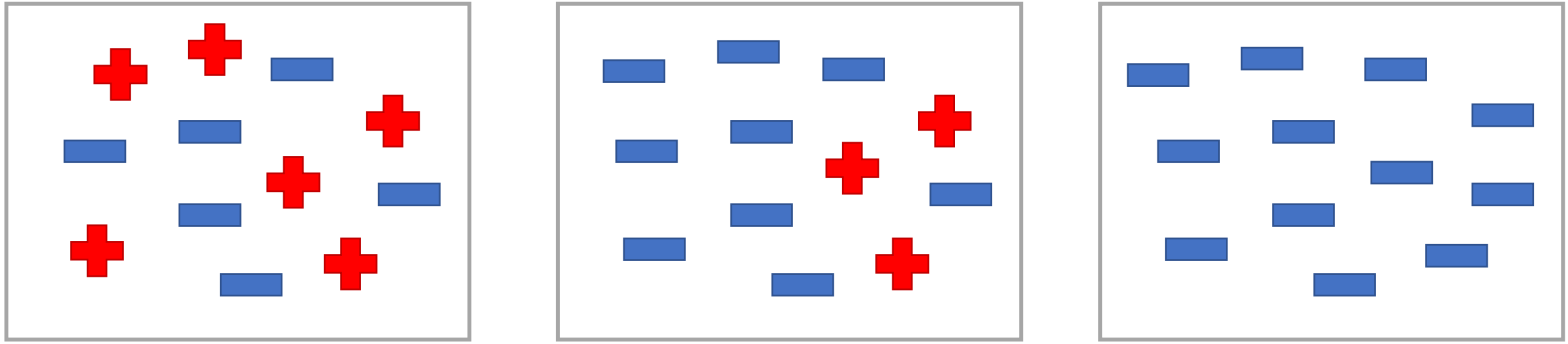
Split Option 2:



Recap - Decision Trees

Impurity Reduction (1)

Leaf Nodes

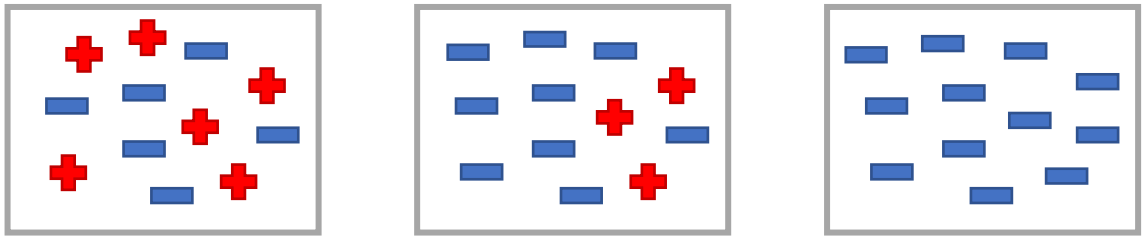


Maximum Impurity → Minimum Impurity

Recap - Decision Trees

Impurity Reduction (2)

Impurity Measures



We will use this one today. →

Entropy impurity $$-\sum_j P(w_j) log_2\left(P(w_j)\right)$$

Gini impurity $$\sum_{i \neq j} P(w_i)P(w_j) = \frac{1}{2}\left[1 - \sum_j P^2(w_j)\right]$$

Misclassification impurity $$1 - \max_j P(w_j)$$

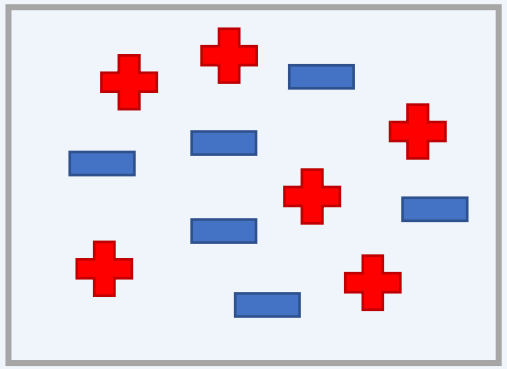
Recap - Decision Trees

Impurity Reduction (3)

$$Entropy\ Impurity = i(N) = -\sum_j P(w_j) log_2\left(P(w_j)\right)$$

$w_1 =$ ▬

$w_2 =$ ✚

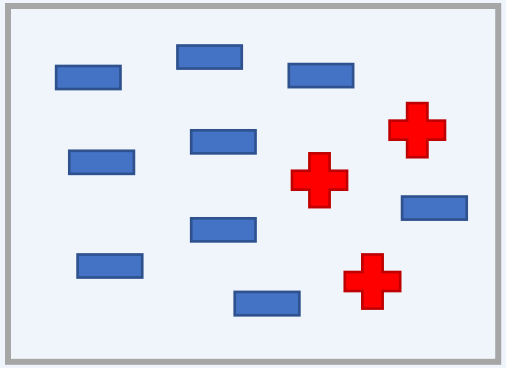


Maximum Impurity

$$P(w_1) = \frac{6}{12} = 0.5 \ , \ P(w_2) = \frac{6}{12} = 0.5$$

$$i(N) = -(0.5 \cdot \log(0.5) + 0.5 \cdot \log(0.5))$$

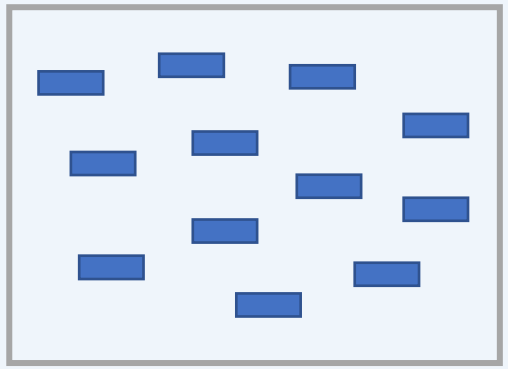
$$= 1$$



$$P(w_1) = \frac{9}{12} = 0.75 \ , \ P(w_2) = \frac{3}{12} = 0.25$$

$$i(N) = -(0.75 \cdot \log(0.75) + 0.25 \cdot \log(0.25))$$

$$= 0.81$$



Minimum Impurity

$$P(w_1) = \frac{12}{12} = 1 \ , \ P(w_2) = \frac{0}{12} = 0$$

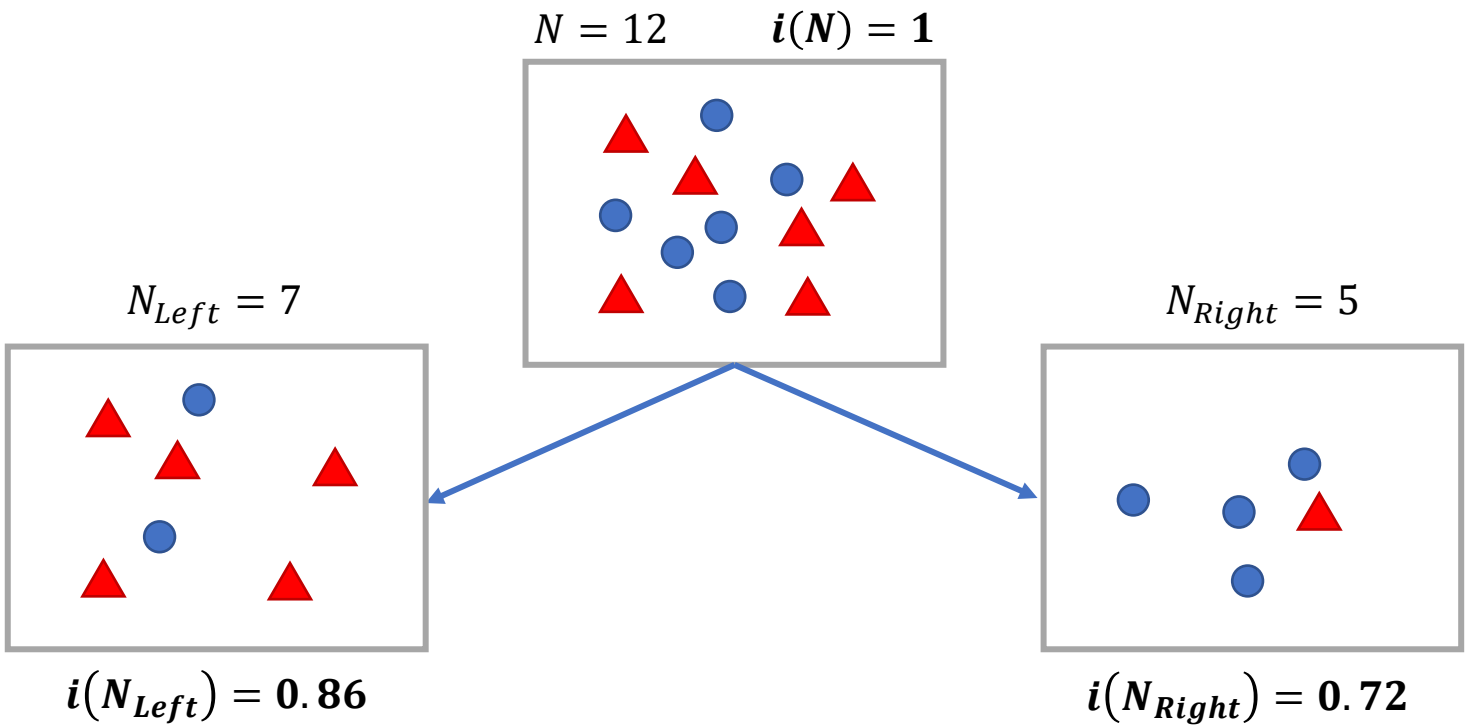
$$i(N) = -(1 \cdot \log(1) + 0 \cdot \log(0))$$

