

6.036: Introduction to Machine Learning

Anonymous
mid-semester survey:
<https://bit.ly/3Bwr53z>

Lecture start: Tuesdays 9:35am

Who's talking? Prof. Tamara Broderick

Questions? Ask on Piazza: "lecture (week) 9" folder

Materials: slides, video will all be available on Canvas

Live Zoom feed: <https://mit.zoom.us/j/94238622313>

Last Time(s)

- I. Neural networks
 - 2 layers
 - More layers:
Convolutional neural nets

Today's Plan

- I. Decision trees
- II. Bagging/Ensembling
- III. Random forests

Predictive performance and beyond

Predictive performance and beyond

- There's more to machine learning than predictive performance

Predictive performance and beyond

- There's more to machine learning than predictive performance

How some election officials are trying to verify the vote more easily

Oct 29, 2020 6:20 PM EST

Predictive performance and beyond

- There's more to machine learning than predictive performance

How some election officials are trying to verify the vote more easily

Oct 29, 2020 6:20 PM EST

“The choices were made to simplify the exposition and implementation: methods need to be transparent to be adopted as part of the election process and to inspire public confidence. [An alternative] might be more efficient, but because of its complexity would likely meet resistance from elections officials and voting rights groups.” — Philip Stark, 2008

Predictive performance and beyond

- There's more to machine learning than predictive performance

How some election officials are trying to verify the vote more easily

Oct 29, 2020 6:20 PM EST

“The choices were made to simplify the exposition and implementation: methods need to be transparent to be adopted as part of the election process and to inspire public confidence. [An alternative] might be more efficient, but because of its complexity would likely meet resistance from elections officials and voting rights groups.” — Philip Stark, 2008

Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission

Rich Caruana
Microsoft Research
rcaruana@microsoft.com

Paul Koch
Microsoft Research
paulkoch@microsoft.com

Yin Lou
LinkedIn Corporation
ylou@linkedin.com

Marc Sturm
NewYork-Presbyterian Hospital
mas9161@nyp.org

Johannes Gehrke
Microsoft
johannes@microsoft.com

Noémie Elhadad
Columbia University
noemie.elhadad@columbia.edu

Predictive performance and beyond

- There's more to machine learning than predictive performance

How some election officials are trying to verify the vote more easily

Oct 29, 2020 6:20 PM EST

“The choices were made to simplify the exposition and implementation: methods need to be transparent to be adopted as part of the election process and to inspire public confidence. [An alternative] might be more efficient, but because of its complexity would likely meet resistance from elections officials and voting rights groups.” — Philip Stark, 2008

Intelligible Models for HealthCare: Predicting Pneumonia Risk and Hospital 30-day Readmission

Rich Caruana
Microsoft Research
rcaruana@microsoft.com

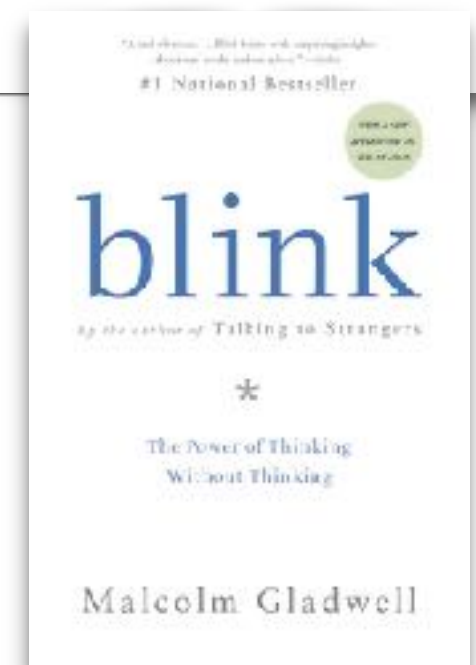
Paul Koch
Microsoft Research
paulkoch@microsoft.com

Yin Lou
LinkedIn Corporation
ylou@linkedin.com

Marc Sturm
NewYork-Presbyterian Hospital
mas9161@nyp.org

Johannes Gehrke
Microsoft
johannes@microsoft.com

Noémie Elhadad
Columbia University
noemie.elhadad@columbia.edu



Predictive performance and beyond

Predictive performance and beyond

- Even if all you care about is vanilla predictive performance, you should care about trees

Predictive performance and beyond

- Even if all you care about is vanilla predictive performance, you should care about trees

kaggle

Predictive performance and beyond

- Even if all you care about is vanilla predictive performance, you should care about trees

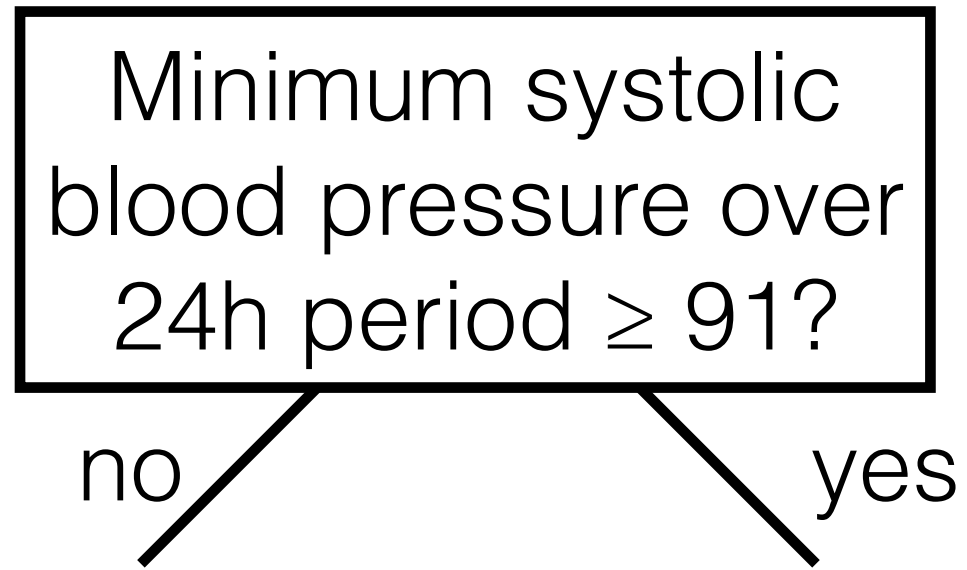
The Kaggle logo is displayed in a large, light blue, lowercase sans-serif font.

A look at Mathurin's toolkit, which he keeps coming back to:

- **Packages:** scikit learn, pandas, numpy
- **Frameworks:** Keras, Tensorflow, Pytorch and Fastai
- **Algorithms:** lightgbm, xgboost, catboost
- **AutoML tools:** Prevision.io, h2o and other open sources such as TPOT, auto sklearn
- **Cloud services:** Google colab and kaggle kernels

Decision tree

Decision tree



Decision tree

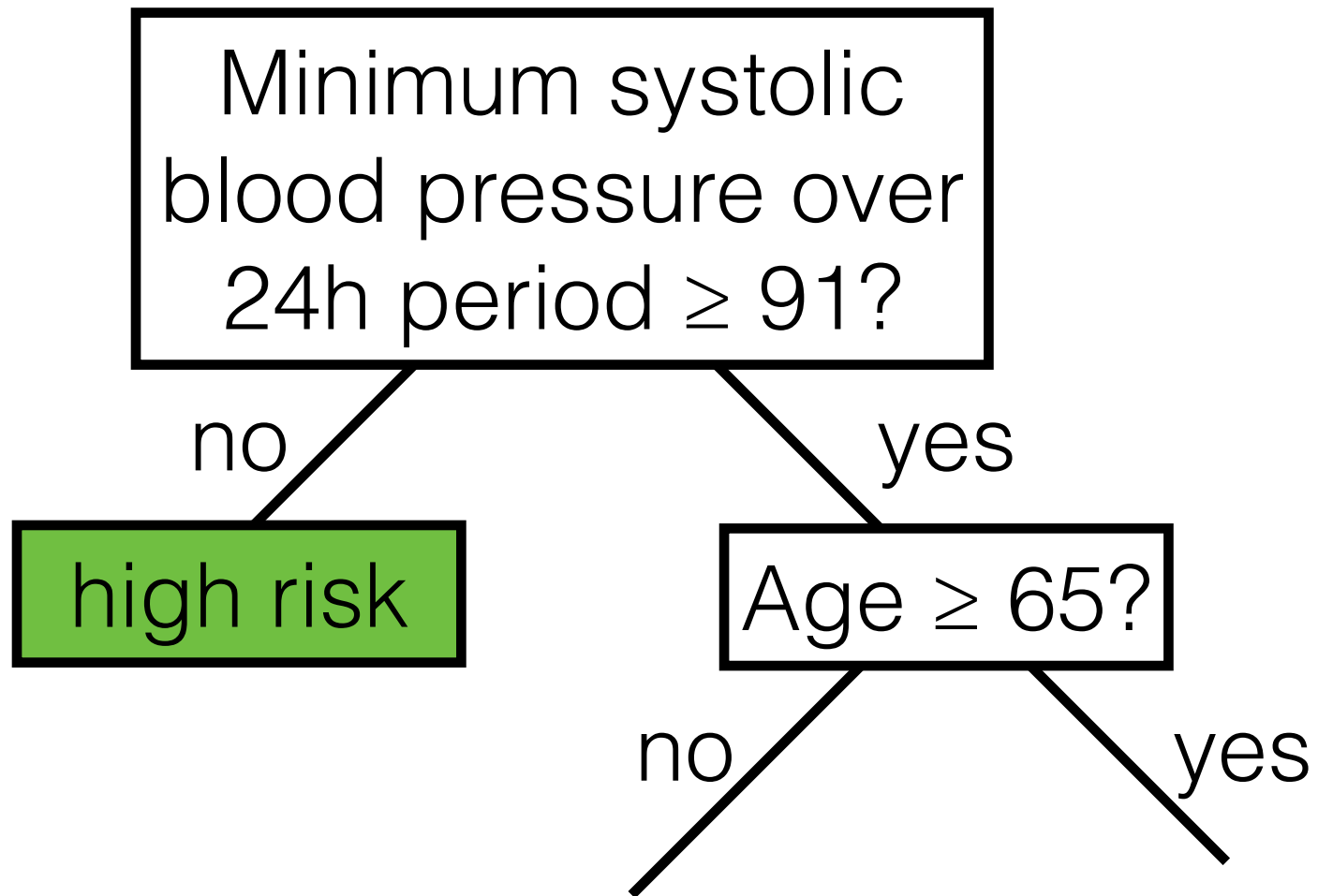
Minimum systolic
blood pressure over
24h period ≥ 91 ?

no

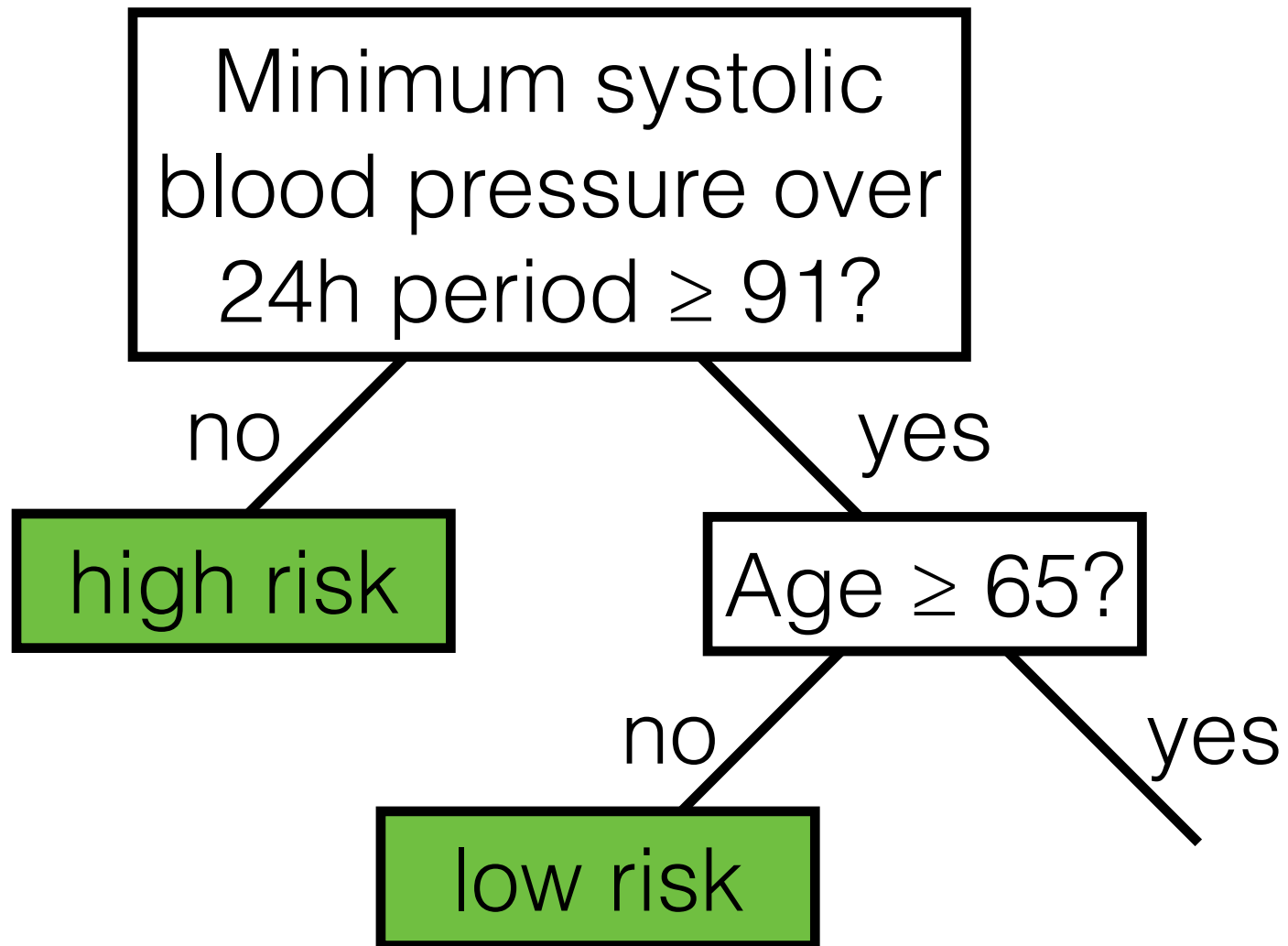
yes

high risk

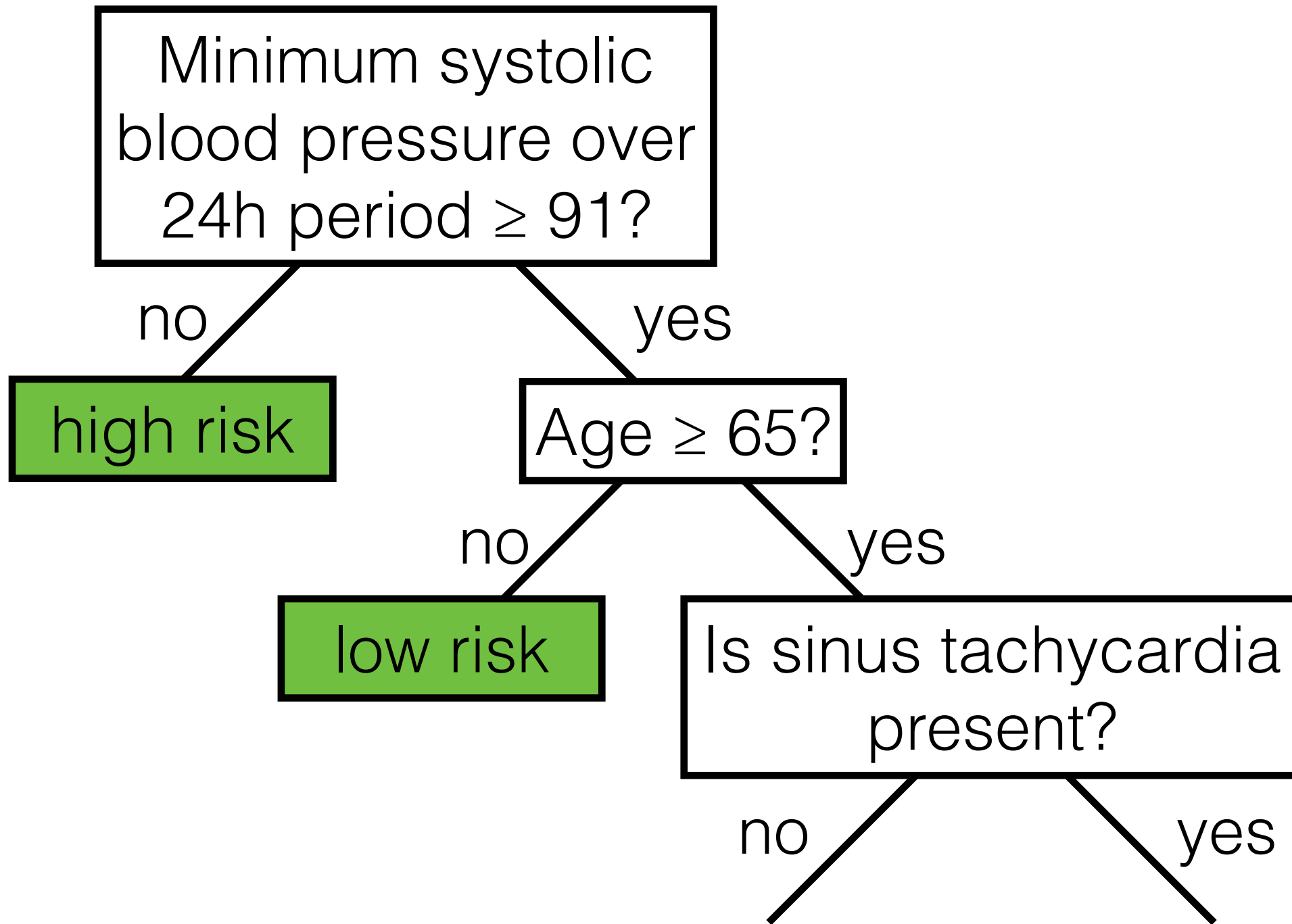
Decision tree



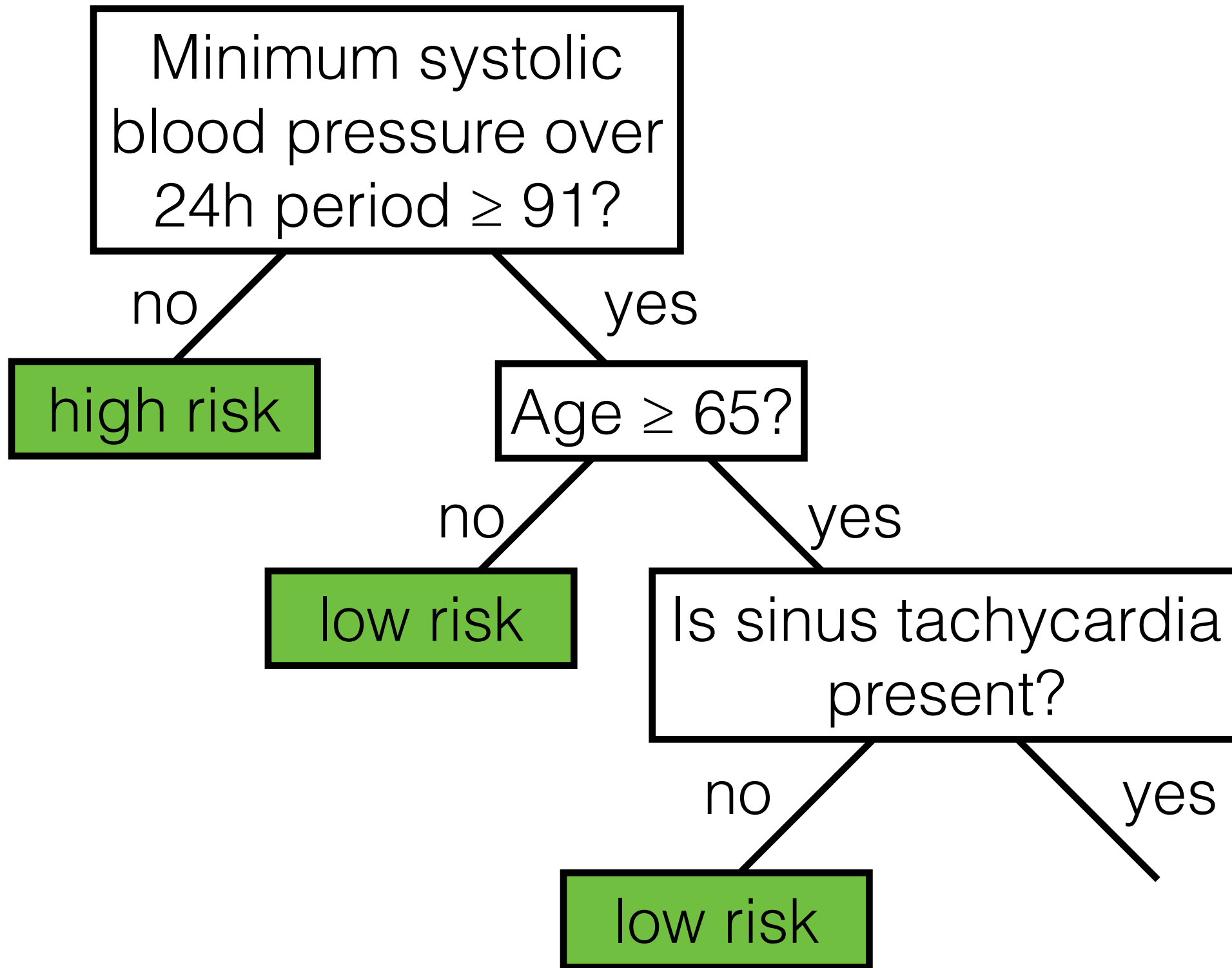
Decision tree



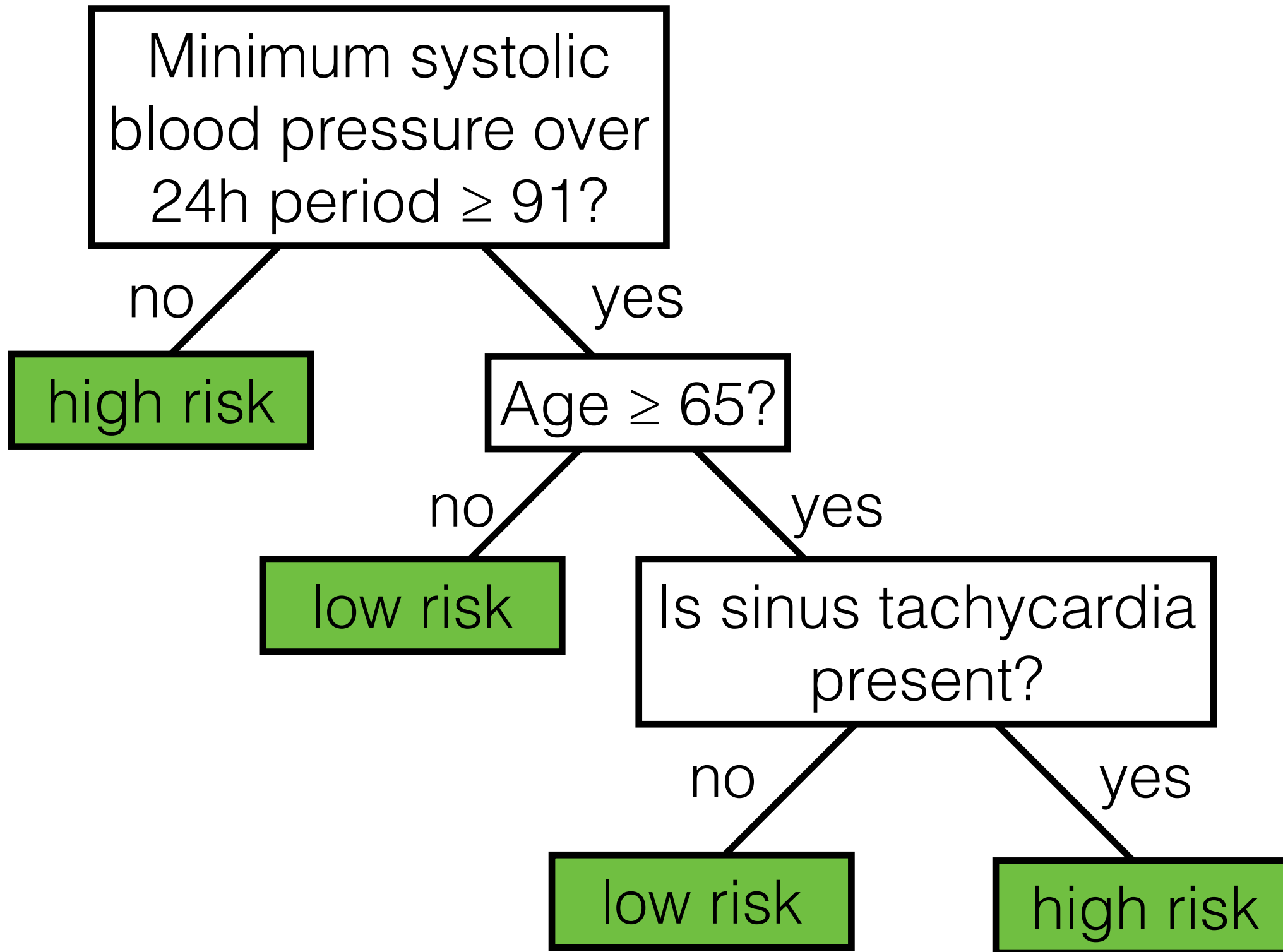
Decision tree



Decision tree



Decision tree



Decision tree

features:

x_1 : date

x_2 : age

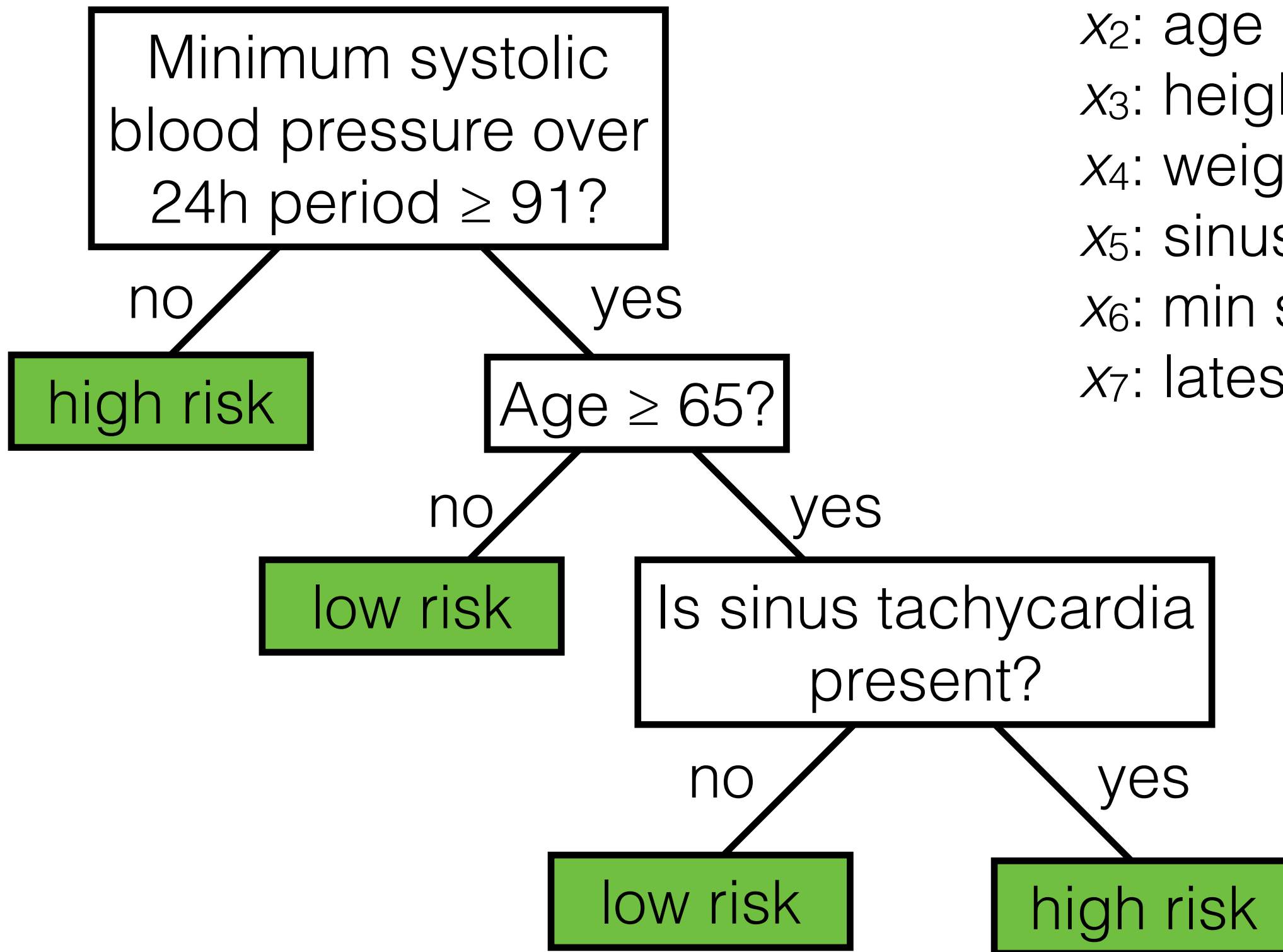
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

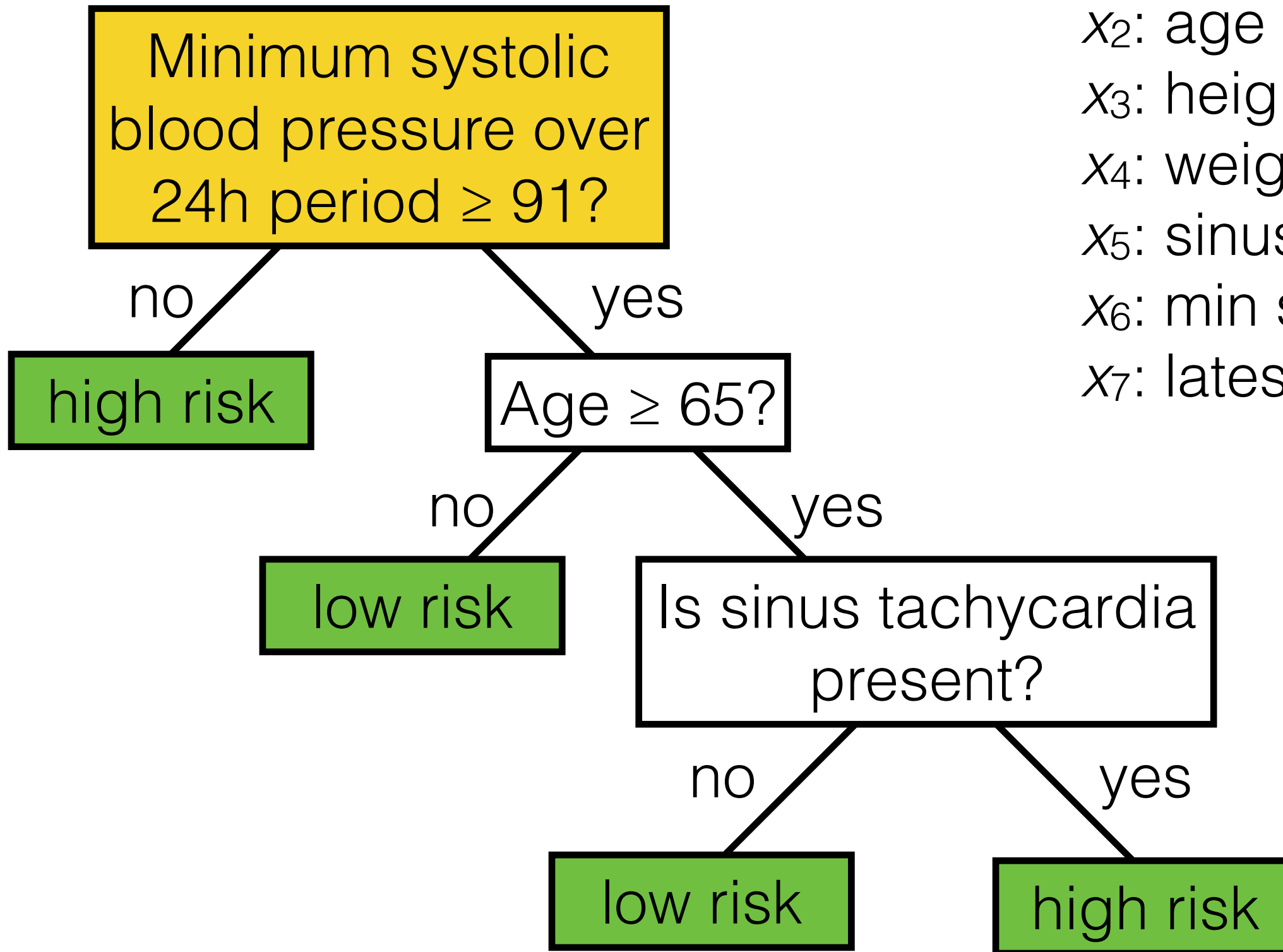
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

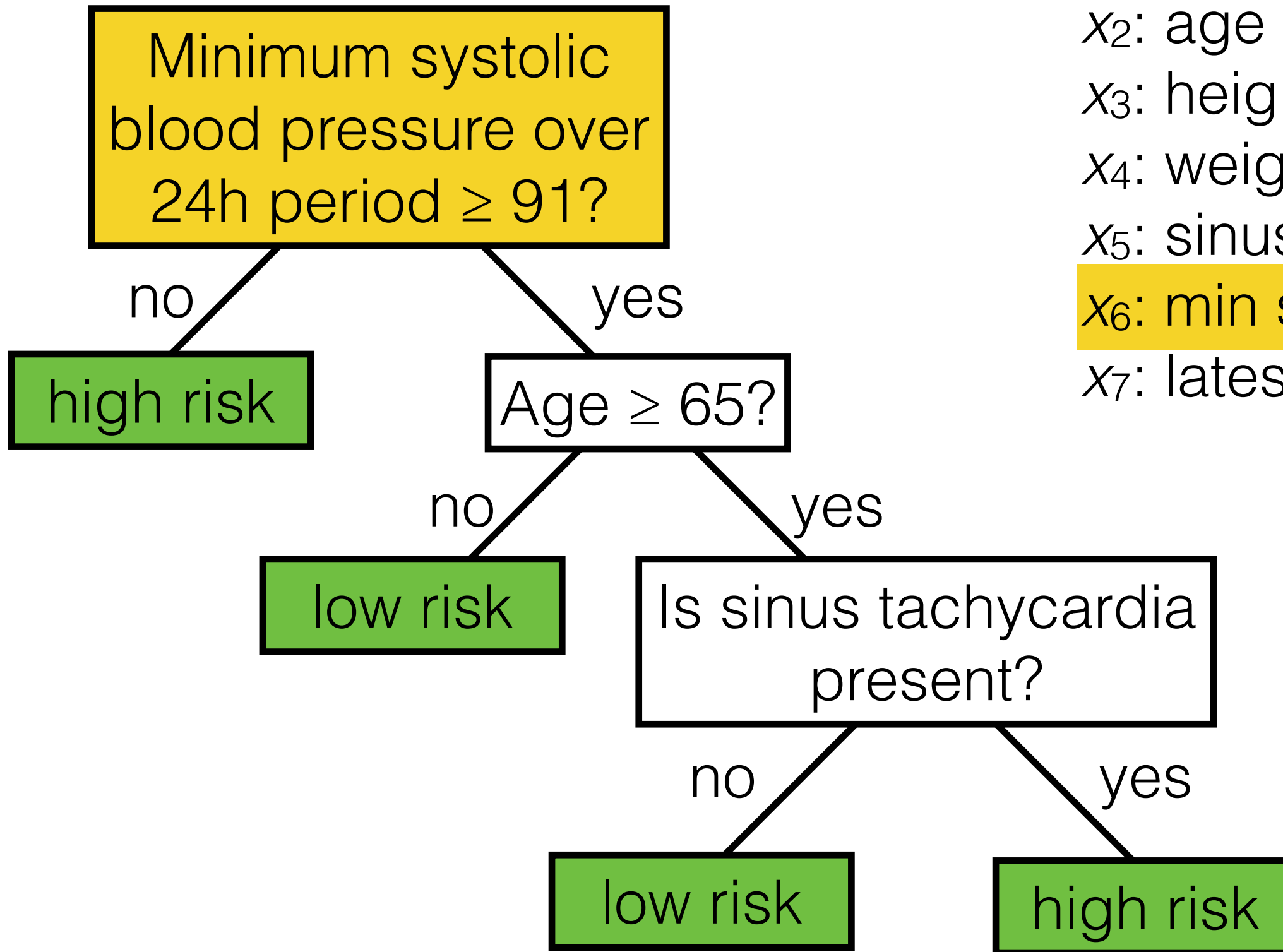
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

