Non-Parametric Models

Rina BUOY, PhD

ChatGPT 4.0

Disclaimer

Adopted from

6.390

Introduction to Machine Learning
(Fall 2024)

https://introml.mit.edu/fall24

Outline

- Recap: parameterized models
- Non-parametric models
  - Interpretability
  - Ease of use and simplicity
- Decision Tree
  - `BuildTree`
- Nearest Neighbor

Recall

Hypothesis class $\mathcal{H}$: set of h

A linear regression hypothesis when $d = 1$:

$h\left(x; \theta, \theta_0\right) = \theta x + \theta_0$

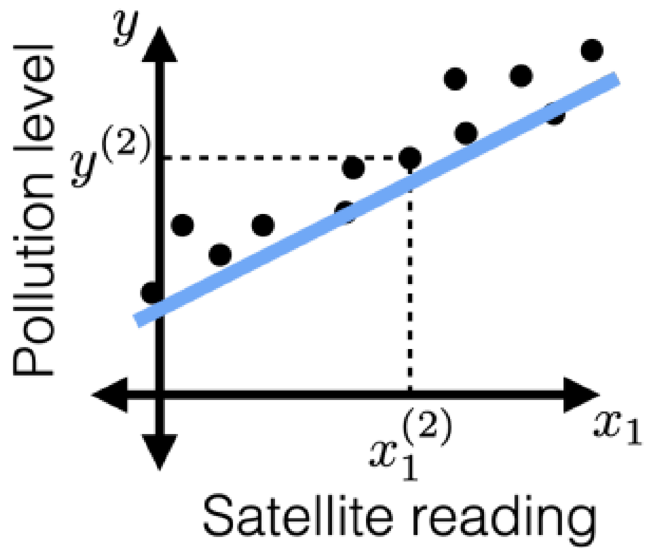
2 scalars

A linear reg. hypothesis when $d \geq 1$:

$h\left(x; \theta, \theta_0\right) = \theta_1 x_1 + \cdots + \theta_d x_d + \theta_0$

$= \theta^\top x + \theta_0$

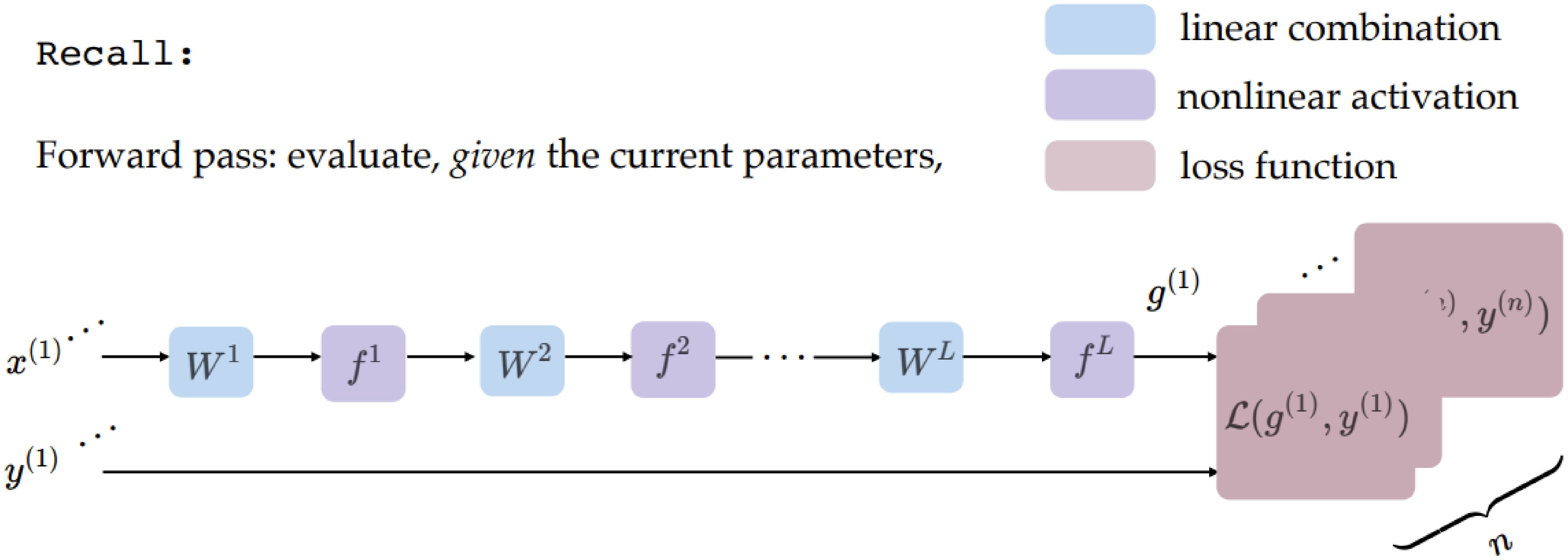
$(d+1)$ scalars



size of parameter is independent of n,
the number of data points

`Recall:`

Forward pass: evaluate, *given* the current parameters,



- the model output $g^{(i)} = f^L\left(\ldots f^2\left(f^1(\mathbf{x}^{(i)}; \mathbf{W}^1); \mathbf{W}^2\right); \ldots \mathbf{W}^L\right)$
- the loss incurred on the current data $\mathcal{L}(g^{(i)}, y^{(i)})$
- the training error $J = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(g^{(i)}, y^{(i)})$

Adopted from : https://introml.mit.edu

Outline

- Recap: parameterized models
- Non-parametric models
  - Interpretability
  - Ease of use and simplicity
- Decision Tree
  - `BuildTree`
- Nearest Neighbor

Non-parametric models

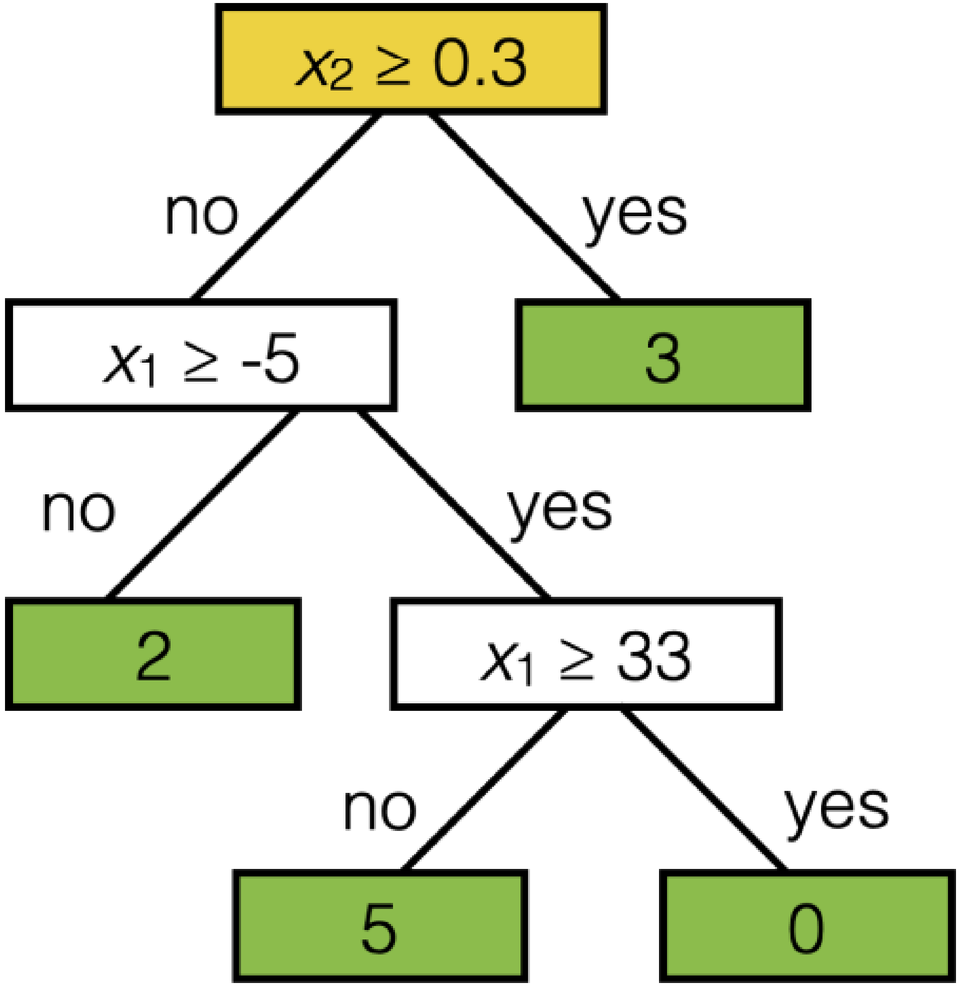
- does not mean "no parameters"
- there are still parameters to be learned to build a hypothesis/model.
- just that, the model/hypothesis does not have a fixed parameterization.
- (e.g. even the number of parameters can change.)

- they are usually fast to implement / train and often very effective.
- often a good baseline (especially when the data doesn't involve complex structures like image or languages)

Outline

- Recap: parameterized models
- Non-parametric models
  - Interpretability
  - Ease of use and simplicity
- Decision Tree
  - `BuildTree`
- Nearest Neighbor

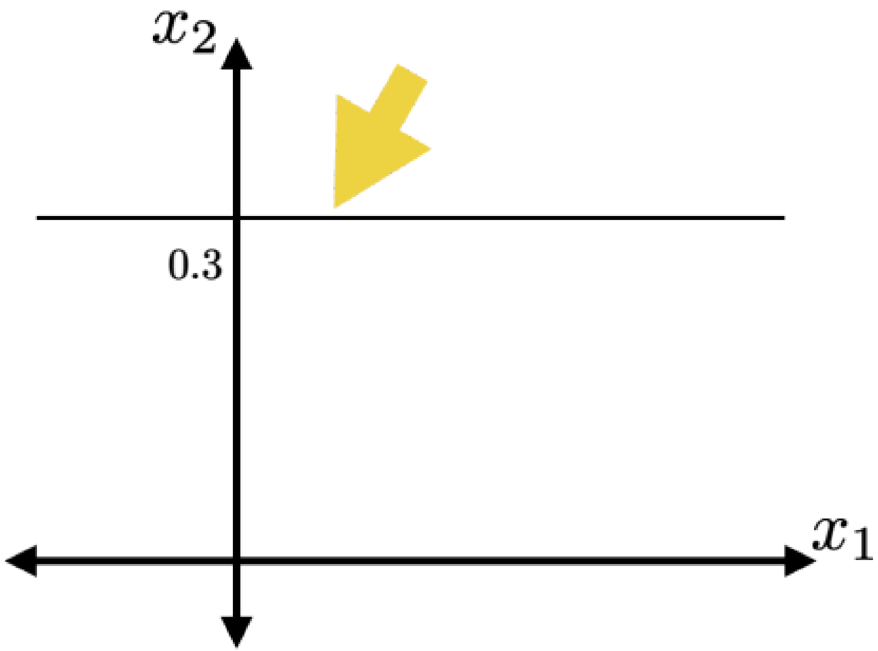
Adopted from : https://introml.mit.edu



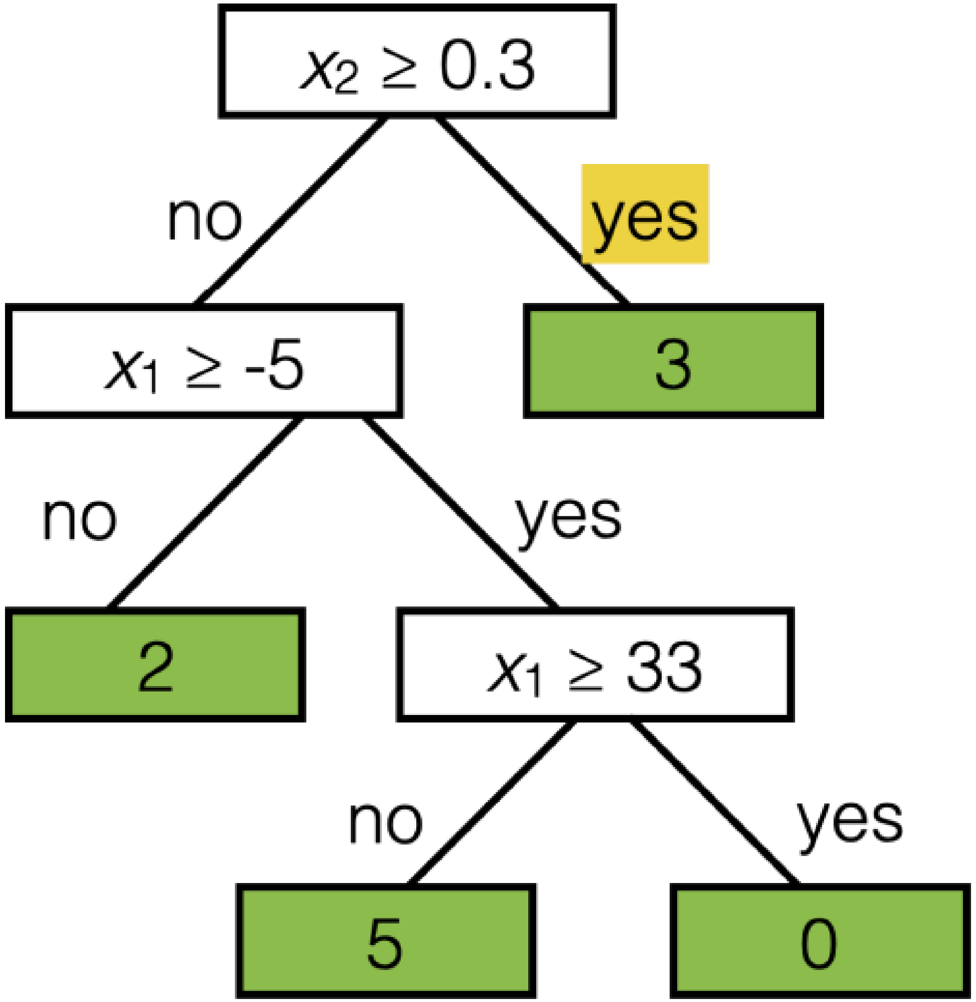
features:

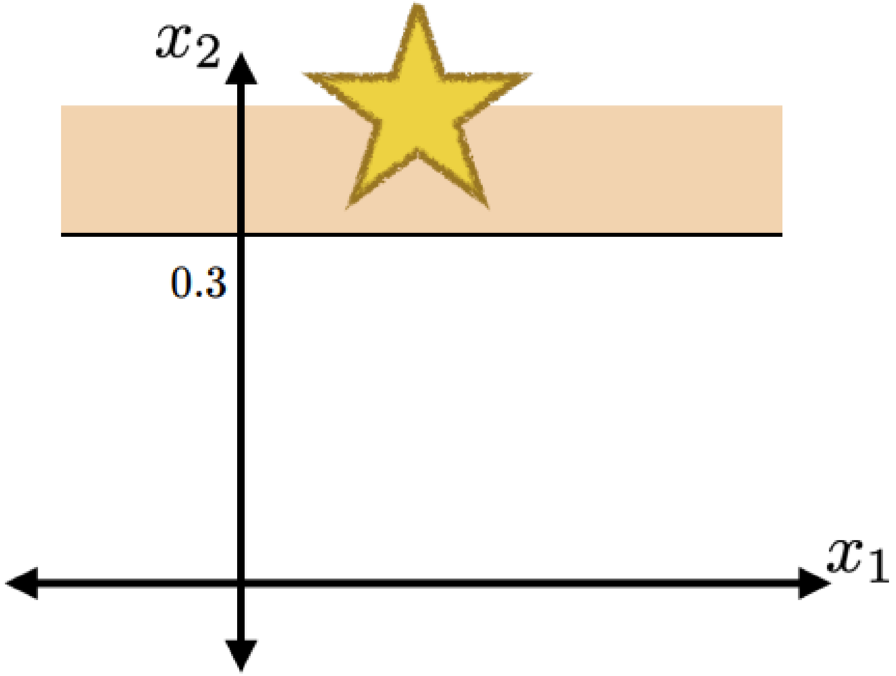
- x_1: temperature (deg C)
- x_2: precipitation (cm/hr)