$$= 0$$

Recap - Decision Trees

Impurity Reduction (3)

$$Purity\ Gain = \Delta i(N) = i(N) - \left[\frac{N_{Left}}{N} i\left(N_{Left}\right) + \frac{N_{Right}}{N} i\left(N_{Right}\right)\right]$$



$w_1 =$ ●

$w_2 =$ ▲

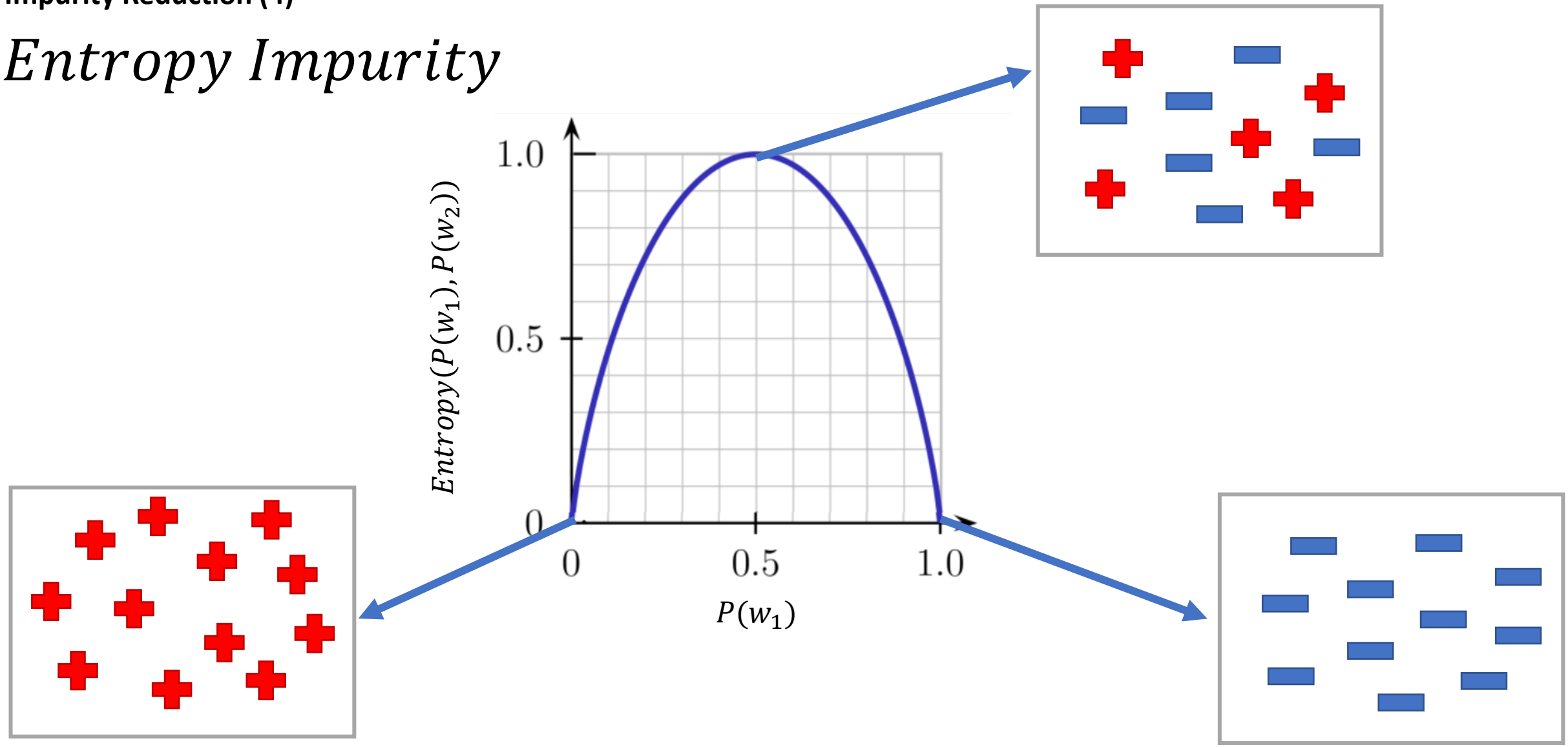
$$\Delta \boldsymbol{i}(\boldsymbol{N}) = 1 - \left[\frac{7}{12} \cdot 0.86 + \frac{5}{12} \cdot 0.72\right] = 0.20$$

$$i(N) = -\sum_j P(w_j) log_2\left(P(w_j)\right)$$

Recap - Decision Trees

Impurity Reduction (4)

Entropy Impurity



Decision Trees – Toy Example

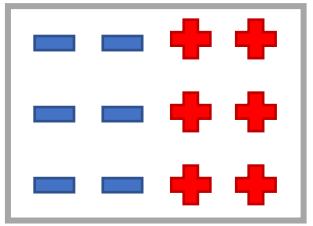
Heart Disease – UCI Dataset

https://www.kaggle.com/ronitf/heart-disease-uci

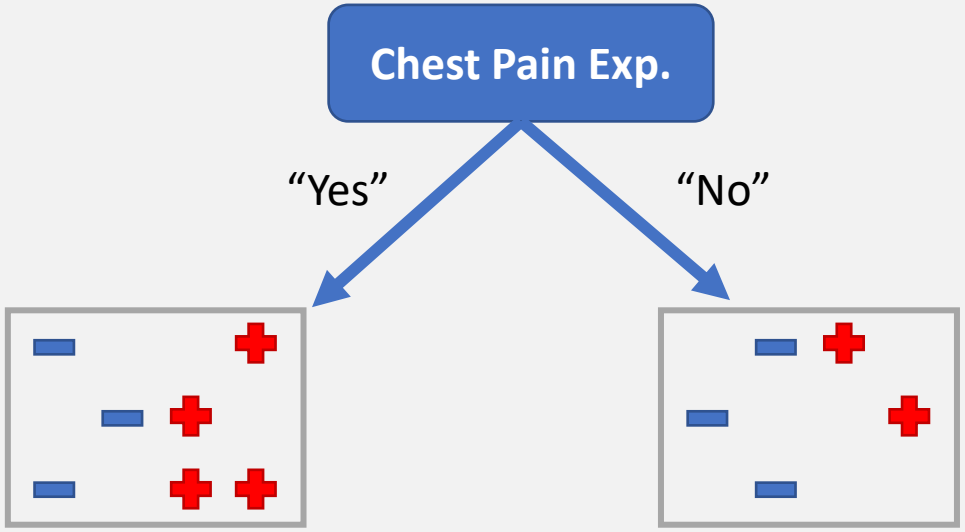
Chest Pain Exp.	Resting Blood Pressure	Cholesterol	Target (y)
Yes	160	273	0
No	120	219	0
Yes	140	335	0
No	128	216	0
Yes	125	254	0
No	160	203	0
No	130	269	1
Yes	120	215	1
Yes	134	271	1
Yes	134	301	1
No	142	302	1
Yes	130	284	1

Decision Trees – Toy Example

Categorical / Boolean Feature



$N = 12$
$Values = [6,6]$
$i(N) = 1.0$



$Values = [3,4]$
$i(N_{Left}) = 0.98$

$Values = [3,2]$
$i(N_{Right}) = 0.97$

$$\boldsymbol{Purity\ Gain} = \Delta i(N) = i(N) - \left[\frac{N_{Left}}{N} i(N_{Left}) + \frac{N_{Right}}{N} i(N_{Right})\right]$$