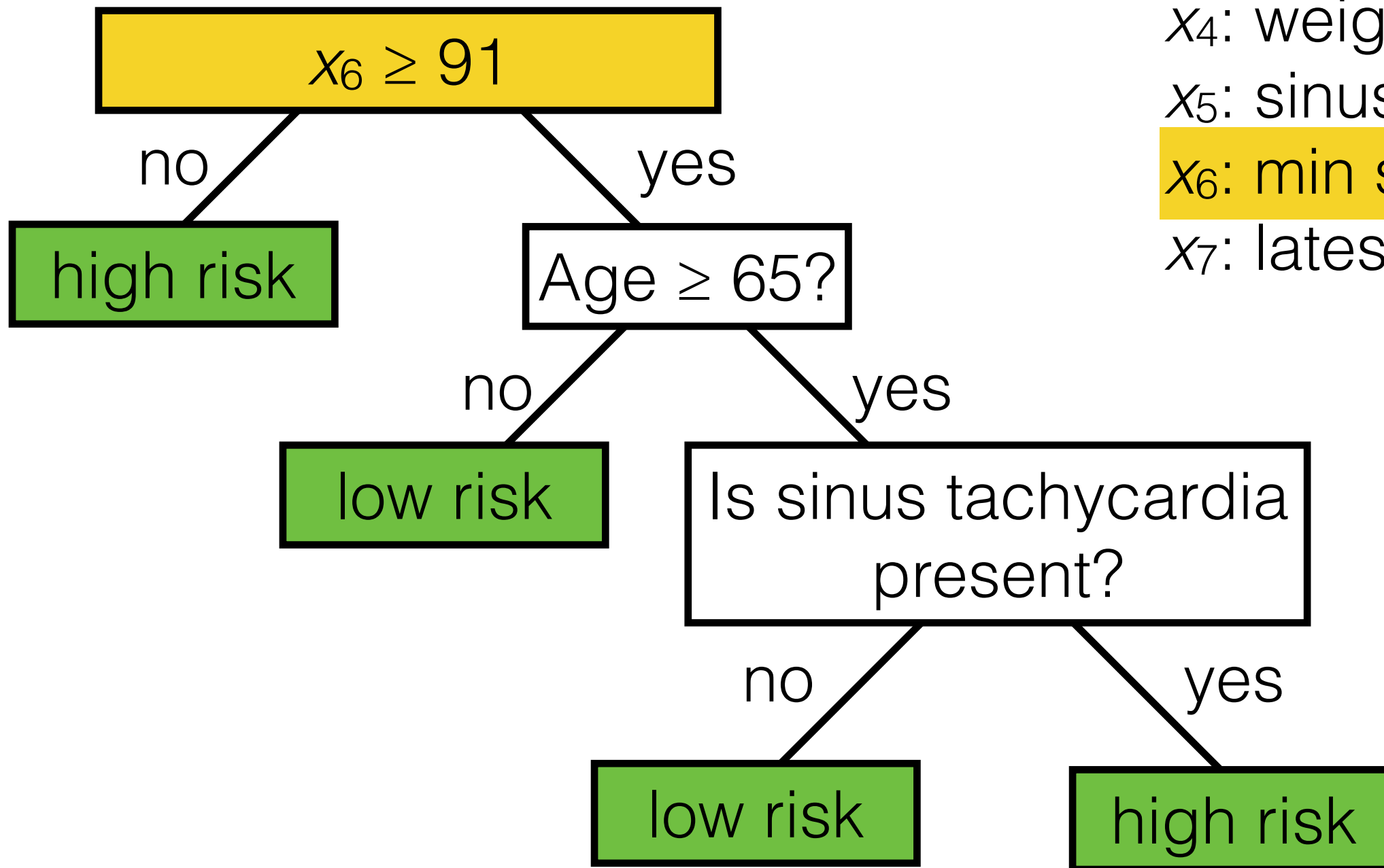
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

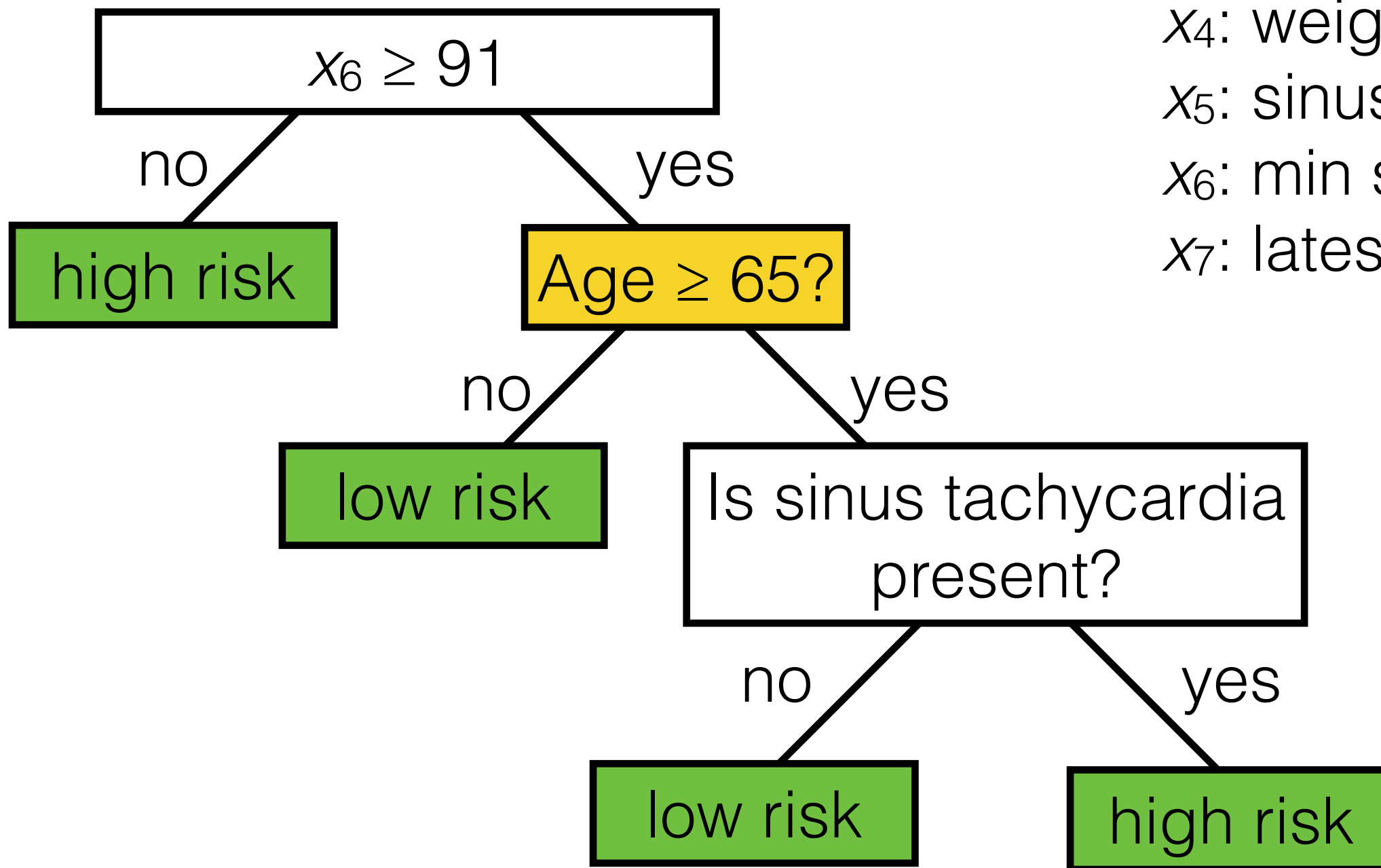
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

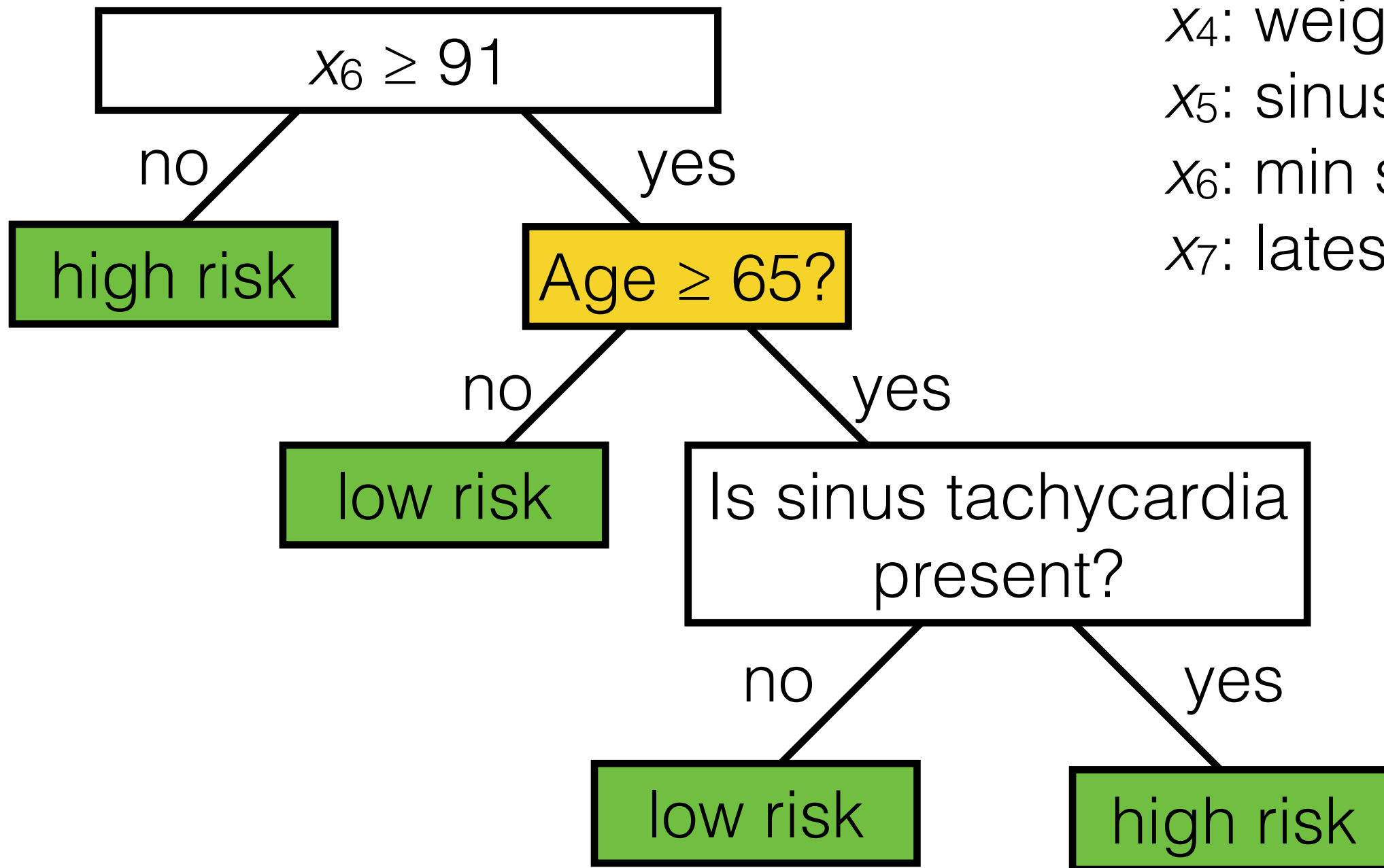
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

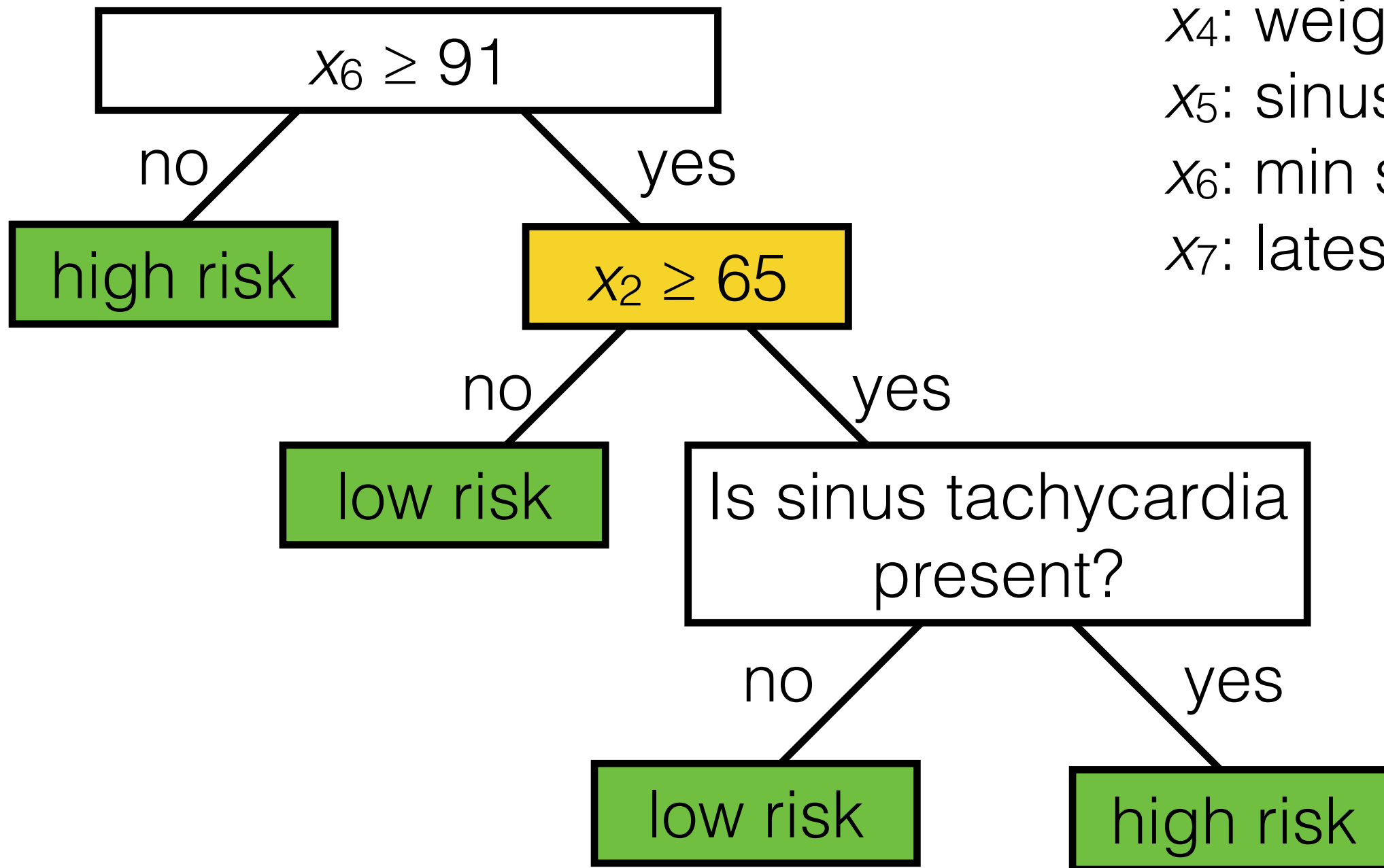
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

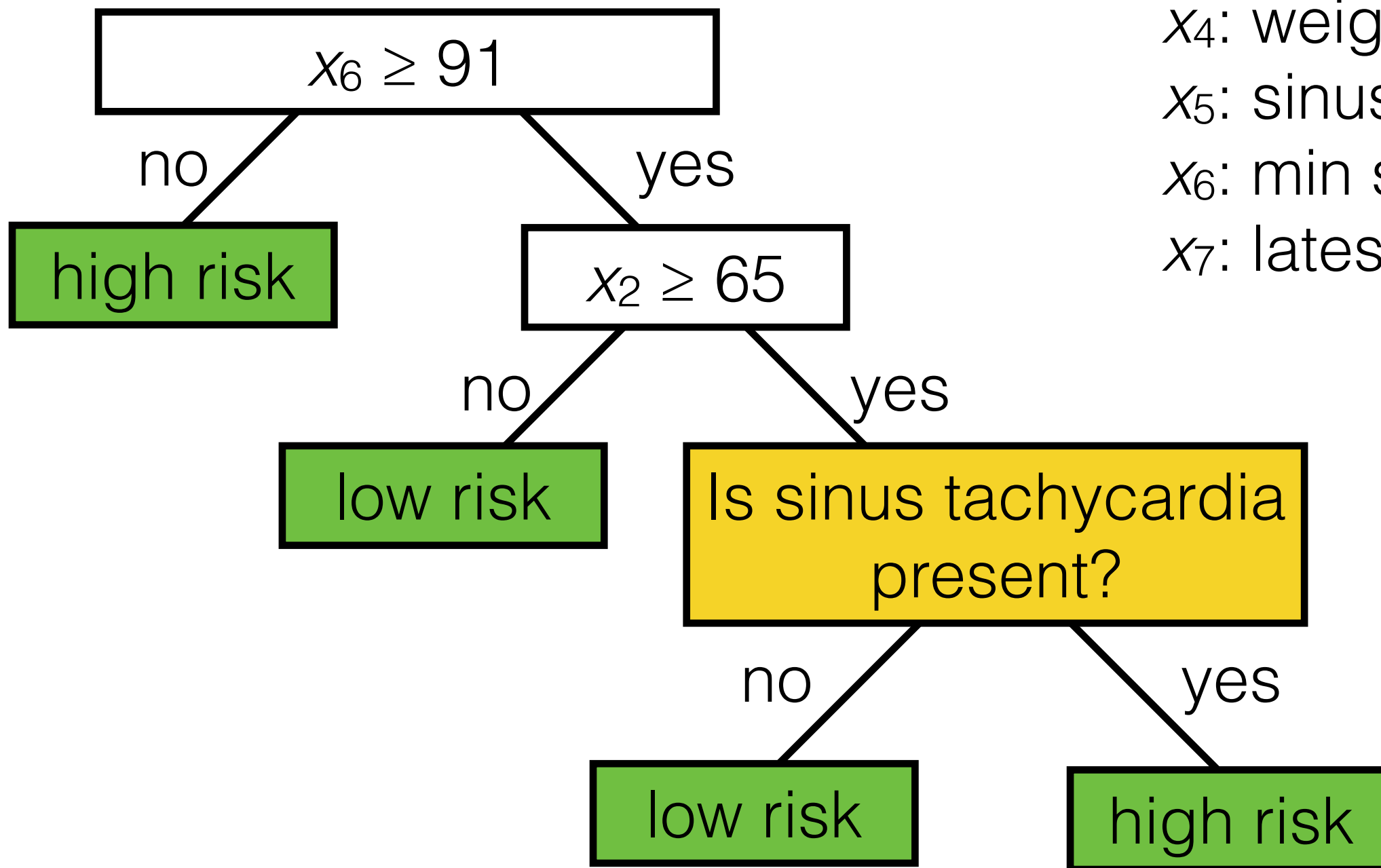
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

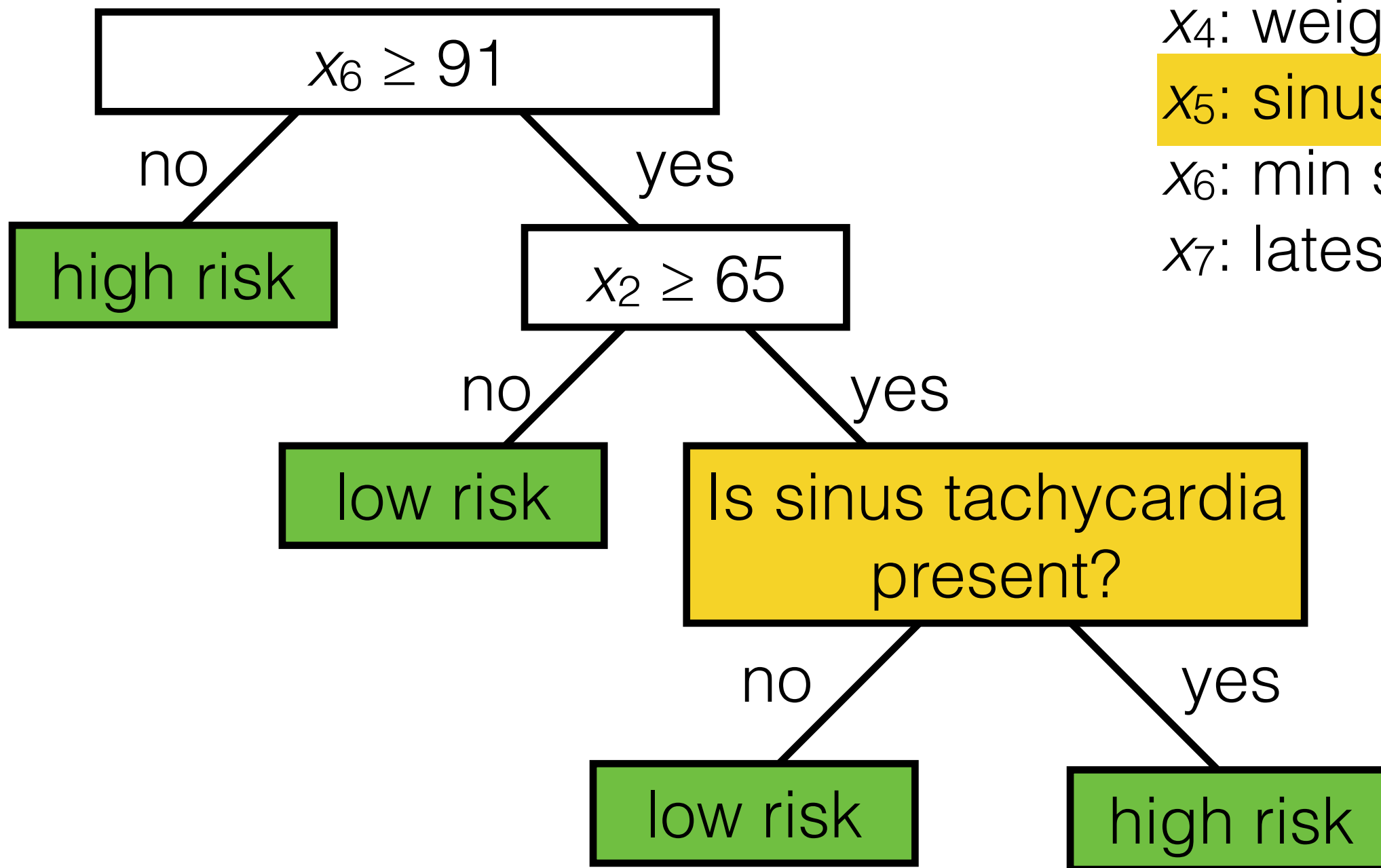
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

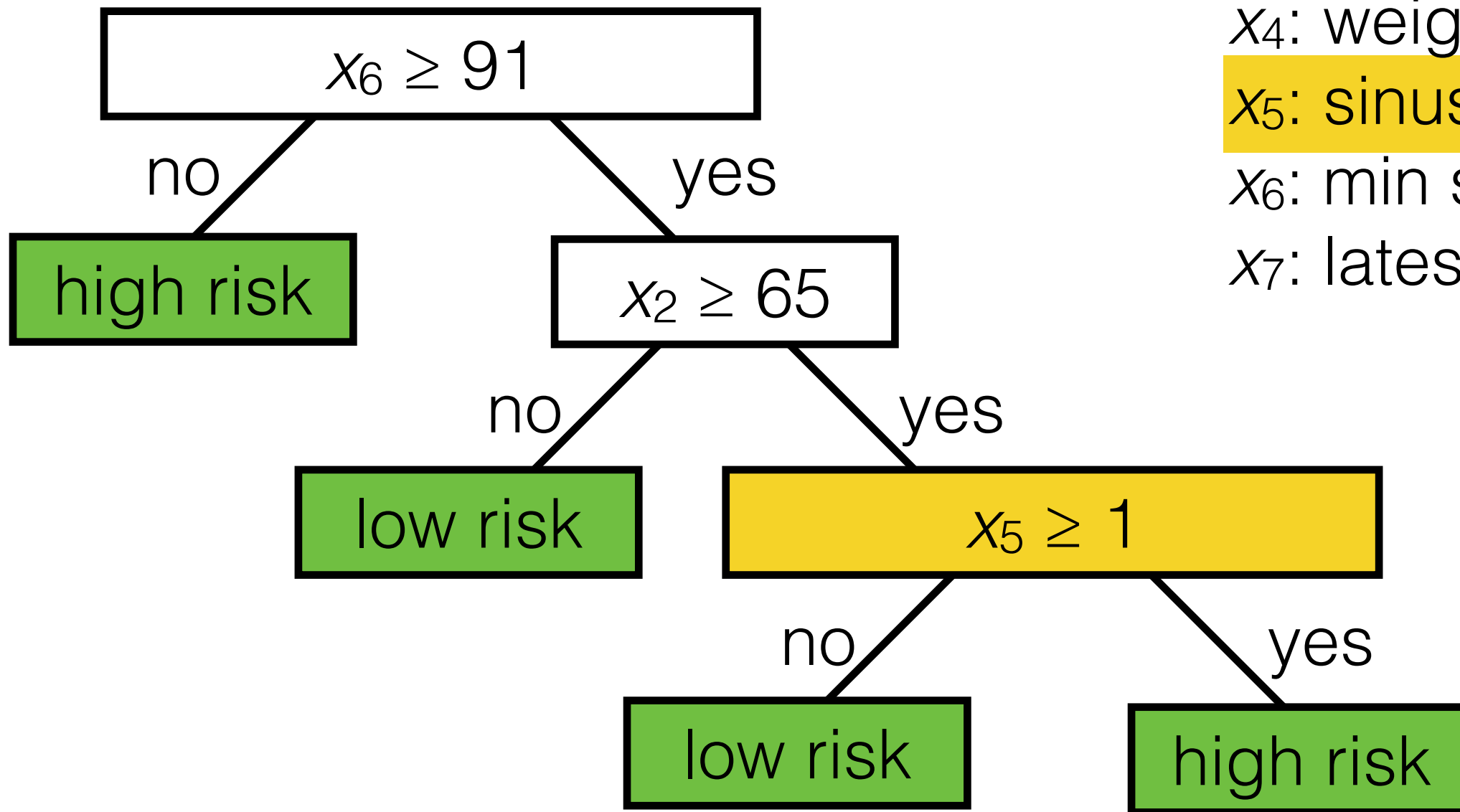
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

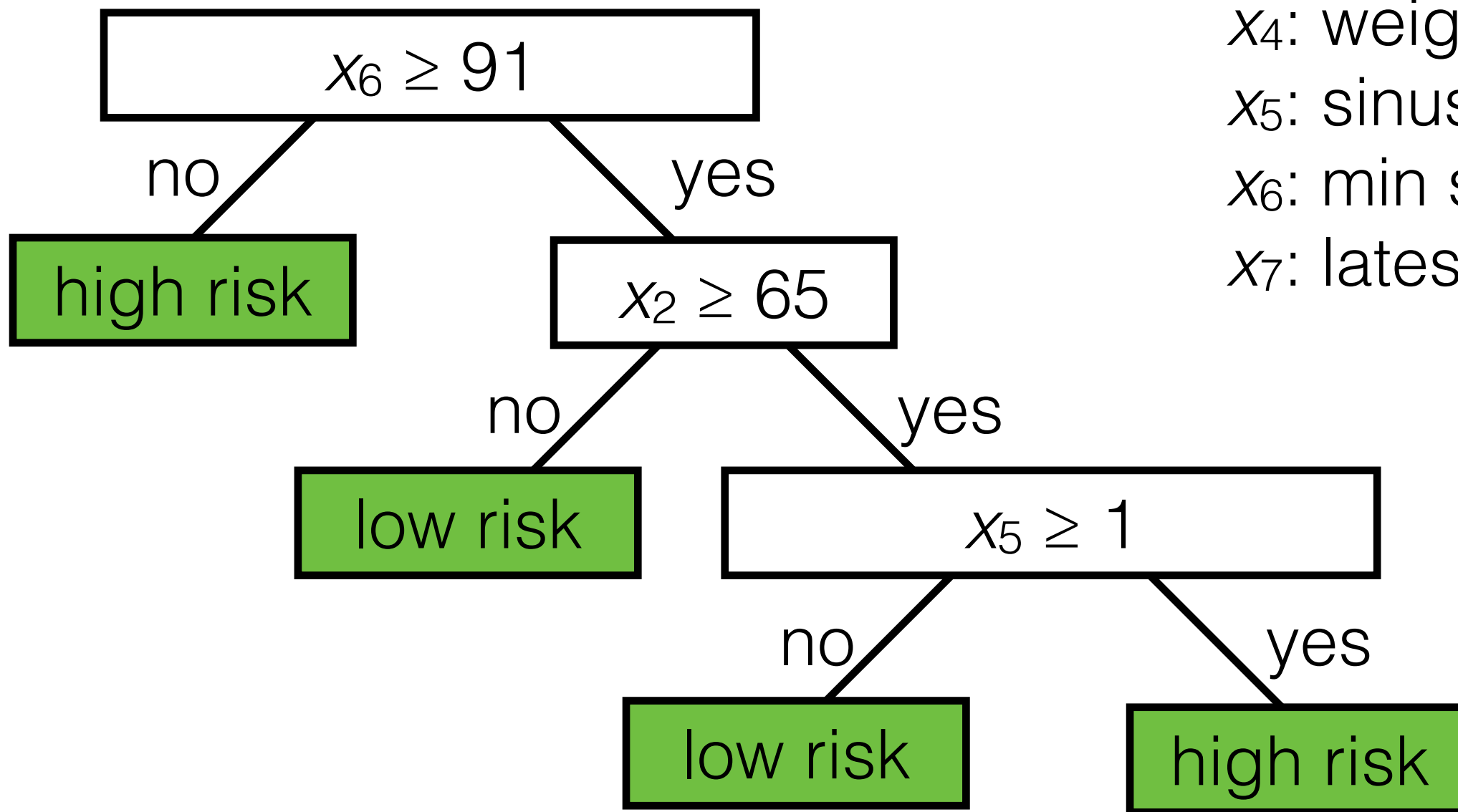
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



Decision tree

features:

x_1 : date

x_2 : age

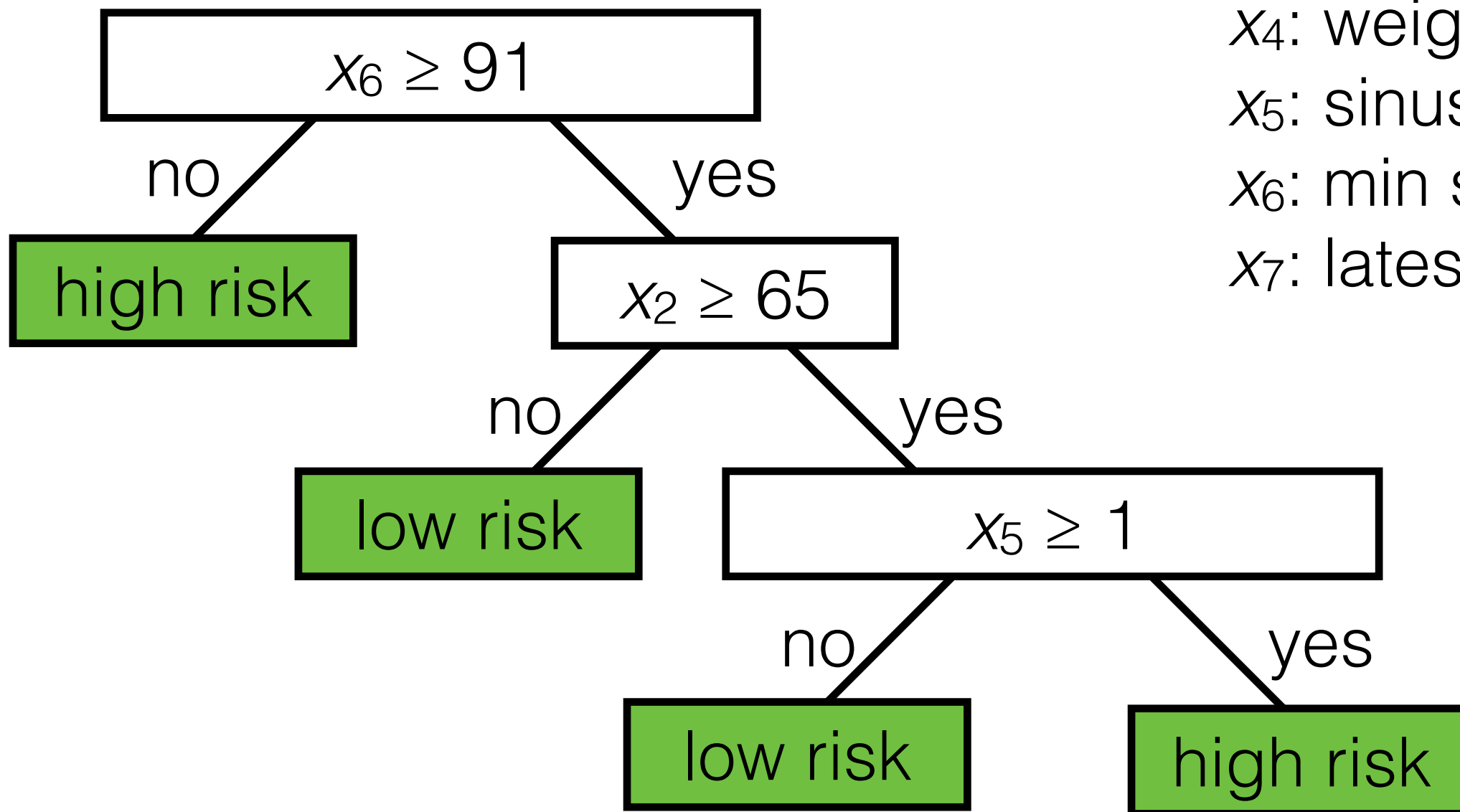
x_3 : height

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

Decision tree

features:

x_1 : date

x_2 : age

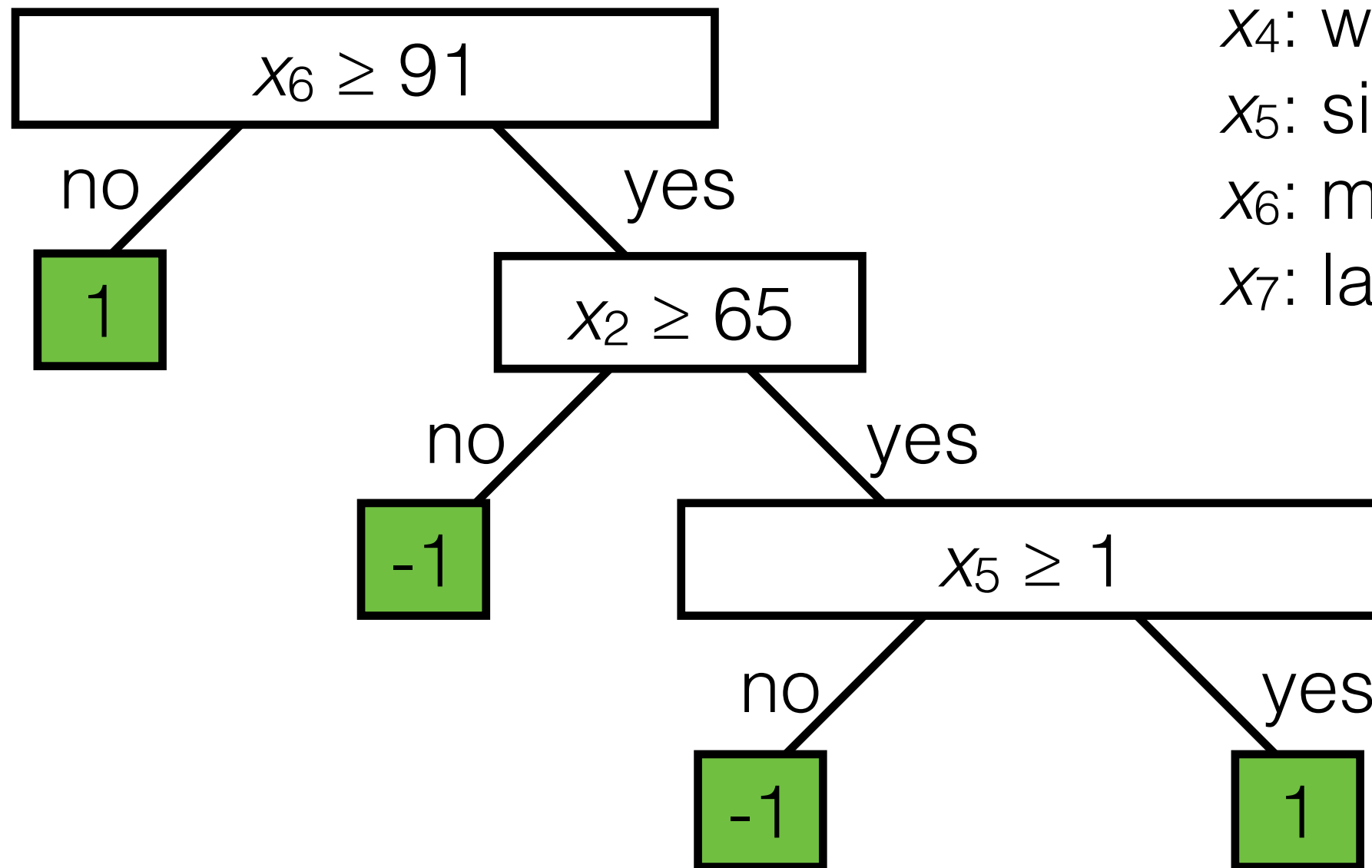
x_3 : height

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

Decision tree

features:

x_1 : date

x_2 : age

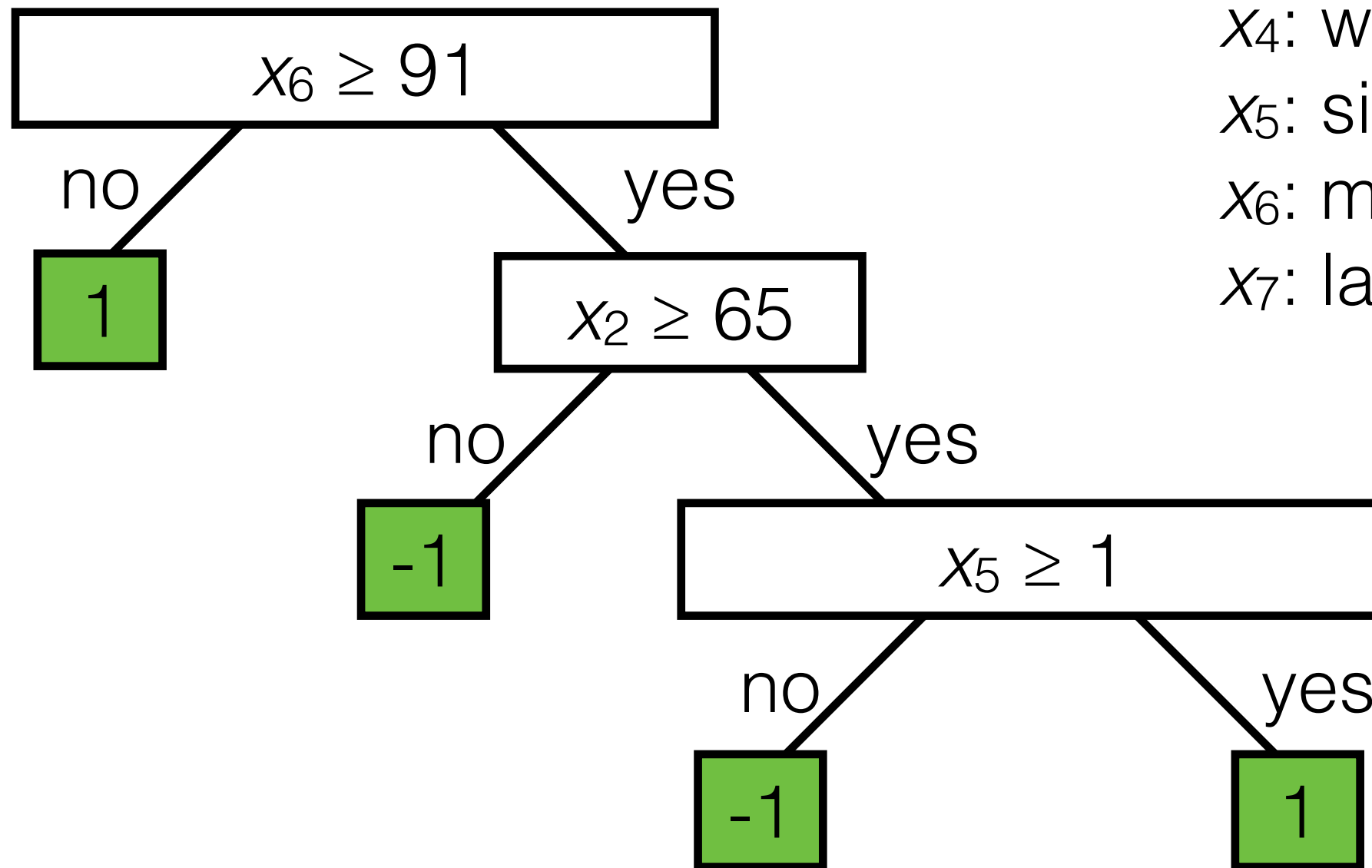
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



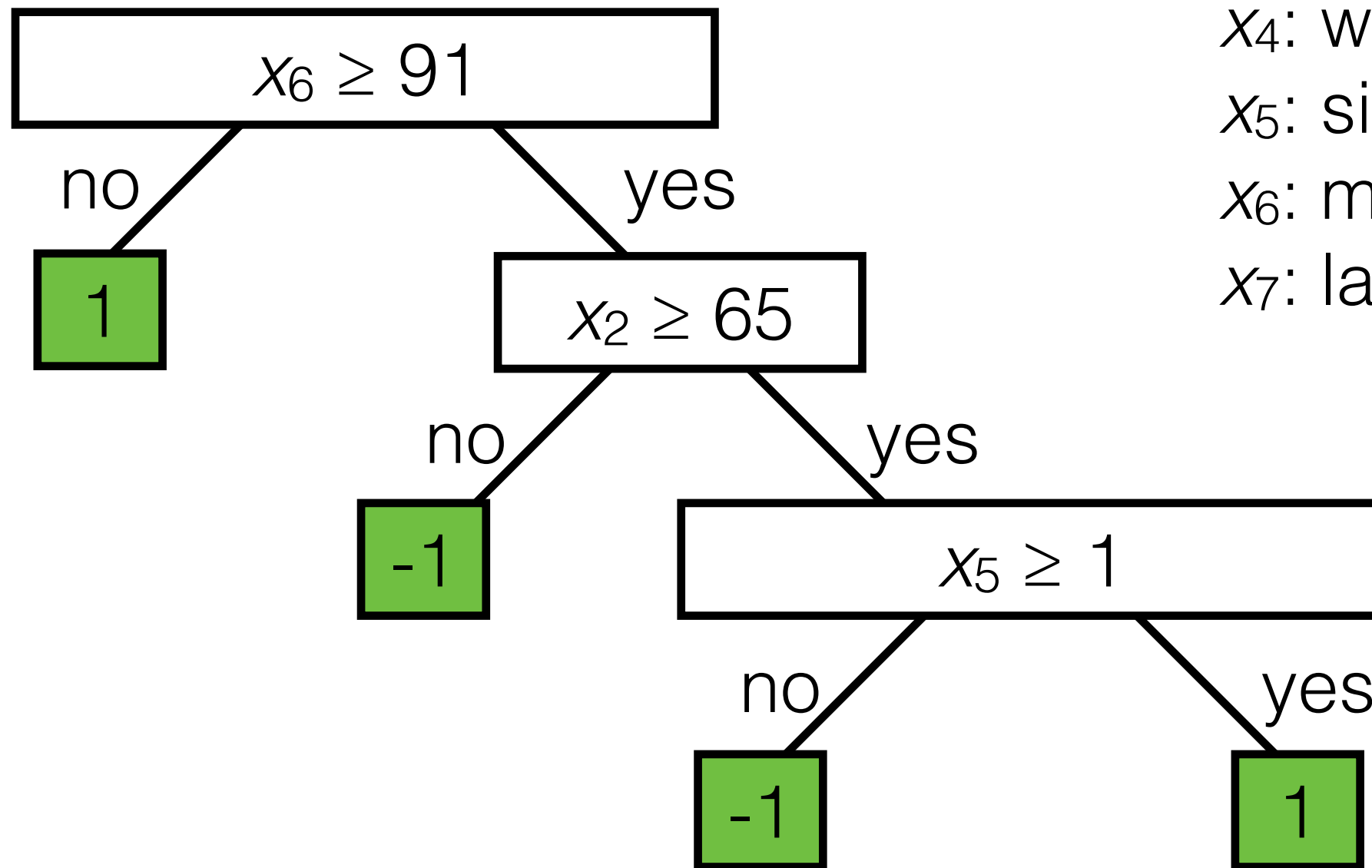
labels y :

1: high risk

-1: low risk

Decision tree

internal node:



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

Decision tree

internal node:

- dimension index j ; split value s

features:

x_1 : date

x_2 : age

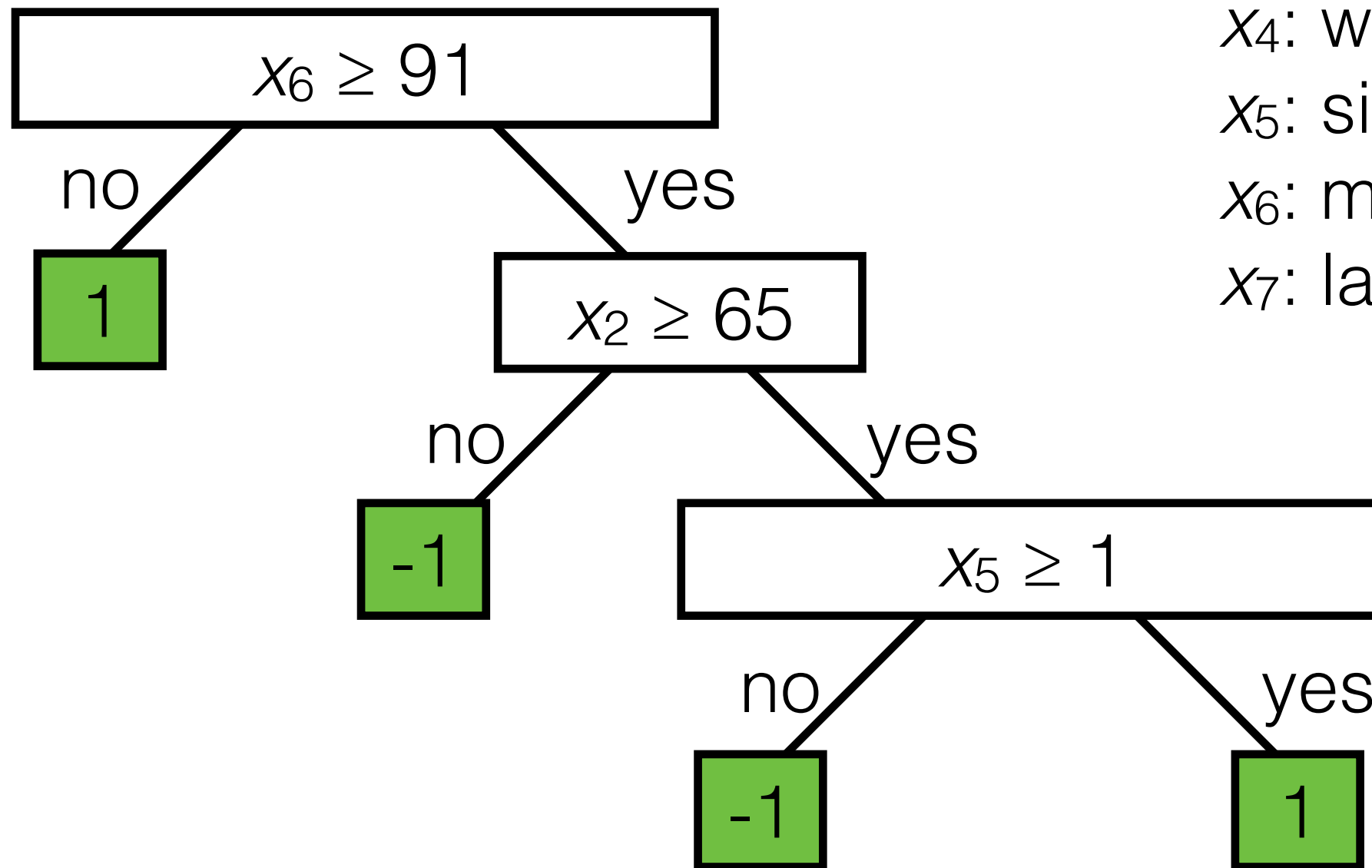
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



labels y :

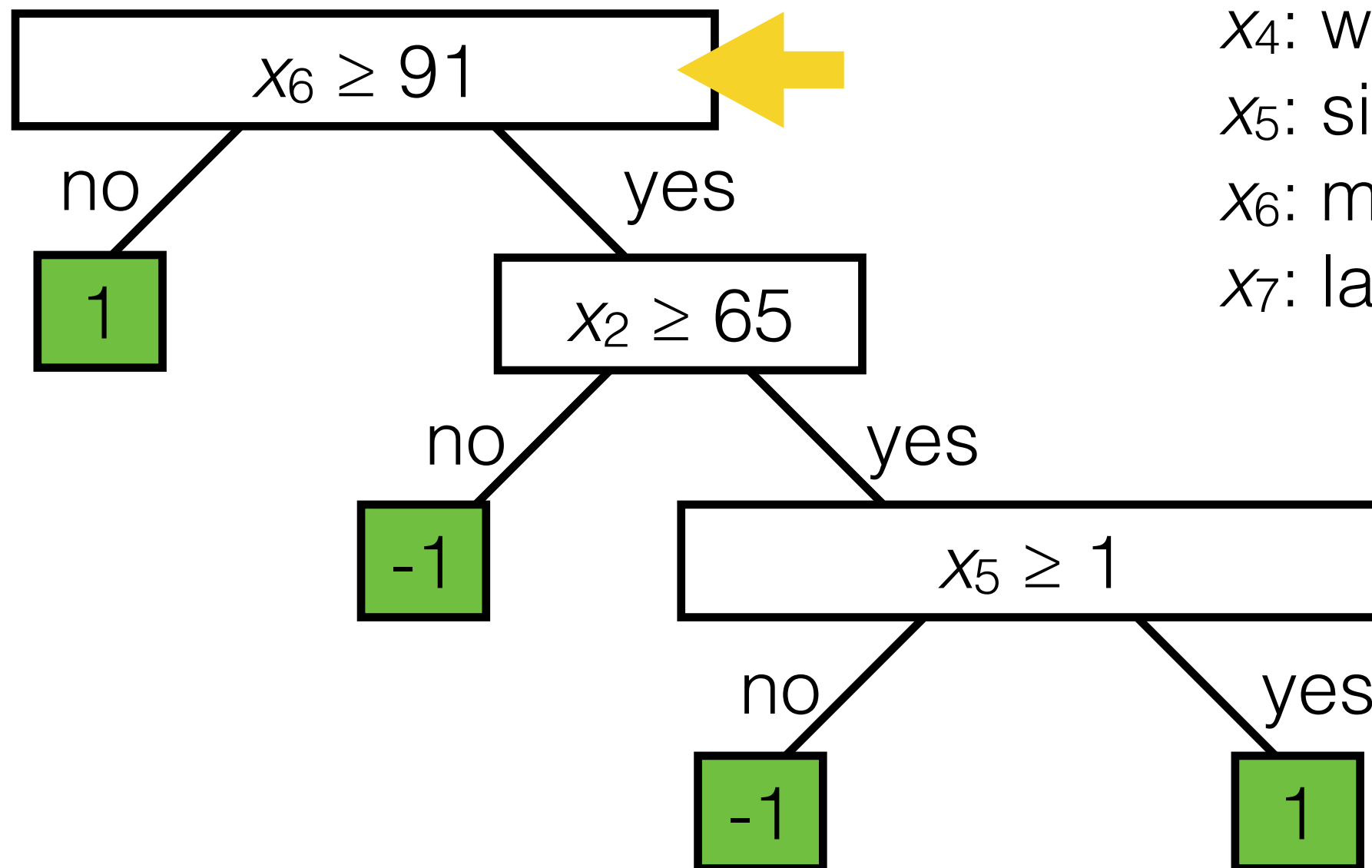
1: high risk

-1: low risk

Decision tree

internal node:

- dimension index j ; split value s



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 :

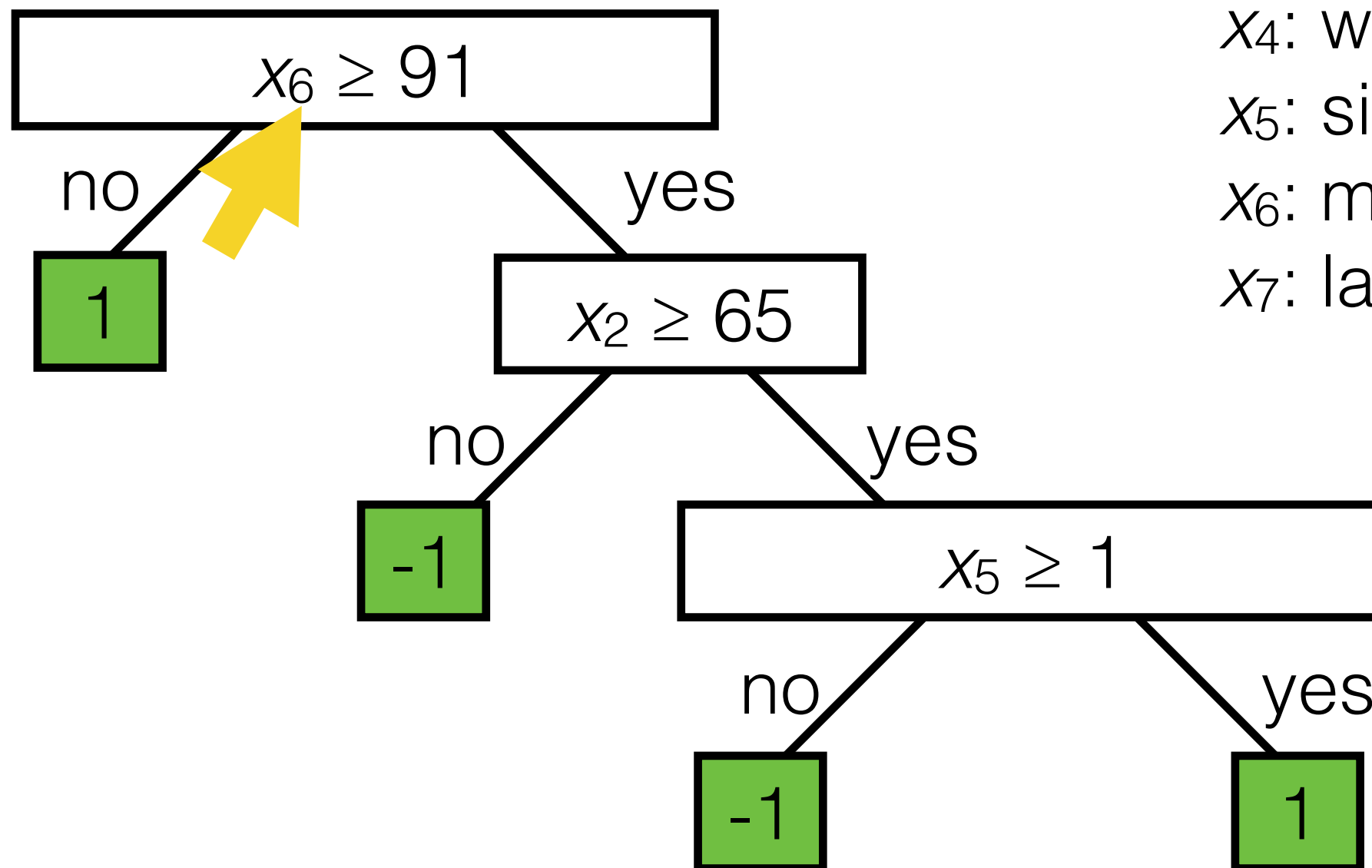
1: high risk

-1: low risk

Decision tree

internal node:

- dimension index j ; split value s



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 :

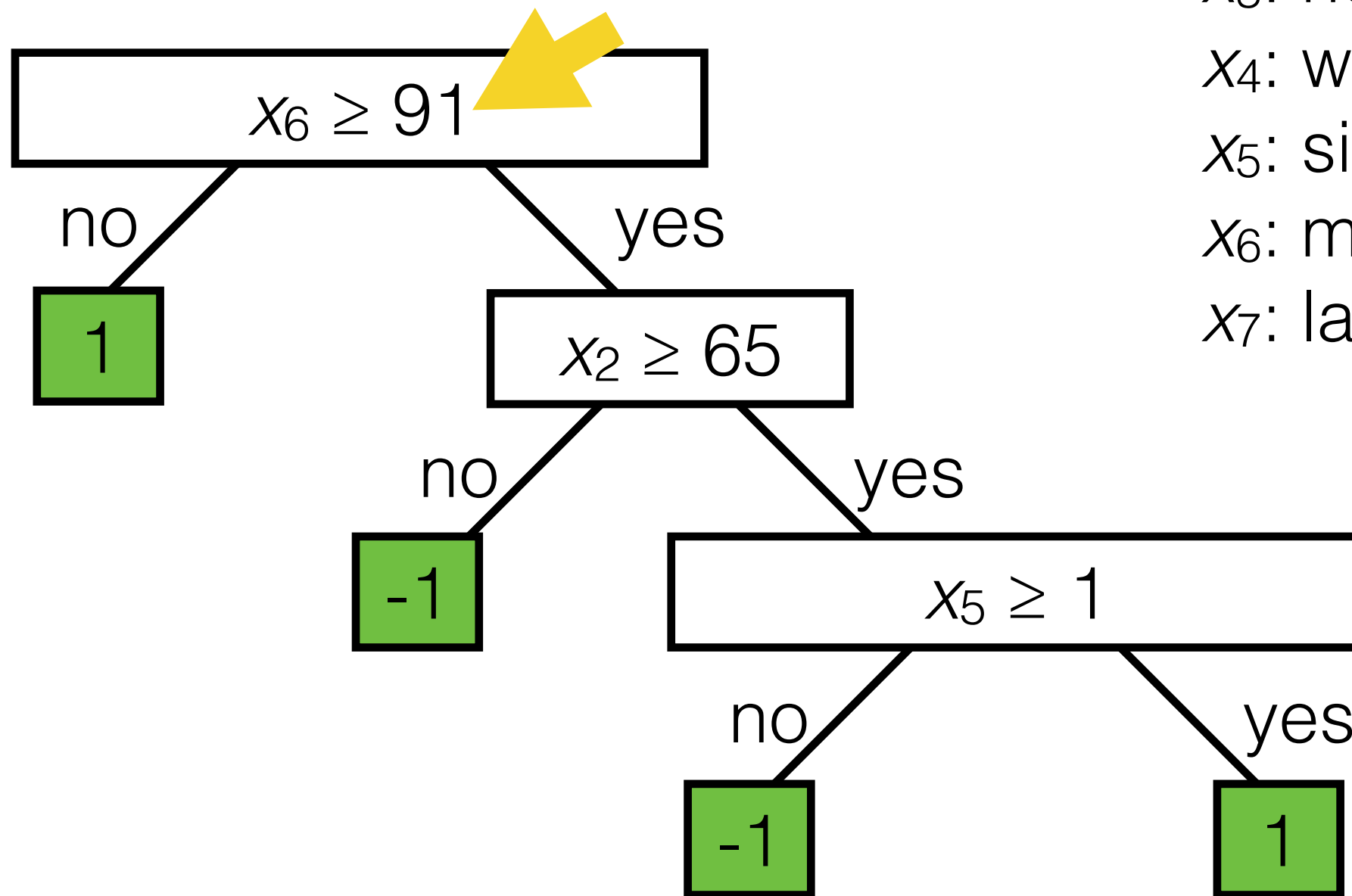
1: high risk

-1: low risk

Decision tree

internal node:

- dimension index j ; split value s



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 :

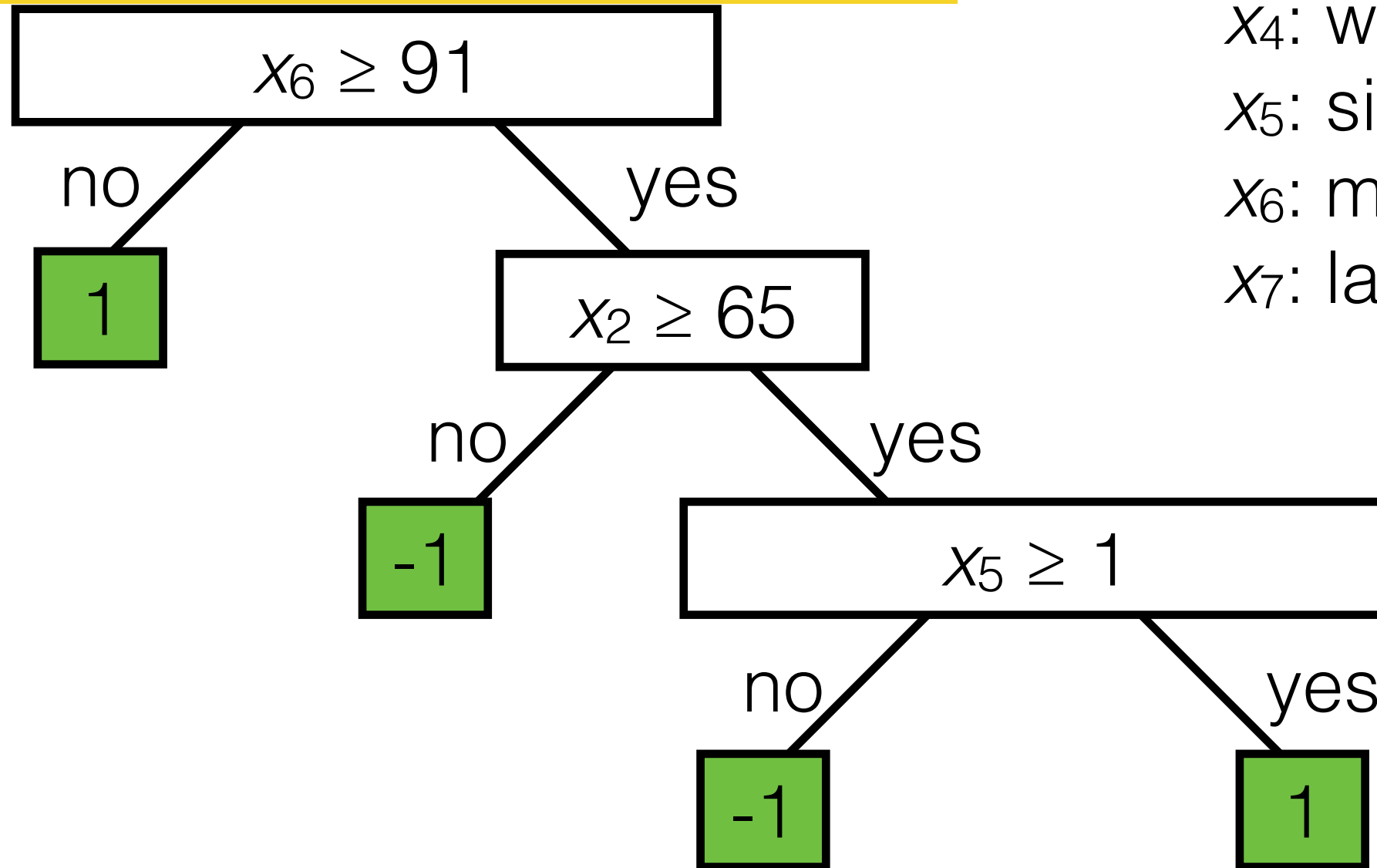
1: high risk

-1: low risk

Decision tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



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 :

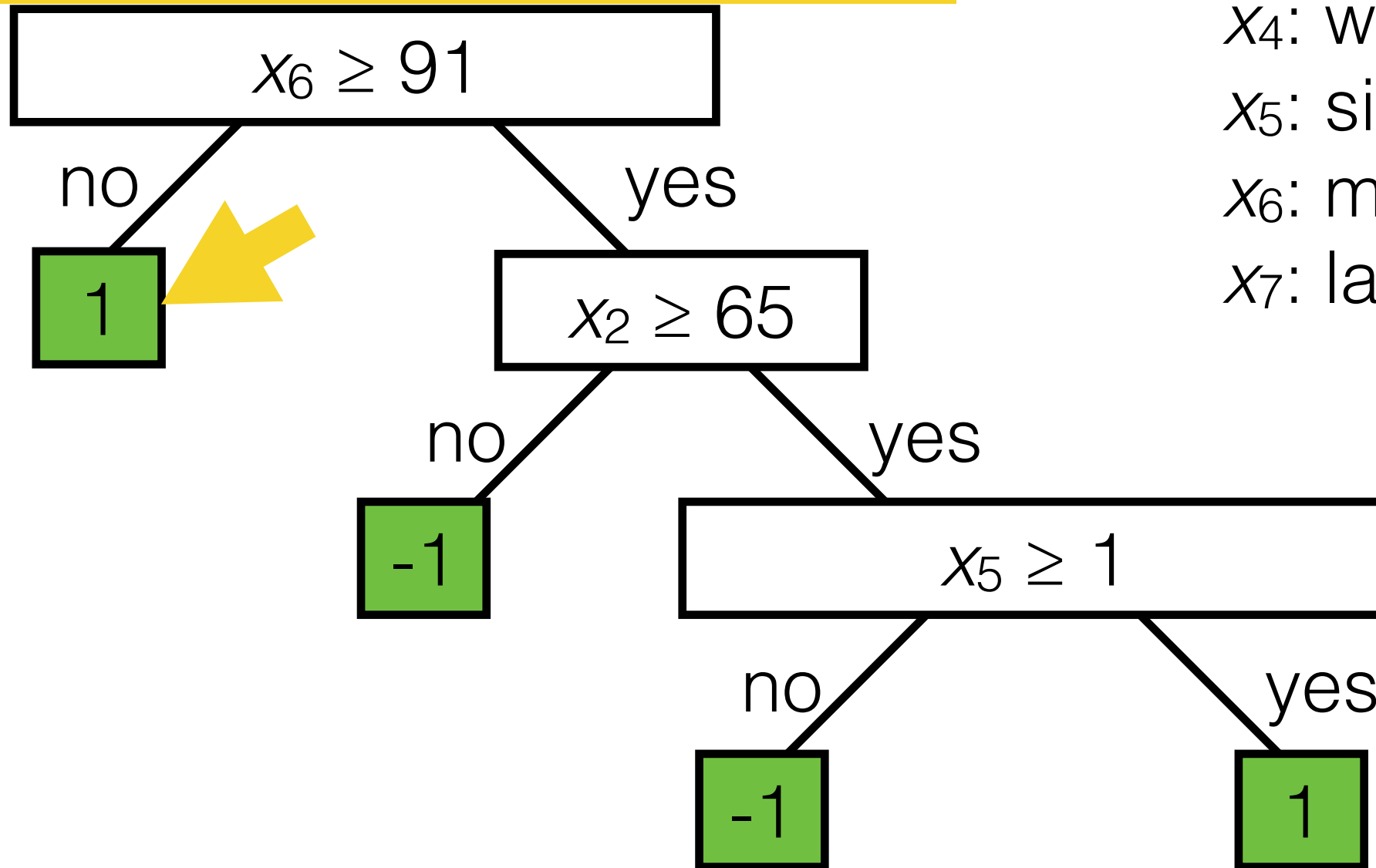
1: high risk

-1: low risk

Decision tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



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 :

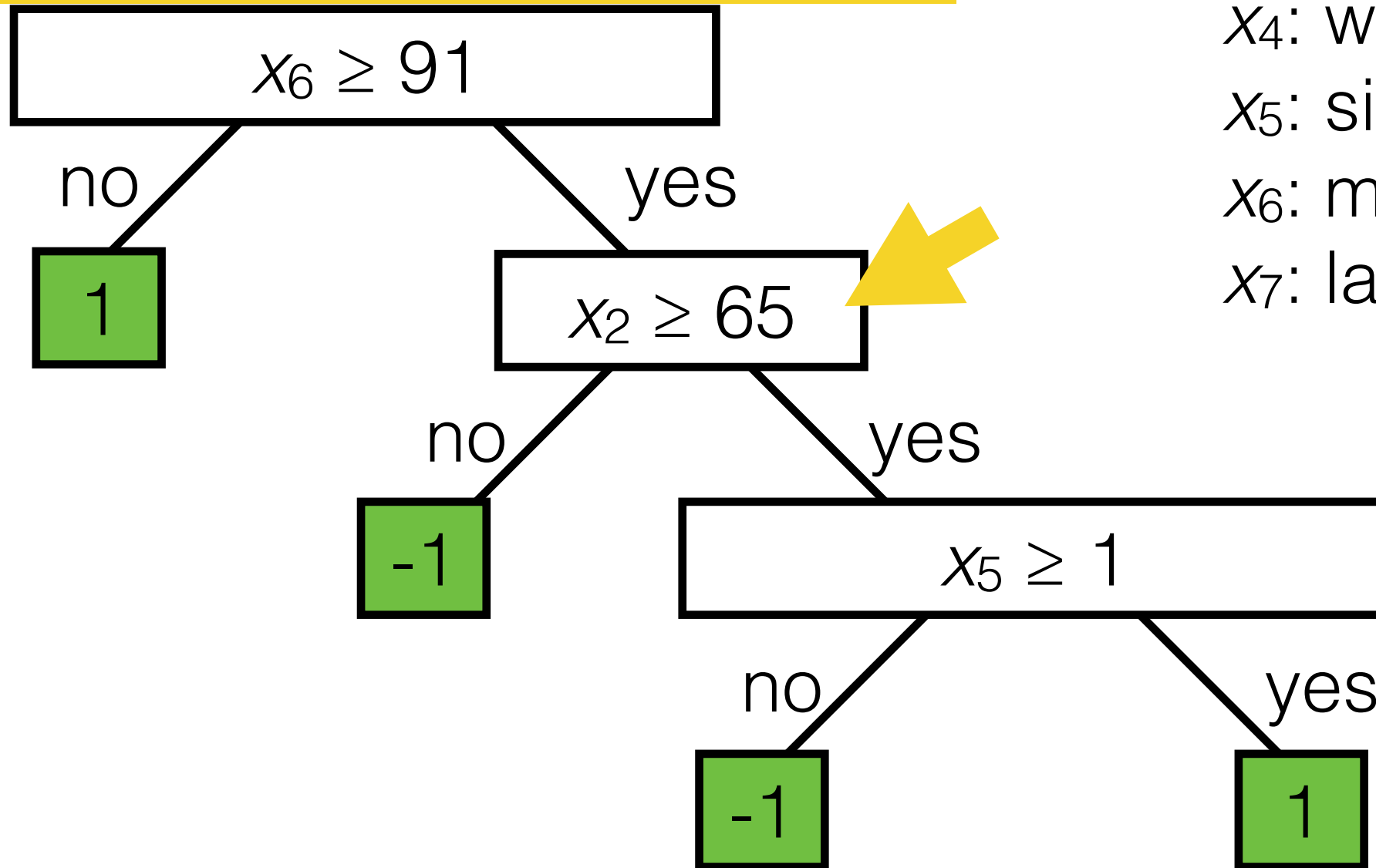
1: high risk

-1: low risk

Decision tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



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 :