label: km run

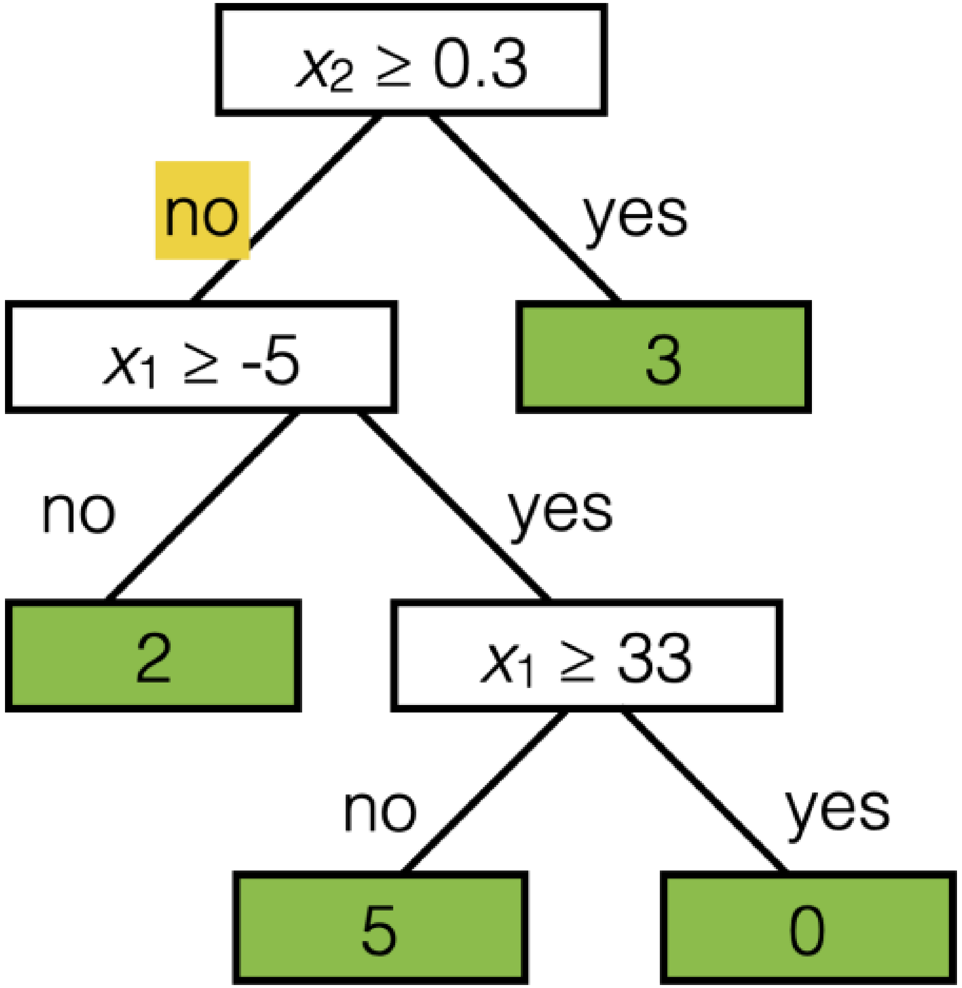


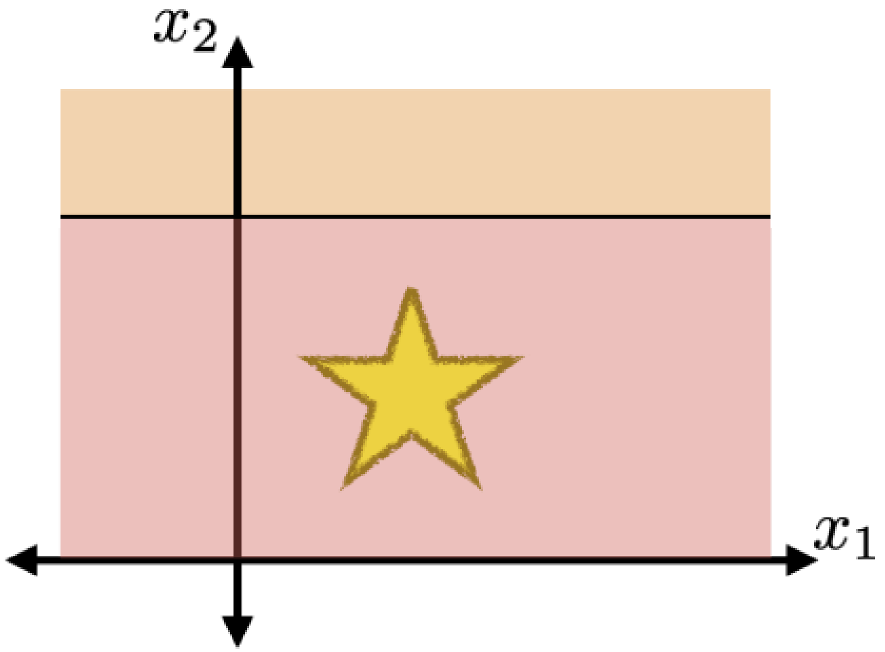
Adopted from : https://introml.mit.edu



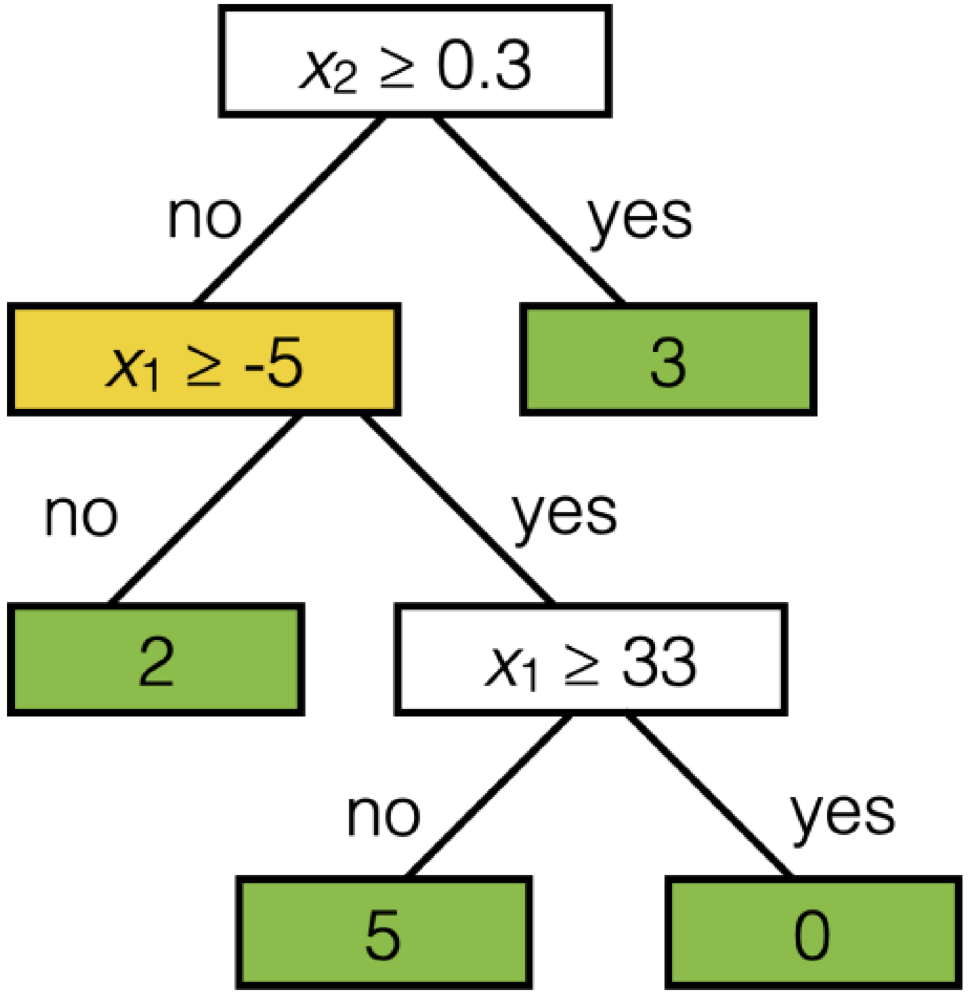


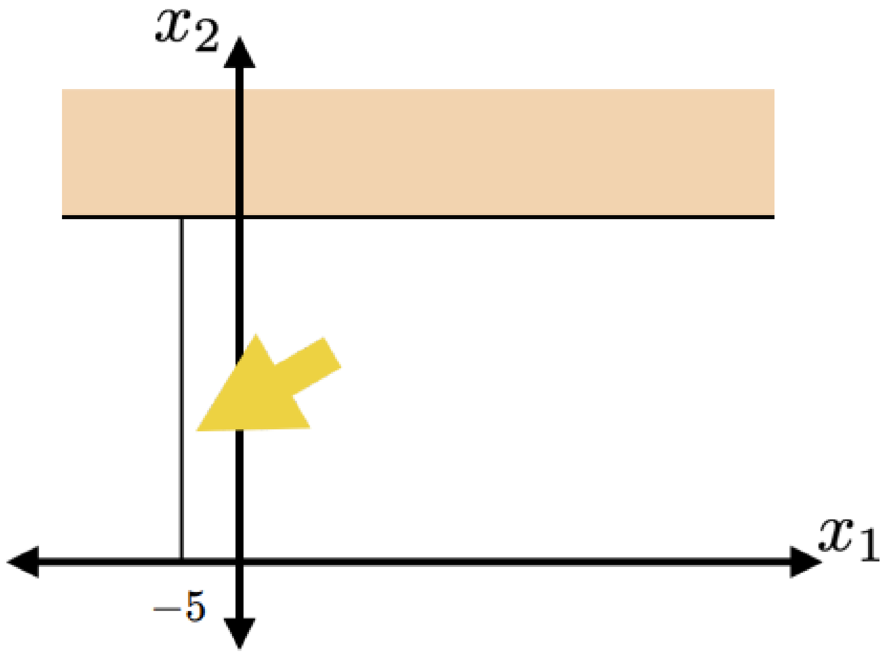
Adopted from : https://introml.mit.edu



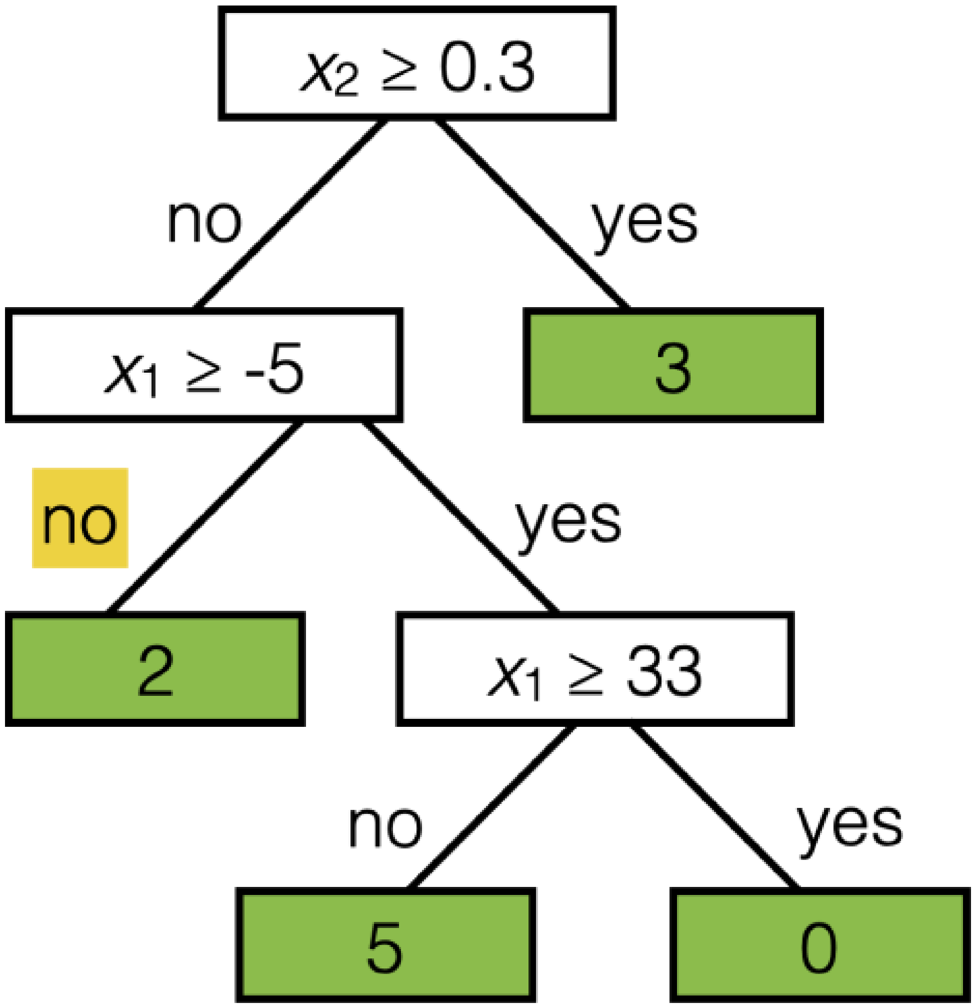


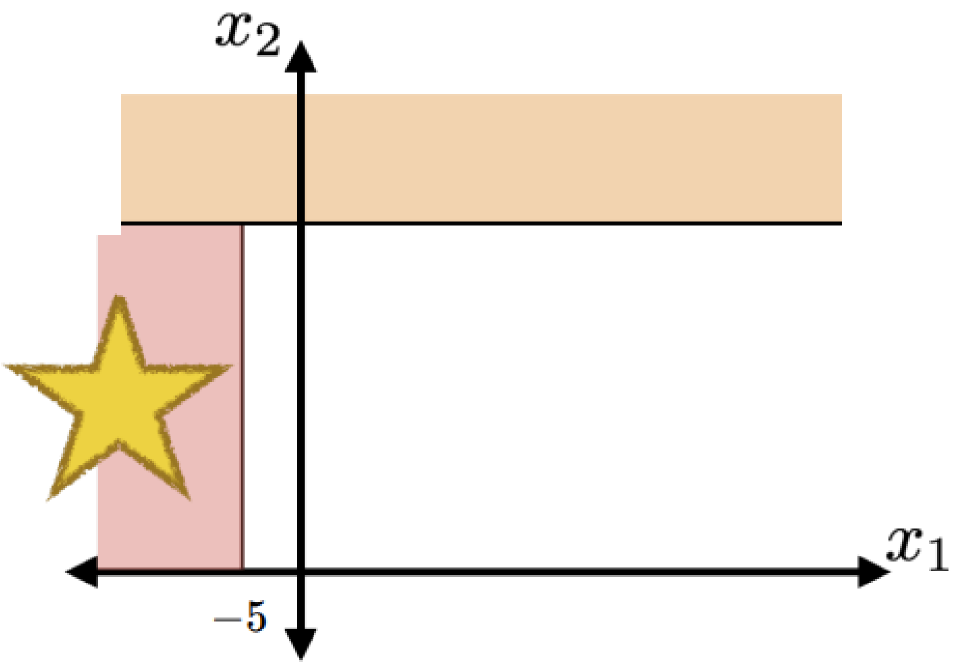
Adopted from : https://introml.mit.edu

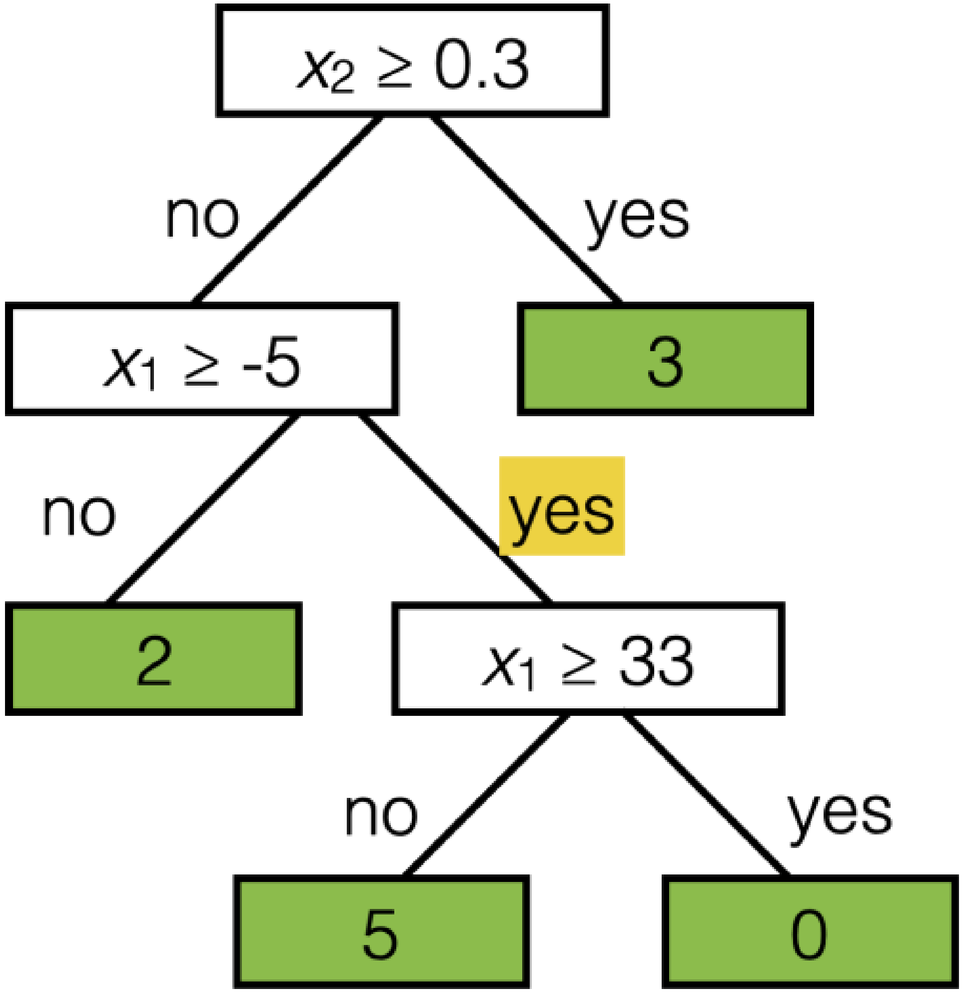


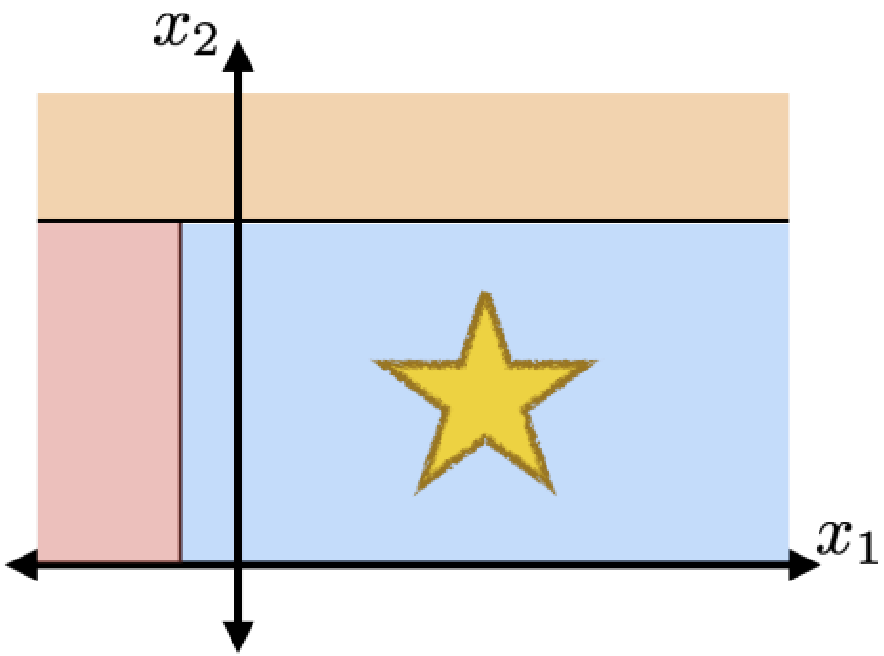


Adopted from : https://introml.mit.edu

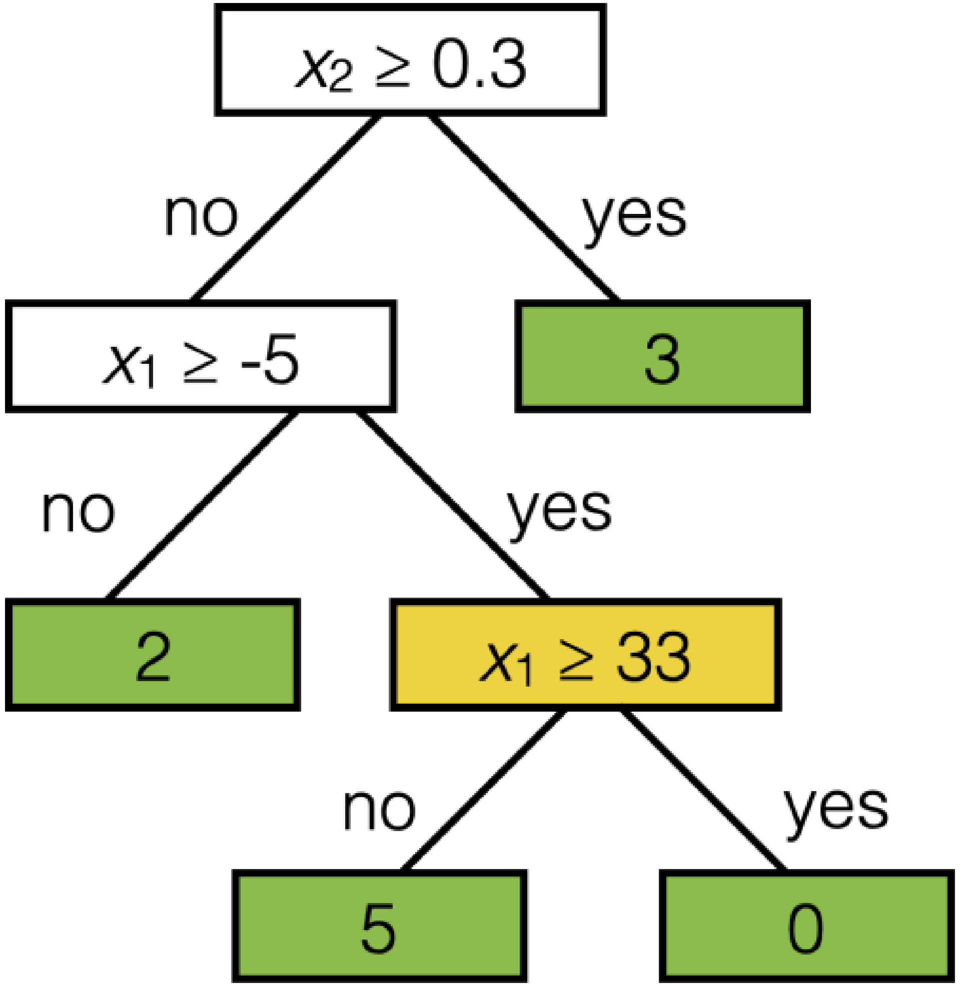


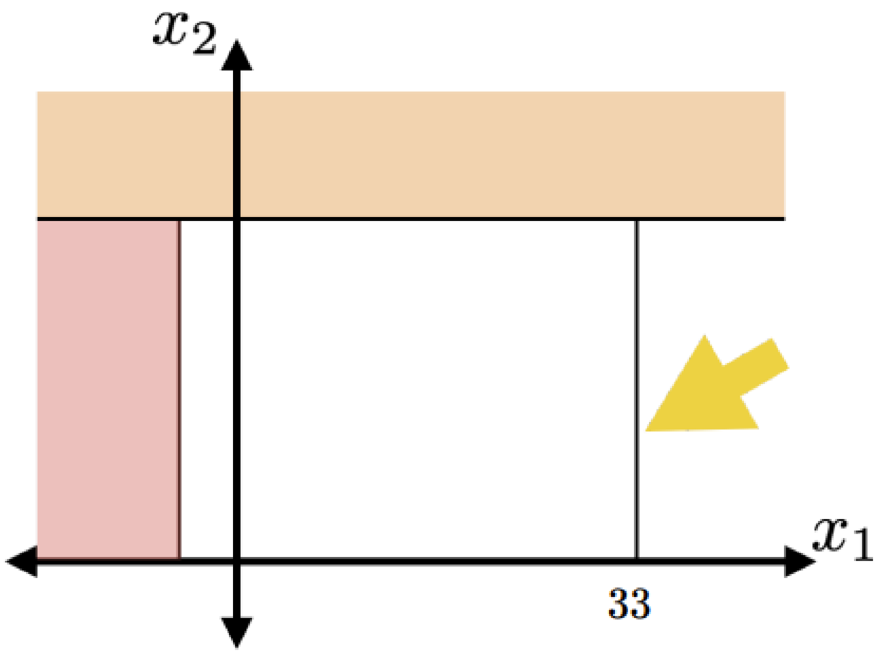


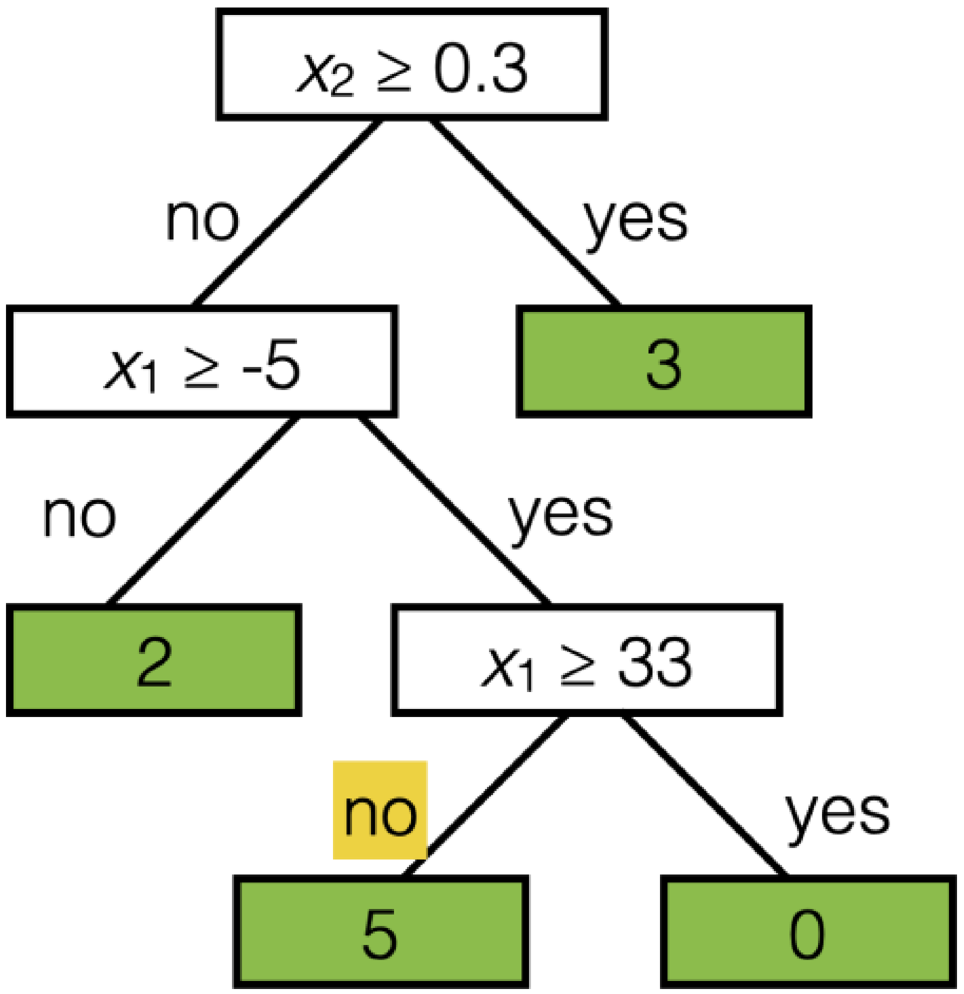

$x_2 \geq 0.3$
no
yes
$x_1 \geq -5$
3
no
yes
2
$x_1 \geq 33$
no
yes
5
0