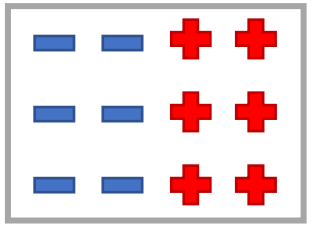
Heart Disease – UCI Dataset

https://www.kaggle.com/ronitf/heart-disease-uci

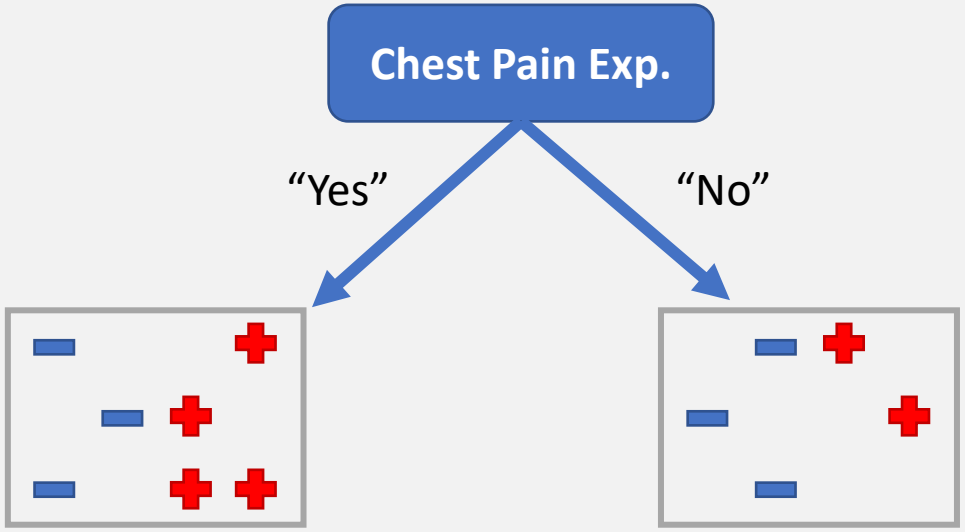
Chest Pain Exp.	Resting Blood Pressure	Cholesterol	Target (y)
Yes	160	273	0
No	120	219	0
Yes	140	335	0
No	128	216	0
Yes	125	254	0
No	160	203	0
No	130	269	1
Yes	120	215	1
Yes	134	271	1
Yes	134	301	1
No	142	302	1
Yes	130	284	1

Decision Trees – Toy Example

Categorical / Boolean Feature



$N = 12$
$Values = [6,6]$
$i(N) = 1.0$



$Values = [3,4]$
$i(N_{Left}) = 0.98$

$Values = [3,2]$
$i(N_{Right}) = 0.97$

$$\Delta \boldsymbol{i}(\boldsymbol{N}) = 1 - \left[\frac{7}{12} \cdot 0.98 + \frac{5}{12} \cdot 0.97\right] = 0.02$$

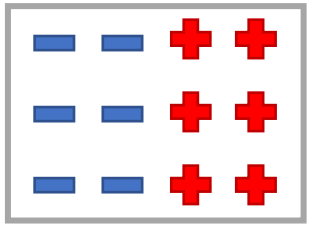
Heart Disease – UCI Dataset

https://www.kaggle.com/ronitf/heart-disease-uci

Chest Pain Exp.	Resting Blood Pressure	Cholesterol	Target (y)
Yes	160	273	0
No	120	219	0
Yes	140	335	0
No	128	216	0
Yes	125	254	0
No	160	203	0
No	130	269	1
Yes	120	215	1
Yes	134	271	1
Yes	134	301	1
No	142	302	1
Yes	130	284	1

Decision Trees – Toy Example

Numerical Feature



$N = 12$
$Values = [6,6]$
$i(N) = 1.0$

Resting Blood Pressure

1. Sort the feature by its values

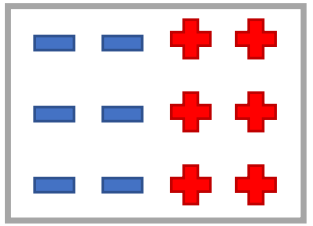
Heart Disease – UCI Dataset

https://www.kaggle.com/ronitf/heart-disease-uci

Chest Pain Exp.	Resting Blood Pressure	Cholesterol	Target (y)
Yes	160	273	0
No	120	219	0
Yes	140	335	0
No	128	216	0
Yes	125	254	0
No	160	203	0
No	130	269	1
Yes	120	215	1
Yes	134	271	1
Yes	134	301	1
No	142	302	1
Yes	130	284	1

Decision Trees – Toy Example

Numerical Feature



$N = 12$
$Values = [6,6]$
$i(N) = 1.0$

Heart Disease – UCI Dataset
https://www.kaggle.com/ronitf/heart-disease-uci

Resting Blood Pressure

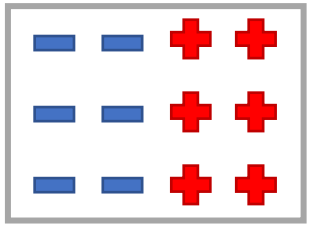
1. Sort the feature by its values
2. Calculate mean between two different values
3. Calculate Purity Gain for each average point

122.5
126.5
129.5
132
137
141
151

Resting Blood Pressure	Target (y)
120	0
120	1
125	0
128	0
130	1
130	1
134	1
134	1
140	0
142	1
160	0
160	0

Decision Trees – Toy Example

Numerical Feature



$N = 12$
$Values = [6,6]$
$i(N) = 1.0$

Heart Disease – UCI Dataset

https://www.kaggle.com/ronitf/heart-disease-uci

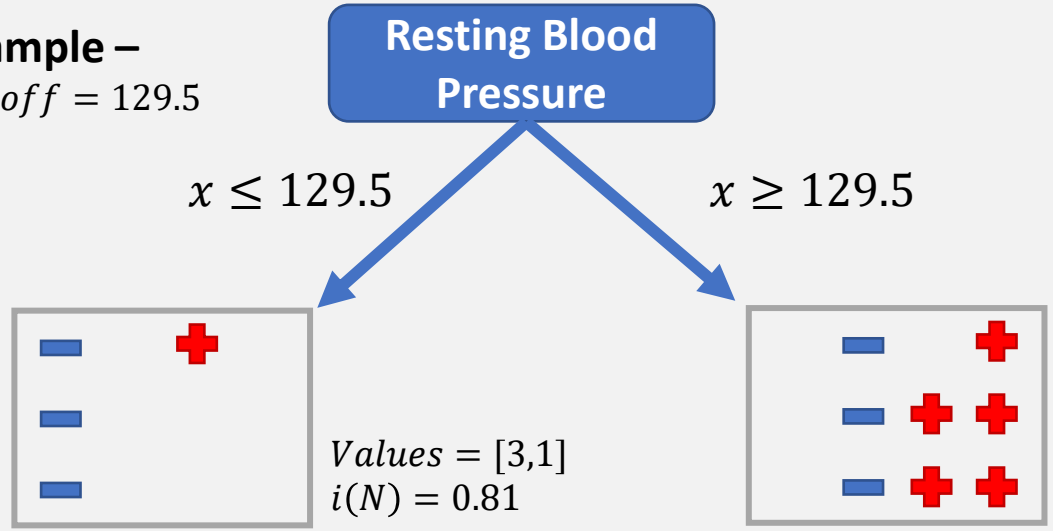
Resting Blood Pressure	Target (y)
120	0
120	1
125	0
128	0
130	1
130	1
134	1
134	1
140	0
142	1
160	0
160	0

122.5
126.5
129.5
132
137
141
151

$\Delta \boldsymbol{i}(\boldsymbol{N}) = 0.09$

3. Calculate Purity Gain for each average point

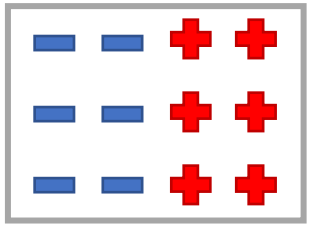
Example –
$Cutoff = 129.5$



$$\boxed{\Delta \boldsymbol{i}(\boldsymbol{N}) = 1 - \left[\frac{4}{12} \cdot 0.81 + \frac{8}{12} \cdot 0.95\right] = 0.09}$$