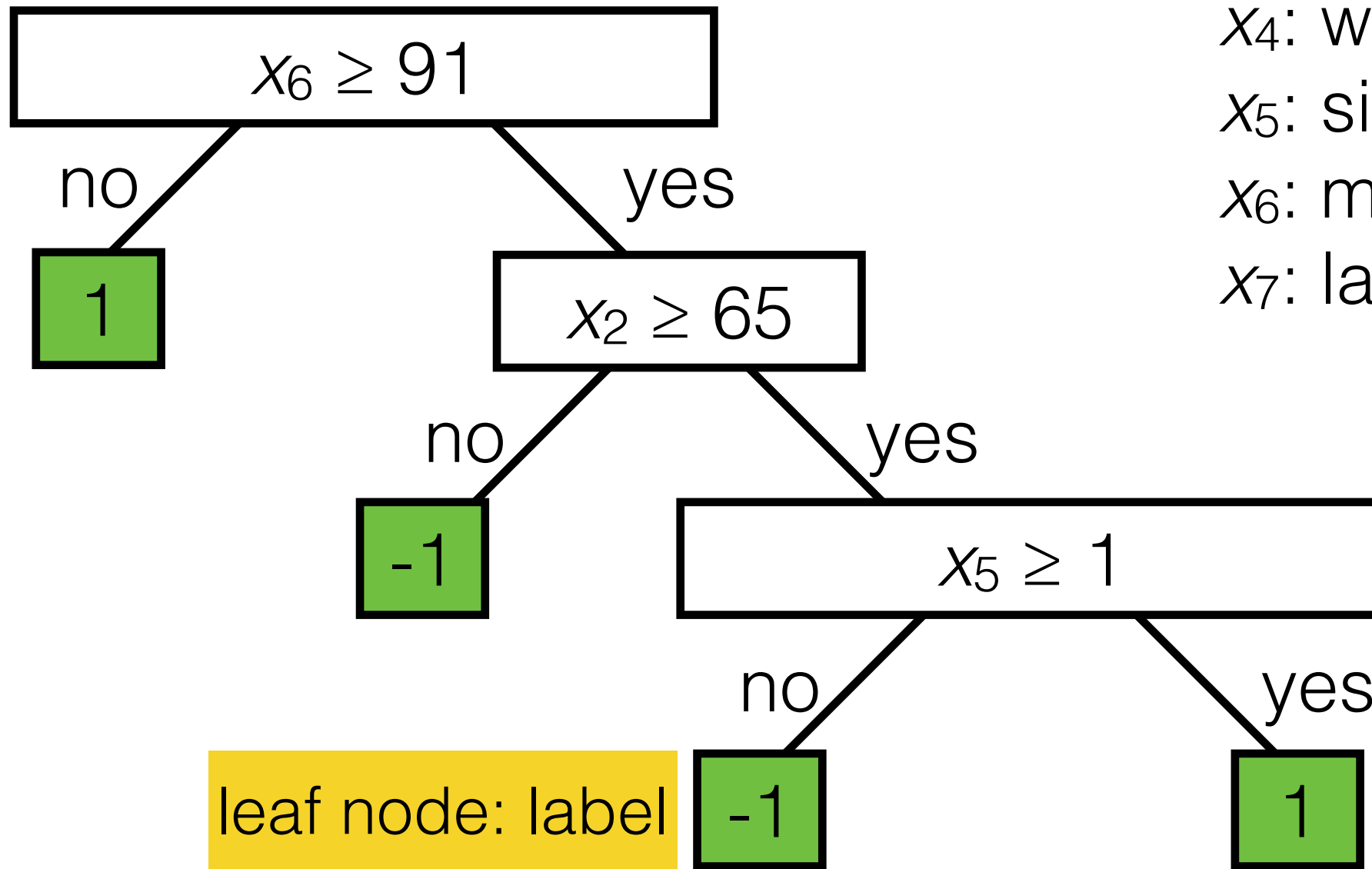
1: high risk

-1: low risk

Decision tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



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

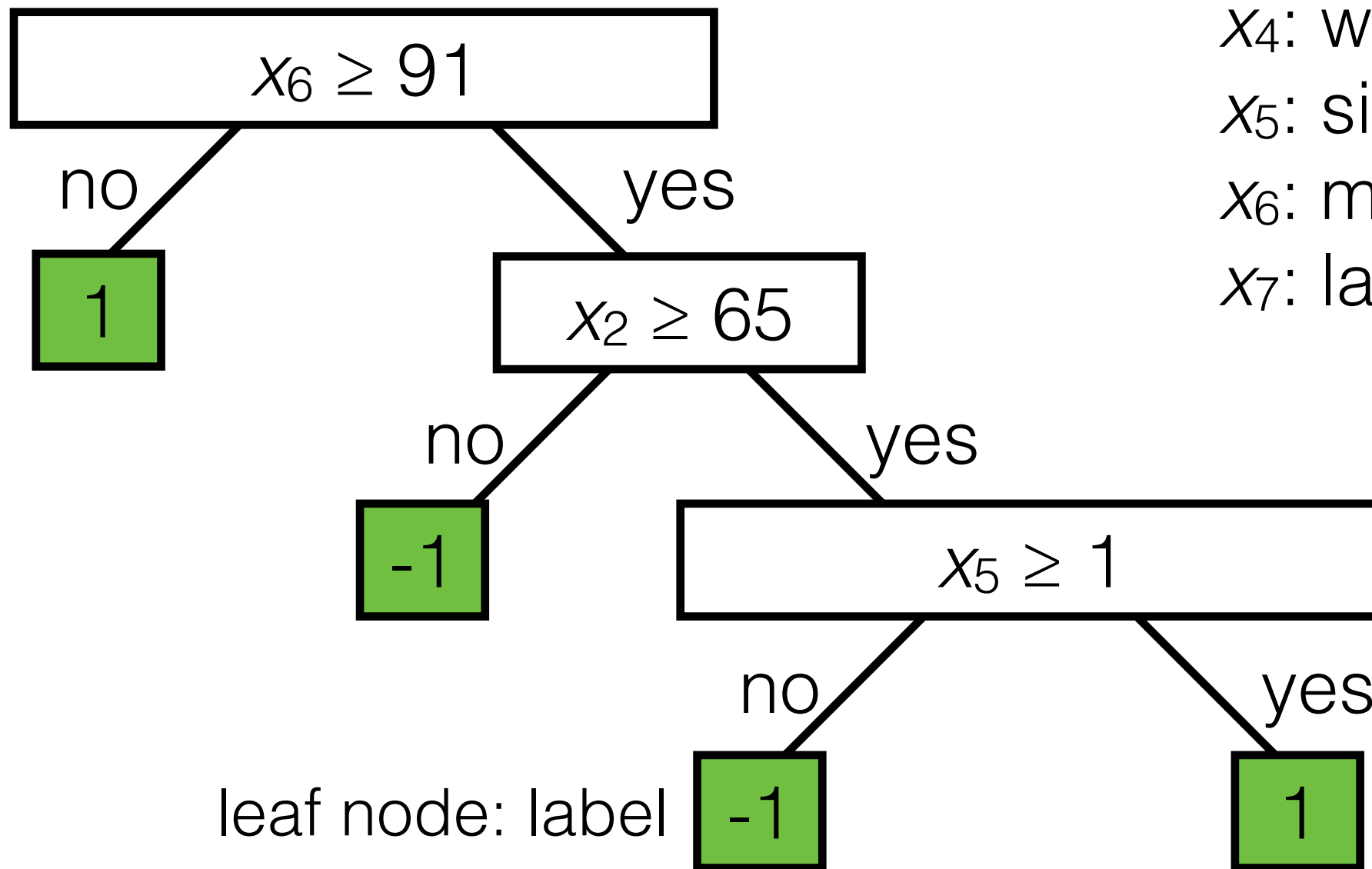
-1: low risk

leaf node: label

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



leaf node: label

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 :

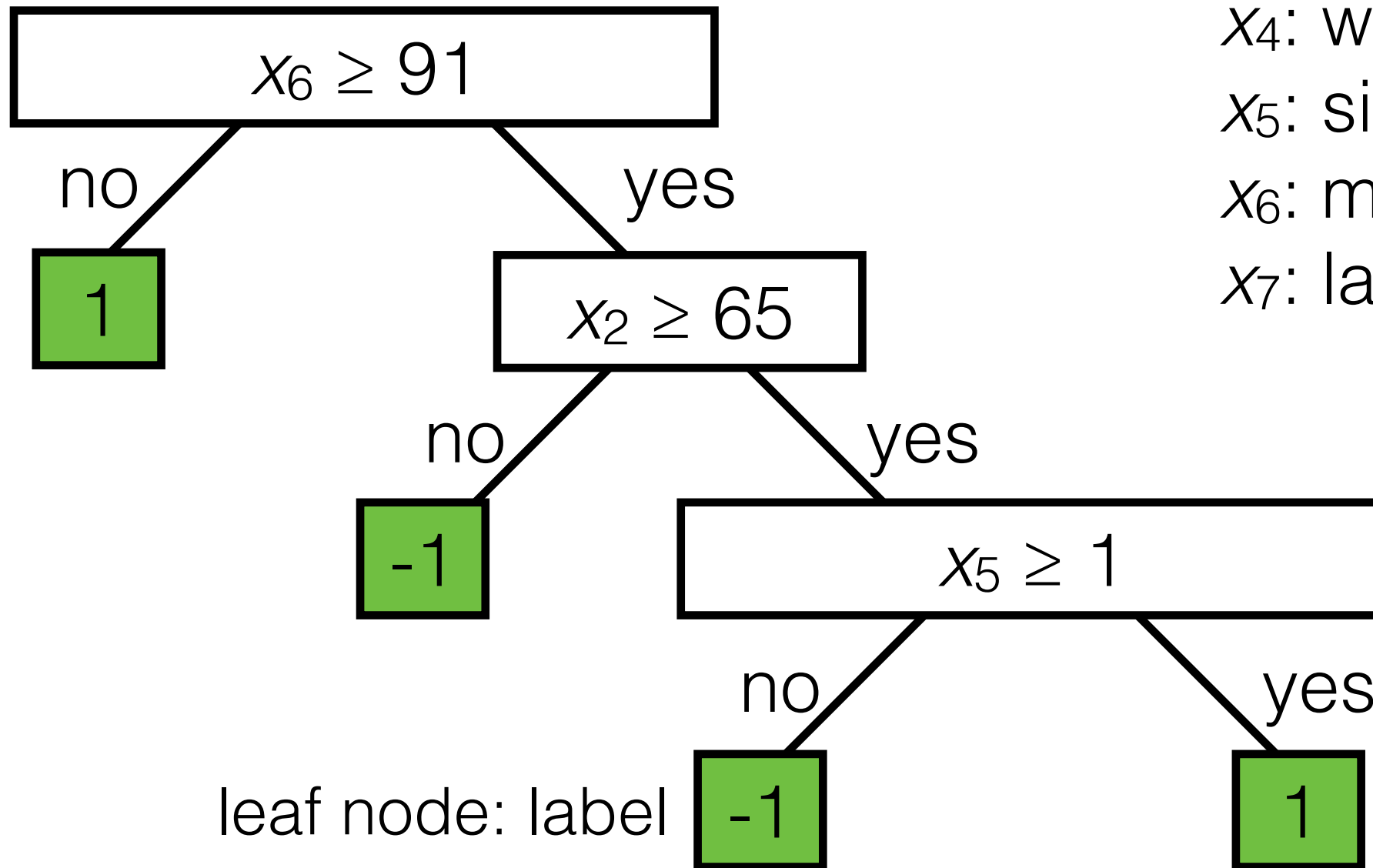
1: high risk

-1: low risk

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



leaf node: label

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

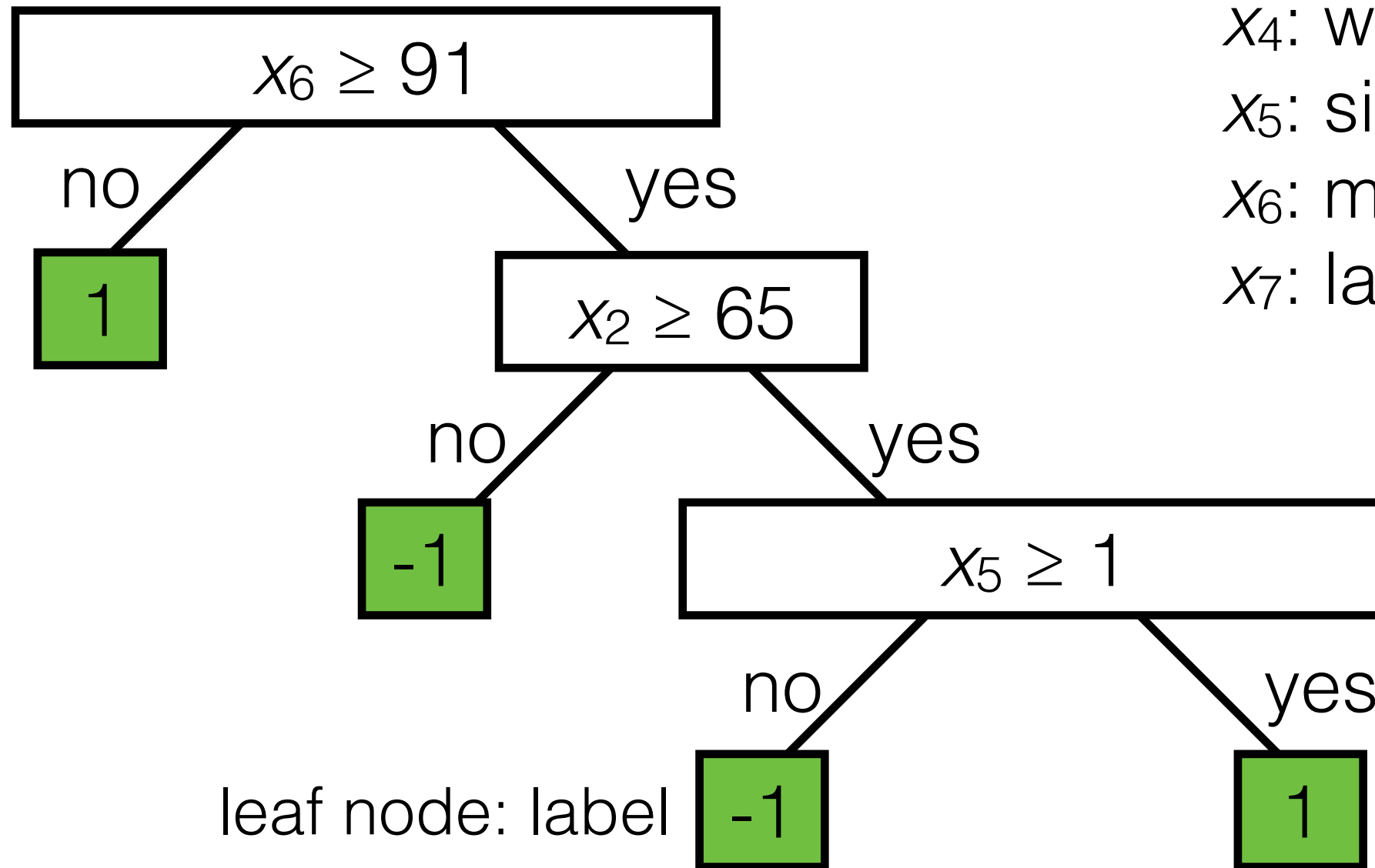
-1: low risk

$x^{(1)}$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



leaf node: label

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

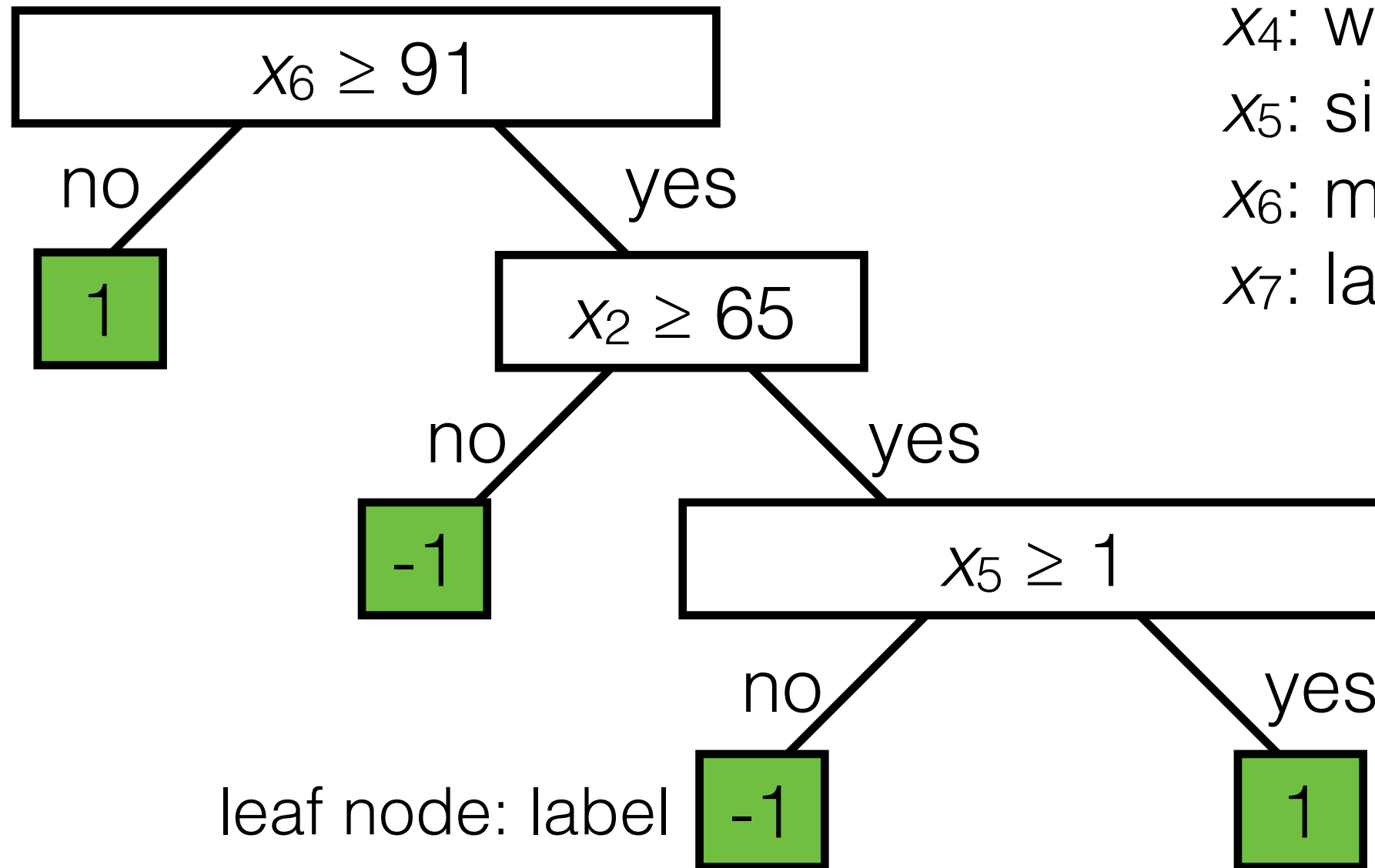
-1: low risk

$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



leaf node: label

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

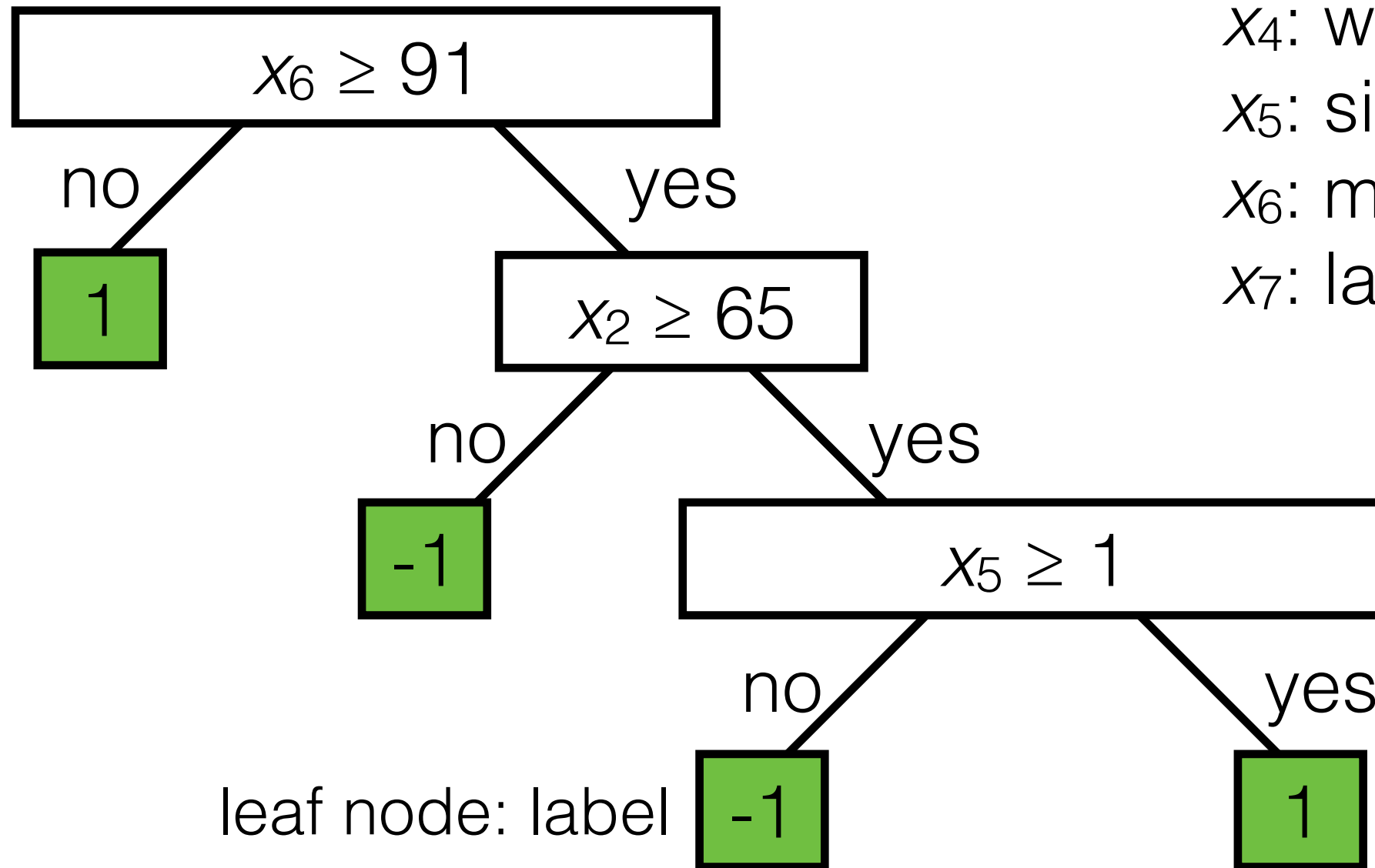
$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)})$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



leaf node: label

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

$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) =$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf

features:

x_1 : date

x_2 : age

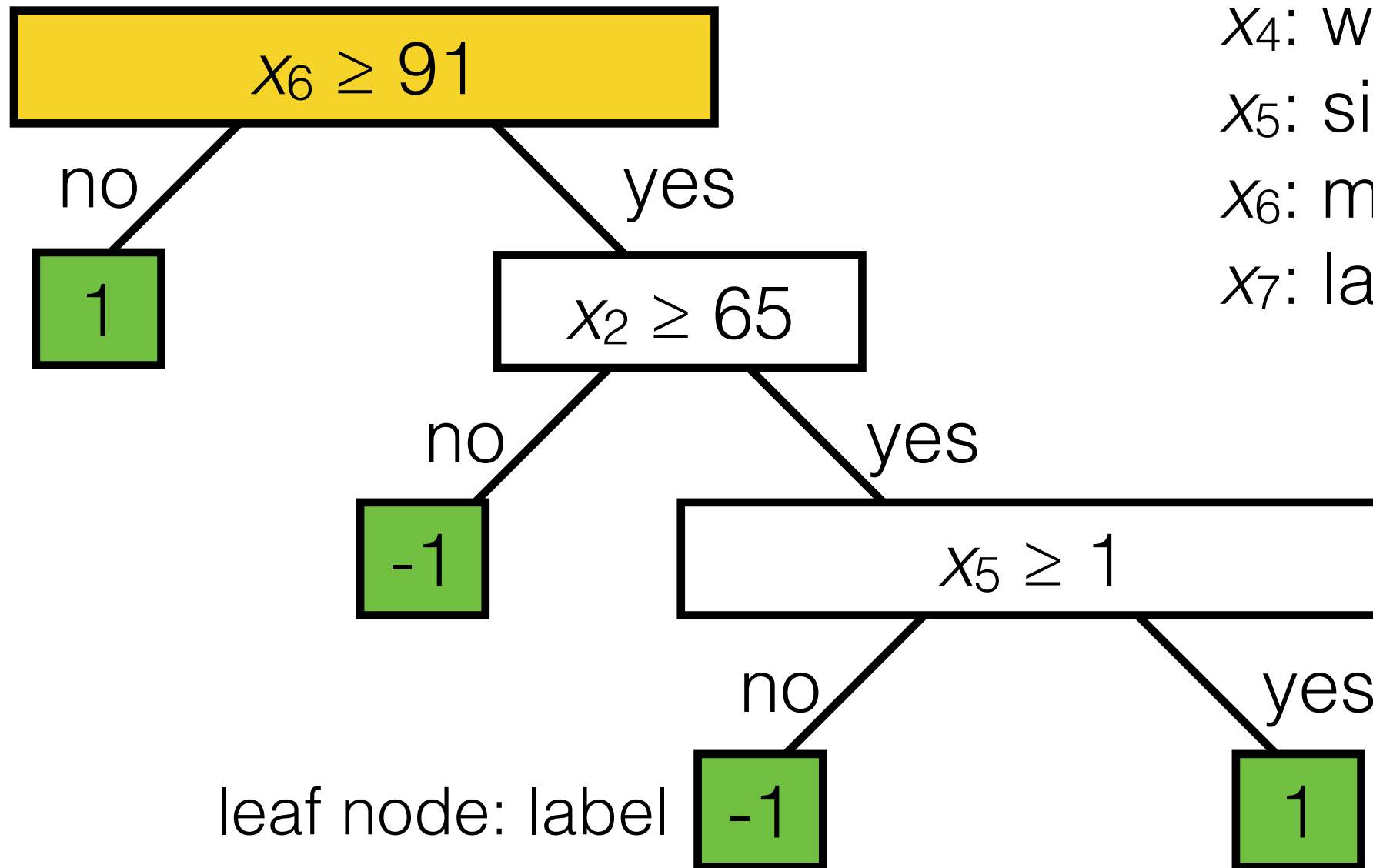
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



leaf node: label

labels y :

1: high risk

-1: low risk

$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) =$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf

features:

x_1 : date

x_2 : age

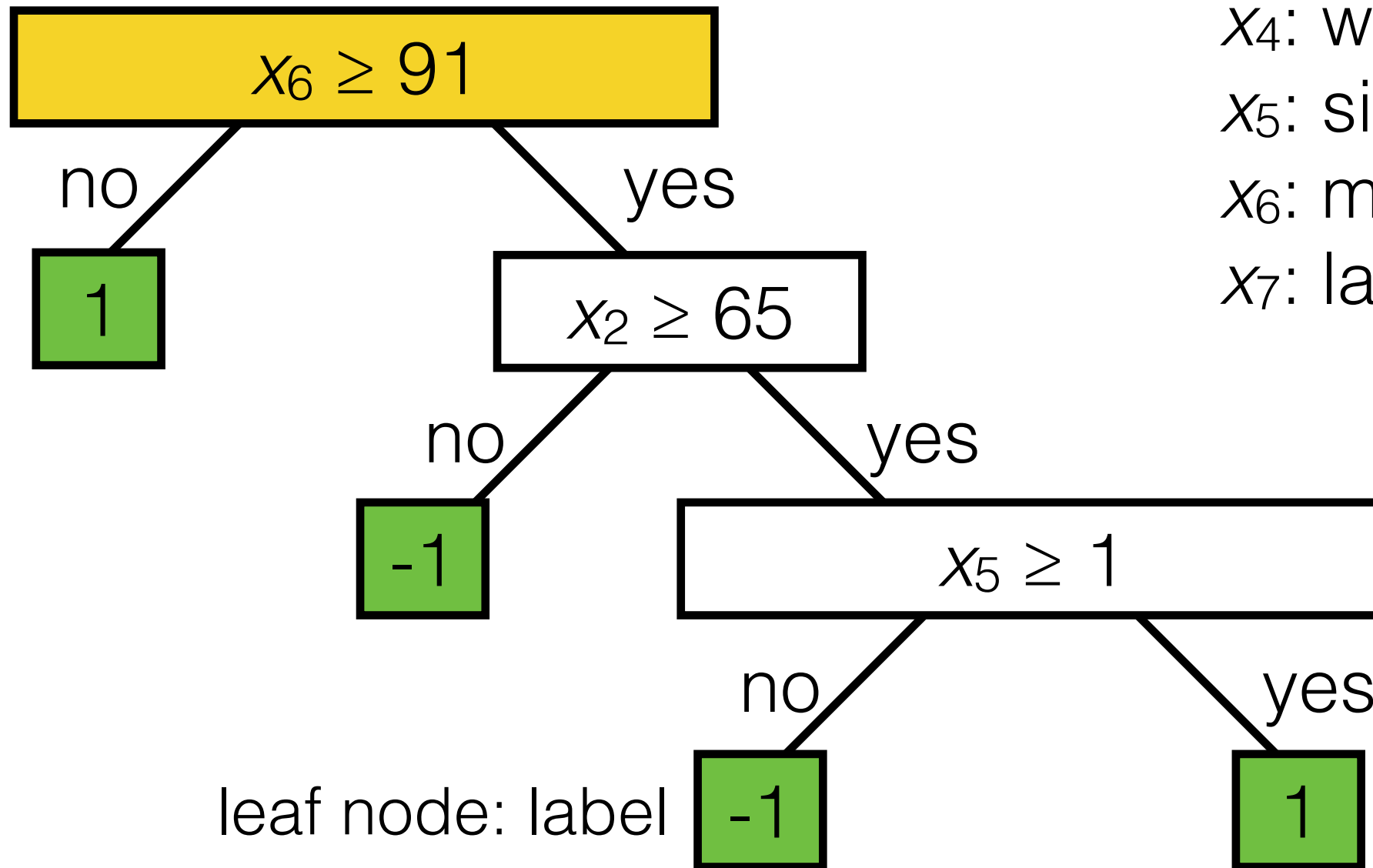
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



leaf node: label

labels y :

1: high risk

-1: low risk

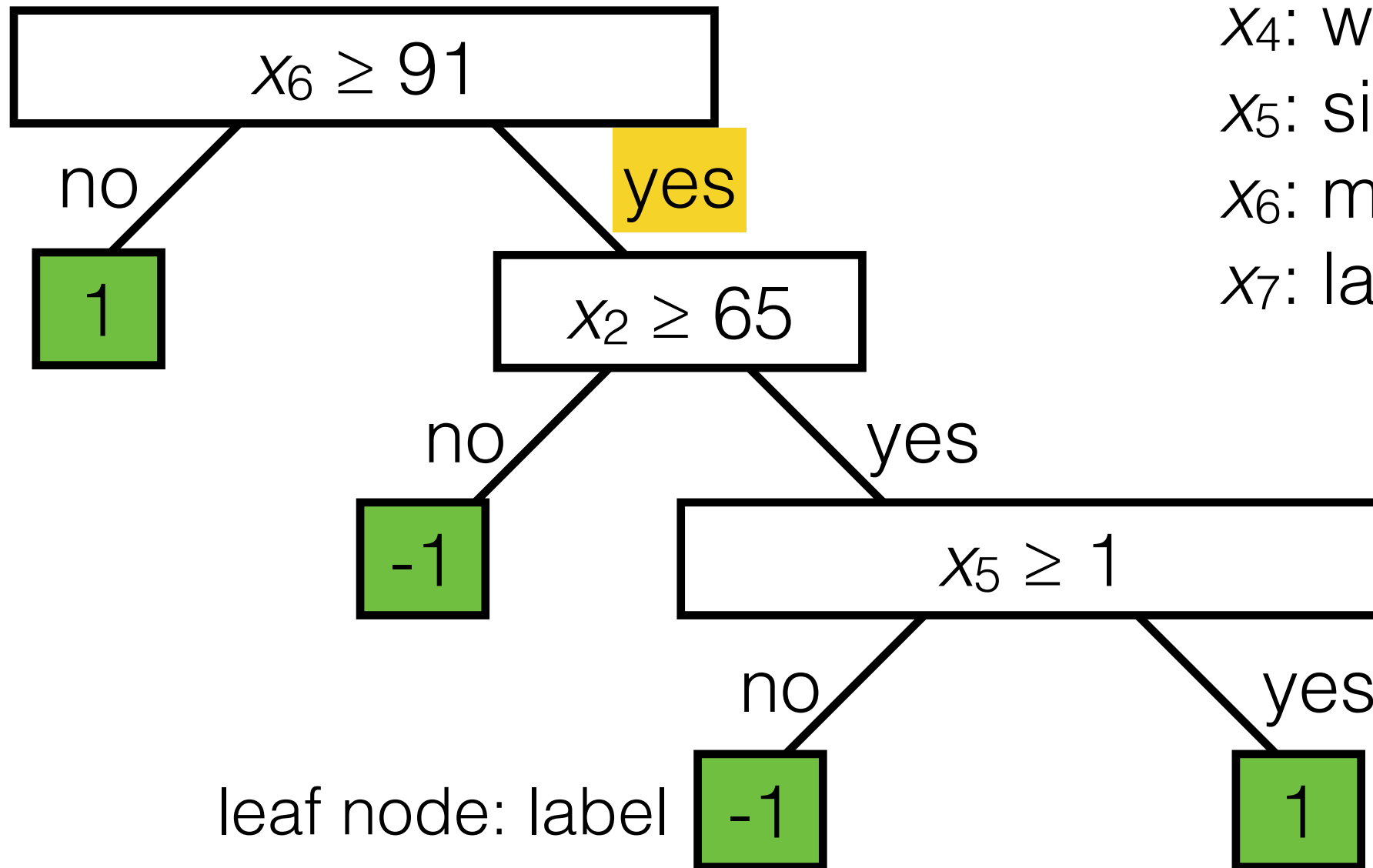
$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) =$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