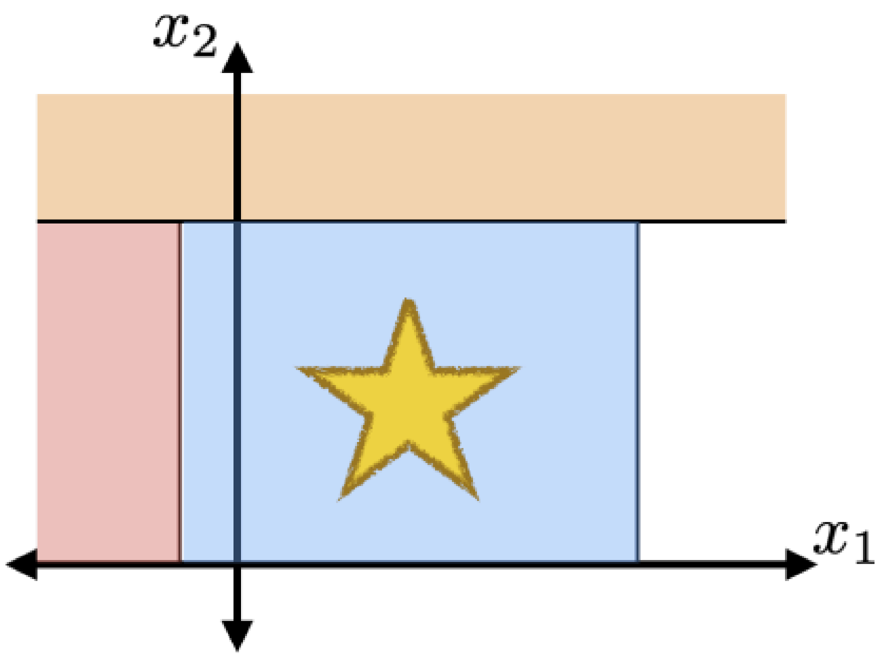

x_2
x_1

Adopted from : https://introml.mit.edu

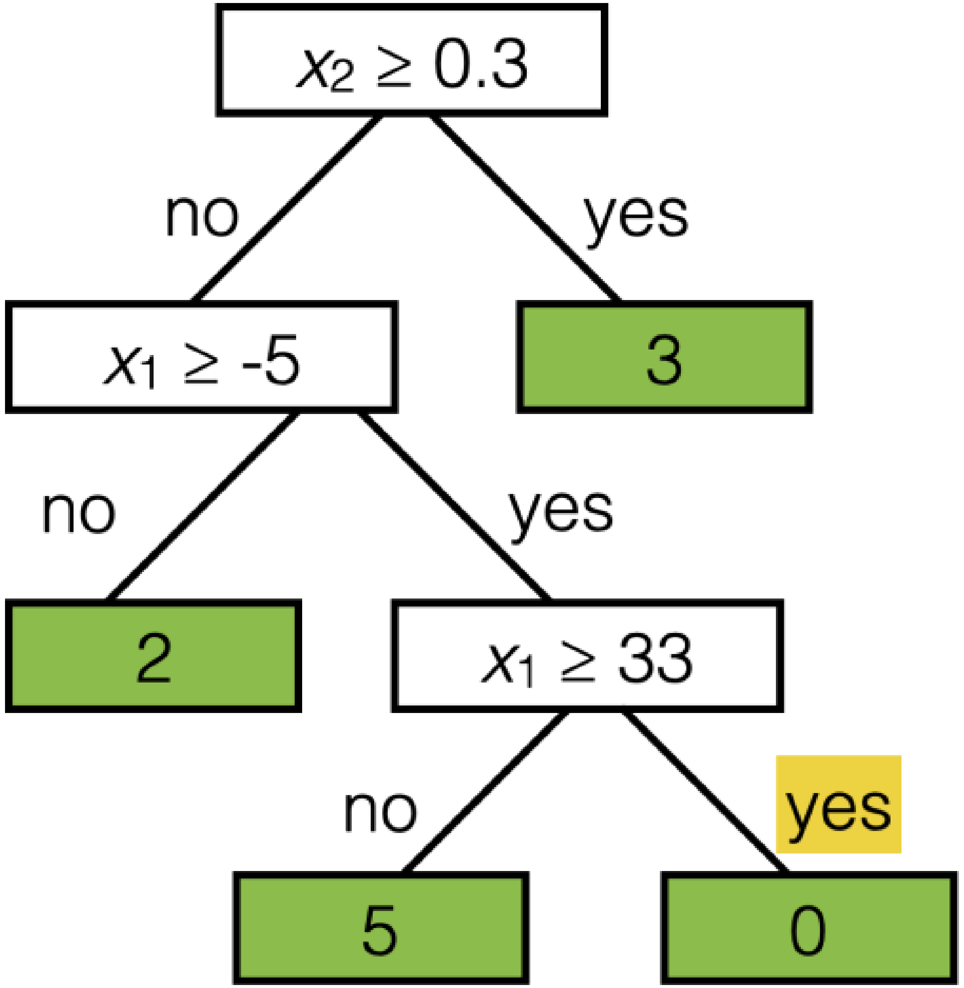


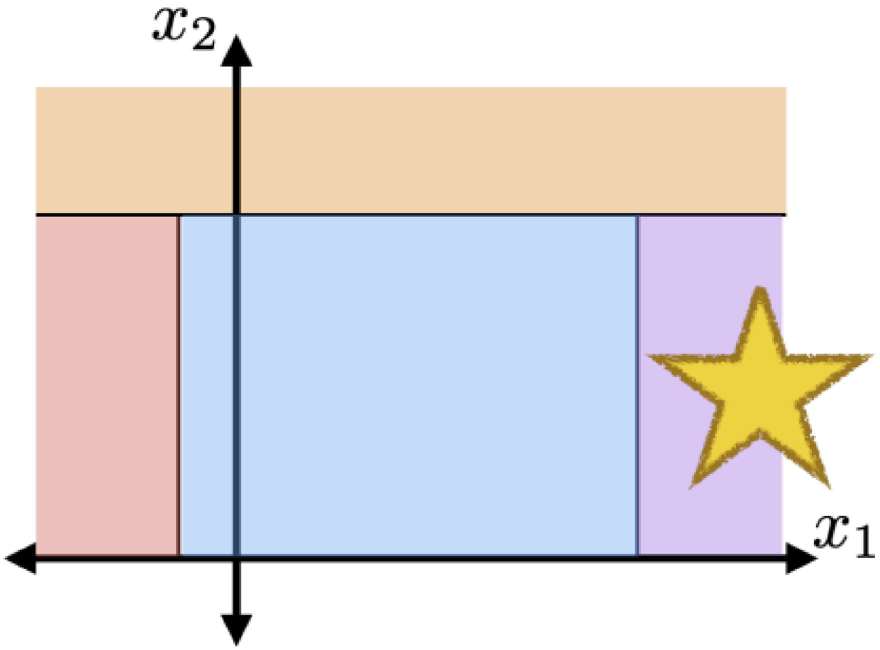


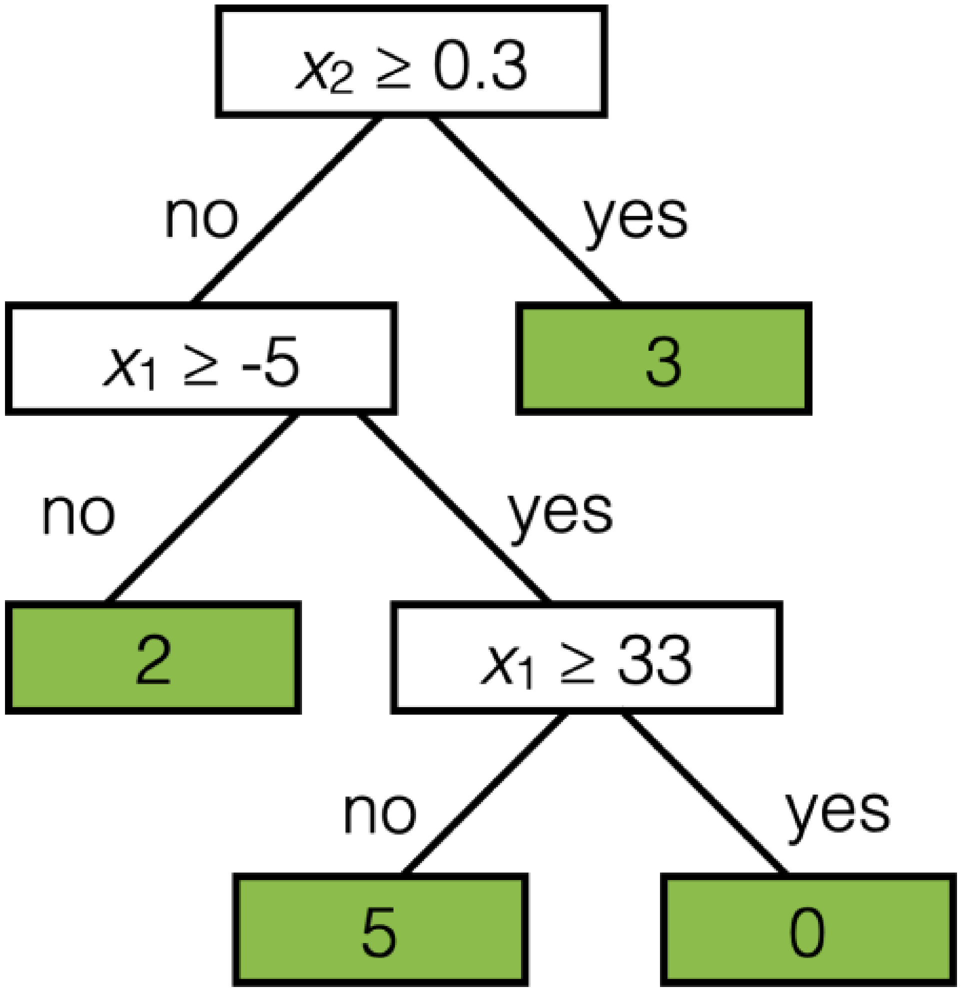

$x_2 \geq 0.3$
no
yes
$x_1 \geq -5$
3
no
yes
2
$x_1 \geq 33$
no
yes
5
0


x_2
x_1

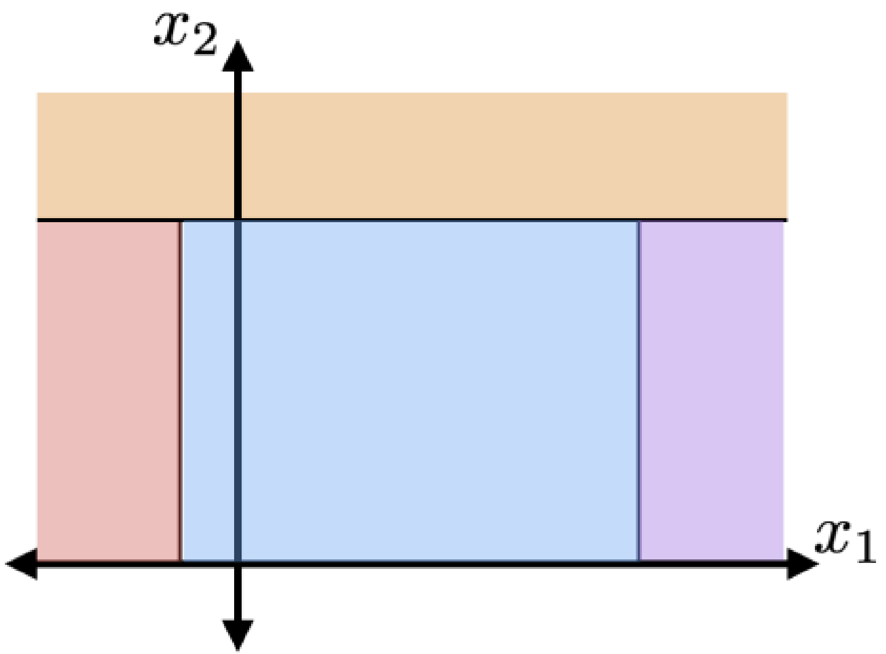
Adopted from : https://introml.mit.edu

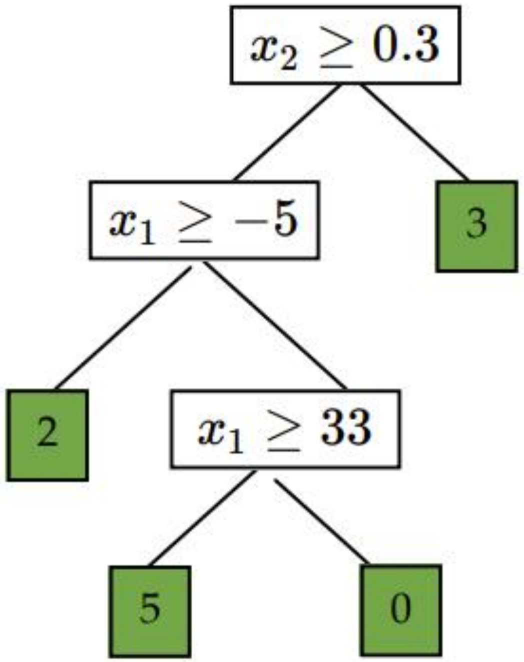


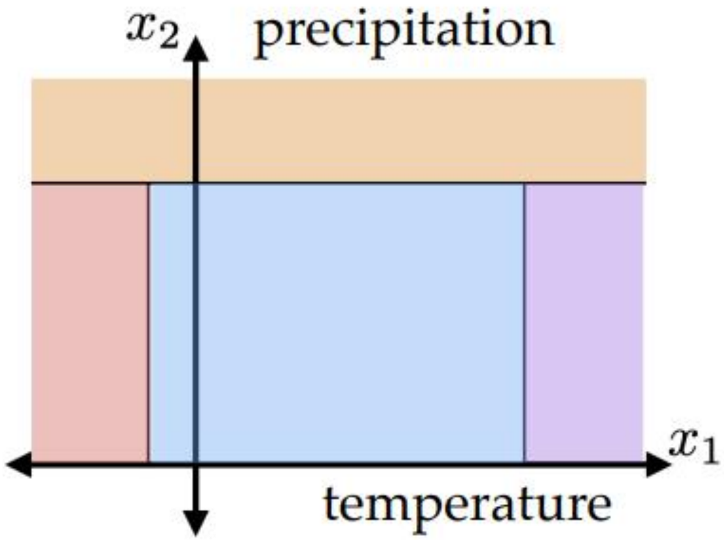




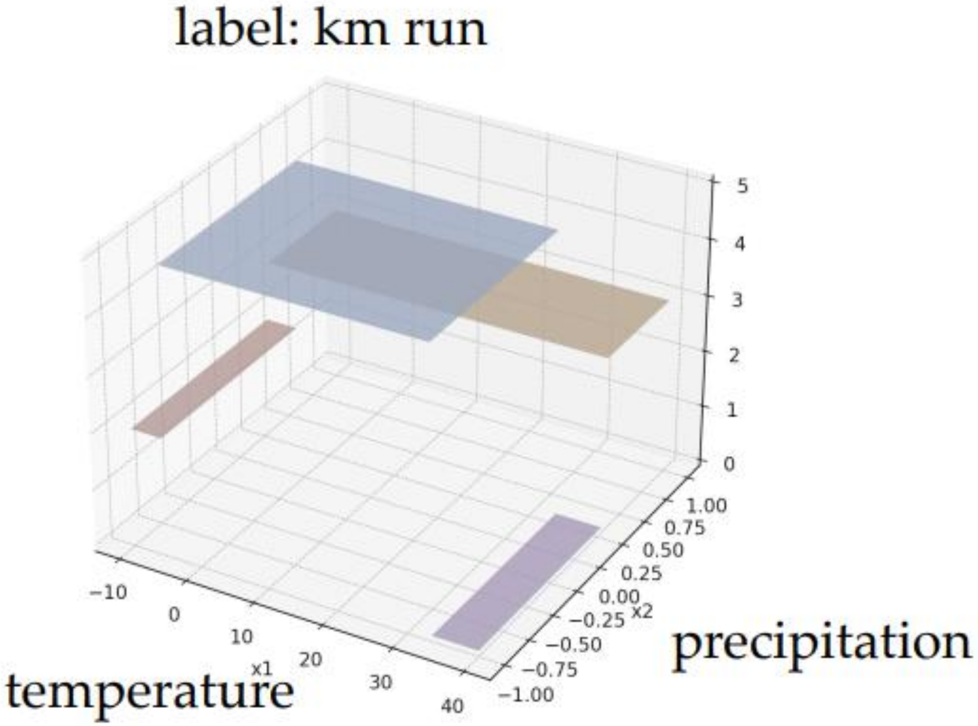
The same prediction applies to an axis-aligned 'box' or 'volume' in the feature space





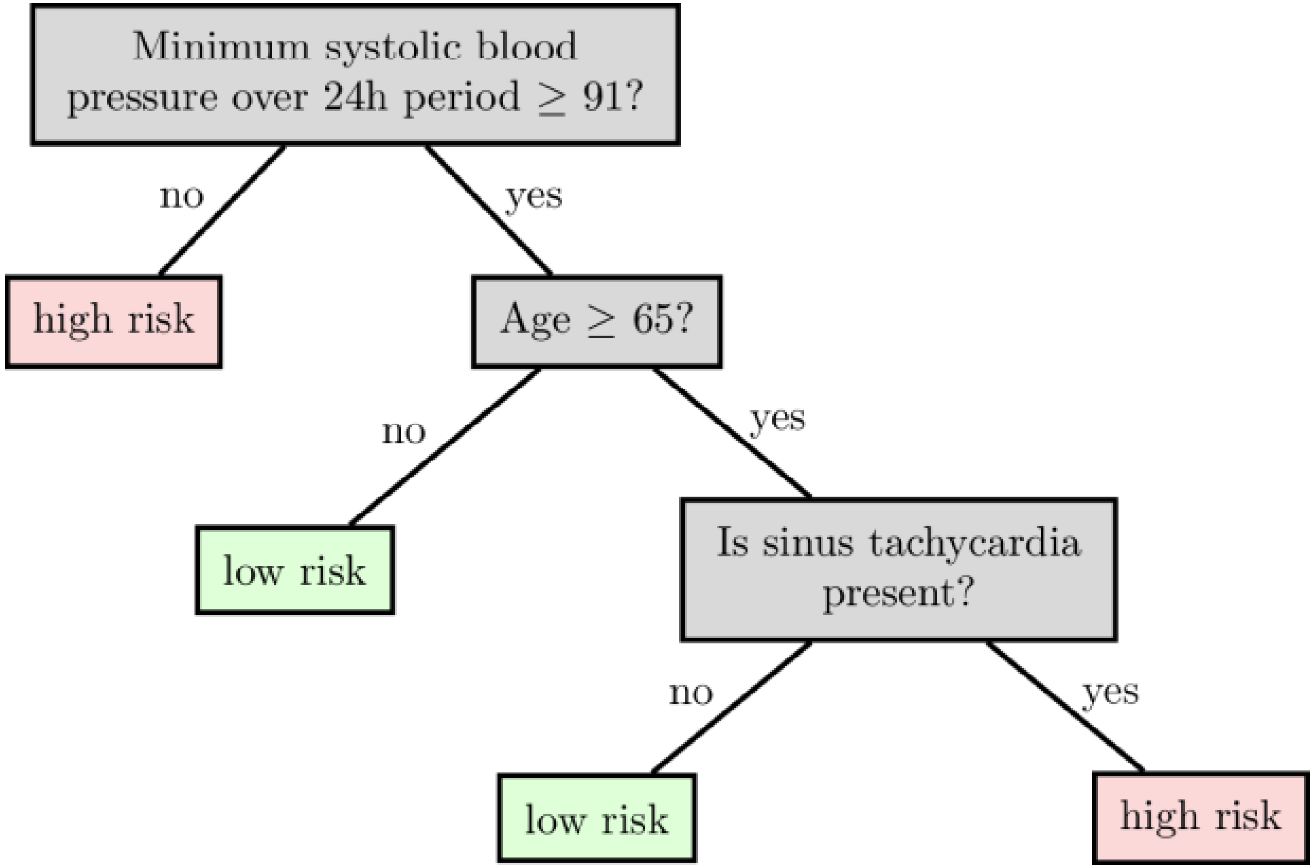


The same prediction applies to an axis-aligned ‘box’ or ‘volume’ in the feature space