Decision Trees – Toy Example

Numerical Feature



$N = 12$
$Values = [6,6]$
$i(N) = 1.0$

Heart Disease – UCI Dataset

https://www.kaggle.com/ronitf/heart-disease-uci

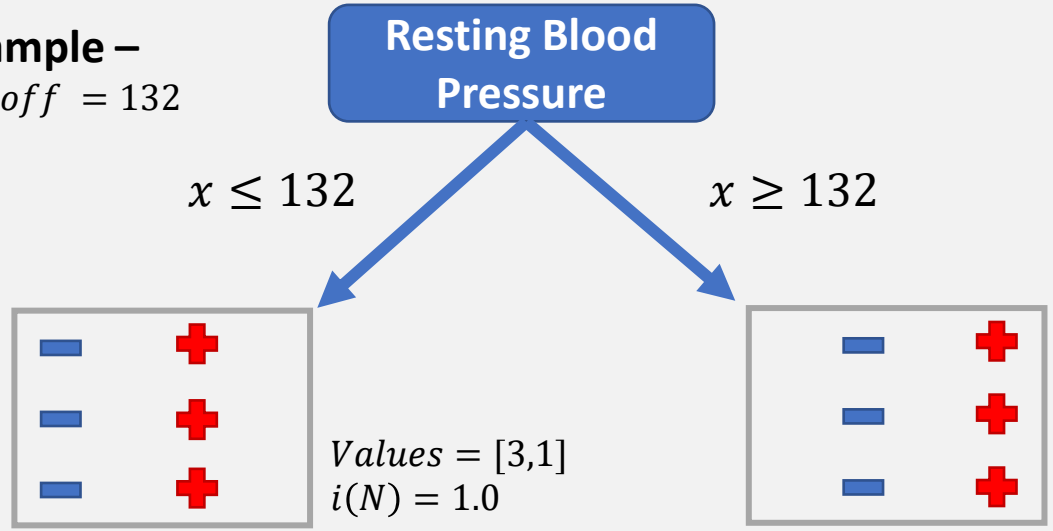
	Resting Blood Pressure	Target (y)
	120	0
122.5	120	1
126.5	125	0
129.5	128	0
	130	1
	130	1
132	134	1
137	134	1
141	140	0
151	142	1
	160	0
	160	0

$\Delta \boldsymbol{i}(\boldsymbol{N}) = 0.09$

$\Delta \boldsymbol{i}(\boldsymbol{N}) = 0$

3. Calculate Purity Gain for each average point

Example –
$Cutoff = 132$



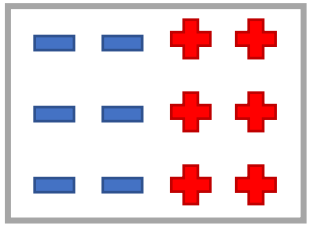
$Values = [3,1]$
$i(N) = 1.0$

$Values = [3,5]$
$i(N) = 1.0$

$$\boxed{\Delta \boldsymbol{i}(\boldsymbol{N}) = 1 - \left[\frac{6}{12} \cdot 1.0 + \frac{6}{12} \cdot 1.0\right] = 0}$$

Decision Trees – Toy Example

Numerical Feature



$N = 12$
$Values = [6,6]$
$i(N) = 1.0$

Heart Disease – UCI Dataset
https://www.kaggle.com/ronitf/heart-disease-uci

Resting Blood Pressure

1. Sort the feature by its values
2. Calculate mean between two different values
3. Calculate Purity Gain for each average point

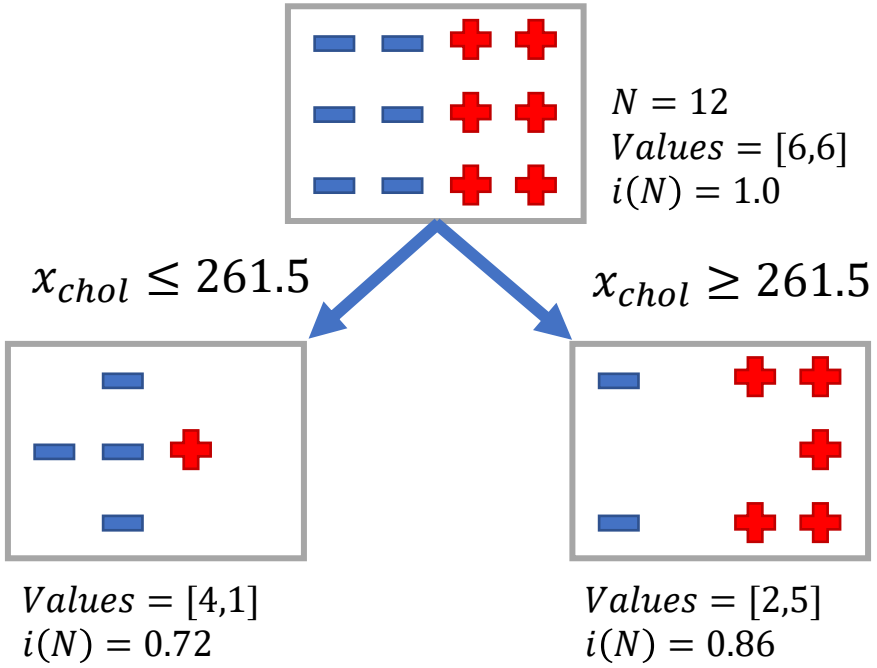
Resting Blood Pressure	Target (y)
120	0
120	1
125	0
128	0
130	1
130	1
134	1
134	1
140	0
142	1
160	0
160	0

$\Delta i(N) = 0$ ← 122.5
$\Delta i(N) = 0.01$ ← 126.5
$\Delta i(N) = 0.09$ ← 129.5
$\Delta i(N) = 0$ ← 132
$\Delta i(N) = 0.09$ ← 137
$\Delta i(N) = 0.10$ ← 141
$\Delta i(N) = 0.19$ ← 151

This is the Purity Gain Value we will compare with the other features

Decision Trees – Toy Example

After calculating the Purity Gain in all of the features, we found that the best split would be for **Cholesterol** with $Cutoff = 216.5$, and Purity Gain of $\boxed{\Delta \boldsymbol{i}(\boldsymbol{N}) = 0.20}$



Heart Disease – UCI Dataset

https://www.kaggle.com/ronitf/heart-disease-uci

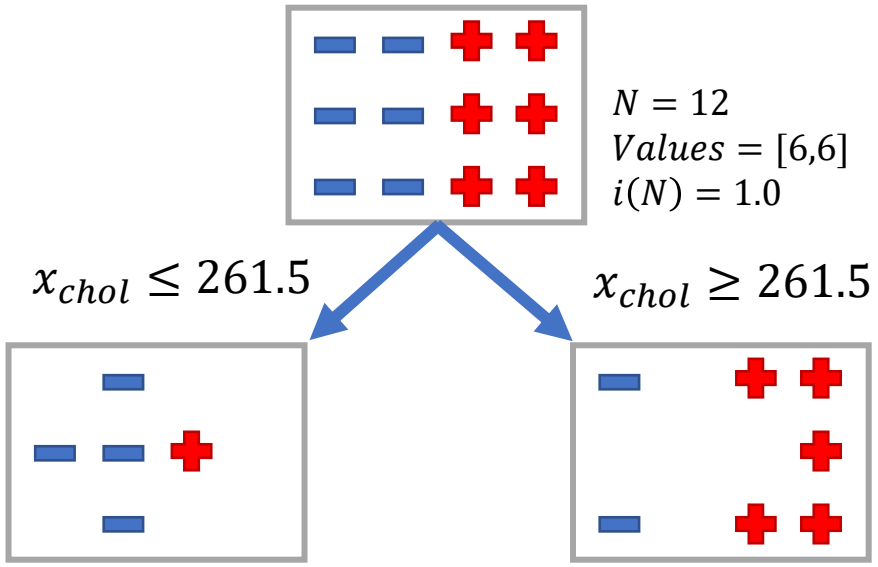
Chest Pain Exp.	Resting Blood Pressure	Cholesterol	Target (y)
Yes	160	273	0
No	120	219	0
Yes	140	335	0
No	128	216	0
Yes	125	254	0
No	160	203	0
No	130	269	1
Yes	120	215	1
Yes	134	271	1
Yes	134	301	1
No	142	302	1
Yes	130	284	1

Decision Trees – Toy Example

Increasing Depth

Each leaf in our model contains a mixture of patients with and without heart disease.

We can continue splitting the sub-trees in order to get fully-pure leaves.