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

$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) =$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf

features:

x_1 : date

x_2 : age

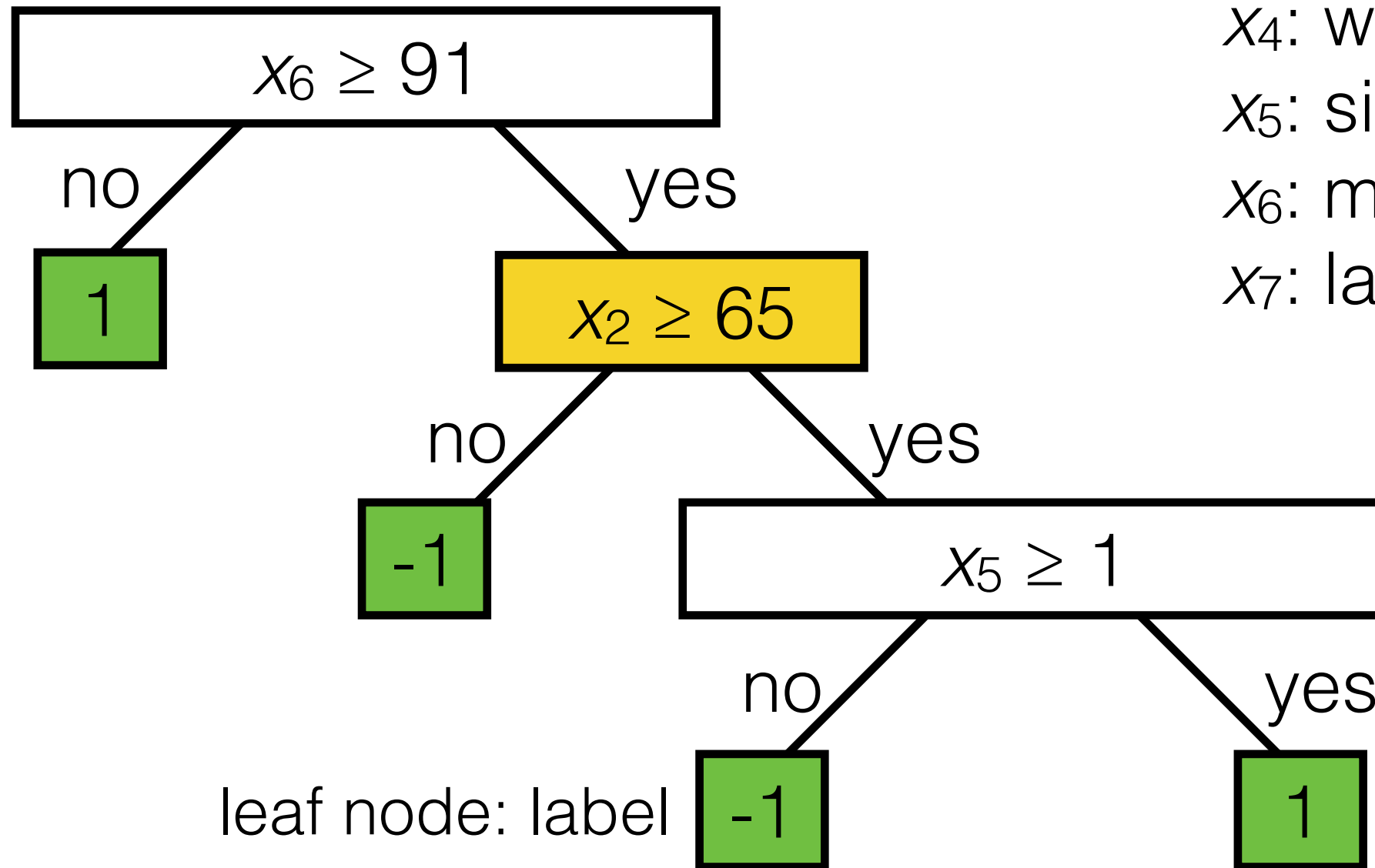
x_3 : height

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

leaf node: label

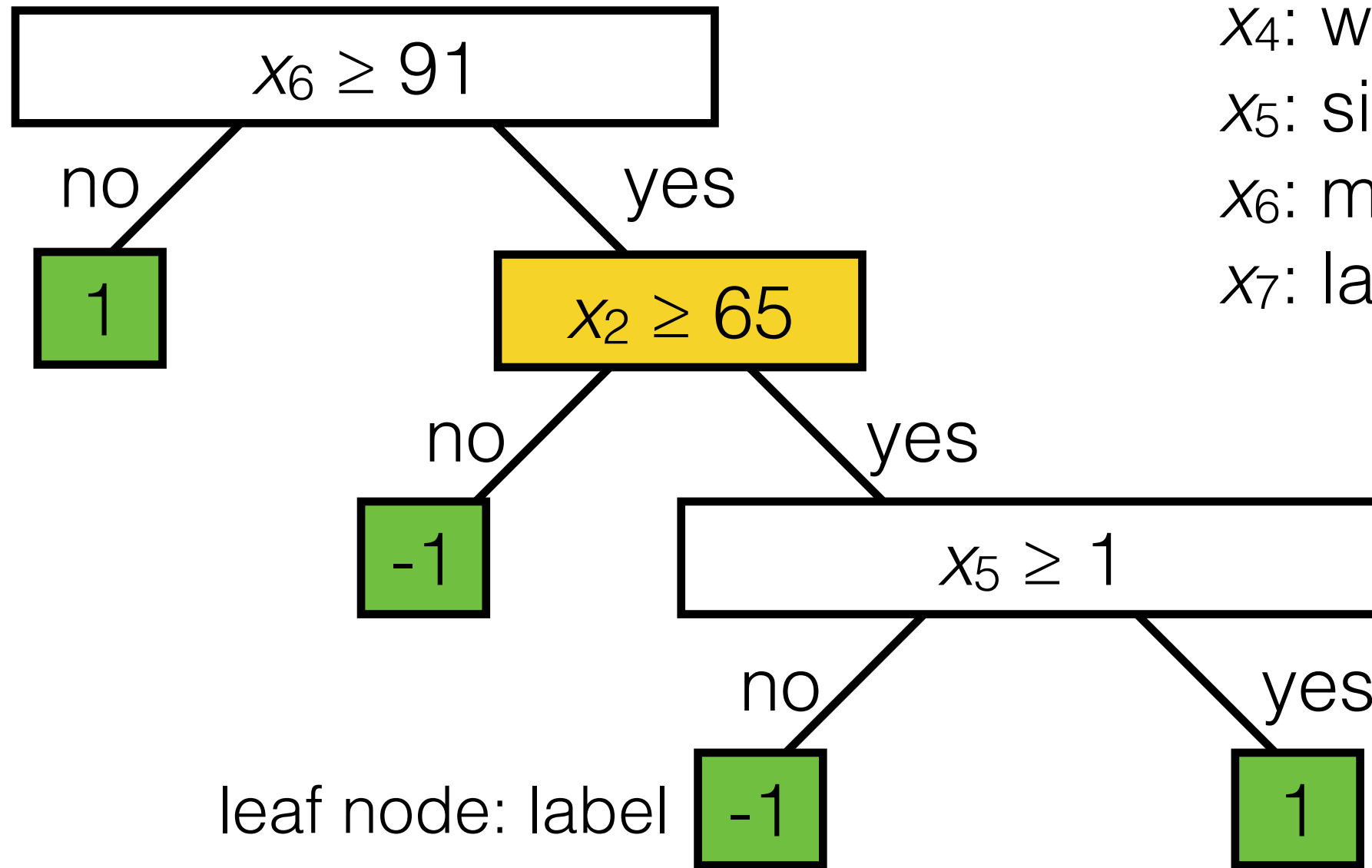
$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) =$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



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

$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) =$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf

features:

x_1 : date

x_2 : age

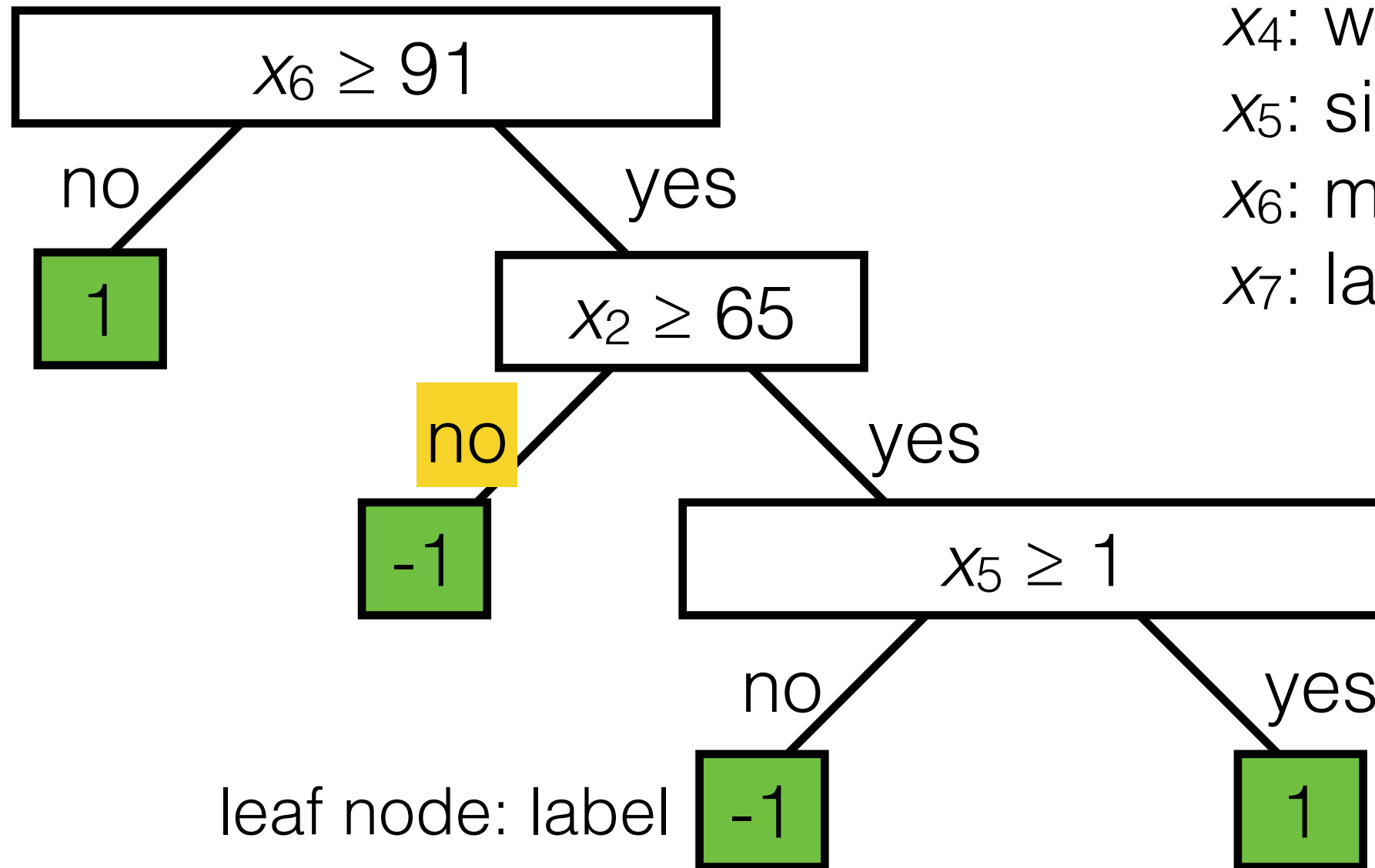
x_3 : height

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

$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) =$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf

features:

x_1 : date

x_2 : age

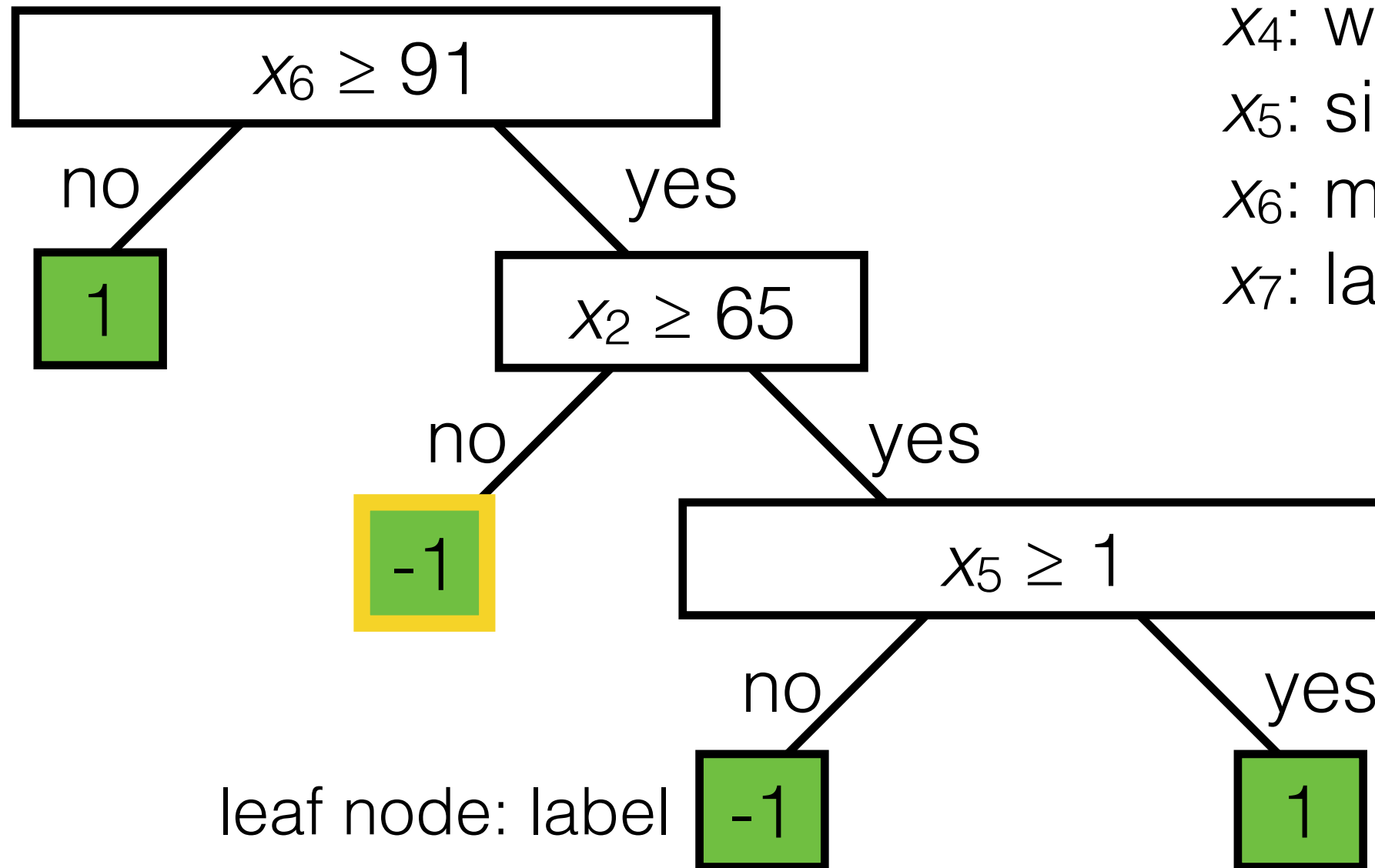
x_3 : height

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

leaf node: label

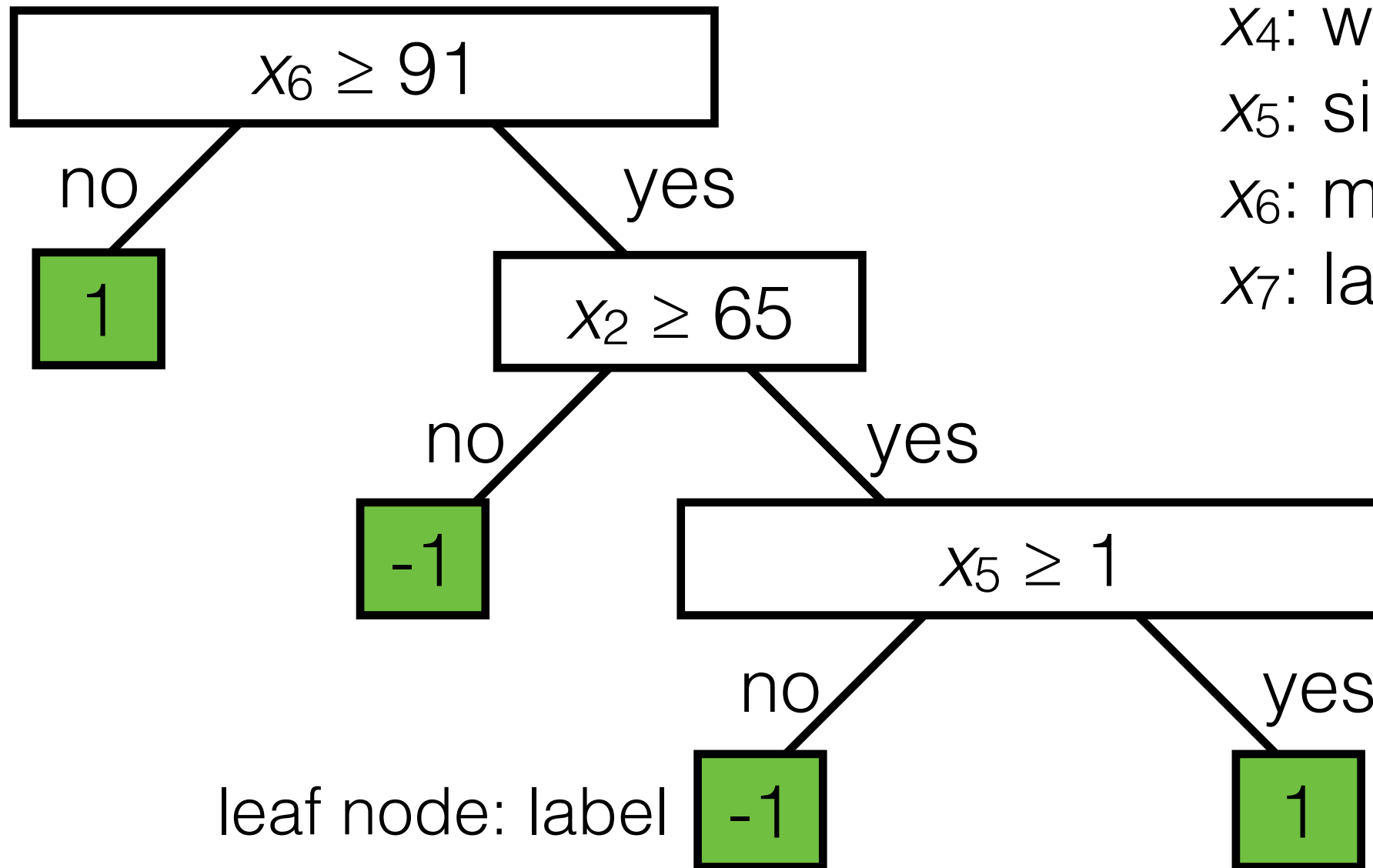
$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) =$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf



leaf node: label

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

$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) = -1$

Classification tree

internal node:

- dimension index j ; split value s
- two child nodes: internal or leaf

features:

x_1 : date

x_2 : age

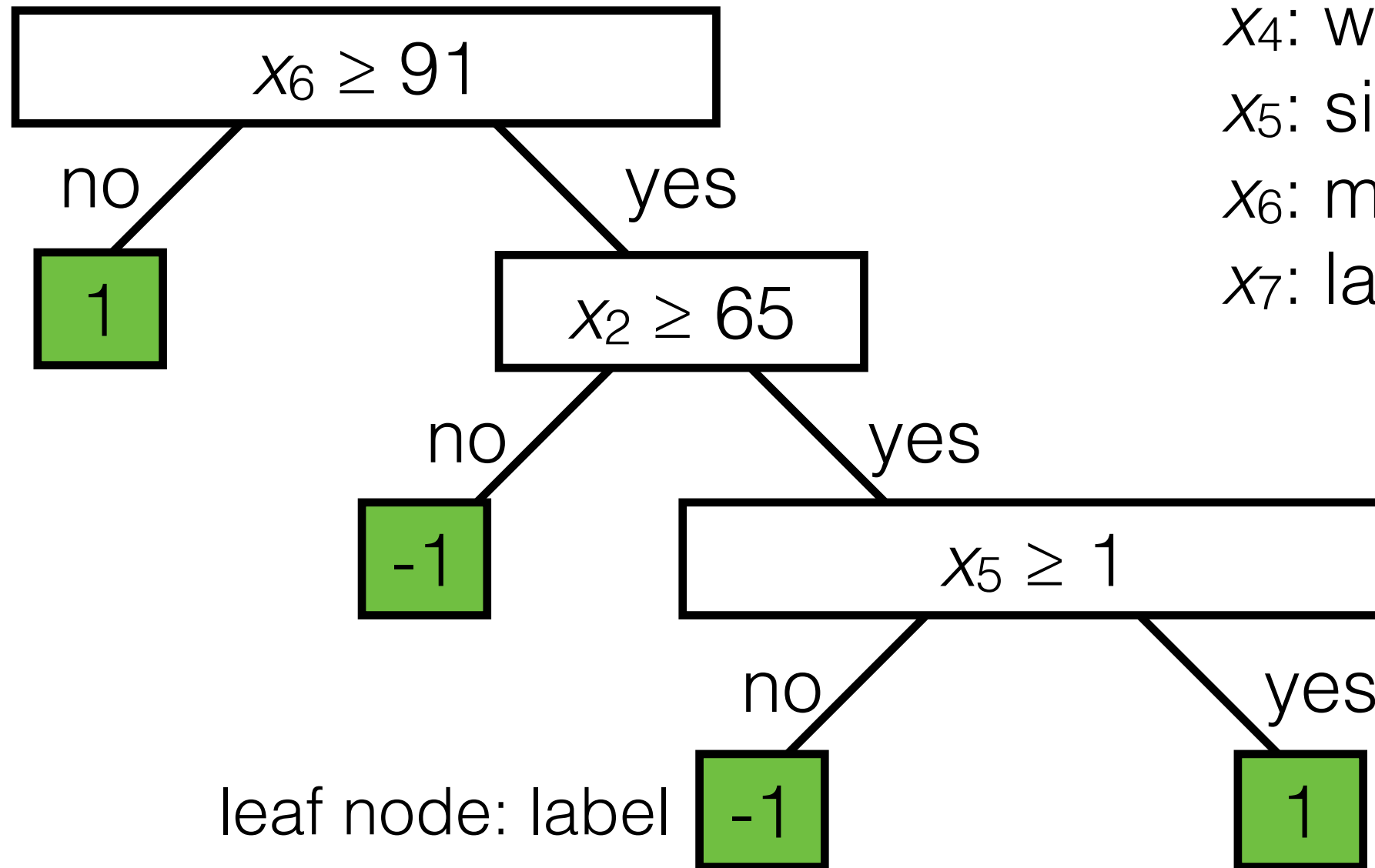
x_3 : height

x_4 : weight

x_5 : sinus tachycardia?

x_6 : min systolic bp, 24h

x_7 : latest diastolic bp



leaf node: label

labels y :

1: high risk

-1: low risk

$x^{(1)}$: 2021/11/08, 49, 172 cm, 70.5 kg, 0, 115, 79

$T(x^{(1)}) = -1$

Regression tree

Regression tree

features:

x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

Regression tree

features:

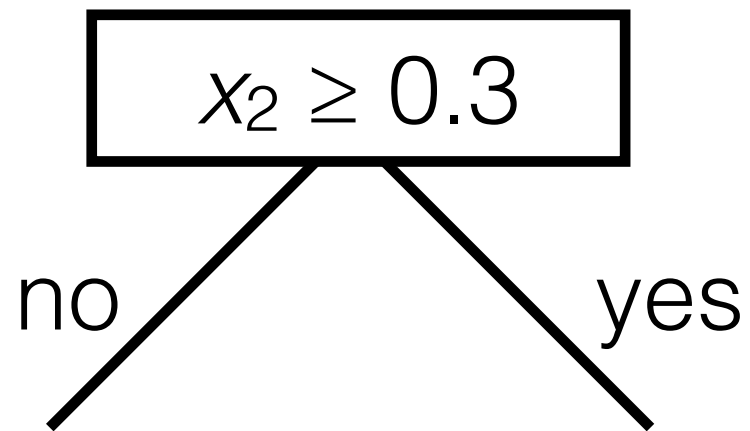
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

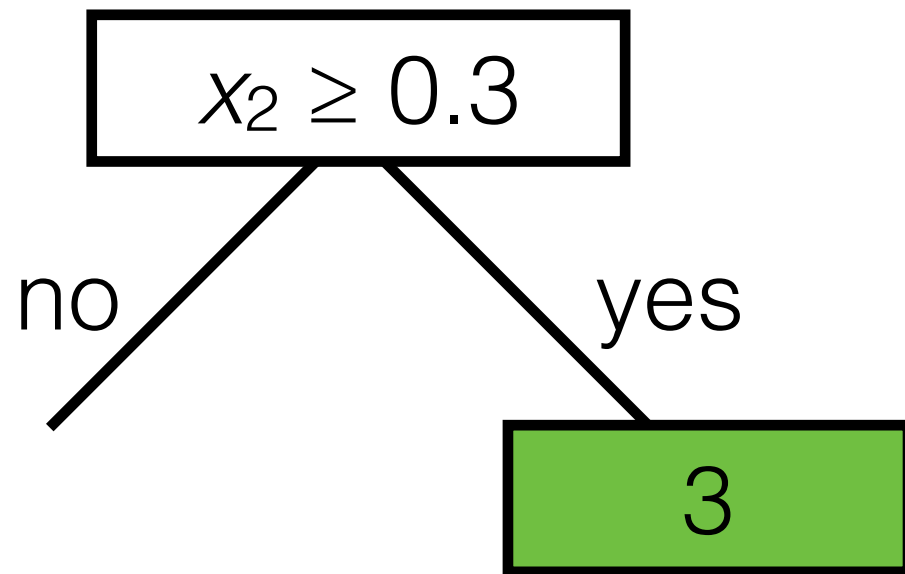
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

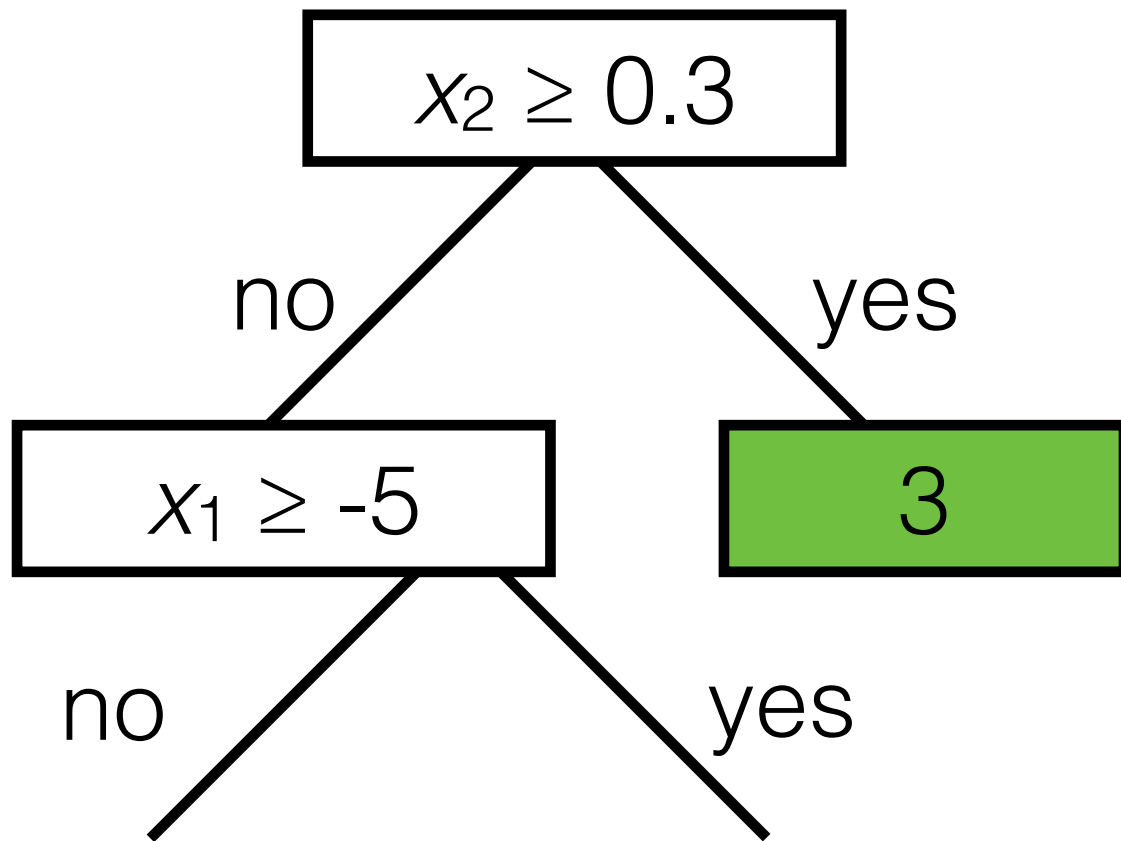
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

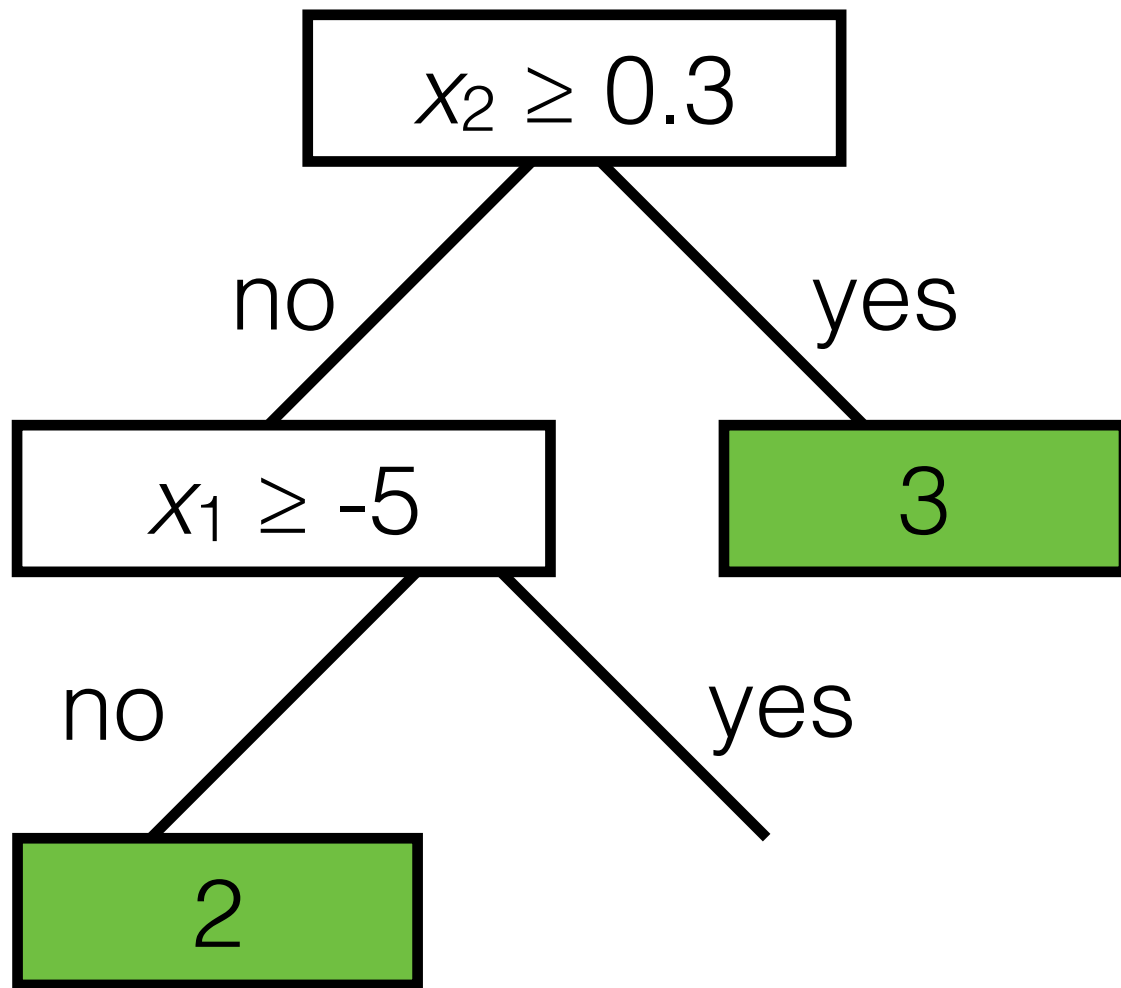
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

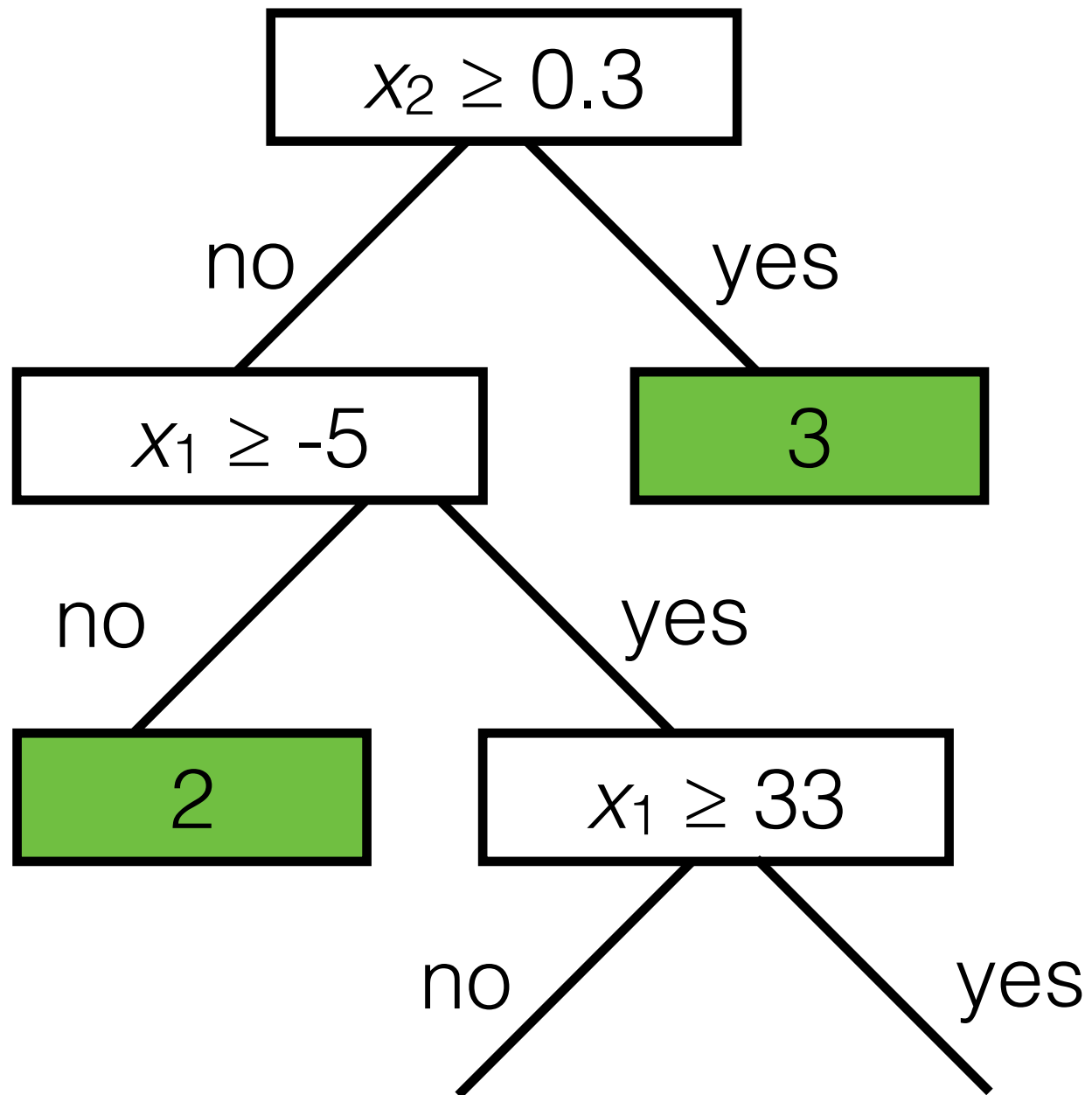
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