Adopted from : https://introml.mit.edu

Decision tree for classification



features:

x_1 : date

x_2 : age

x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

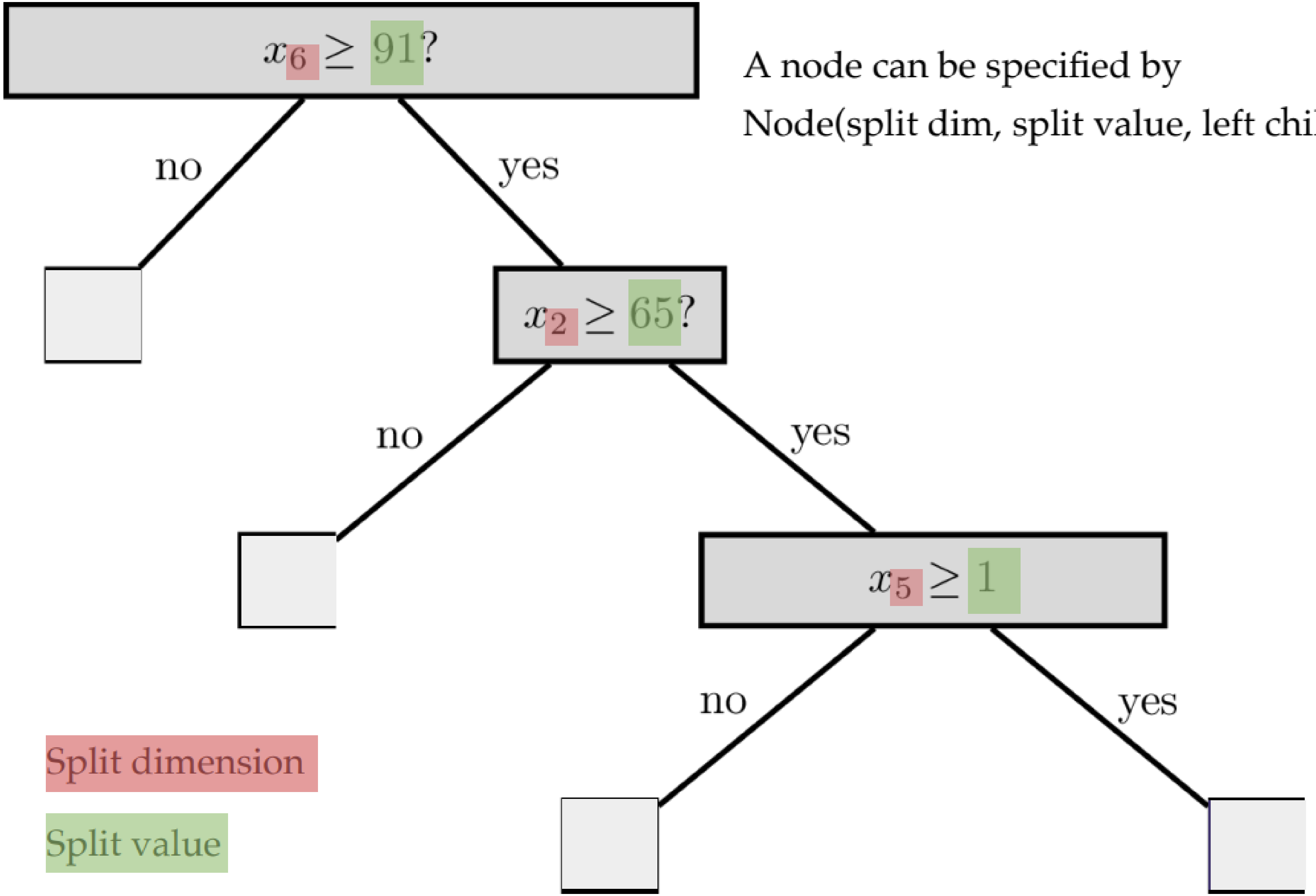
x_6 : min systolic bp, 24h

x_7 : latest diastolic bp

labels y :

1: high risk

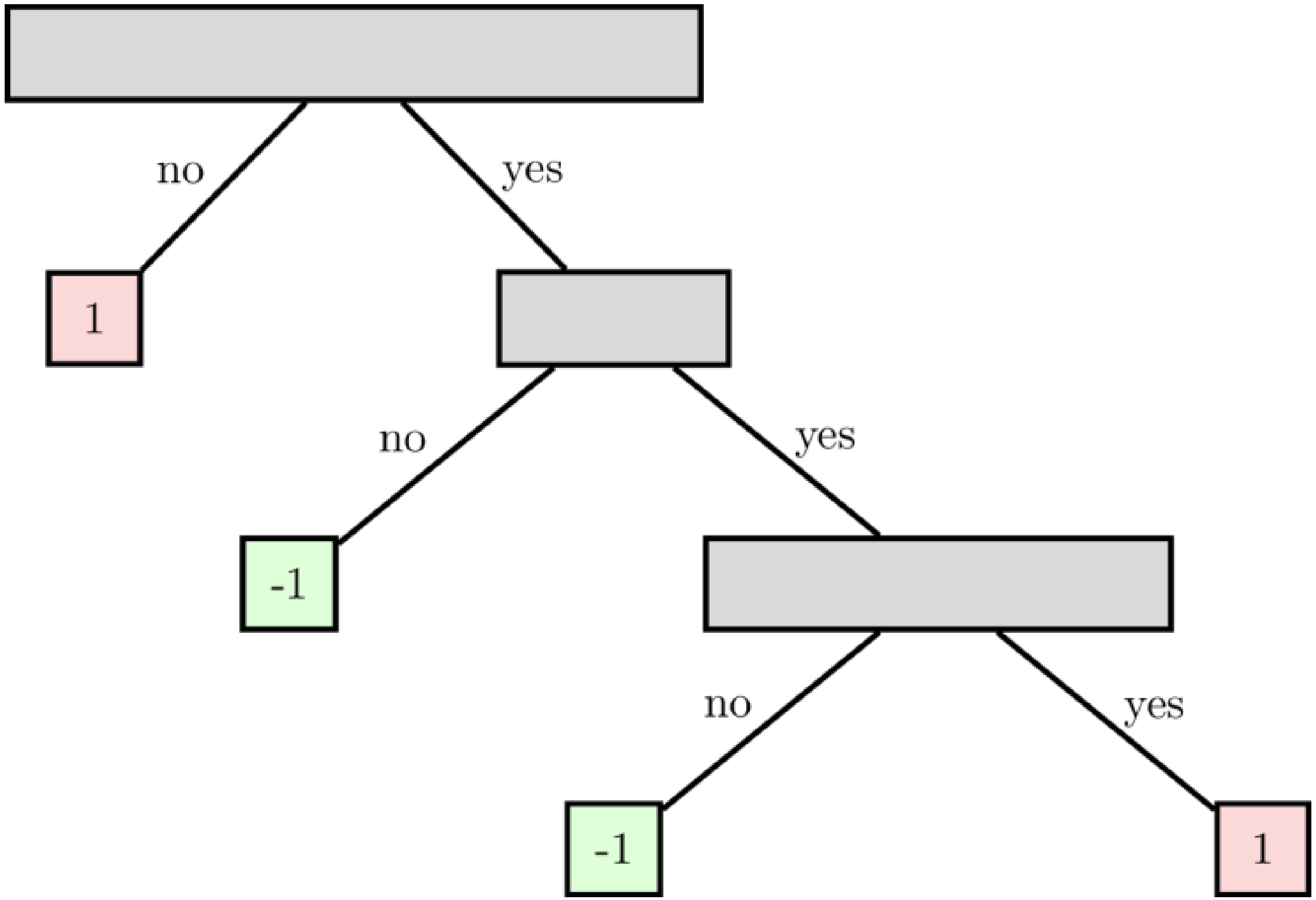
-1: low risk



A node can be specified by
Node(split dim, split value, left child, right child)

Split dimension

Split value



A leaf can be specified by
Leaf(leaf_value)

Adopted from : https://introml.mit.edu

Outline

- Recap: parameterized models
- Non-parametric models
  - Interpretability
  - Ease of use and simplicity
- Decision Tree
  - `BuildTree`
- Nearest Neighbor

Set of indices.