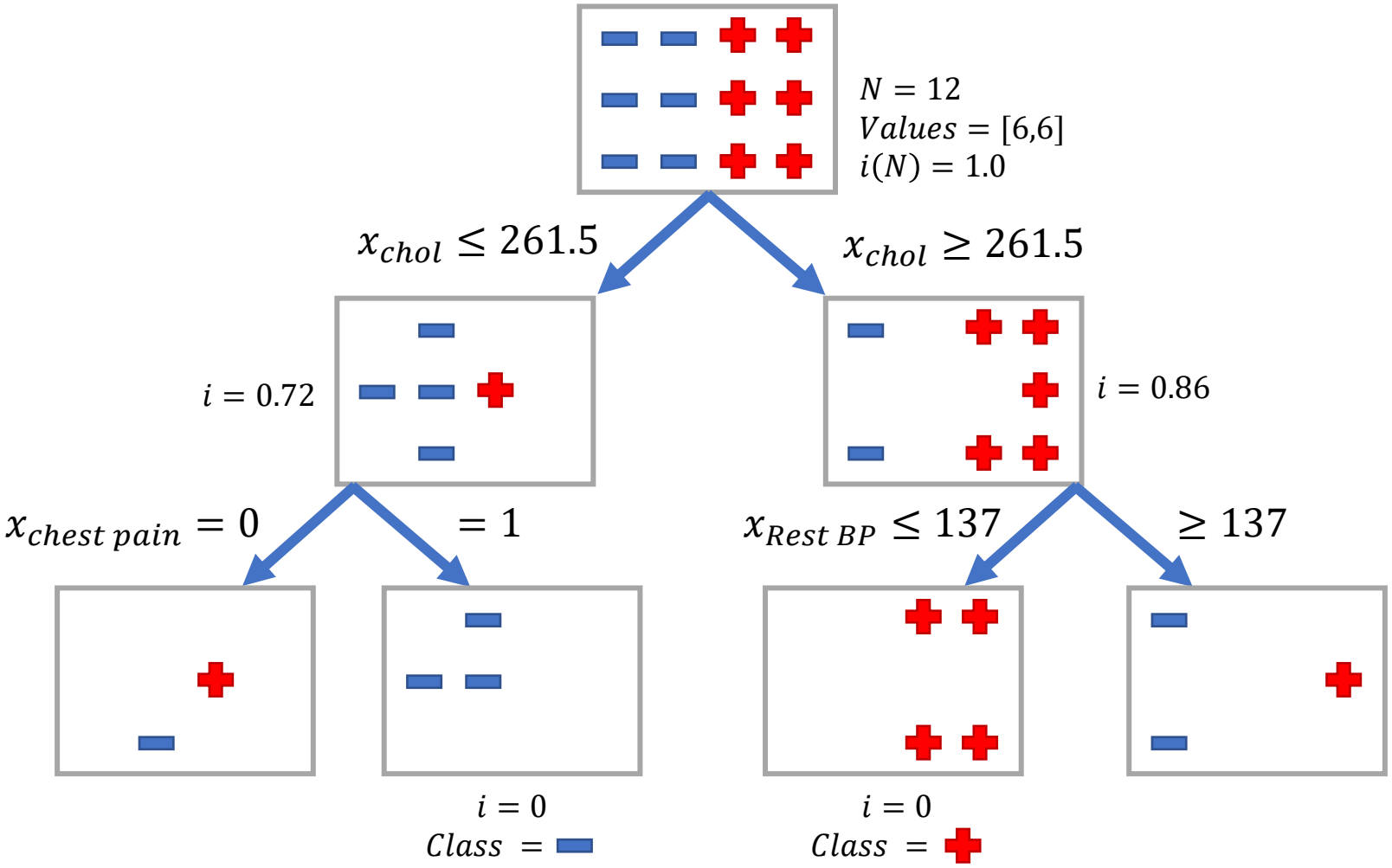


Chest Pain Exp.	Resting Blood Pressure	Cholesterol	Target (y)
No	120	219	0
No	128	216	0
No	160	203	0
Yes	125	254	0
Yes	120	215	1

Chest Pain Exp.	Resting Blood Pressure	Cholesterol	Target (y)
Yes	160	273	0
Yes	140	335	0
No	130	269	1
Yes	134	271	1
Yes	134	301	1
No	142	302	1
Yes	130	284	1