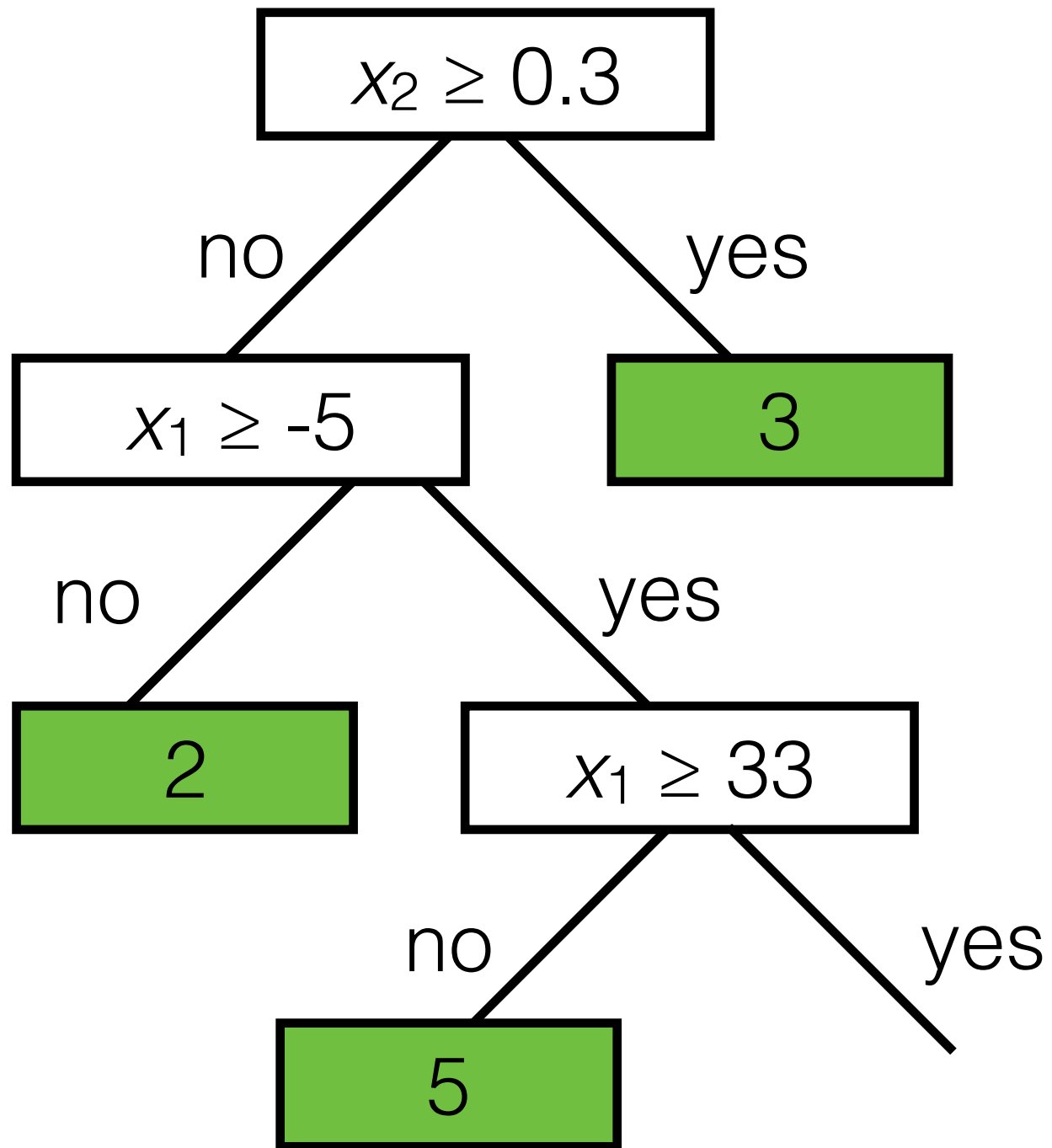
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

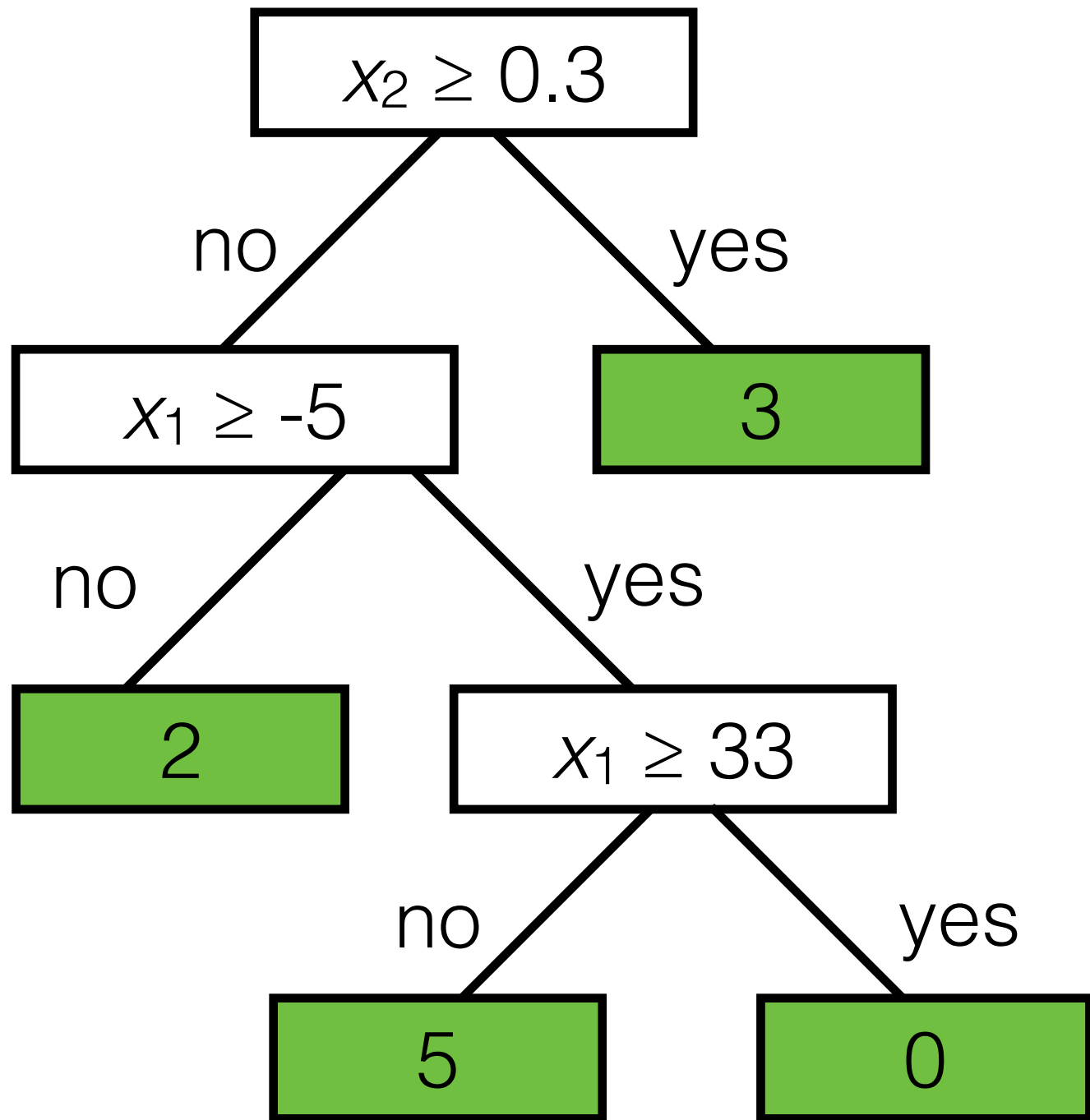
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

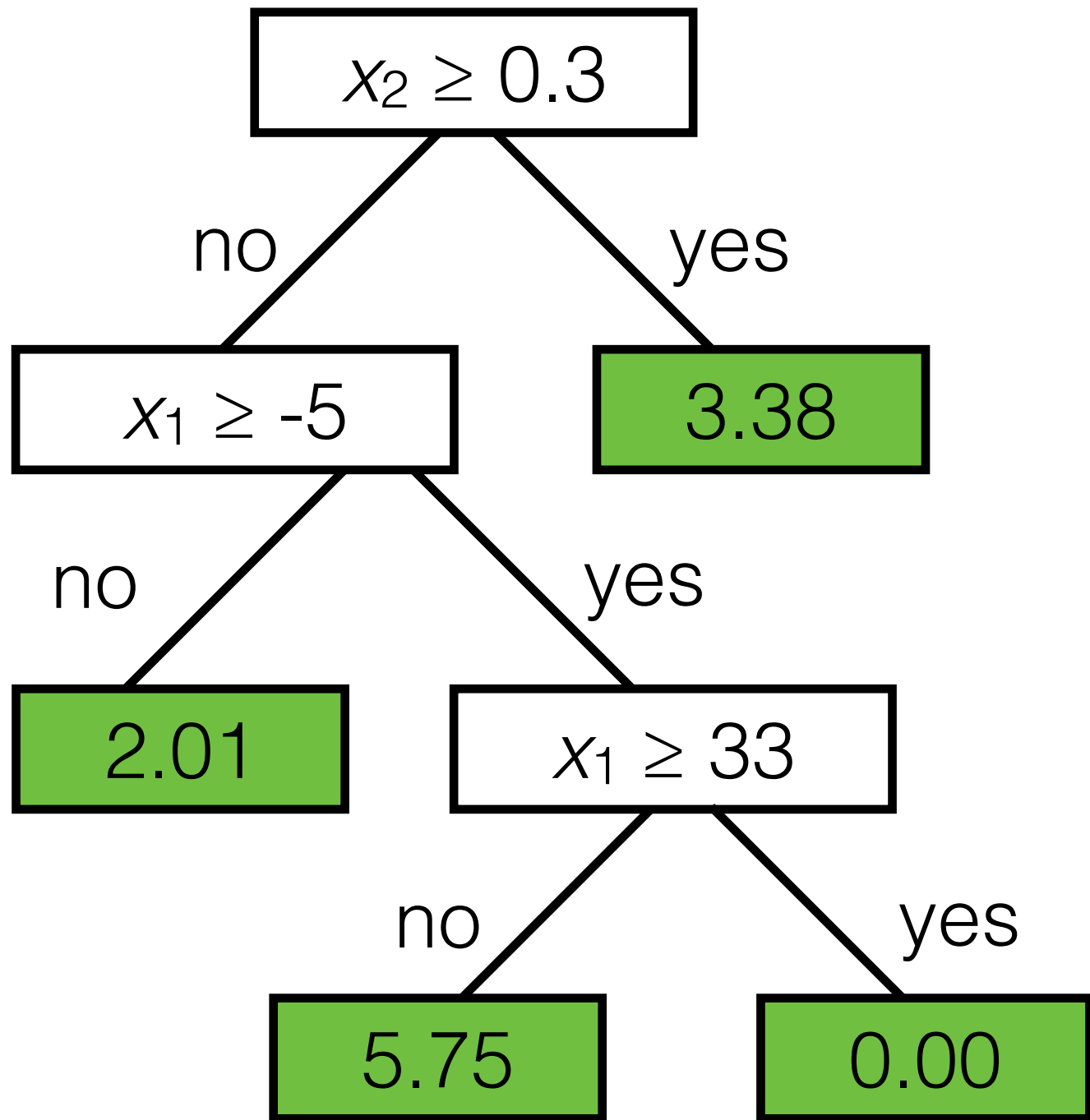
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

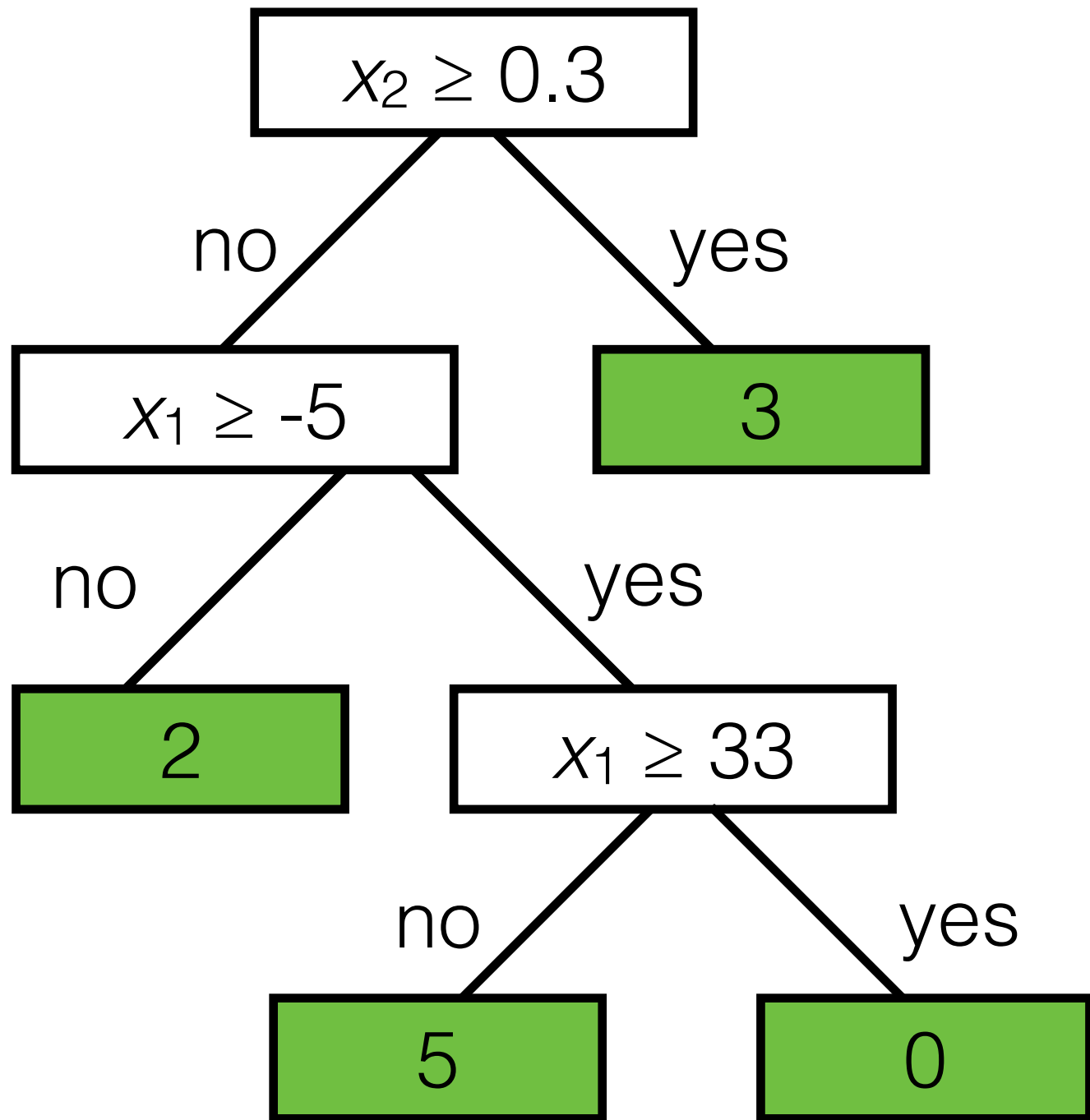
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

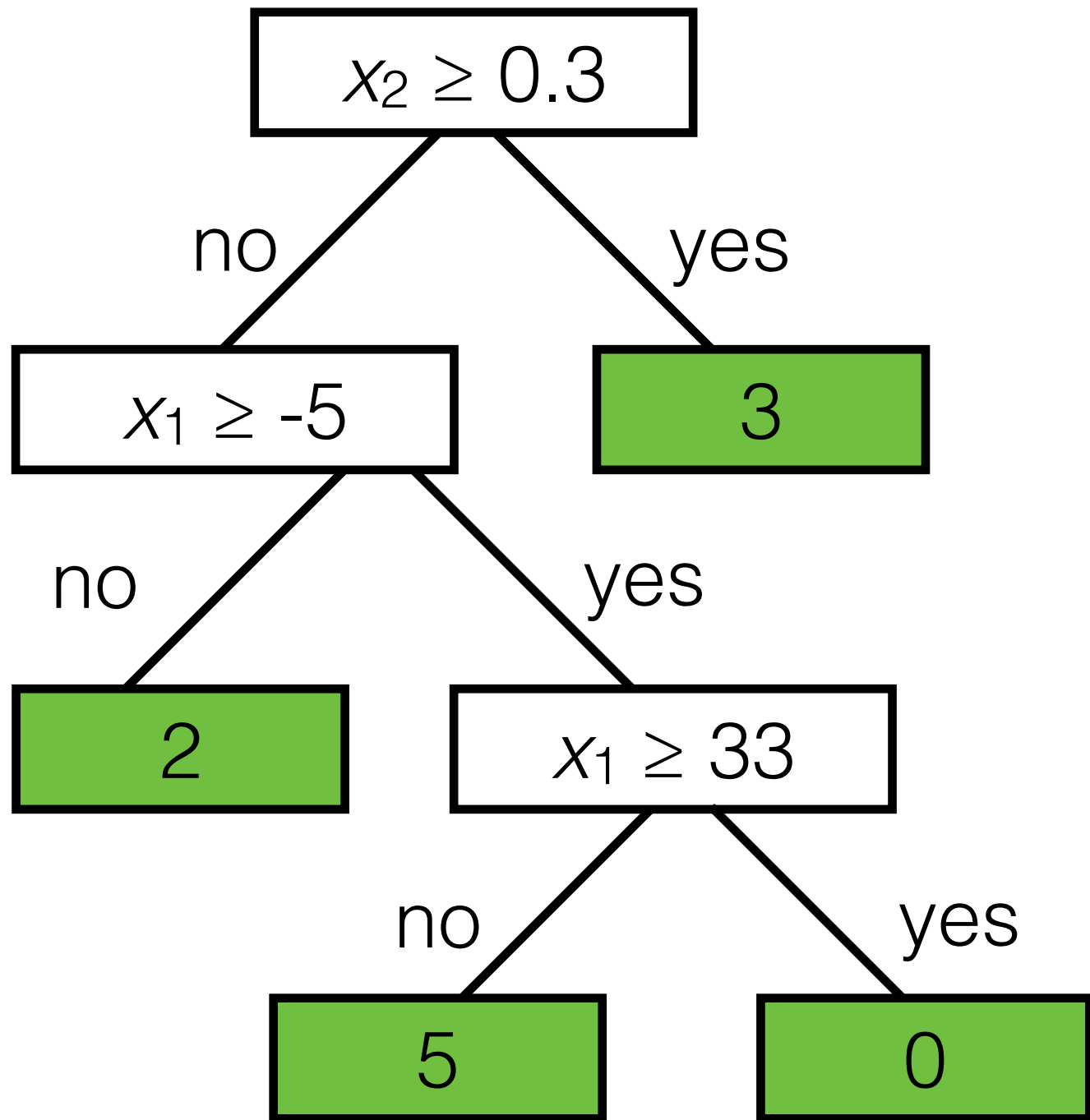
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

labels:

y : km run

Regression tree



features:

x_1 : temperature (deg C)

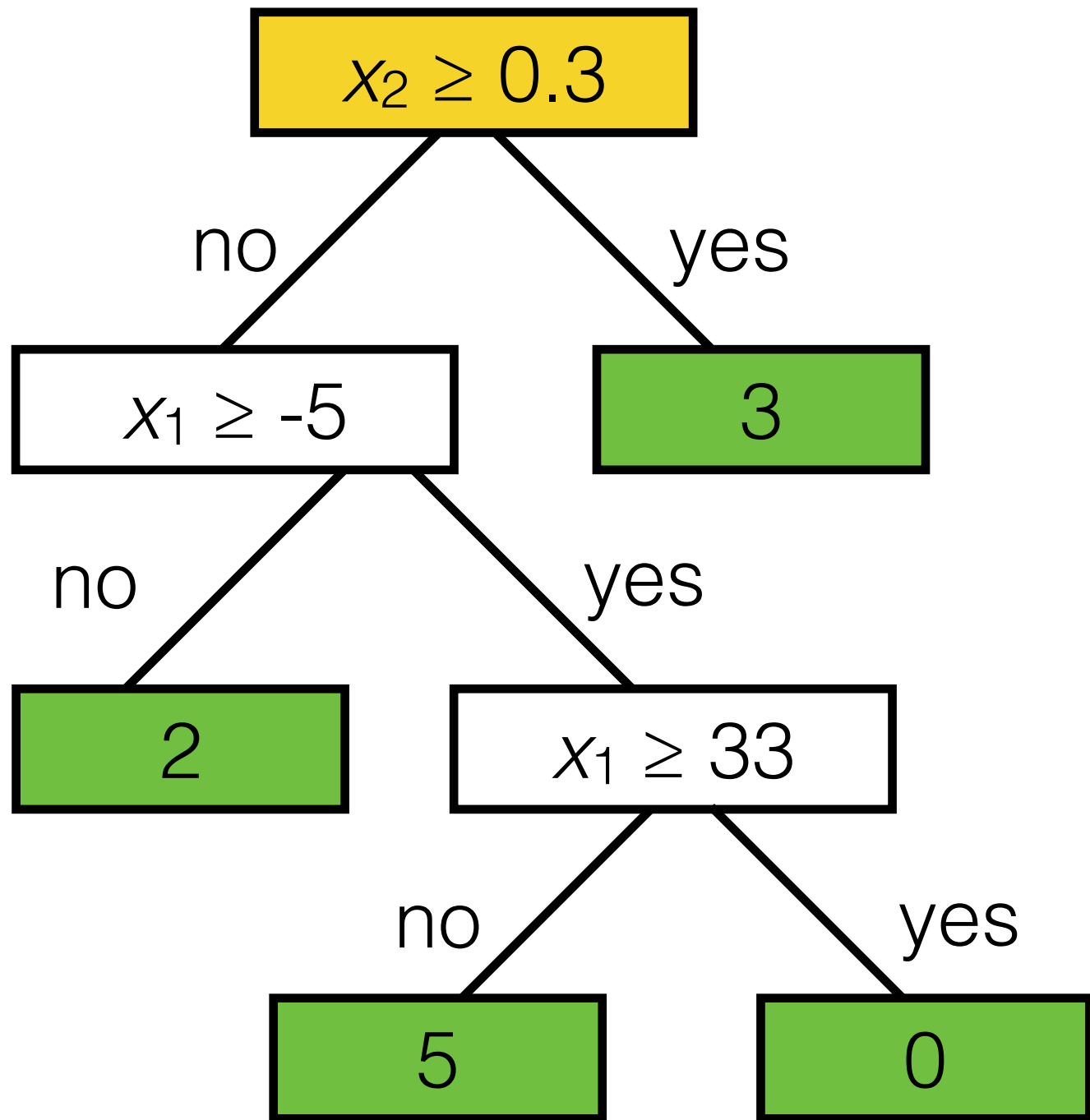
x_2 : precipitation (cm/hr)

labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:

Regression tree



features:

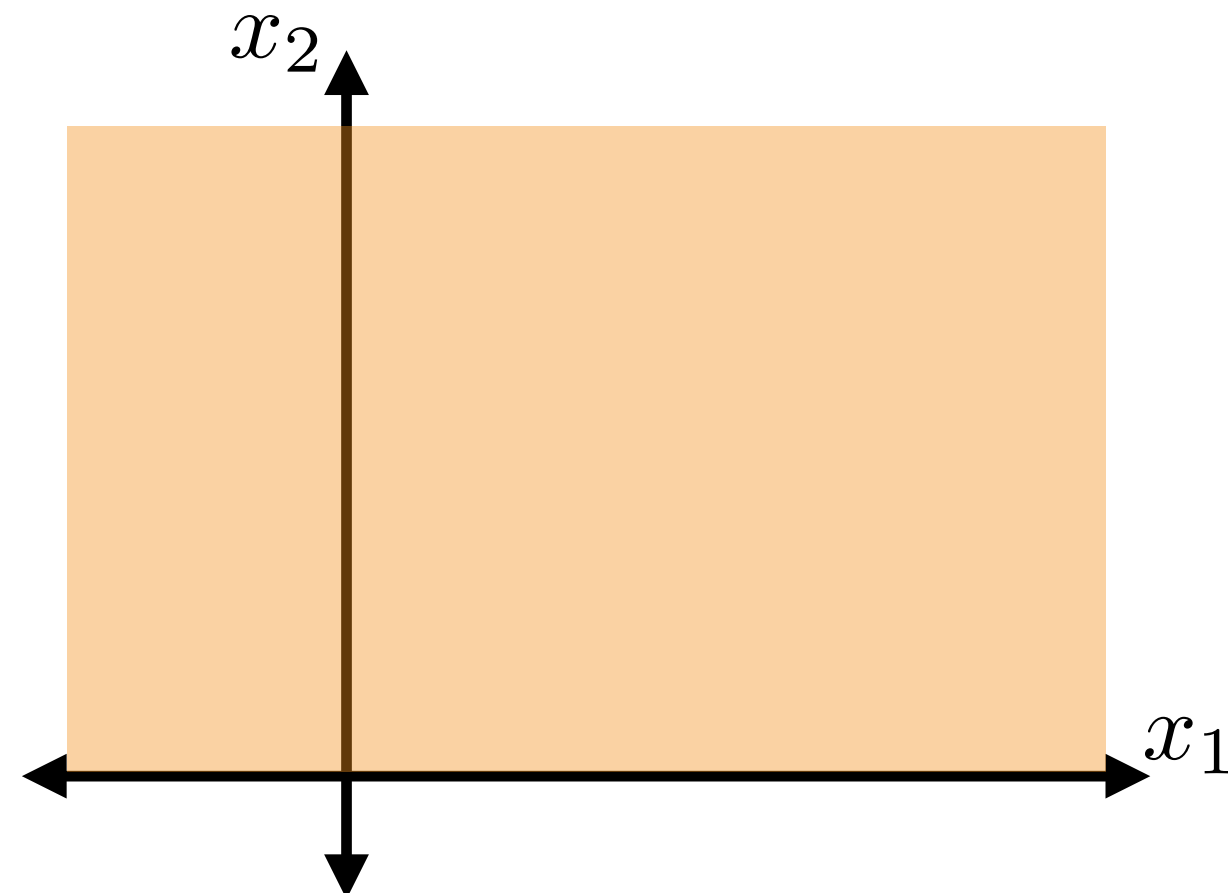
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

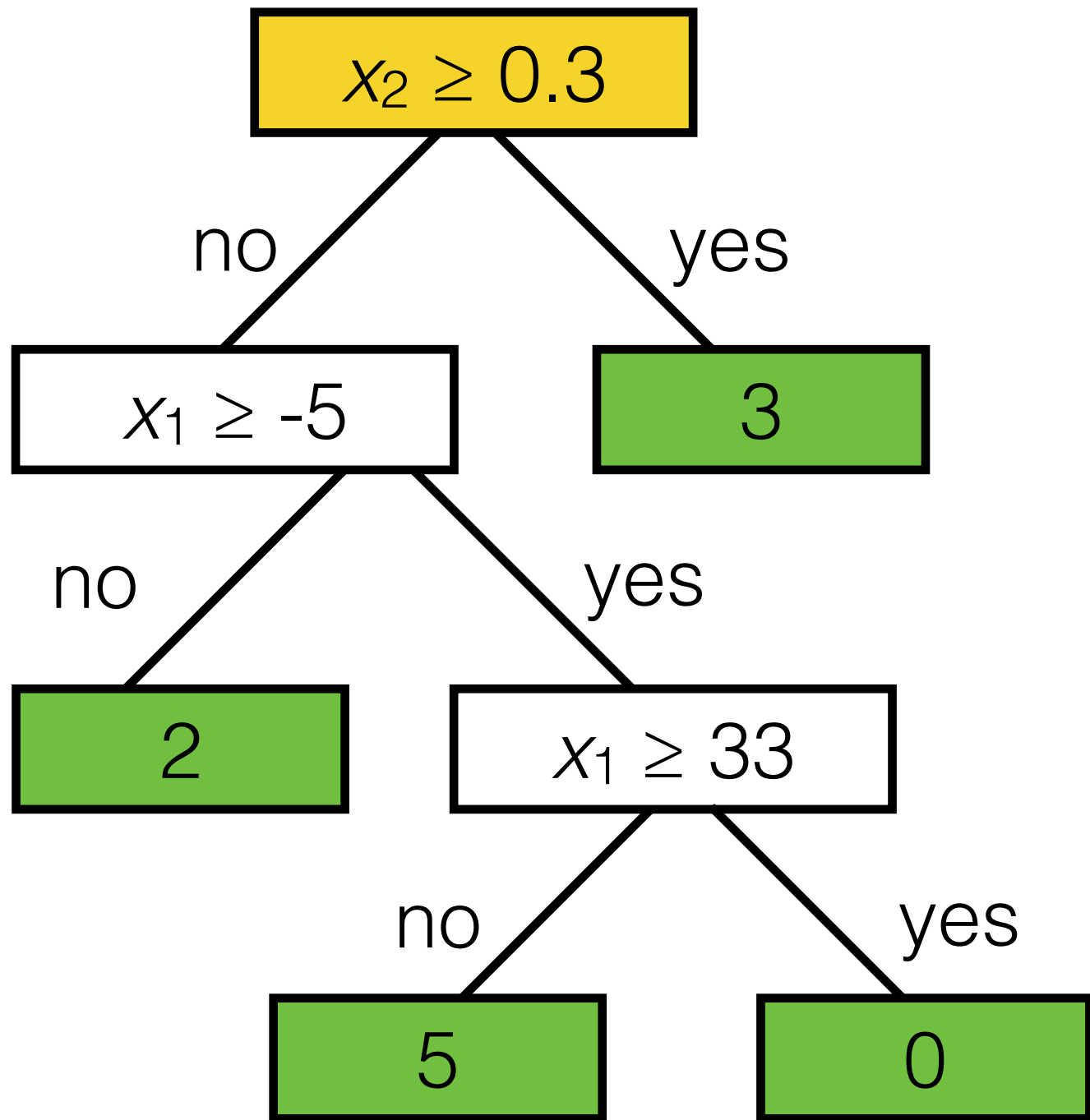
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

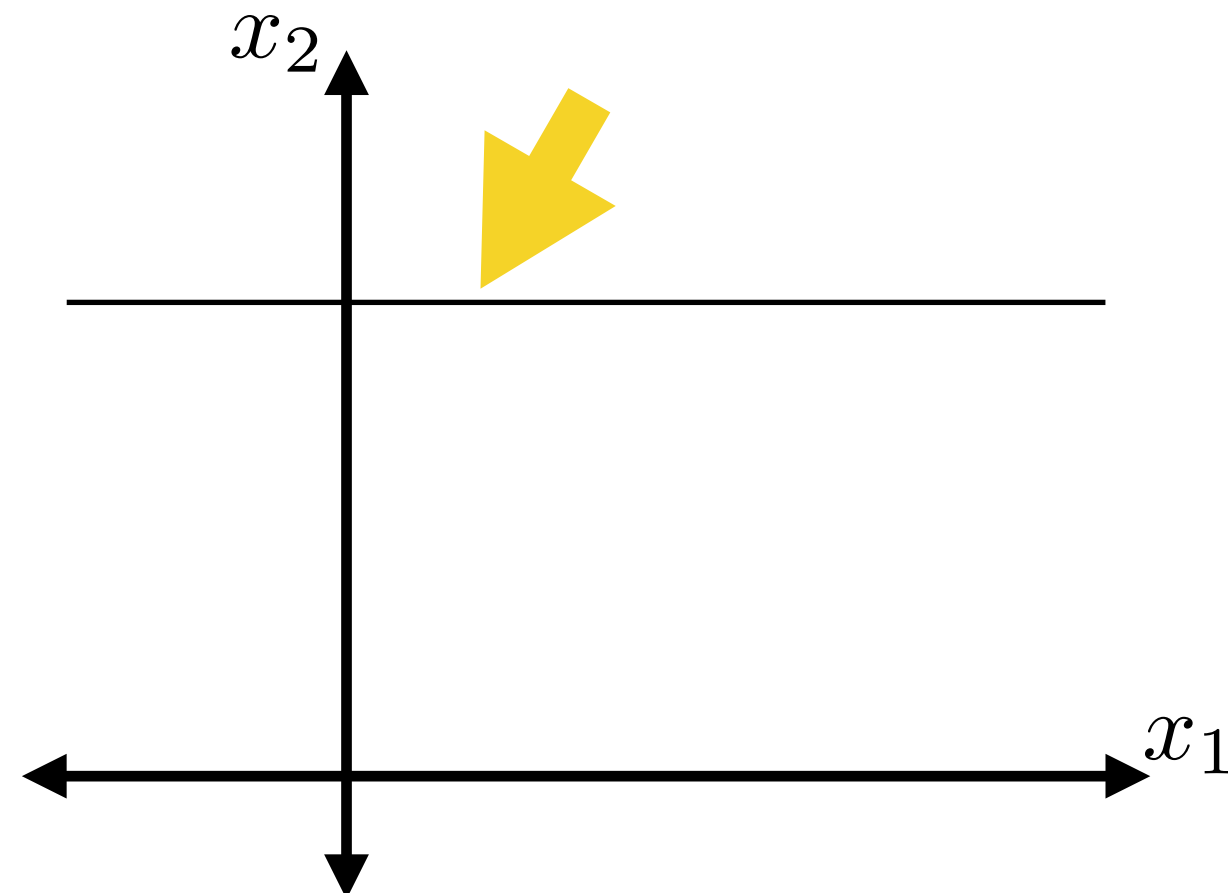
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

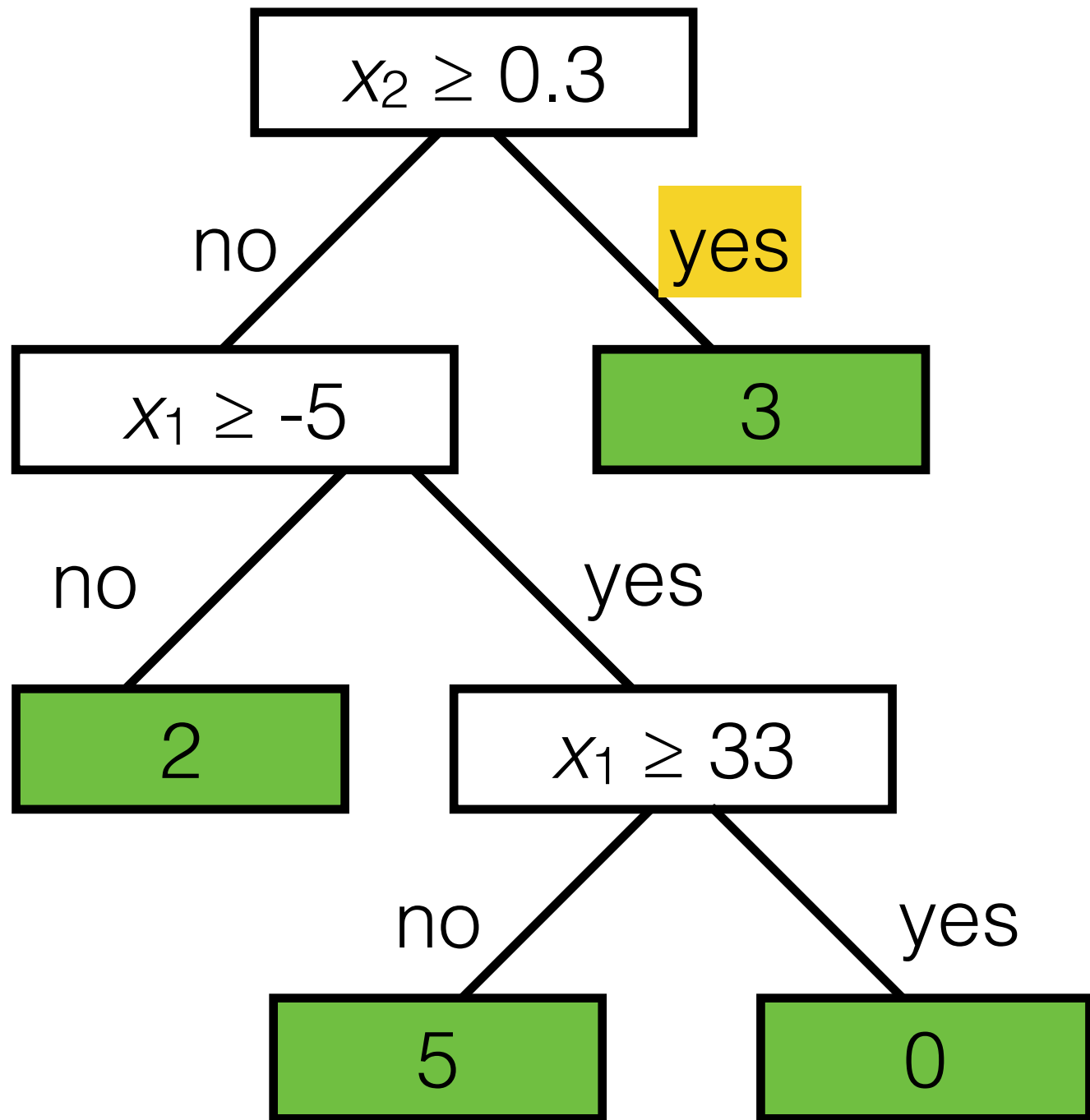
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