Hyper-parameter, largest leaf size (i.e. the maximum number of training data that can "flow" into that leaf).

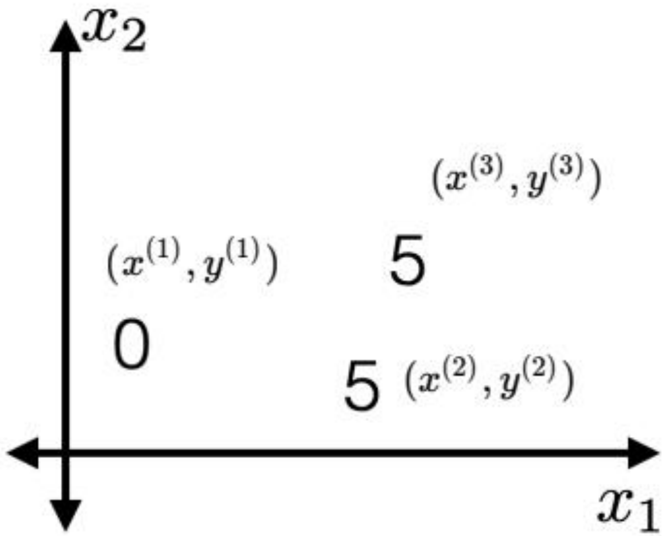
$\text{BuildTree}(I, k, \mathcal{D})$

1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \, y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \, y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+} \right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-} \right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I} \, y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k \right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k \right) \right)$

Adopted from : https://introml.mit.edu

$\text{BuildTree}(I, k, \mathcal{D})$

1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \, y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \, y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+} \right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-} \right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I} \, y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k \right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k \right) \right)$

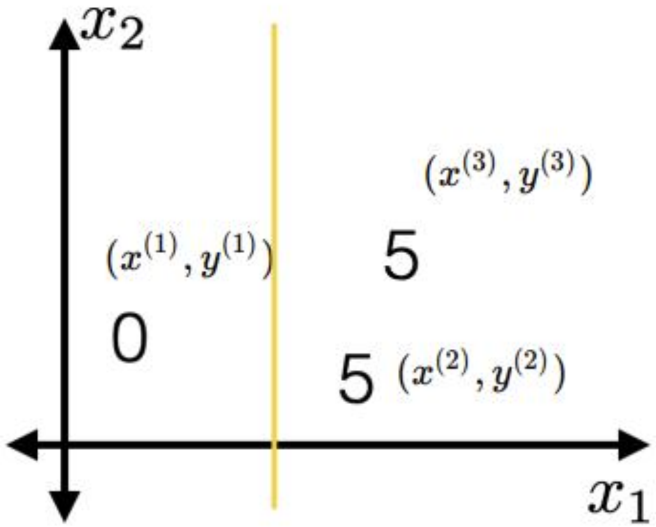


- Choose $k = 2$
- $\text{BuildTree}(\{1, 2, 3\}; 2)$
- Line 1 true

Adopted from : https://introml.mit.edu

$\text{BuildTree}(I, k, \mathcal{D})$

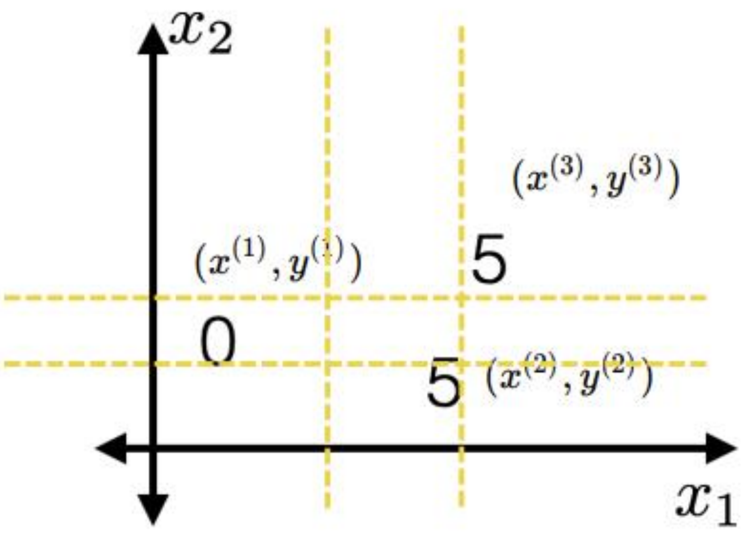
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \; y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \; y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+} \right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-} \right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I} \; y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k \right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k \right) \right)$



- For this fixed (j, s)
  - $I_{j,s}^{+} = \{2, 3\}$
  - $I_{j,s}^{-} = \{1\}$
  - $\hat{y}_{j,s}^{+} = 5$
  - $\hat{y}_{j,s}^{-} = 0$
  - $E_{j,s} = 0$

BuildTree$(I, k, \mathcal{D})$

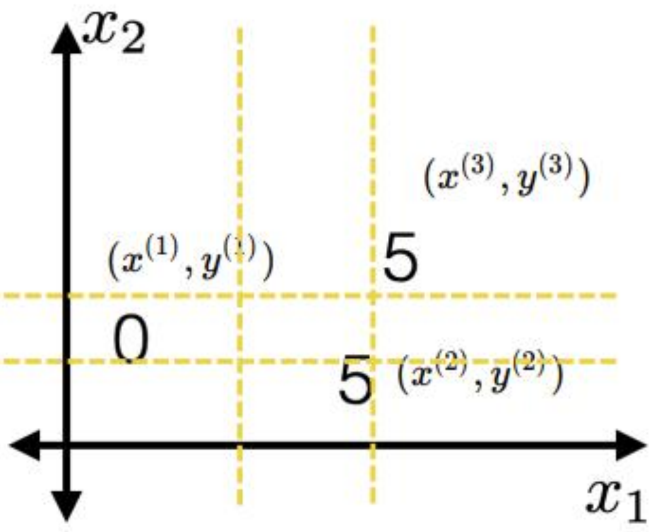
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \, y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \, y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+}\right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-}\right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I} \, y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k\right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k\right)\right)$



- Line 2: a finite number of (j, s) combo suffices (those that split in-between data points)

BuildTree$(I, k, \mathcal{D})$

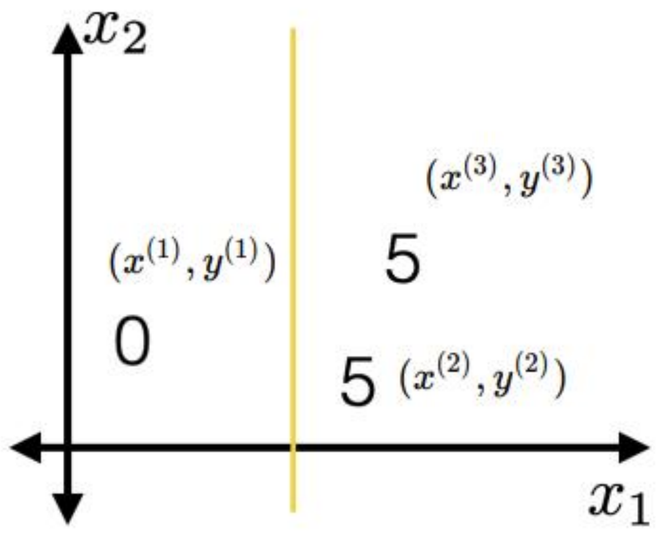
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \, y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \, y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+}\right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-}\right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I} \, y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k\right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k\right)\right)$



- Line 8: picks the "best" among these finite choices of (j, s) combos (random tie-breaking).

BuildTree$(I, k, \mathcal{D})$

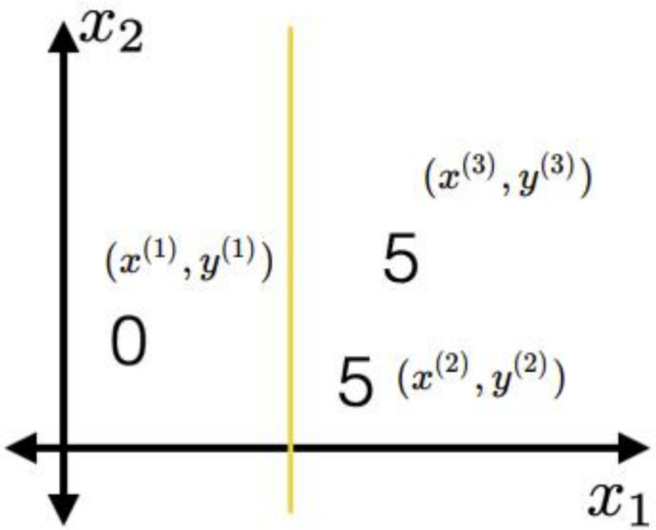
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \; y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \; y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+} \right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-} \right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I} \; y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k \right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k \right) \right)$



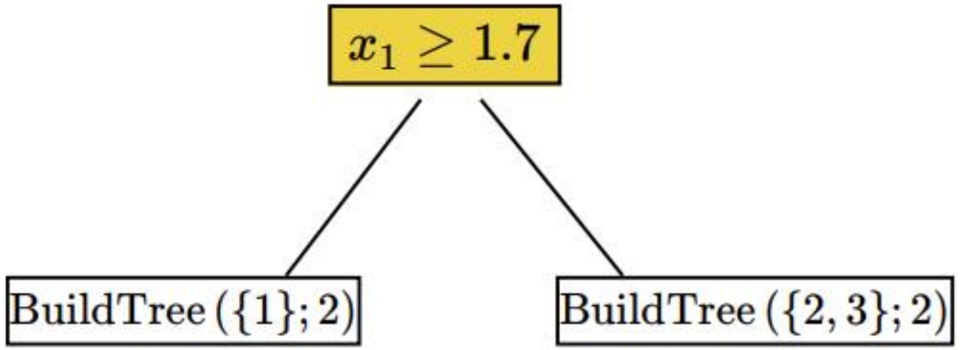
Suppose line 8 sets this $(j^*, s^*) = (1, 1.7)$

BuildTree($I, k, \mathcal{D}$)

1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \; y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \; y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+} \right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-} \right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I} \; y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k \right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k \right) \right)$

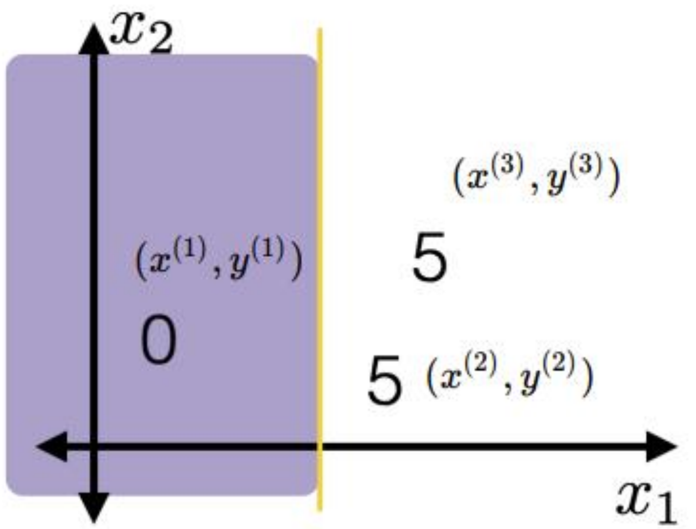


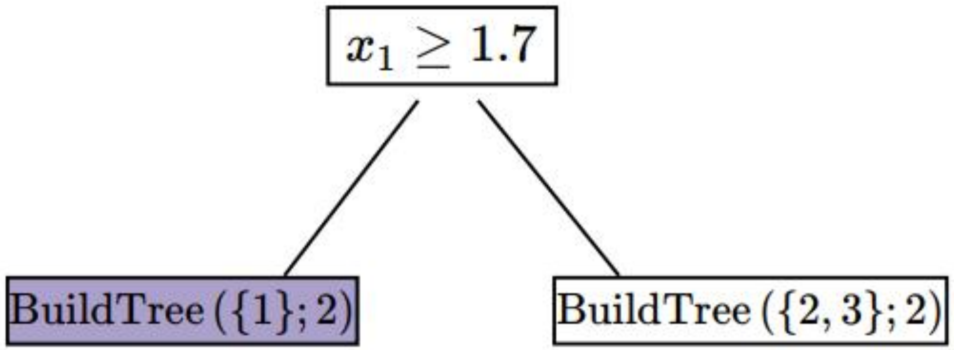
Line 12 recursion



BuildTree$(I, k, \mathcal{D})$

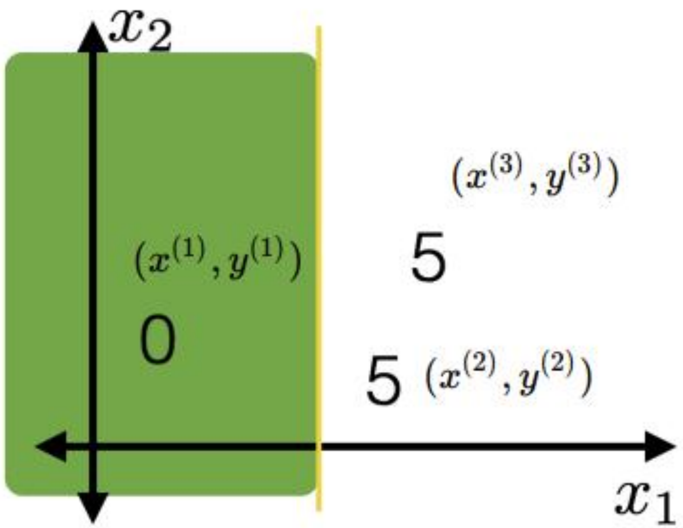
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \, y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \, y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+} \right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-} \right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I} \, y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k \right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k \right) \right)$

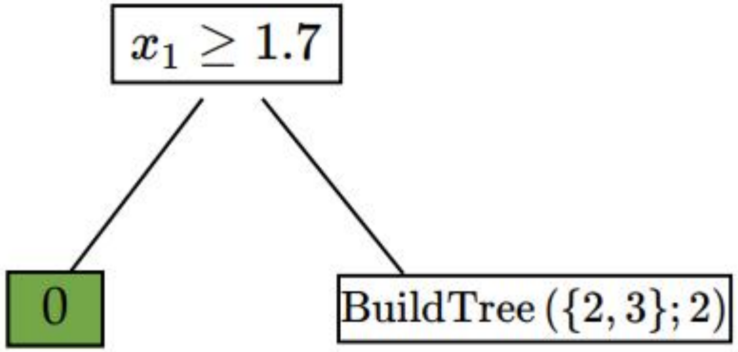




BuildTree$(I, k, \mathcal{D})$

1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \; y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \; y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+} \right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-} \right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I} \; y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k \right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k \right) \right)$