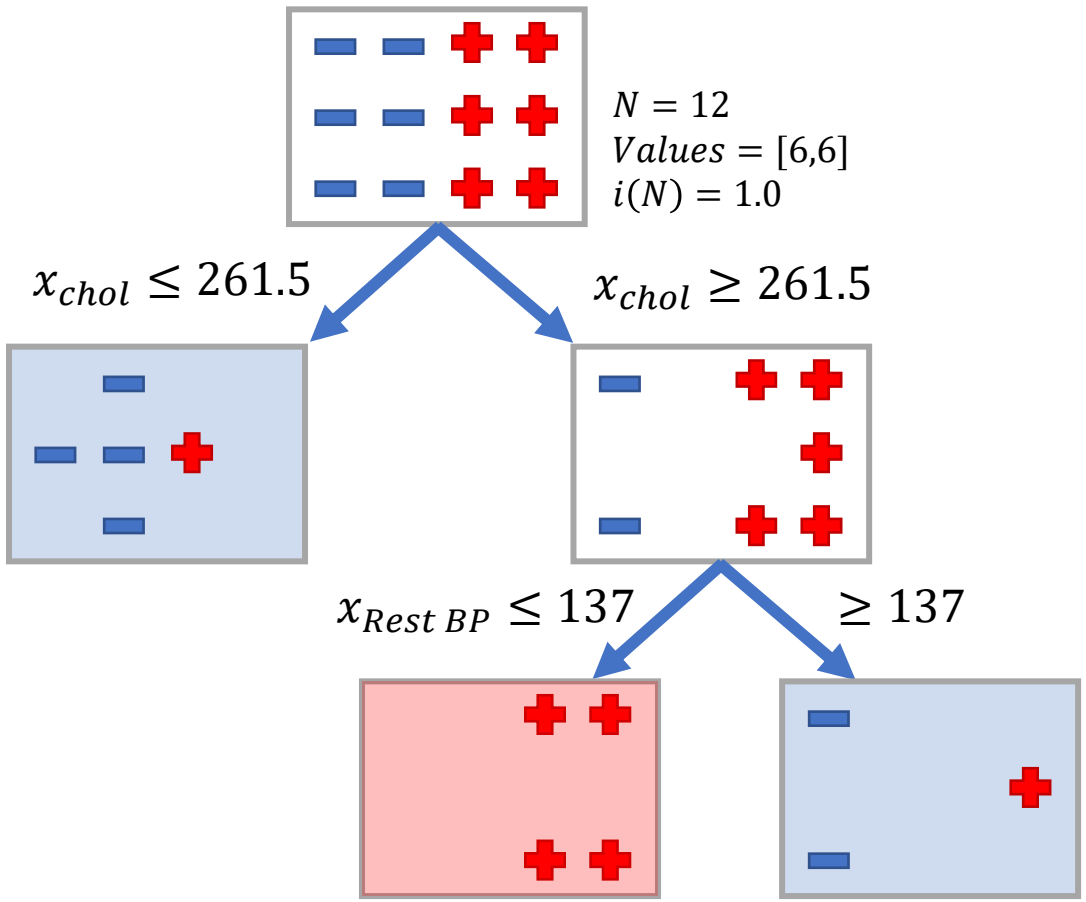
Decision Trees – Toy Example

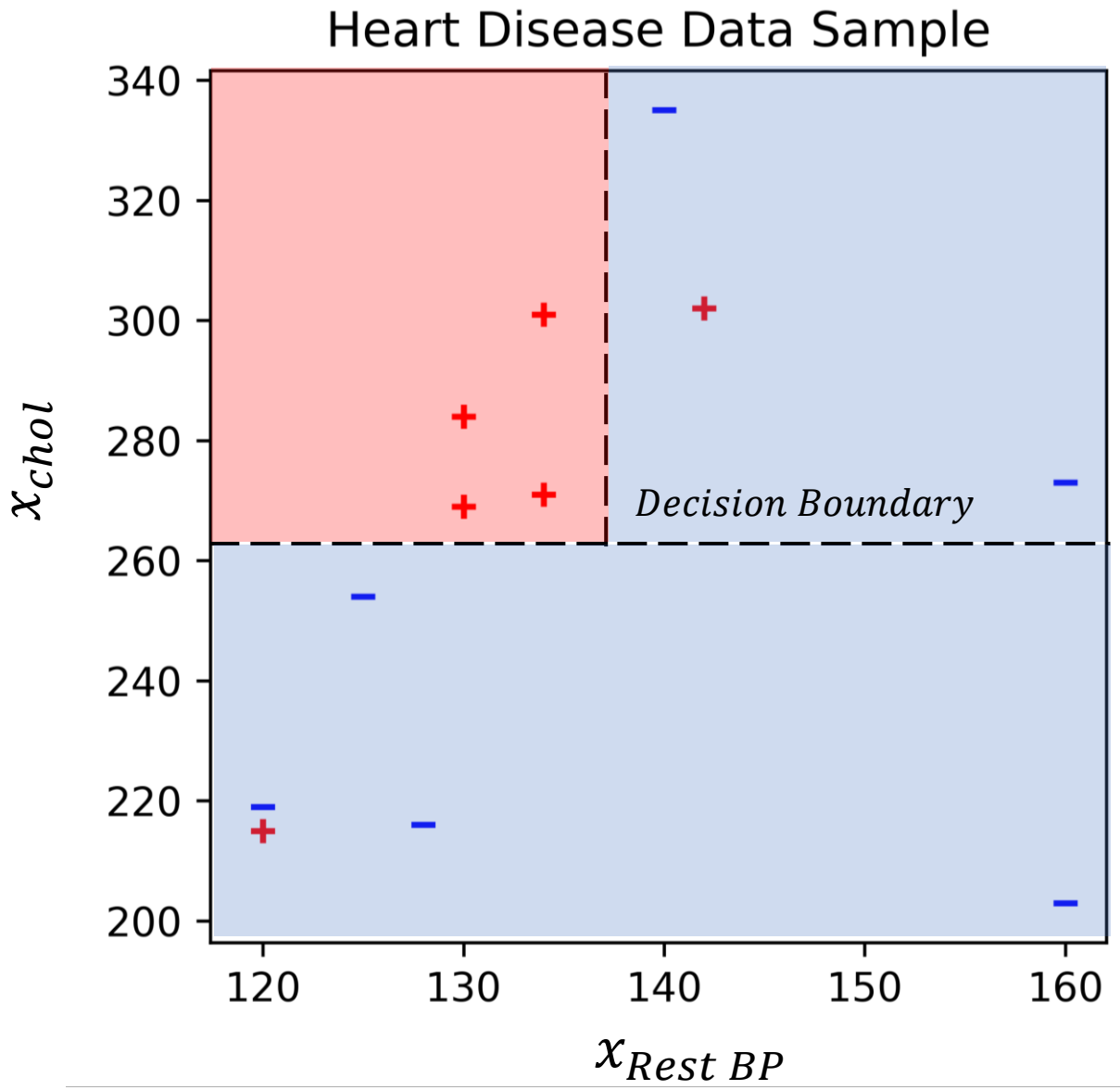
Tree Depth = 2



Decision Trees – Toy Example

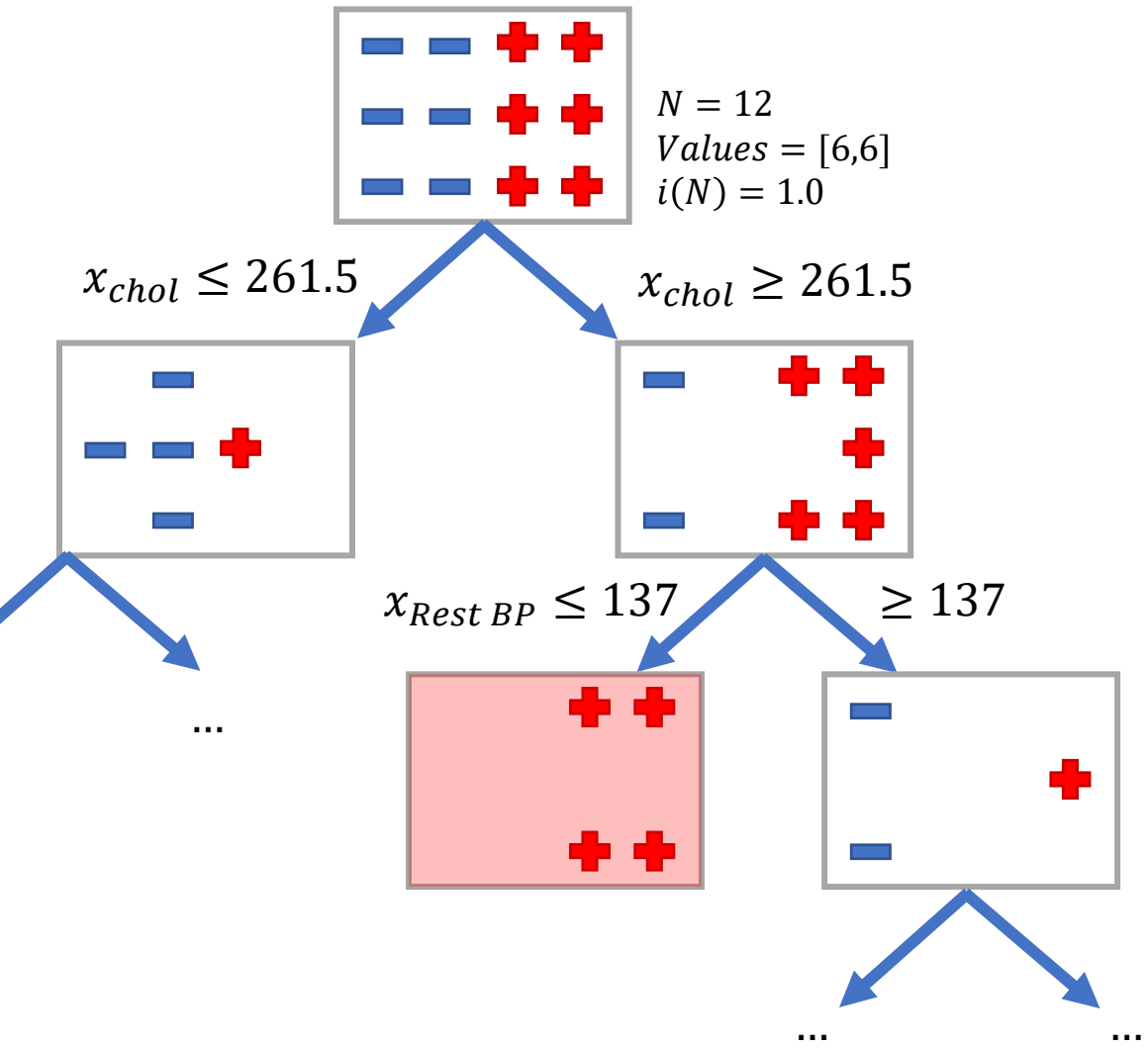
Decision Surface

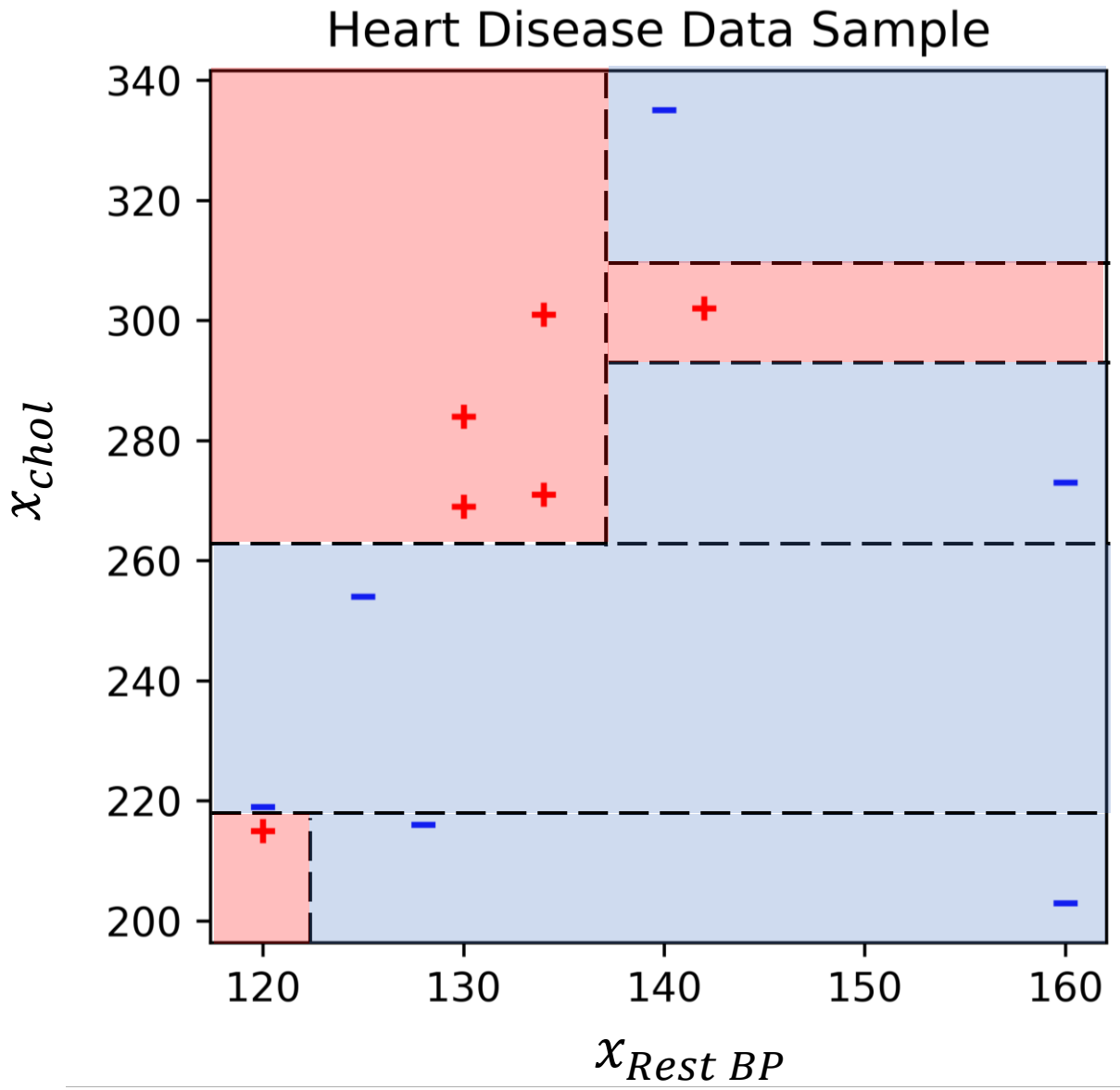




Decision Trees – Toy Example

Overfitting



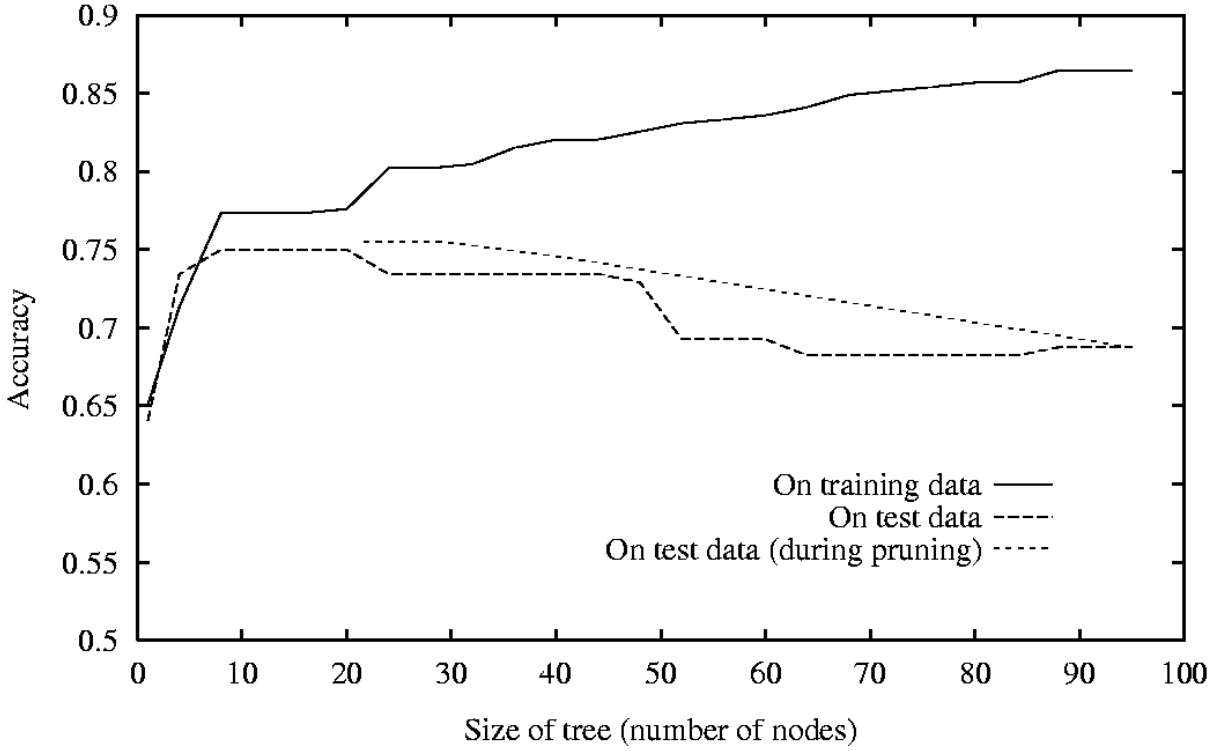


Overfitting in Decision Trees

How can we avoid overfitting?

1. **Setting Constraints on Tree Size**
   - Minimum samples for a node split
   - Maximum depth of tree (vertical depth)
   - Maximum number of terminal nodes

2. **Tree Pruning**
   reduce the size of the Decision Tree by removing sections of the tree that provide little power to classify instances.



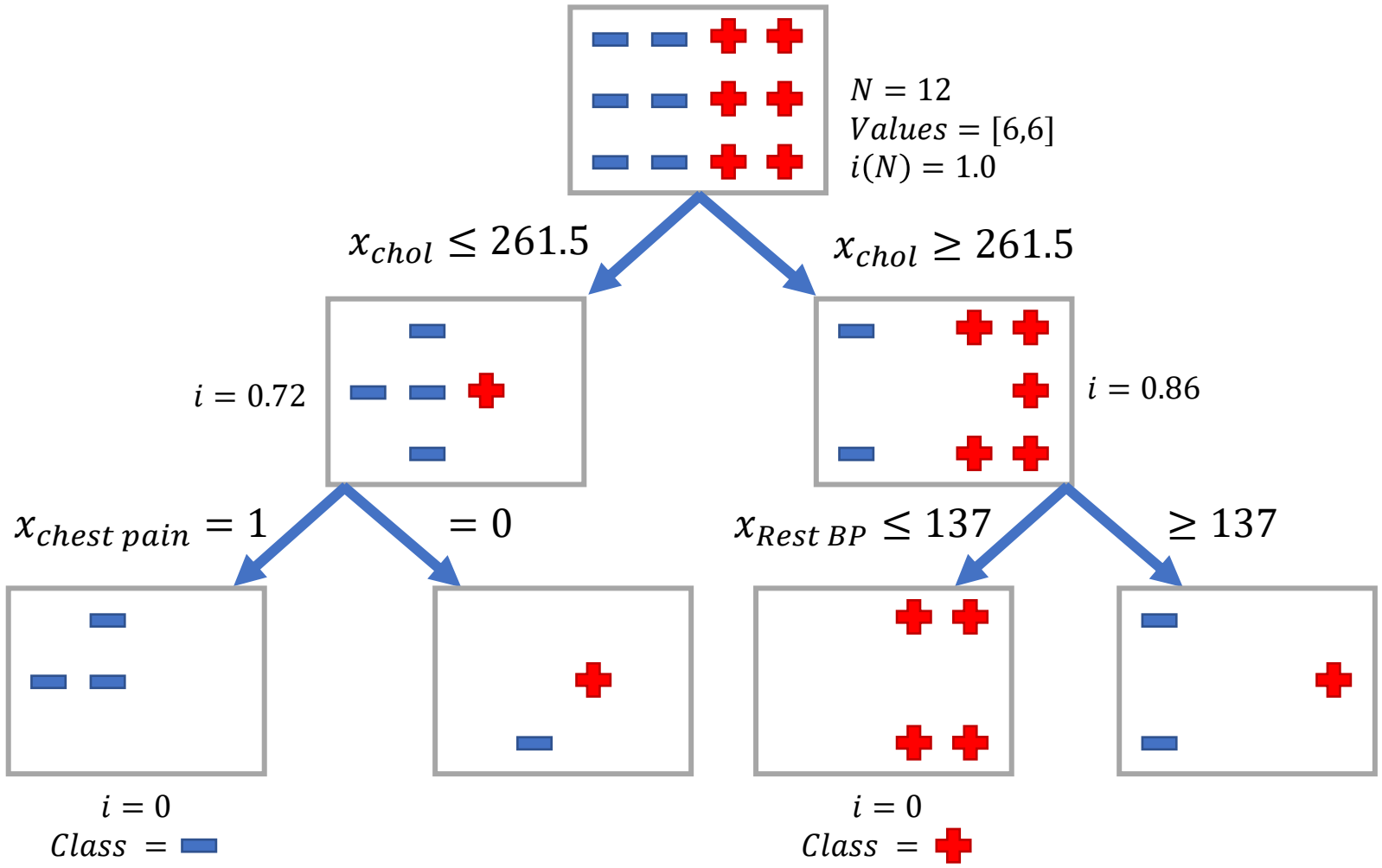
Decision Trees in Scikit-Learn

```
from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier(   criterion='entropy', #Chosen from {"gini", "entropy"}
                                    max_depth=None, #The maximum depth of the tree (None: Unlimited),
                                                        #The higher it gets, MORE fit.
                                    min_samples_split=2,    #What is the minimum samples required to keep splitting.
                                #The higher it gets, LESS fit.
                                    max_features=None,    #How many features are we allowed to use (None: Unlimited),
                                 #The higher it gets, MORE Fit.
                                    max_leaf_nodes=None, #The maximum number of leaves (None: Unlimited)
                                                            #The higher it gets, MORE fit.
                                    min_impurity_split=1e-7 #The higher it gets, the LESS fit.

                                 )
```