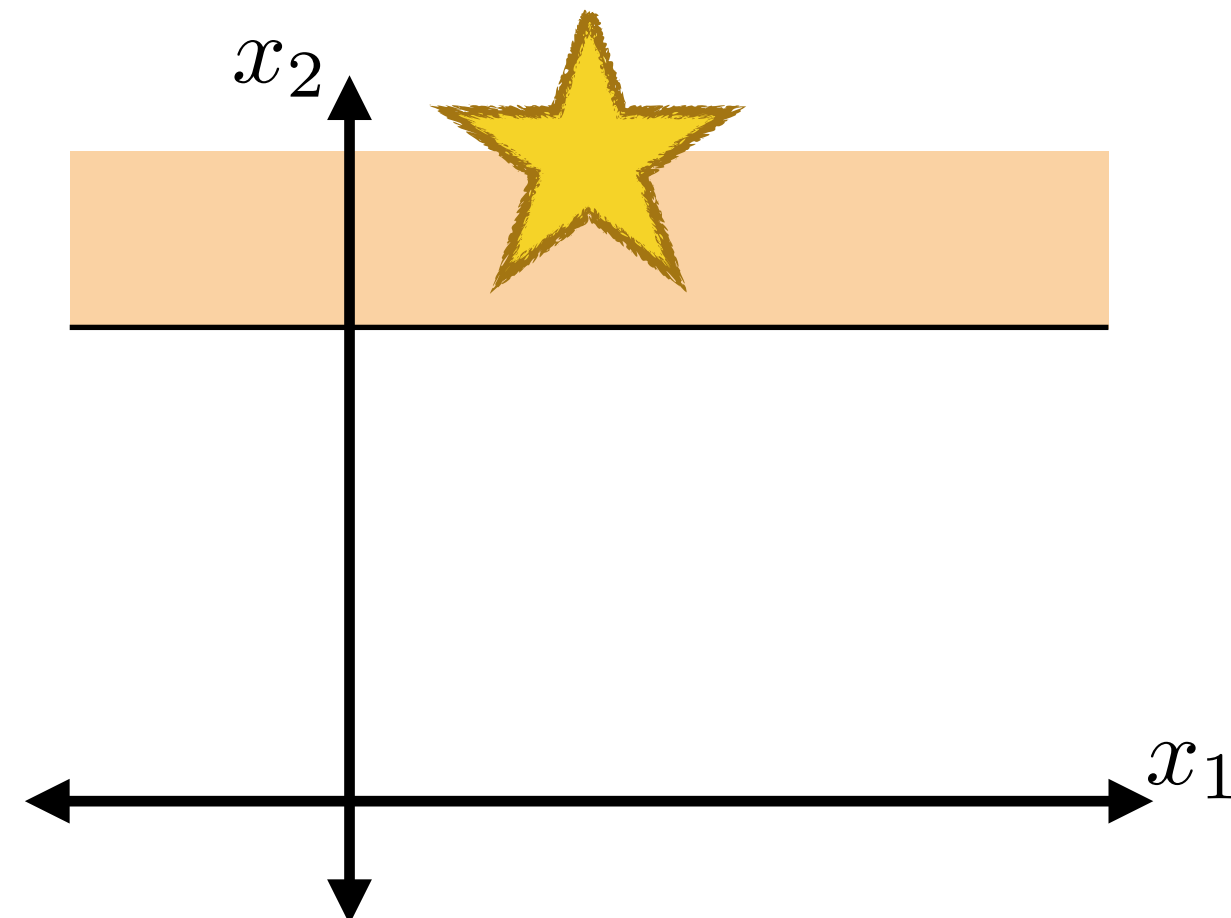
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

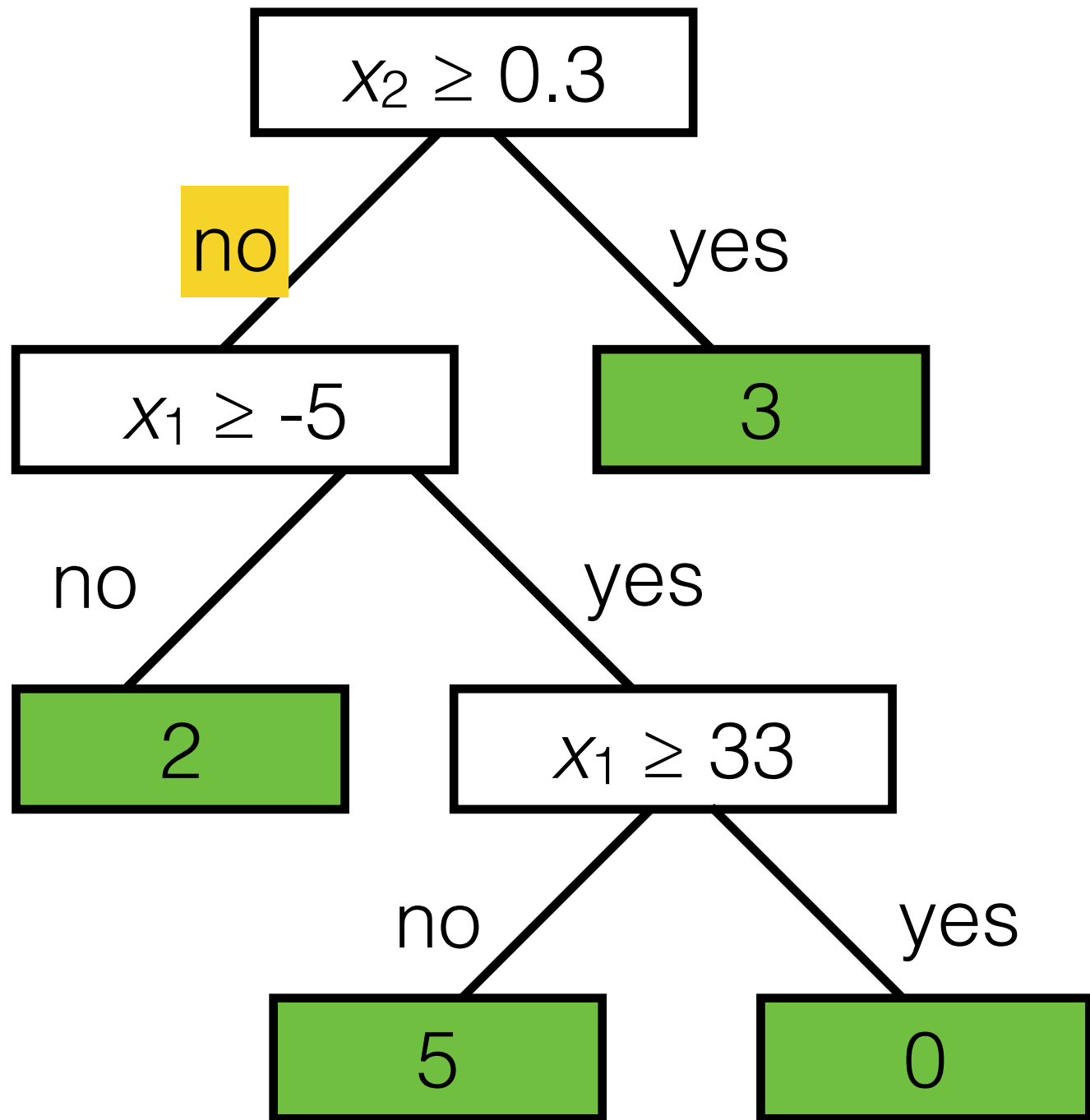
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

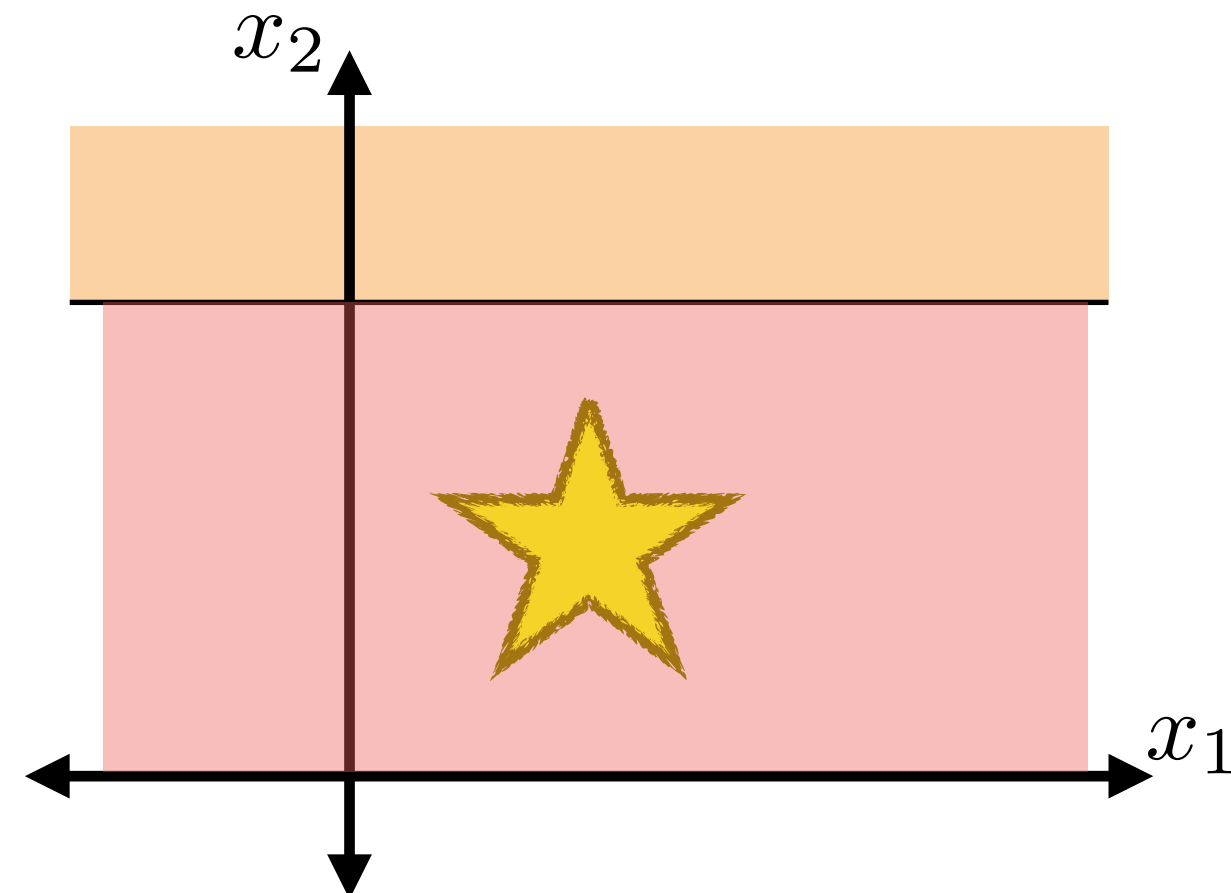
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

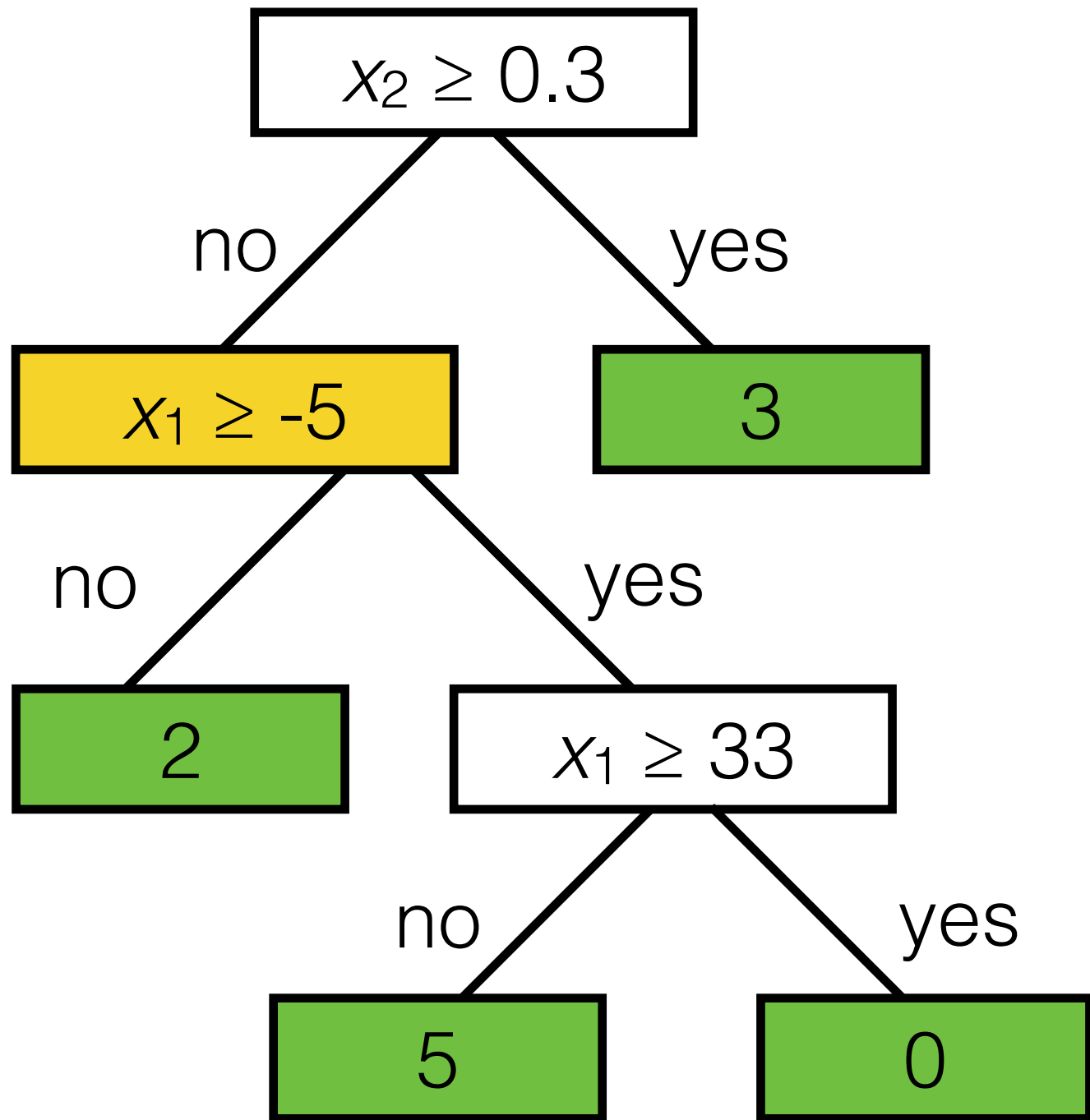
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

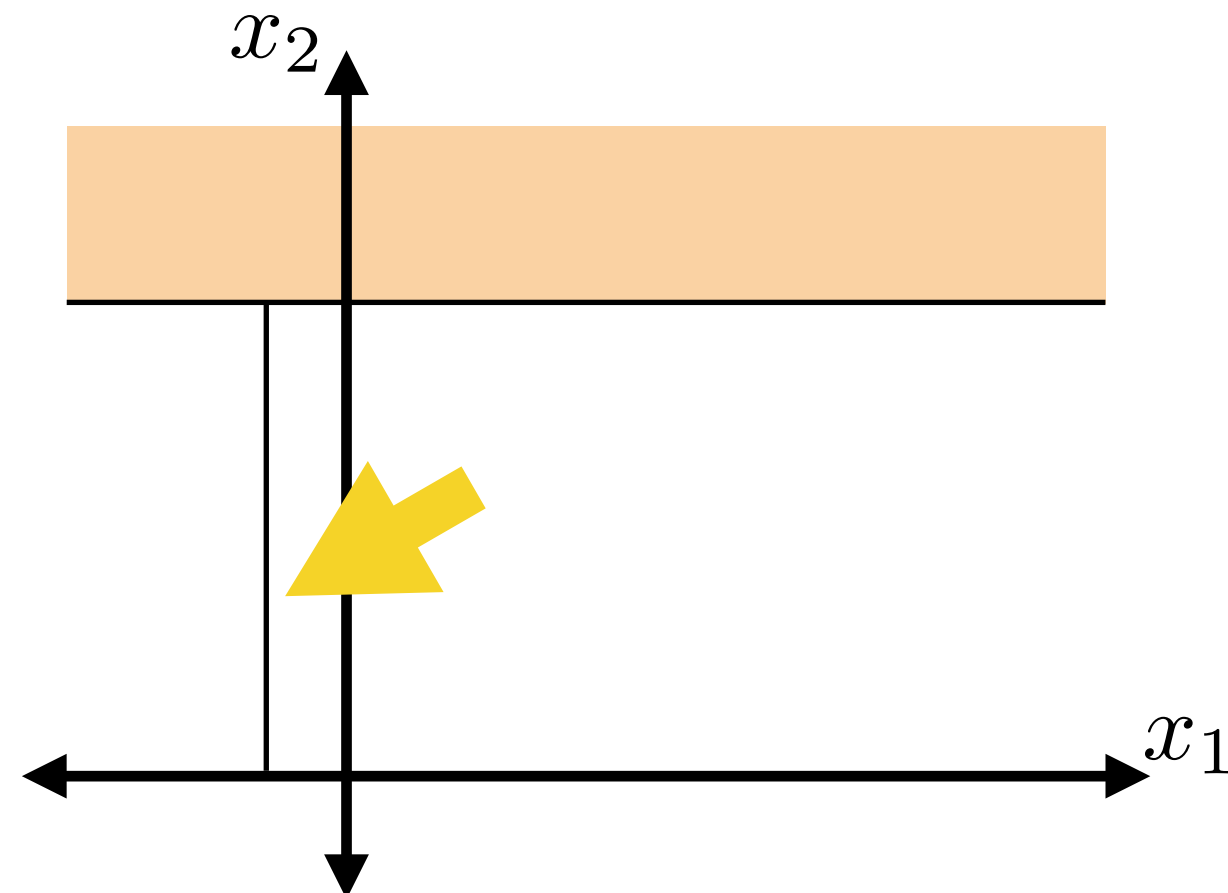
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

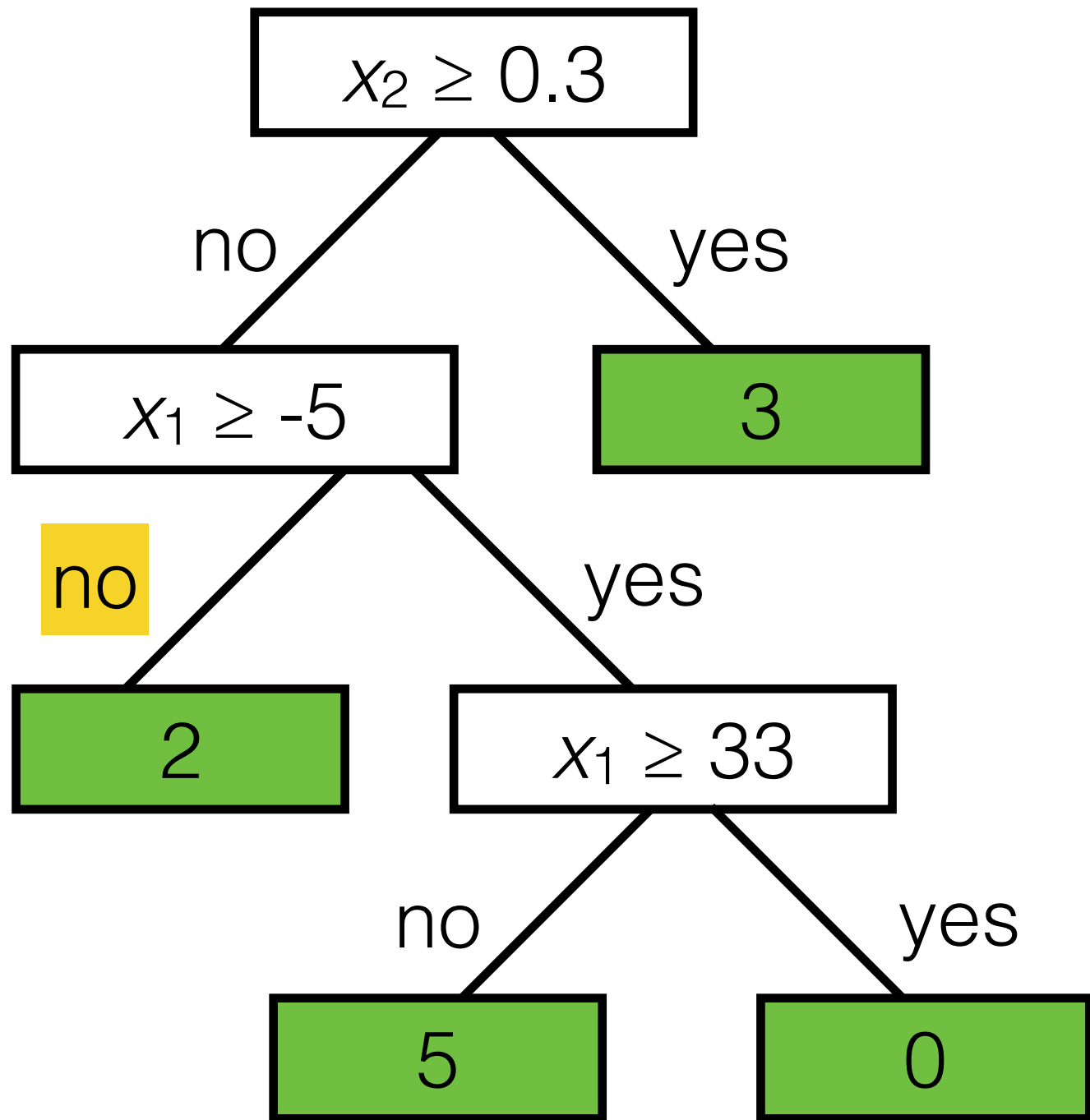
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

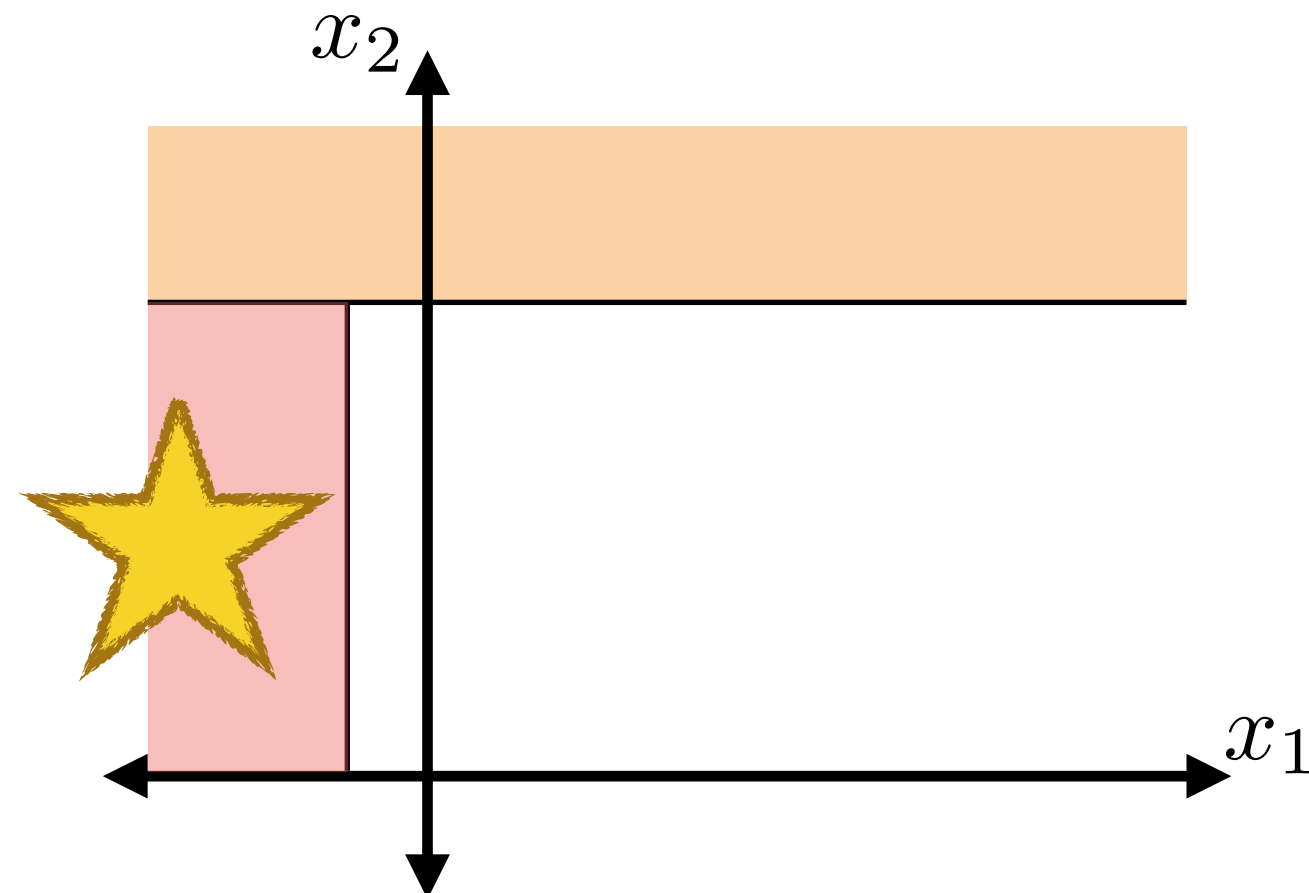
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

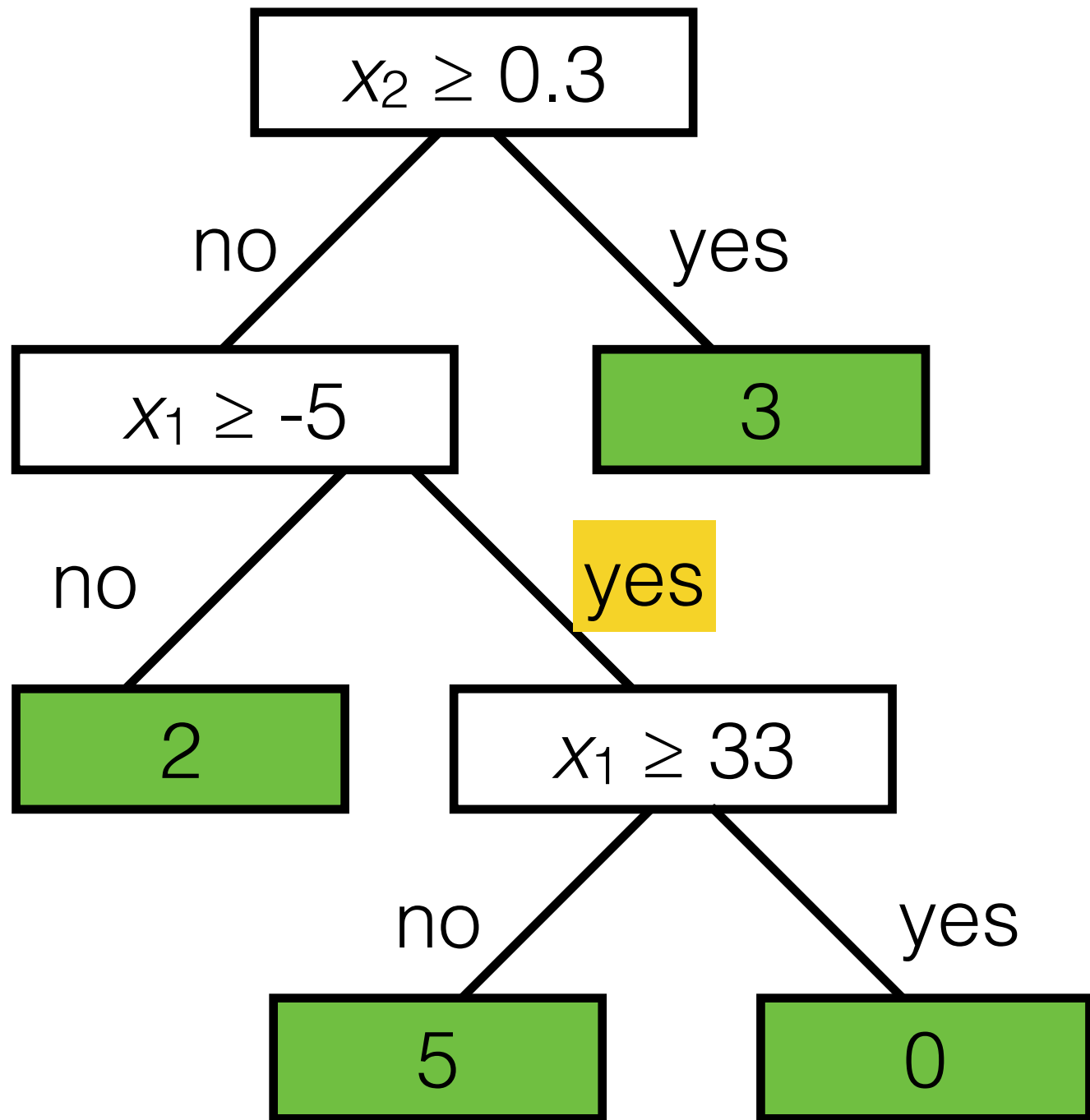
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

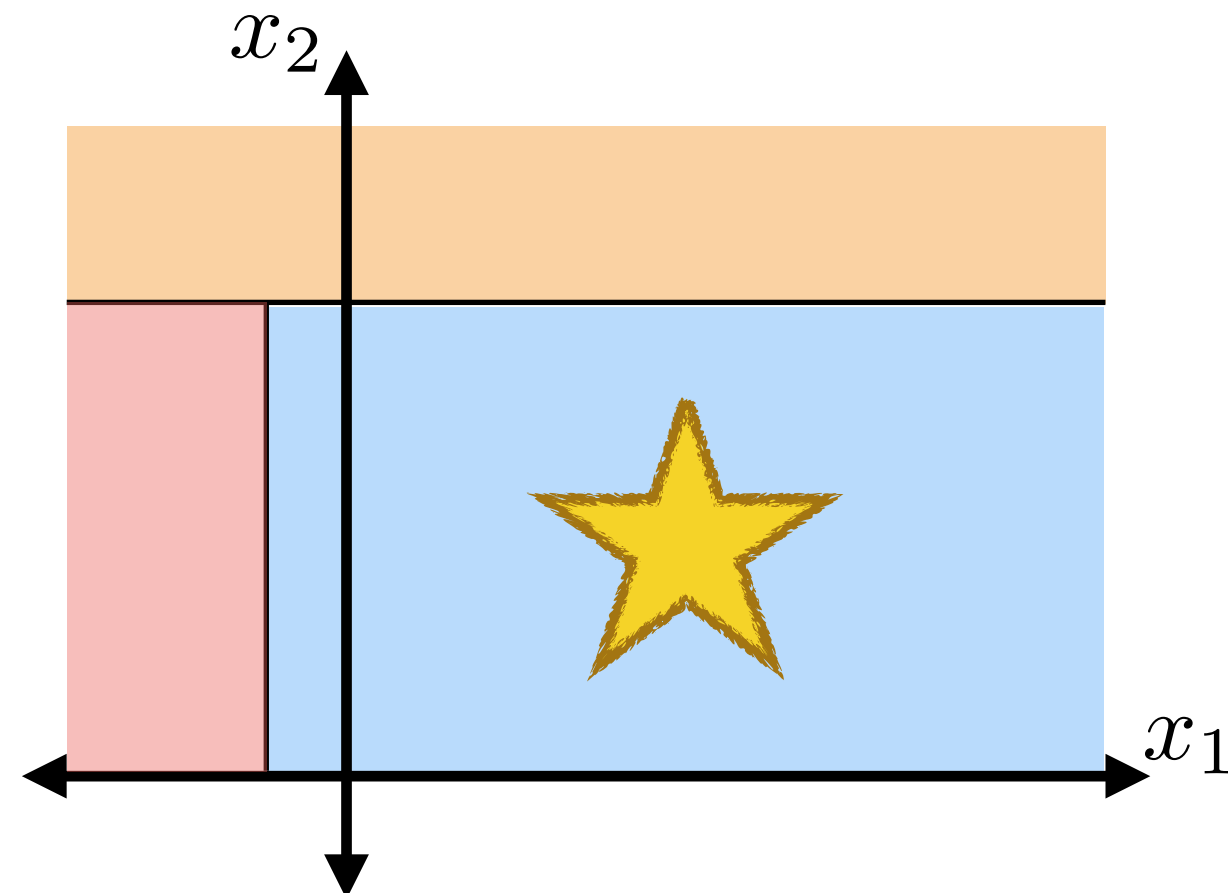
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

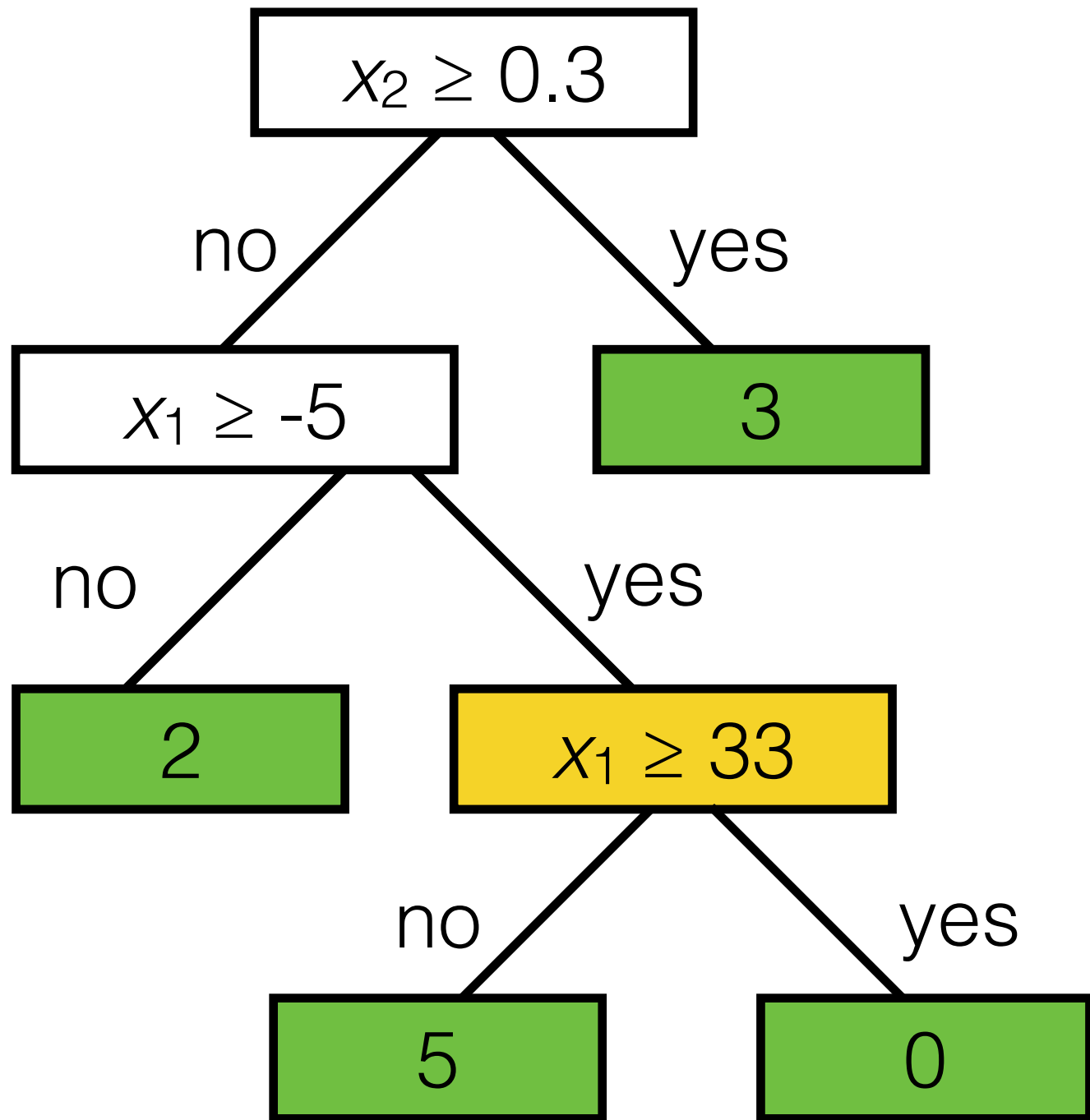
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