$\text{BuildTree}(I, k, \mathcal{D})$

if $|I| > k$

    for each split dim j and split value s

        Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$

        Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$

        Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}} \, y^{(i)}$

        Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}} \, y^{(i)}$

        Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+} \right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-} \right)^2$

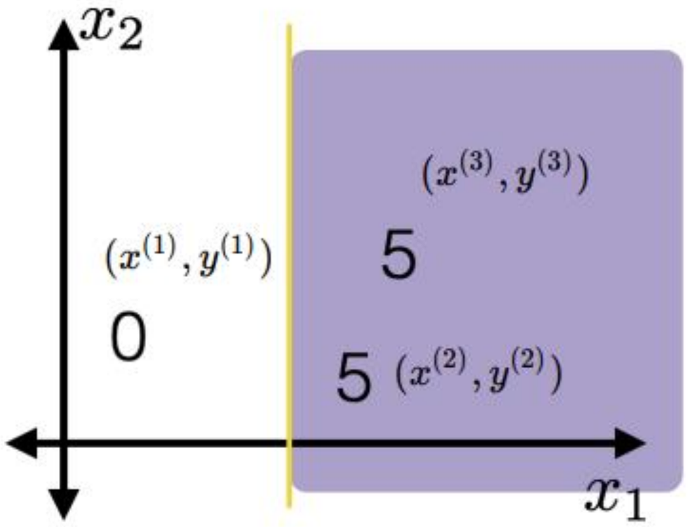
    Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$

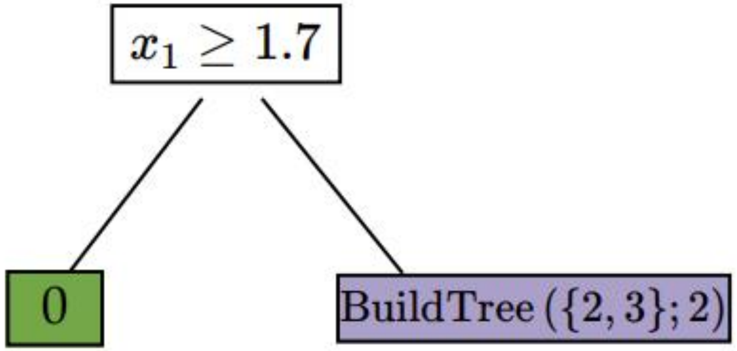
else

    Set $\hat{y} = \text{average}_{\, i \in I} \, y^{(i)}$

    return Leaf(leave_value=$\hat{y}$)

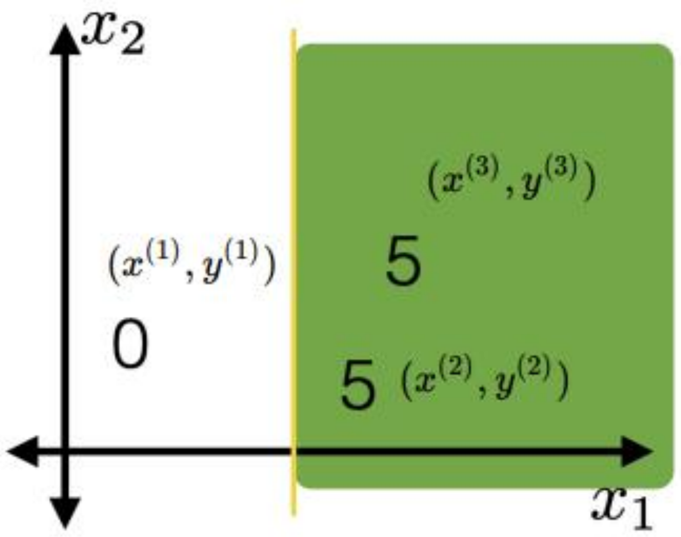
return Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k \right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k \right) \right)$

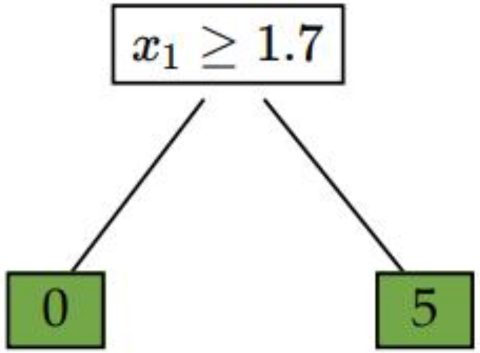




BuildTree$(I, k, \mathcal{D})$

1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\, i \in I_{j,s}^{+}}\, y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{average}_{\, i \in I_{j,s}^{-}}\, y^{(i)}$
7. Set $E_{j,s} = \sum_{i \in I_{j,s}^{+}} \left(y^{(i)} - \hat{y}_{j,s}^{+}\right)^2 + \sum_{i \in I_{j,s}^{-}} \left(y^{(i)} - \hat{y}_{j,s}^{-}\right)^2$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{average}_{\, i \in I}\, y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k\right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k\right)\right)$





For classification

BuildTree$(I, k, \mathcal{D})$

```
if |I| > k
    for each split dim j and split value s
        Set I⁺_{j,s} = { i ∈ I | x_j^(i) ≥ s }
        Set I⁻_{j,s} = { i ∈ I | x_j^(i) < s }
        Set ŷ⁺_{j,s} = majority_{i ∈ I⁺_{j,s}} y^(i)
        Set ŷ⁻_{j,s} = majority_{i ∈ I⁻_{j,s}} y^(i)
        Set E_{j,s} = (|I⁻_{j,s}| / |I|) · H(I⁻_{j,s}) + (|I⁺_{j,s}| / |I|) · H(I⁺_{j,s})
    Set (j*, s*) = arg min_{j,s} E_{j,s}
else
    Set ŷ = majority_{i ∈ I} y^(i)
    return Leaf(leave_value=ŷ)
return Node(j*, s*, BuildTree(I⁻_{j*,s*}, k), BuildTree(I⁺_{j*,s*}, k))
```

$$I^+_{j,s} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}, \quad I^-_{j,s} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$$

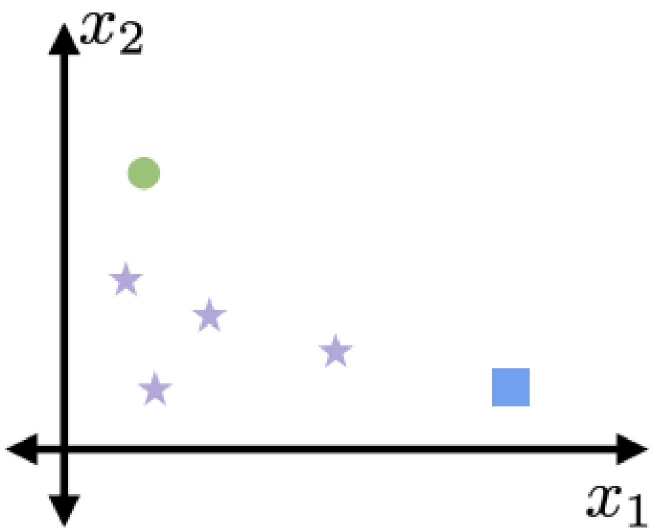
$$\hat{y}^+_{j,s} = \text{majority}_{\, i \in I^+_{j,s}}\, y^{(i)}, \quad \hat{y}^-_{j,s} = \text{majority}_{\, i \in I^-_{j,s}}\, y^{(i)}$$

$$E_{j,s} = \frac{|I^-_{j,s}|}{|I|} \cdot H\left(I^-_{j,s}\right) + \frac{|I^+_{j,s}|}{|I|} \cdot H\left(I^+_{j,s}\right)$$

use majority vote as (intermediate) prediction

use weighted average entropy as performance metric entropy $H = -\sum_{\text{class } c} \hat{P}_c(\log_2 \hat{P}_c)$

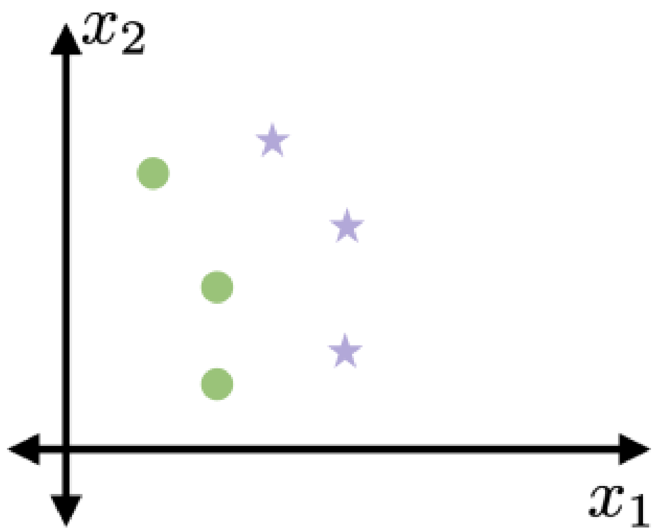
for example: three classes ★ ● ■



$\hat{P}_c$ ★ $\frac{4}{6}$ ● $\frac{1}{6}$ ■ $\frac{1}{6}$

$H = -[\frac{4}{6}\log_2\left(\frac{4}{6}\right) + \frac{1}{6}\log_2\left(\frac{1}{6}\right) + \frac{1}{6}\log_2\left(\frac{1}{6}\right)]$

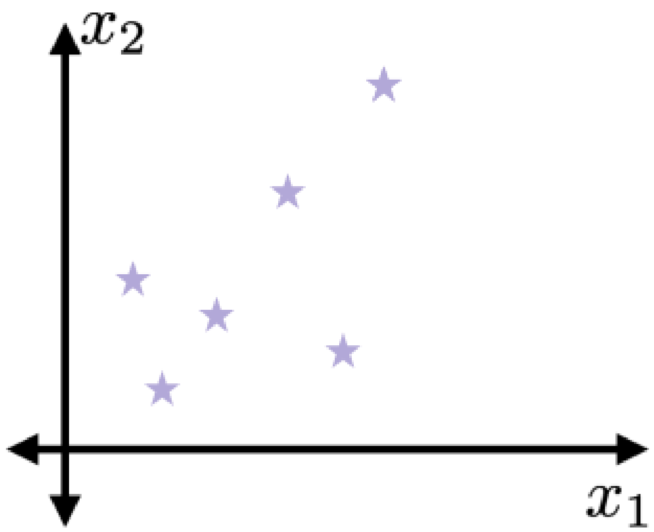
(about 1.252)



$\hat{P}_c$ ★ $\frac{3}{6}$ ● $\frac{3}{6}$ ■ 0

$H = -[\frac{3}{6}\log_2\left(\frac{3}{6}\right) + \frac{3}{6}\log_2\left(\frac{3}{6}\right) + 0]$

(about 1.1)



$\hat{P}_c$ ★ $\frac{6}{6}$ ● 0 ■ 0

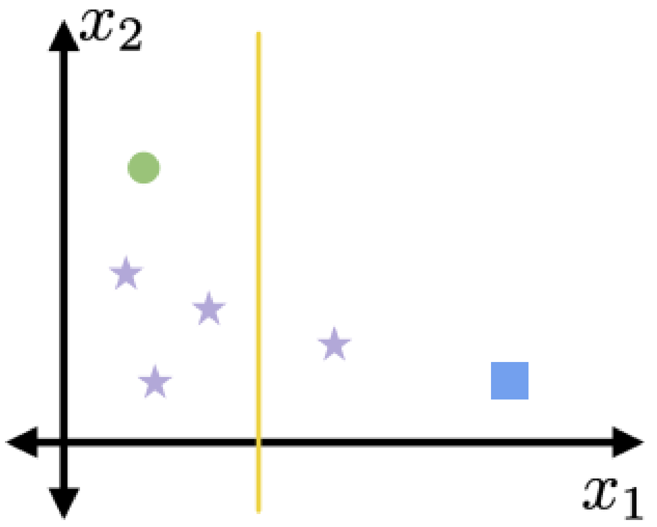
$H = -[\frac{6}{6}\log_2\left(\frac{6}{6}\right) + 0 + 0]$

$(= 0)$

Adopted from : https://introml.mit.edu

BuildTree$(I, k, \mathcal{D})$

```
if |I| > k
    for each split dim j and split value s
        Set I^+_{j,s} = { i ∈ I | x_j^(i) ≥ s }
        Set I^-_{j,s} = { i ∈ I | x_j^(i) < s }
        Set ŷ^+_{j,s} = majority_{i ∈ I^+_{j,s}} y^(i)
        Set ŷ^-_{j,s} = majority_{i ∈ I^-_{j,s}} y^(i)
        Set E_{j,s} = |I^-_{j,s}|/|I| · H(I^-_{j,s}) + |I^+_{j,s}|/|I| · H(I^+_{j,s})
    Set (j*, s*) = arg min_{j,s} E_{j,s}
else
    Set ŷ = majority_{i ∈ I} y^(i)
    return Leaf(leave_value=ŷ)
return Node(j*, s*, BuildTree(I^-_{j*,s*}, k), BuildTree(I^+_{j*,s*}, k))
```



$$\frac{|I^-_{j,s}|}{|I|} \cdot H\left(I^-_{j,s}\right) + \frac{|I^+_{j,s}|}{|I|} \cdot H\left(I^+_{j,s}\right)$$

$$= \frac{4}{6} \cdot H\left(I^-_{j,s}\right) + \frac{2}{6} \cdot H\left(I^+_{j,s}\right)$$

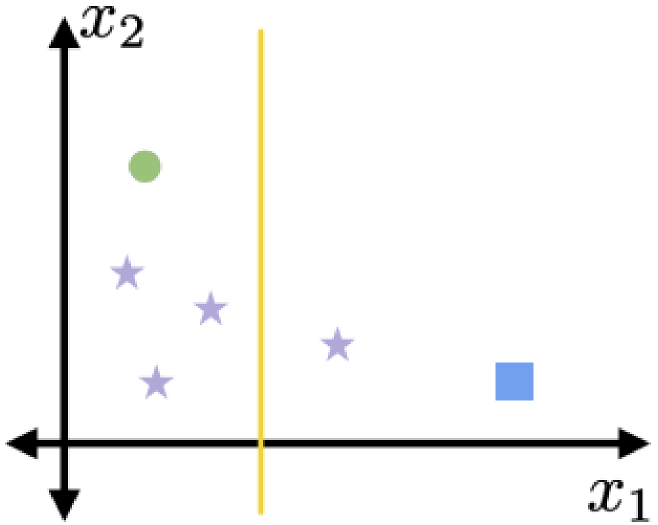
fraction to the left of the split

fraction to the right of the split

Adopted from : https://introml.mit.edu

BuildTree$(I, k, \mathcal{D})$

1. **if** $|I| > k$
2. **for each** split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{majority}_{\, i \in I_{j,s}^{+}} \; y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{majority}_{\, i \in I_{j,s}^{-}} \; y^{(i)}$
7. Set $E_{j,s} = \frac{|I_{j,s}^{-}|}{|I|} \cdot H\left(I_{j,s}^{-}\right) + \frac{|I_{j,s}^{+}|}{|I|} \cdot H\left(I_{j,s}^{+}\right)$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{majority}_{\, i \in I} \; y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k\right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k\right)\right)$



$$-\left[\tfrac{3}{4}\log_2\left(\tfrac{3}{4}\right) + \tfrac{1}{4}\log_2\left(\tfrac{1}{4}\right) + 0\right] \approx 0.811$$

$$\frac{4}{6} \cdot H\left(I_{j,s}^{-}\right) + \frac{2}{6} \cdot H\left(I_{j,s}^{+}\right)$$