https://scikit-learn.org/stable/modules/tree.html#tree

Decision Trees in Scikit-Learn

Plotting the Decision Tree (2)



Decision Trees in Scikit-Learn

Plotting the Decision Tree (1)

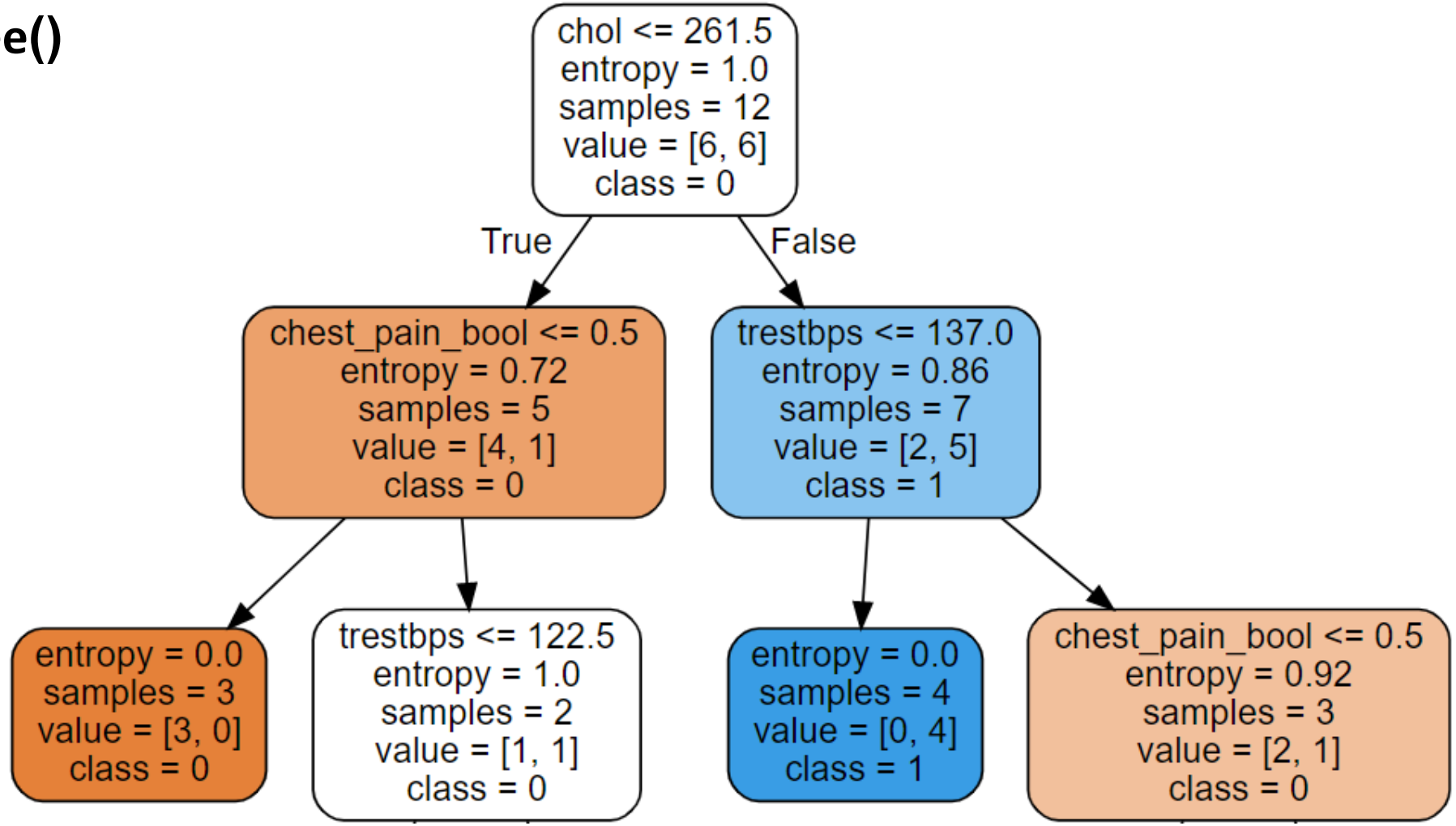
Using plot_tree()

```
from sklearn.tree import plot_tree

plt.figure(figsize=(10,5))
dot_data  = plot_tree(clf,
                      feature_names = X_sample.columns,
                      class_names = ["0","1"],
                      rounded = True, proportion = False, impurity = True,
                      label='all', precision = 2, filled = True)
```

Decision Trees in Scikit-Learn

Plotting the Decision Tree (3)

Using plot_tree()



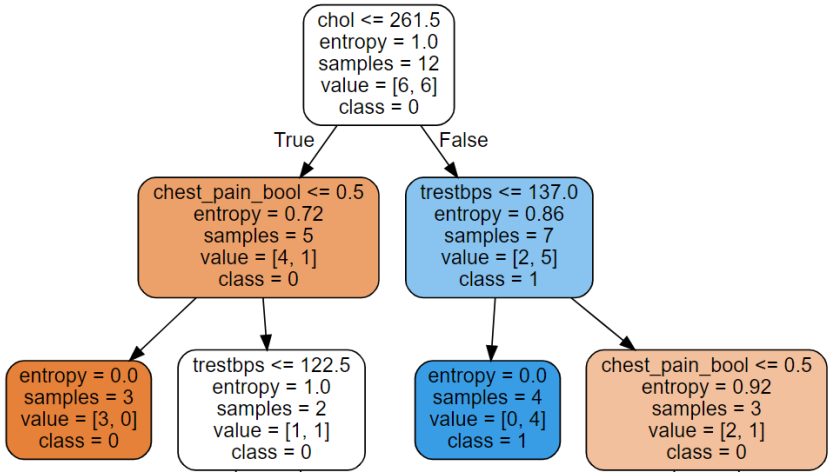
Decision Trees in Scikit-Learn

Plotting the Decision Tree (3)

Using plot_tree()

All the nodes, except the leaf nodes (colored terminal nodes), have 5 parts:

1. Question asked about the data based on a value of a feature. Each question has either a True or False answer that splits the node. Based on the answer to the question, a data point moves down the tree.
2. gini: The Gini Impurity of the node. The average weighted Gini Impurity decreases as we move down the tree.
3. samples: The number of observations in the node.
4. value: The number of samples in each class. For example, the top node has 6 samples in class 0 and 6 samples in class 1.
5. class: The majority classification for points in the node. In the case of leaf nodes, this is the prediction for all samples in the node.



Decision Tree Pros and Cons

- **Pros**:
  - Interpretable: users can easily understand decisions
  - Easily handles irrelevant attributes (Gain = 0) → ignoring noise
  - Can handle missing values
  - Very compact: # nodes << X after pruning
  - Very fast at testing time: O(depth) → Real time
- **Cons**:
  - Only axis-aligned splits of data
  - Greedy (may not find the best tree)
  - Can easily overfit

המחלקה להנדסת תעשייה
הפקולטה להנדסה ע"ש איבי ואלדר פליישמן
אוניברסיטת תל אביב

מבוא ללמידת מכונה

Introduction to Machine Learning

אוניברסיטת תל אביב TEL AVIV UNIVERSITY

המשך שבוע נעים!

אילן וסילבסקי

תשפ"ב 2022