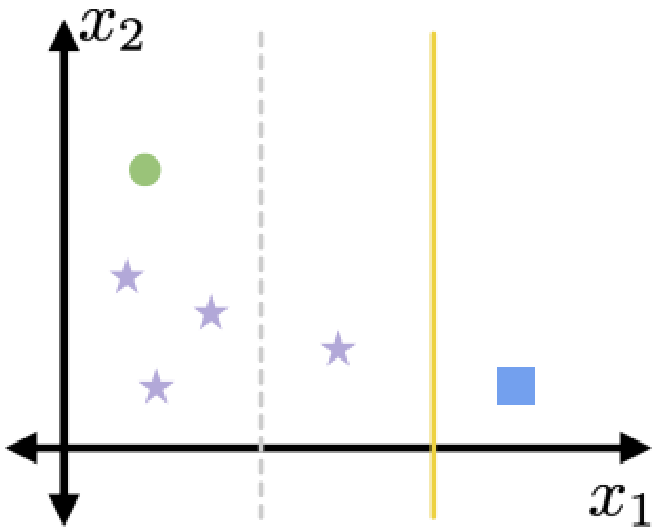
$$-\left[\tfrac{1}{2}\log_2\left(\tfrac{1}{2}\right) + \tfrac{1}{2}\log_2\left(\tfrac{1}{2}\right) + 0\right] = 1$$

(line 7, overall $E_{j,s} \approx 0.874$)

Adopted from : https://introml.mit.edu

BuildTree$(I, k, \mathcal{D})$

1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{majority}_{\, i \in I_{j,s}^{+}}\, y^{(i)}$
6. Set $\hat{y}_{j,s}^{-} = \text{majority}_{\, i \in I_{j,s}^{-}}\, y^{(i)}$
7. Set $E_{j,s} = \frac{|I_{j,s}^{-}|}{|I|} \cdot H\left(I_{j,s}^{-}\right) + \frac{|I_{j,s}^{+}|}{|I|} \cdot H\left(I_{j,s}^{+}\right)$
8. Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
9. **else**
10. Set $\hat{y} = \text{majority}_{\, i \in I}\, y^{(i)}$
11. **return** Leaf(leave_value=$\hat{y}$)
12. **return** Node $\left(j^*, s^*, \text{BuildTree}\left(I_{j^*,s^*}^{-}, k\right), \text{BuildTree}\left(I_{j^*,s^*}^{+}, k\right)\right)$



line 7
$E_{j,s} \approx 0.874$

overall $E_{j,s} \approx 0.602$

line 8, set the better (j, s)

Ensemble

≡ WIRED SUBSCRIBE

ELIOT VAN BUSKIRK BUSINESS 09.22.2009 11:19 AM

How the Netflix Prize Was Won

Like BellKor's Pragmatic Chaos, the winner of the Netflix Prize, second-place The Ensemble was an amalgam of teams which had been competing individually for the million-dollar prize. But it wasn't until leaders joined forces with also-rans that real progress was made in the Netflix movie recommenda [...]

ELSEVIER International Journal of Forecasting

Volume 36, Issue 1, January–March 2020, Pages 54-74

The M4 Competition: 100,000 time series and 61 forecasting methods

Spyros Makridakis [a], Evangelos Spiliotis [b], Vassilios Assimakopoulos [b]

Coronavirus Disease MENU >

CASES, DATA & SURVEILLANCE

Forecasts of COVID-19 Deaths

Updated Nov. 12, 2020

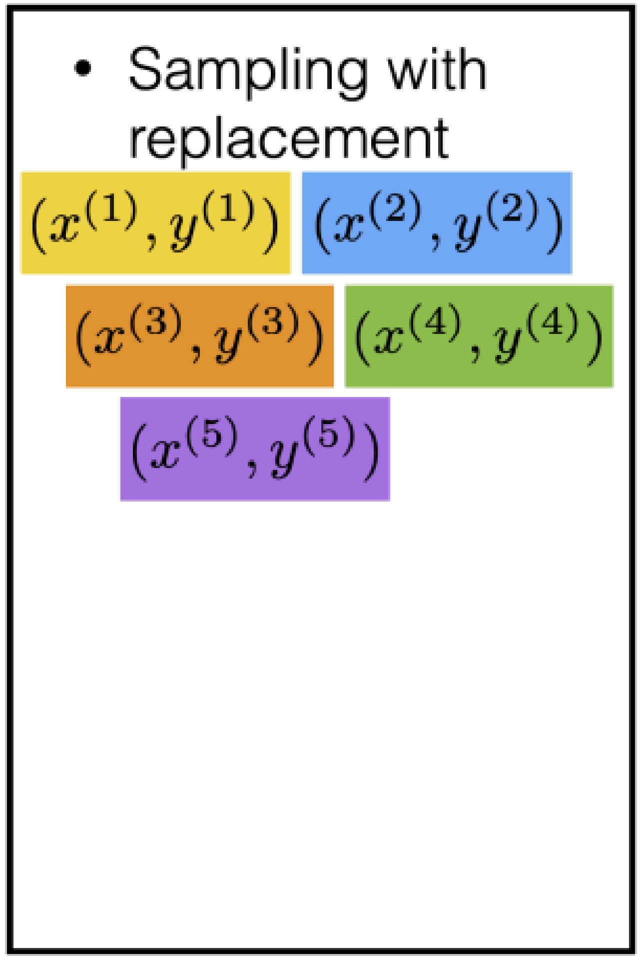
Observed and forecasted new and total reported COVID-19 deaths as of November 9, 2020.

Interpretation of Forecasts of New and Total Deaths

- This week CDC received forecasts of COVID-19 deaths over the next 4 weeks from 36 modeling groups that were included in the ensemble forecast. Of the 36 groups, 33 provided forecasts for both new and total deaths, two groups forecasted total deaths only, and one forecasted new death only.

Bagging

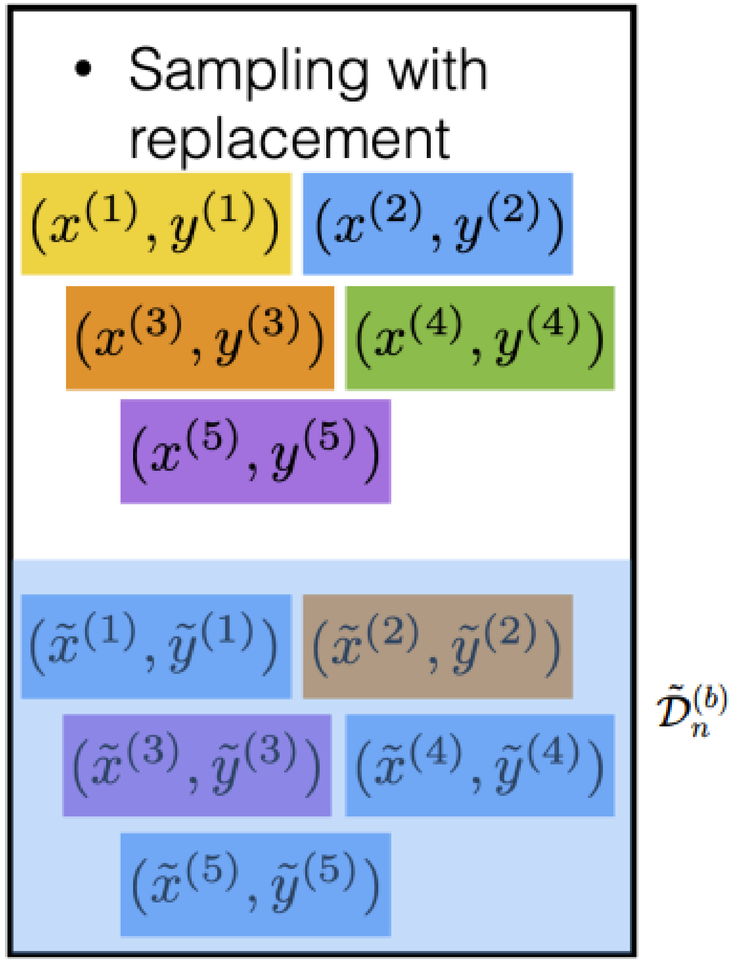
- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regat**ing**
  - Training data $\mathcal{D}_n$



Adopted from : https://introml.mit.edu

Bagging

- Training data $\mathcal{D}_n$
- For $b = 1, \ldots, B$
  - Draw a new "data set" $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling with replacement from $\mathcal{D}_n$
  - Train a predictor $\hat{f}^{(b)}$ on $\tilde{\mathcal{D}}_n^{(b)}$
- Return
  - For regression: $\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{(b)}(x)$
  - For classification: predictor at a point is class with highest vote count at that point



Outline

- Recap: parameterized models
- Non-parametric models
  - Interpretability
  - Ease of use and simplicity
- Decision Tree
  - `BuildTree`
- Nearest Neighbor

Nearest neighbor

- Training: None (or rather: memorize the entire training data)

  Hyper-parameter: k

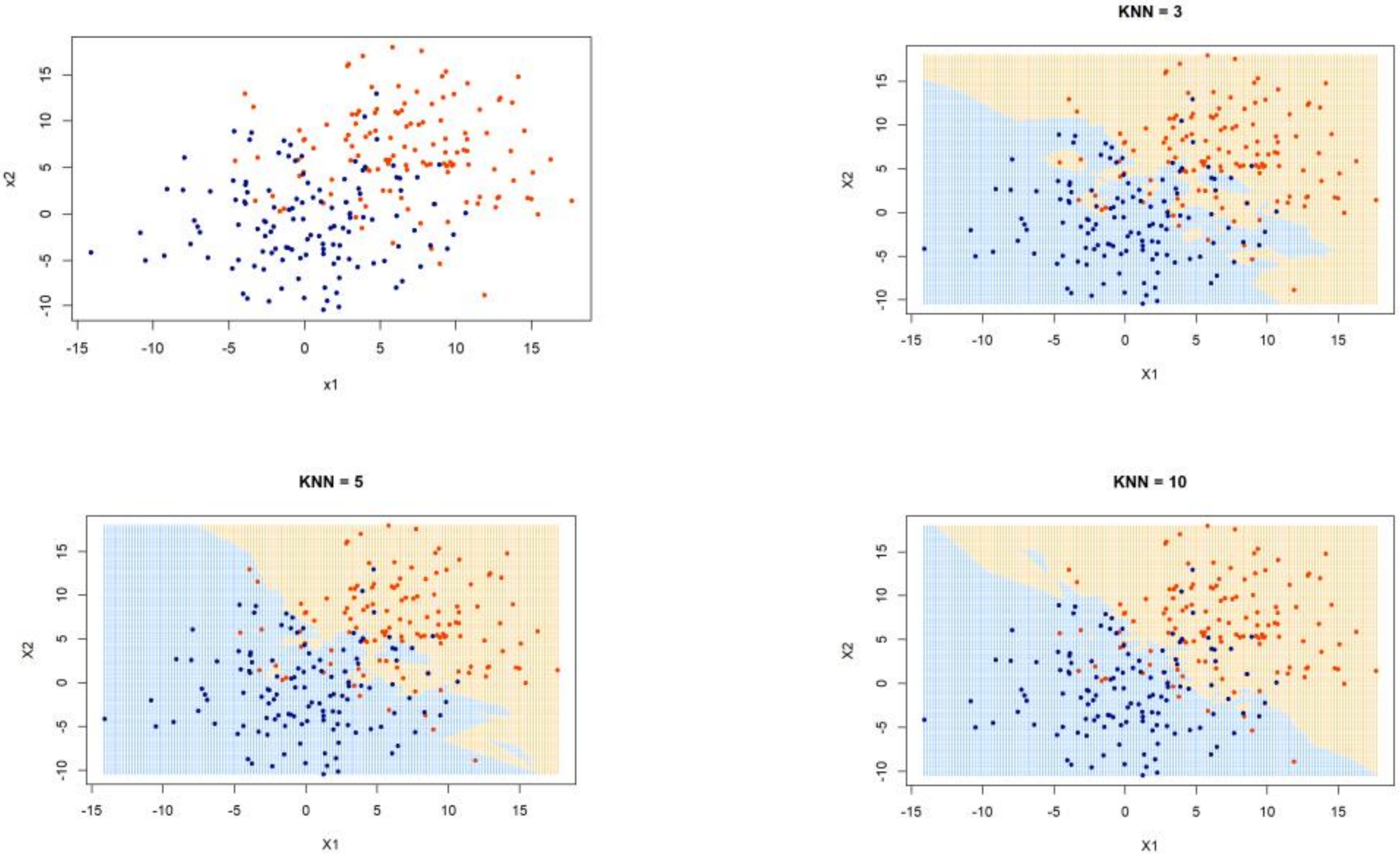
  Distance metric (typically Euclidean or Manhattan distance)

  A tie-breaking scheme (typically at random)

- Predicting (inferencing, testing):

  - for a new data point x_{new} do:
    - find the k points in training data nearest to x_{new}
      - For classification: predict label $\hat{y}_{\text{new}}$ for x_{new} by taking a majority vote of the k neighbors's labels y
      - For regression: predict label $\hat{y}_{\text{new}}$ for x_{new} by taking an average over the k neighbors' labels y

Adopted from : https://introml.mit.edu



Summary

- One really important class of ML models is called “non-parametric”.
- Decision trees are kind of like creating a flow chart. These hypotheses are the most human-understandable of any we have worked with.We regularize by first growing trees that are very big and then “pruning” them.
- Ensembles: sometimes it's useful to come up with a lot of simple hypotheses and then let them “vote” to make a prediction for a new example.
- Nearest neighbor remembers all the training data for prediction. Depends crucially on our notion of “closest” (standardize data is important). Can do fancier things (weighted kNN).Less good in high dimensions (computationally expensive).