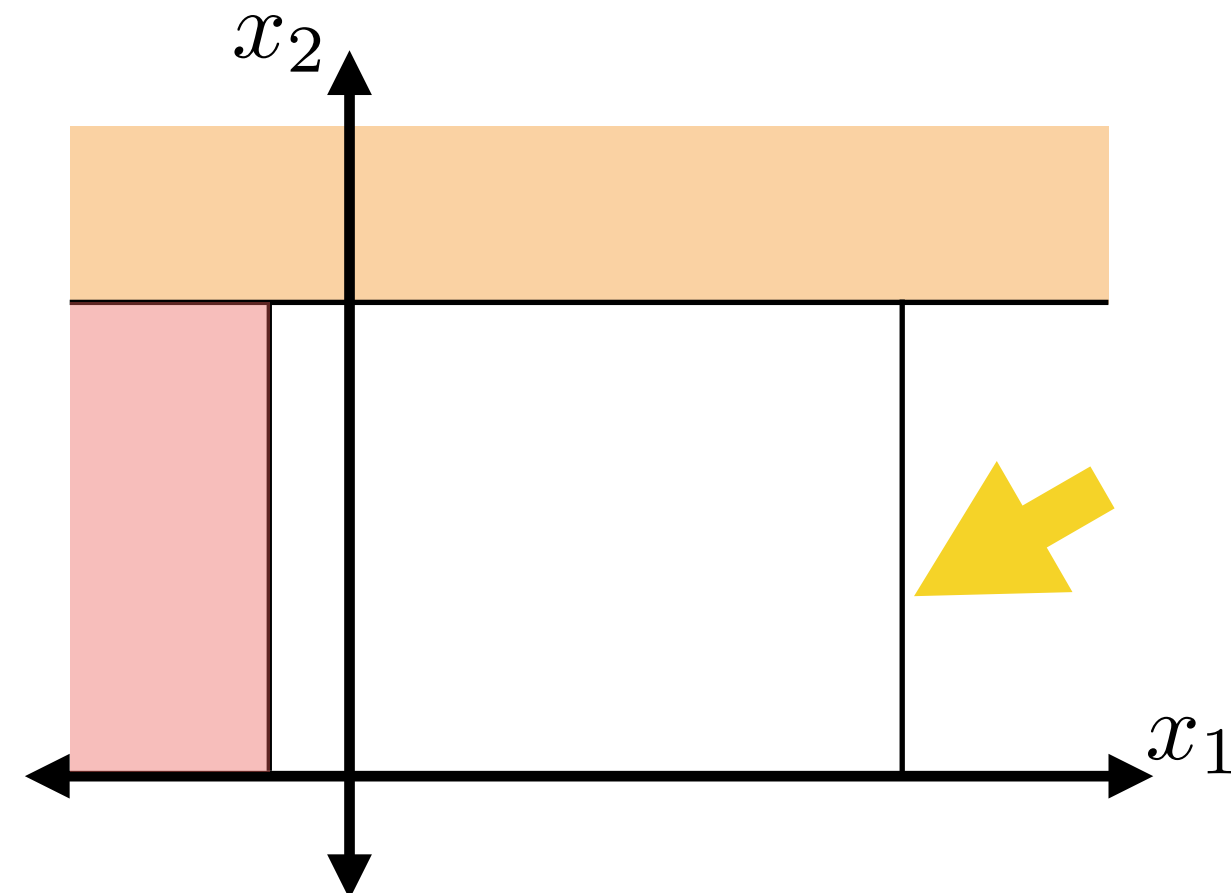
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

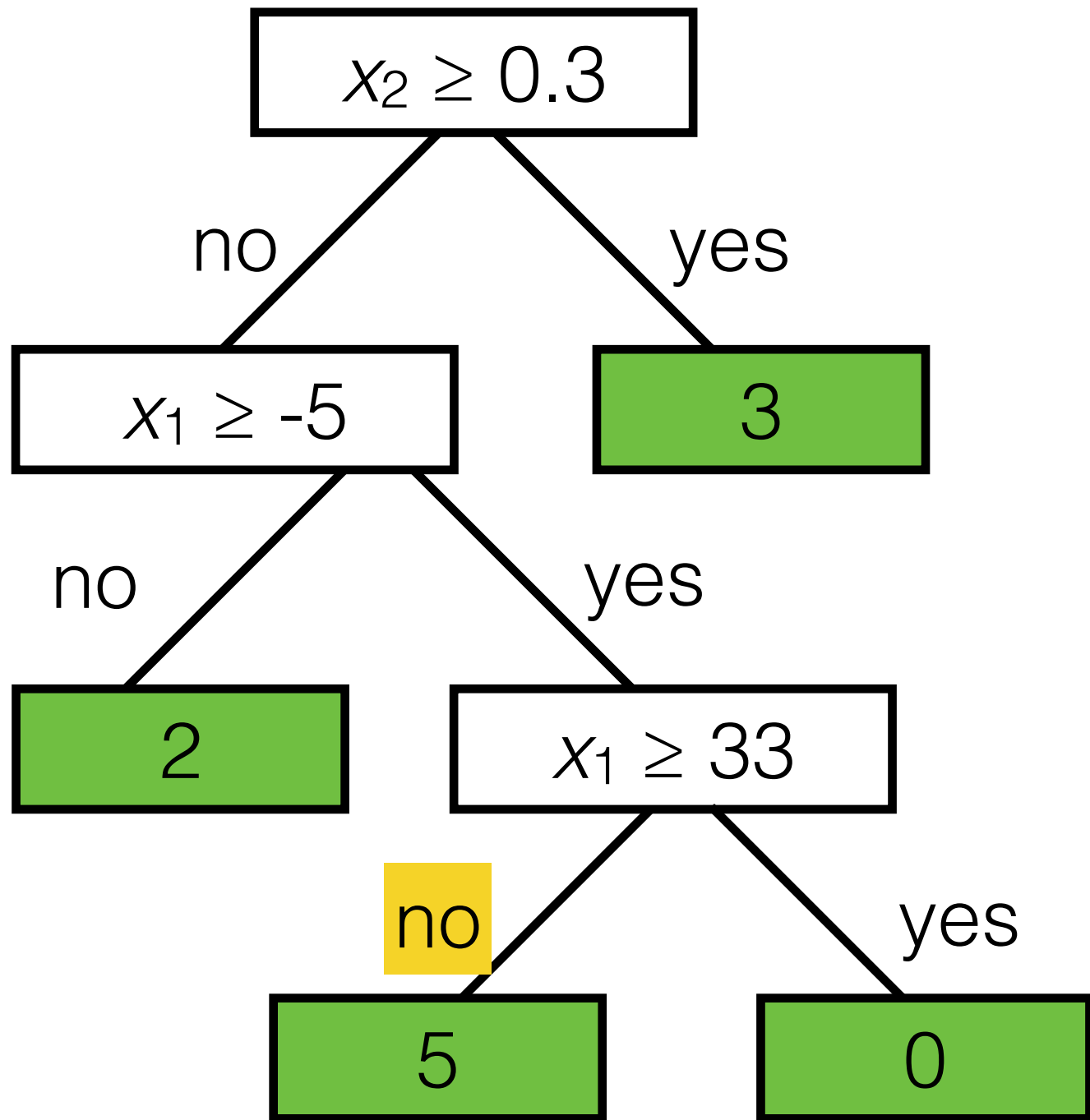
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

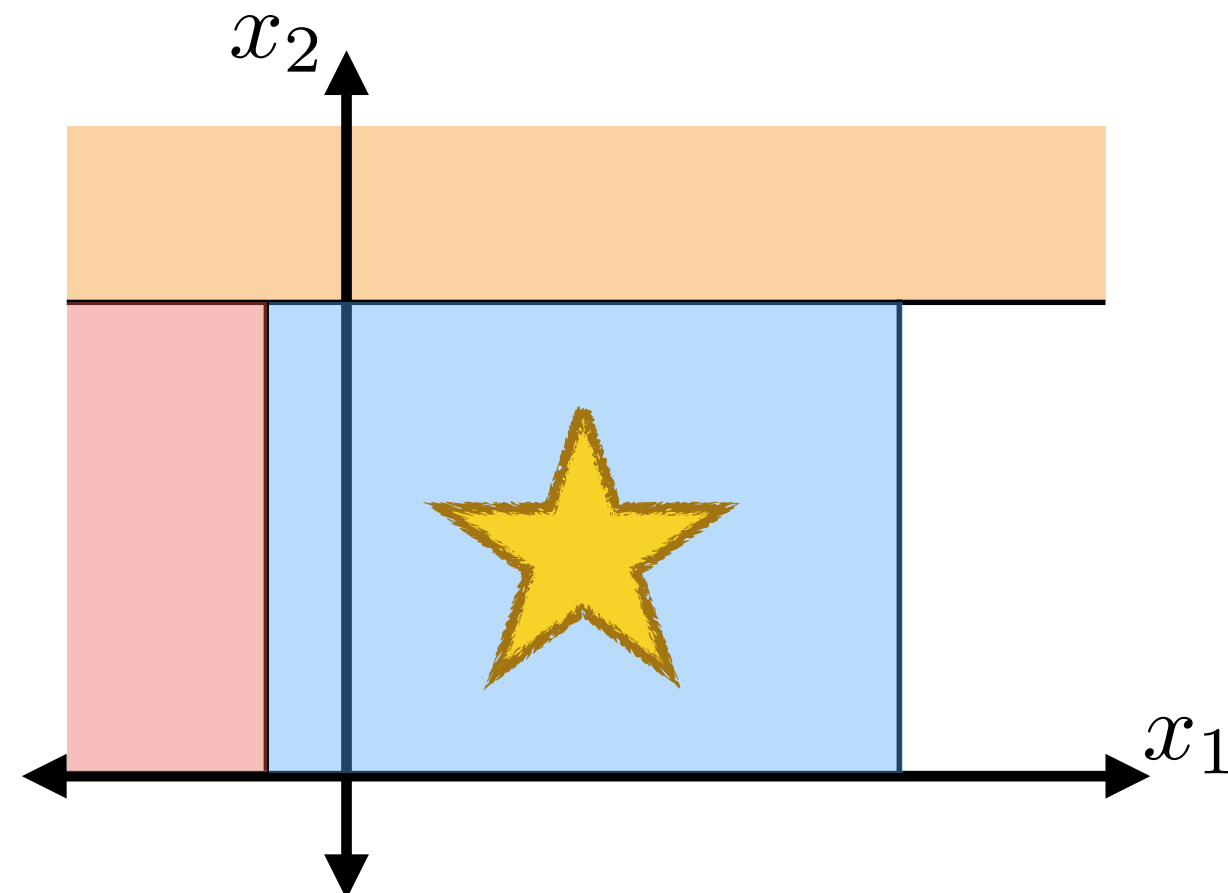
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

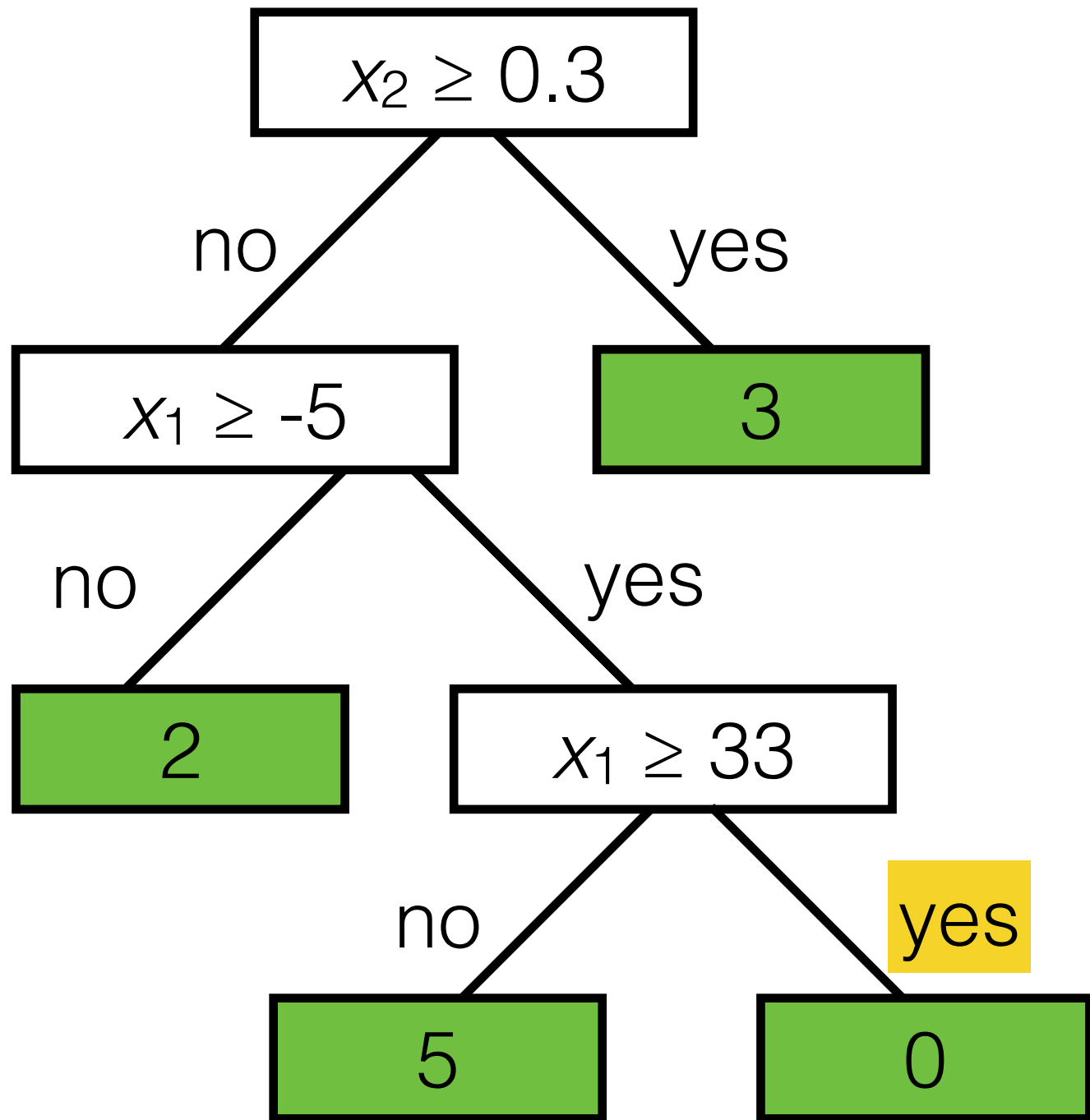
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

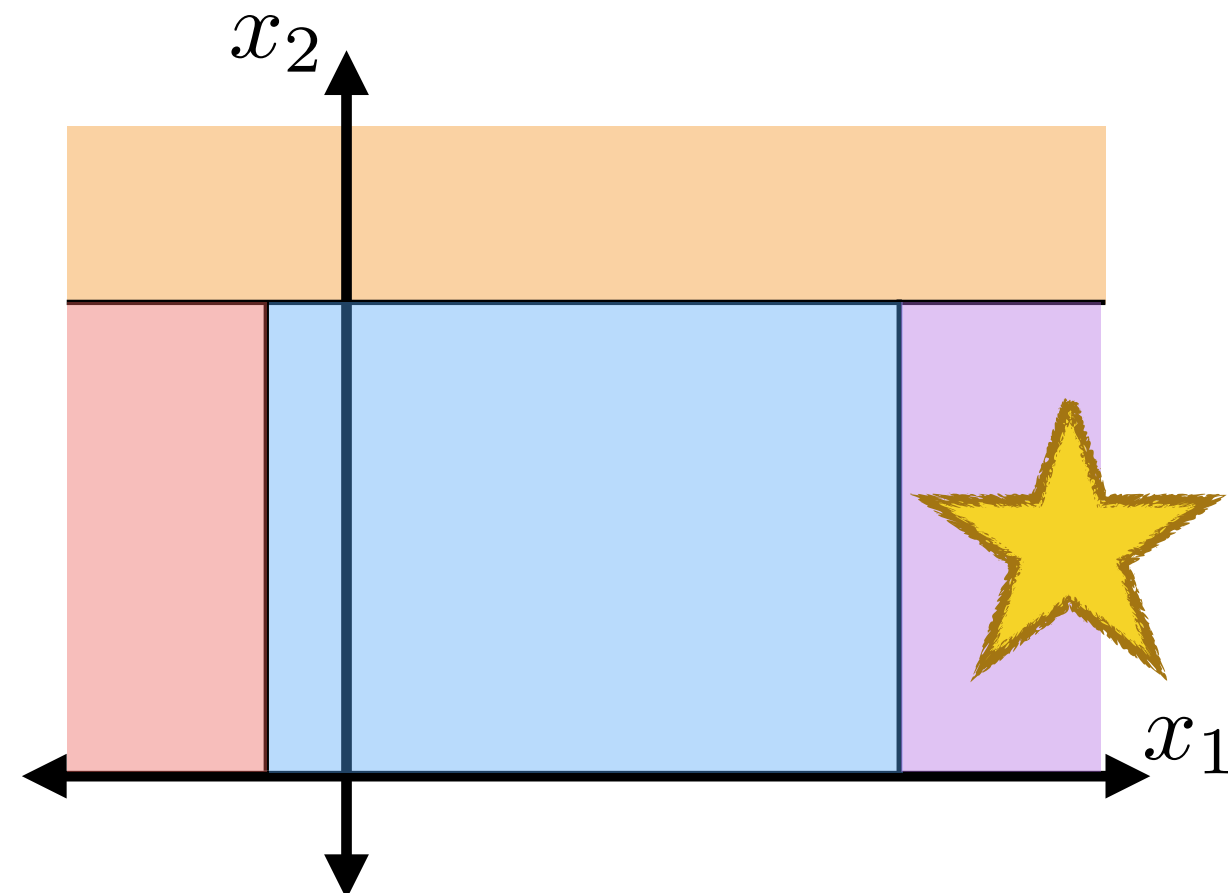
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

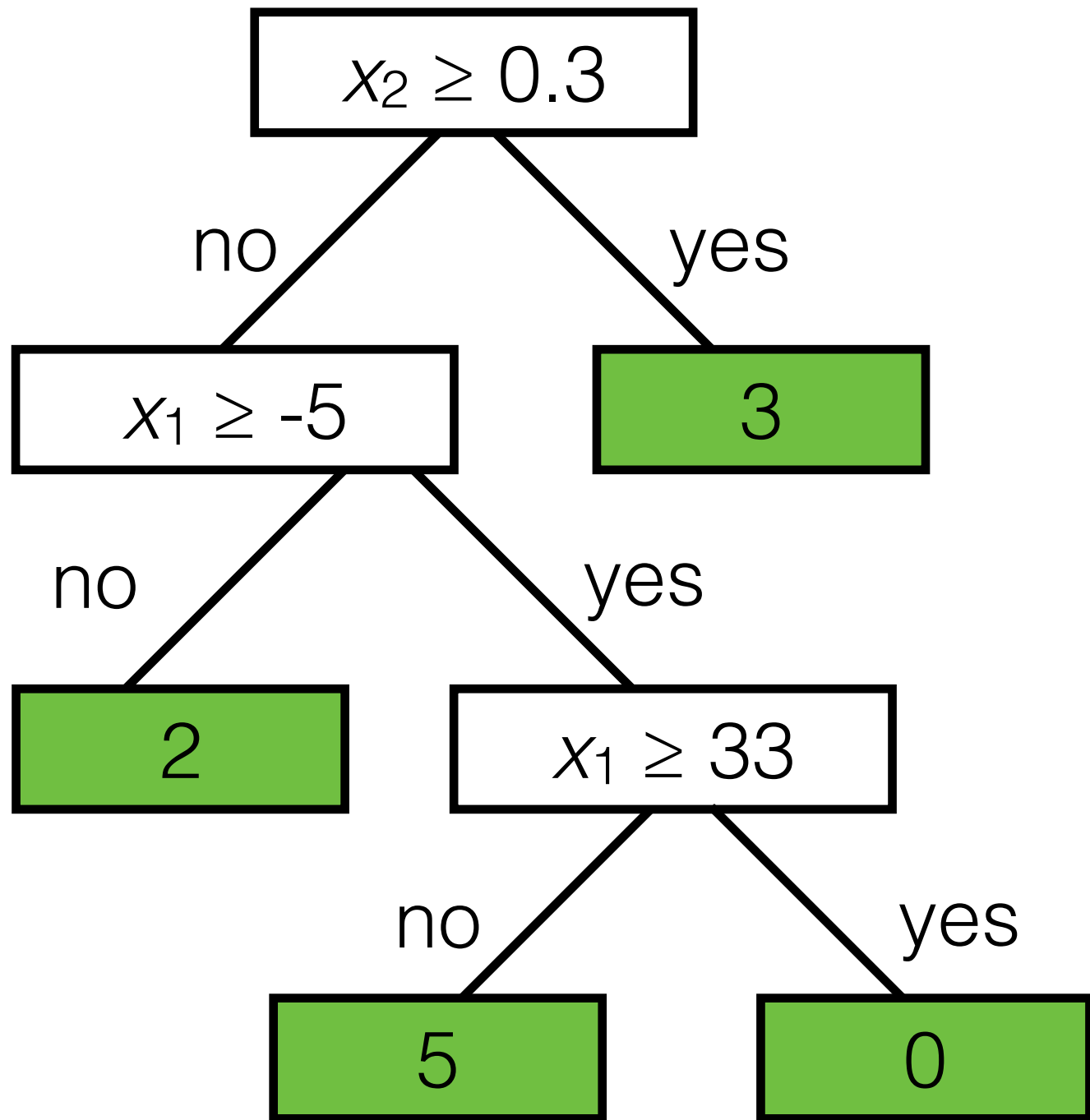
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:



Regression tree



features:

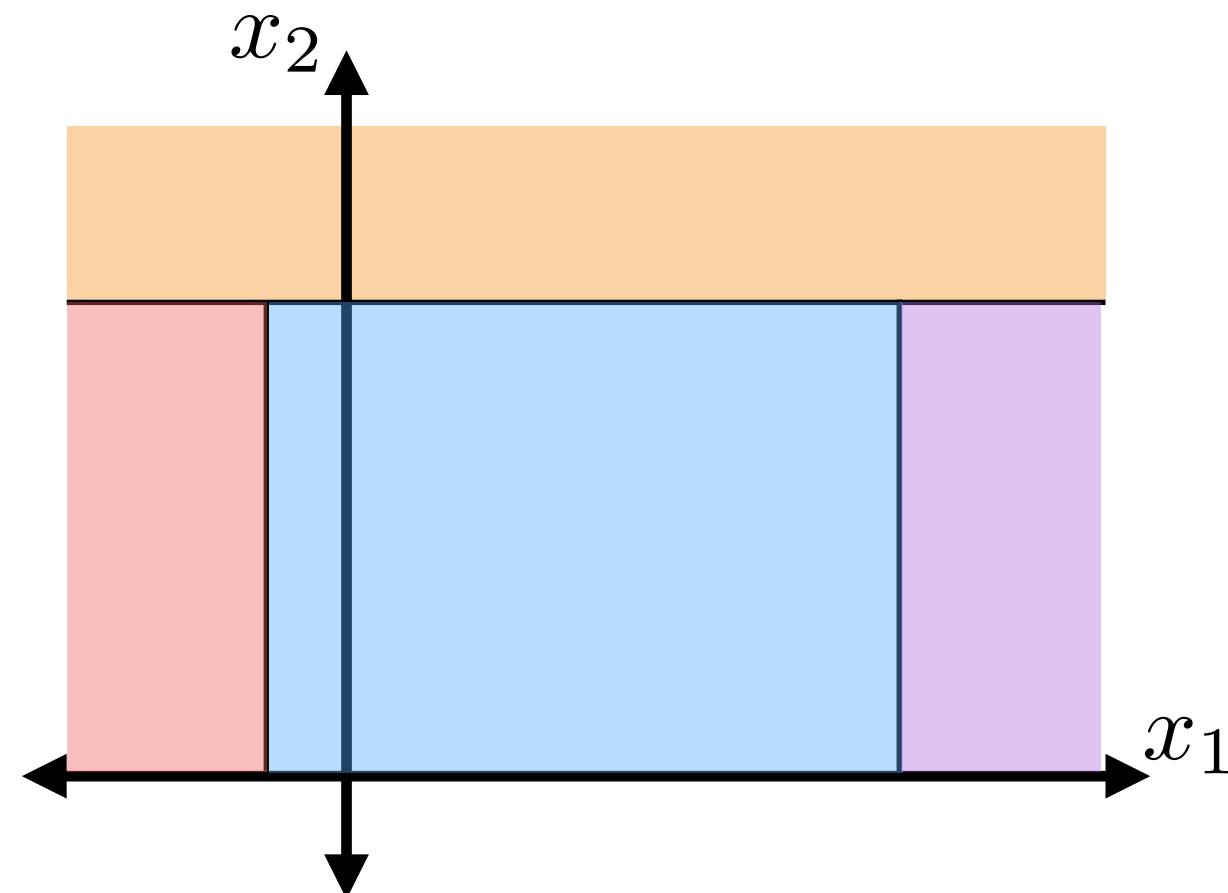
x_1 : temperature (deg C)

x_2 : precipitation (cm/hr)

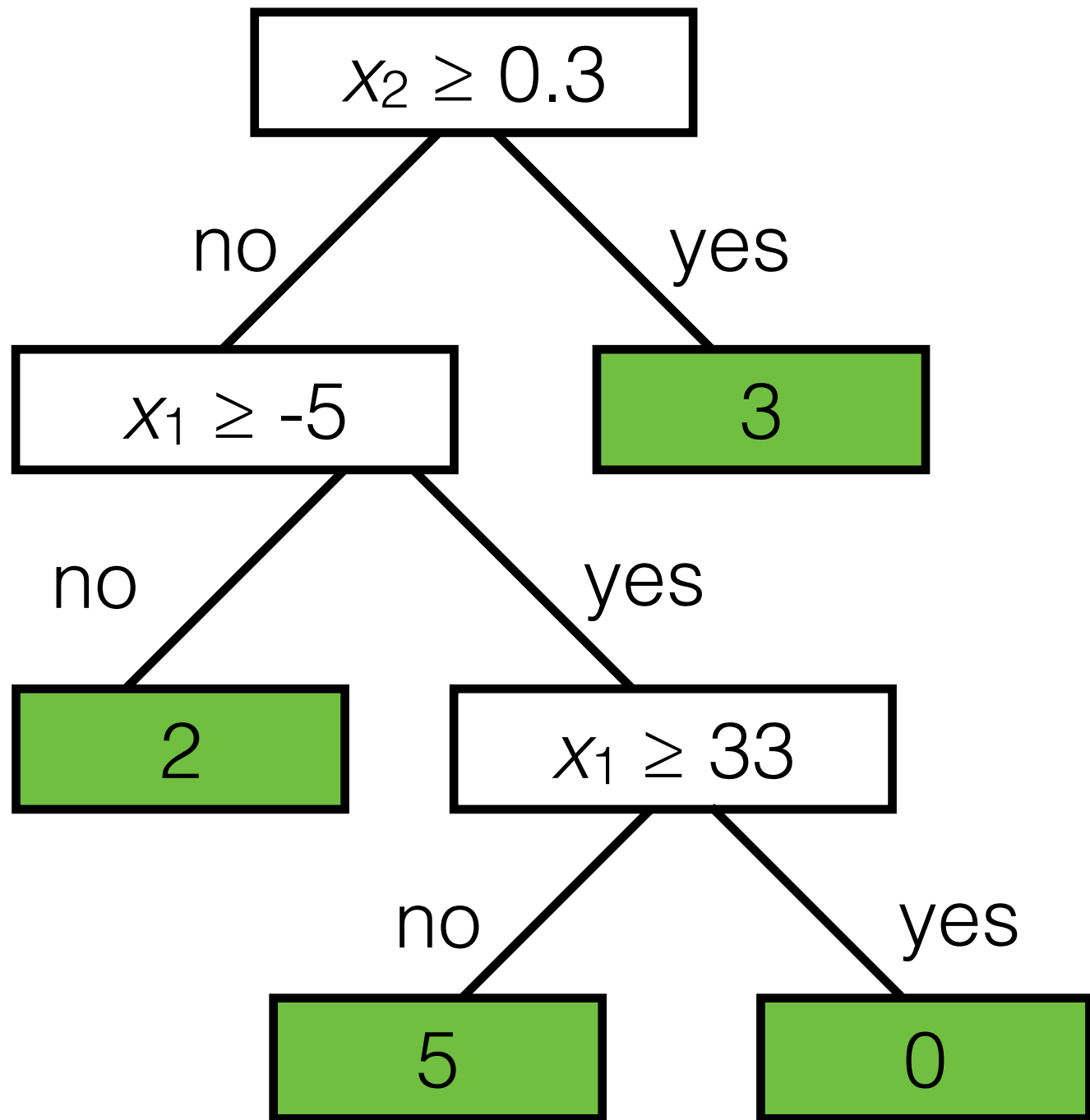
labels:

y : km run

- Tree defines an axis-aligned “partition” of the feature space:

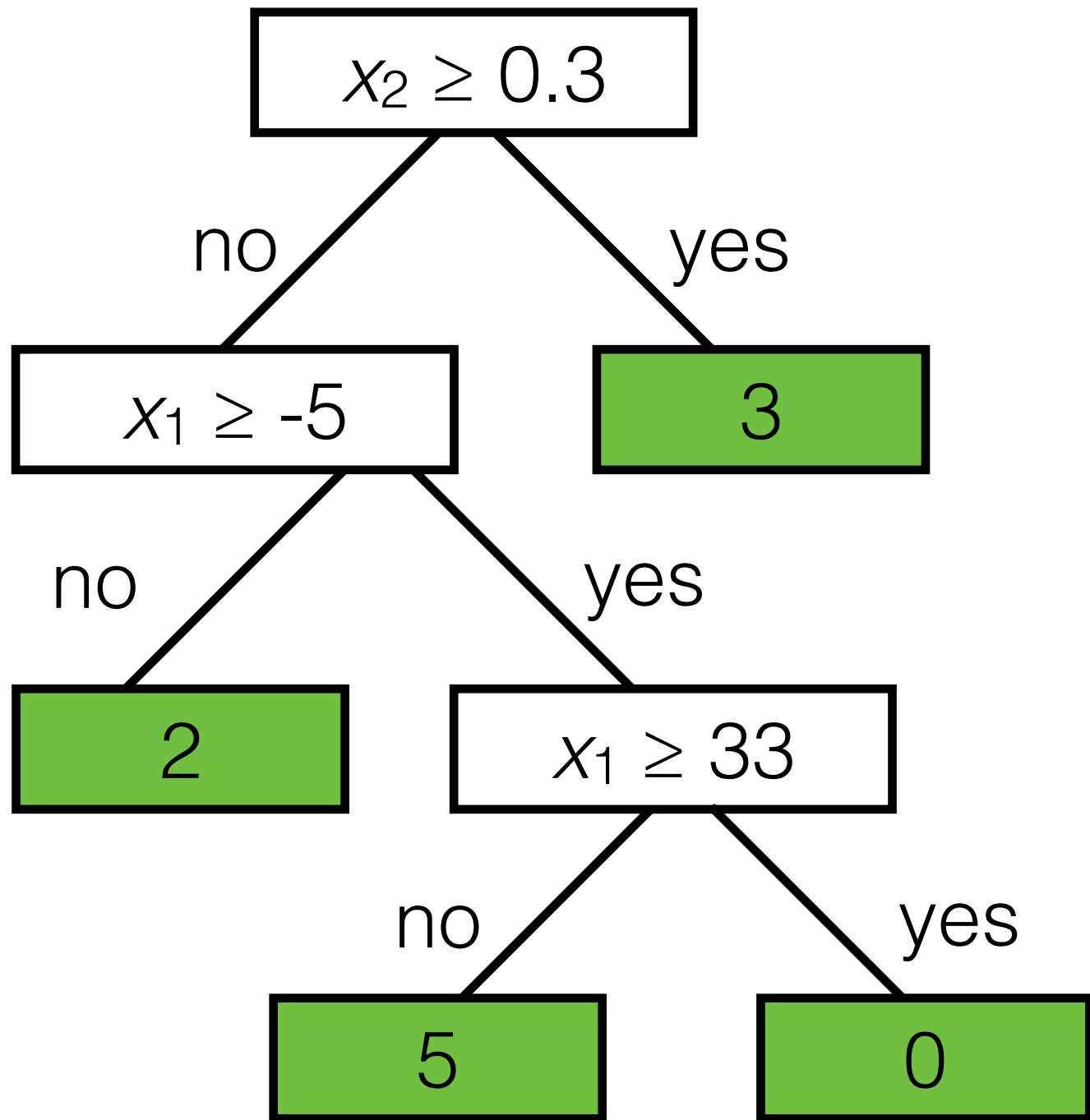


Decision tree

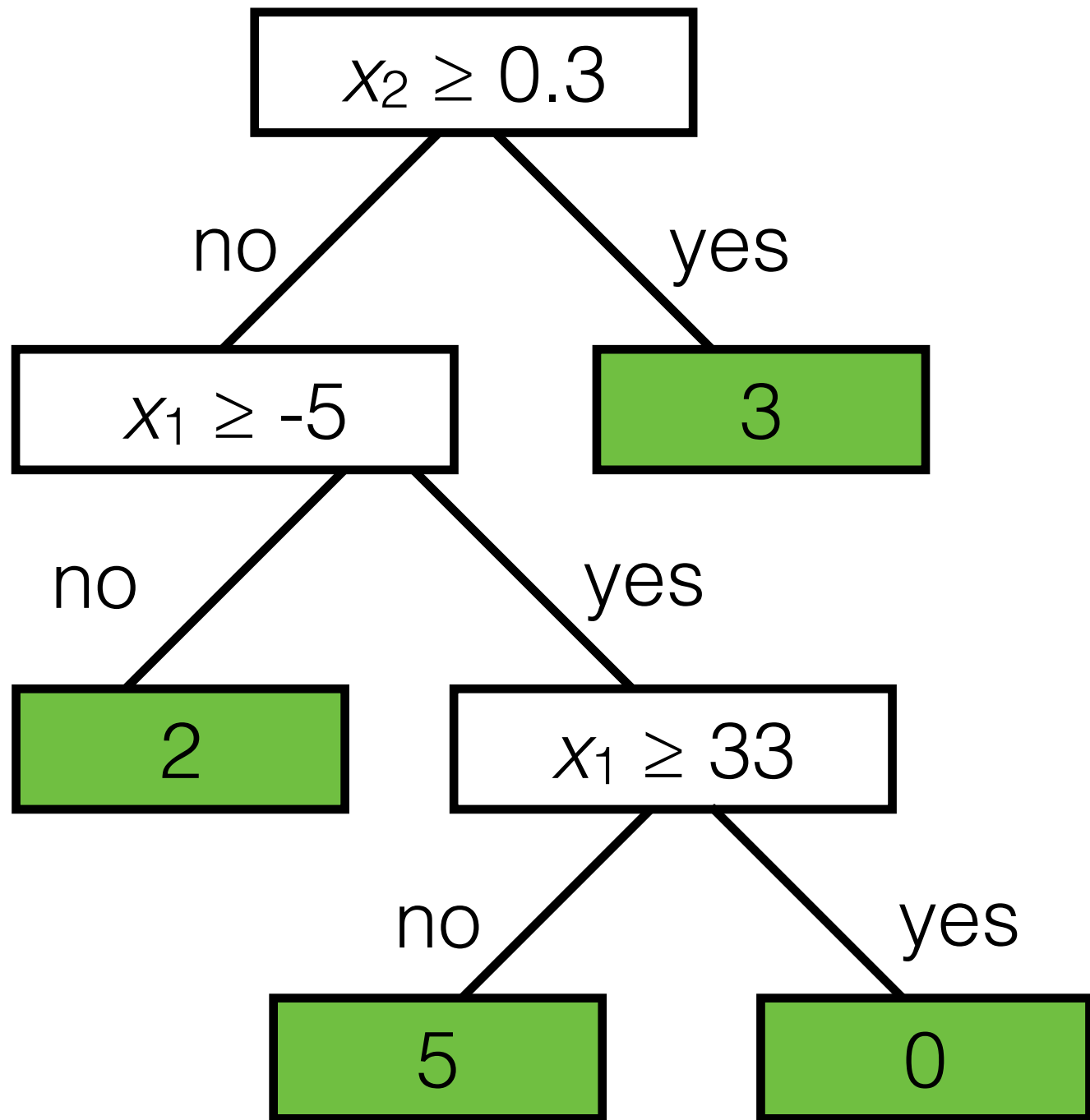


Decision tree

Recall: familiar pattern

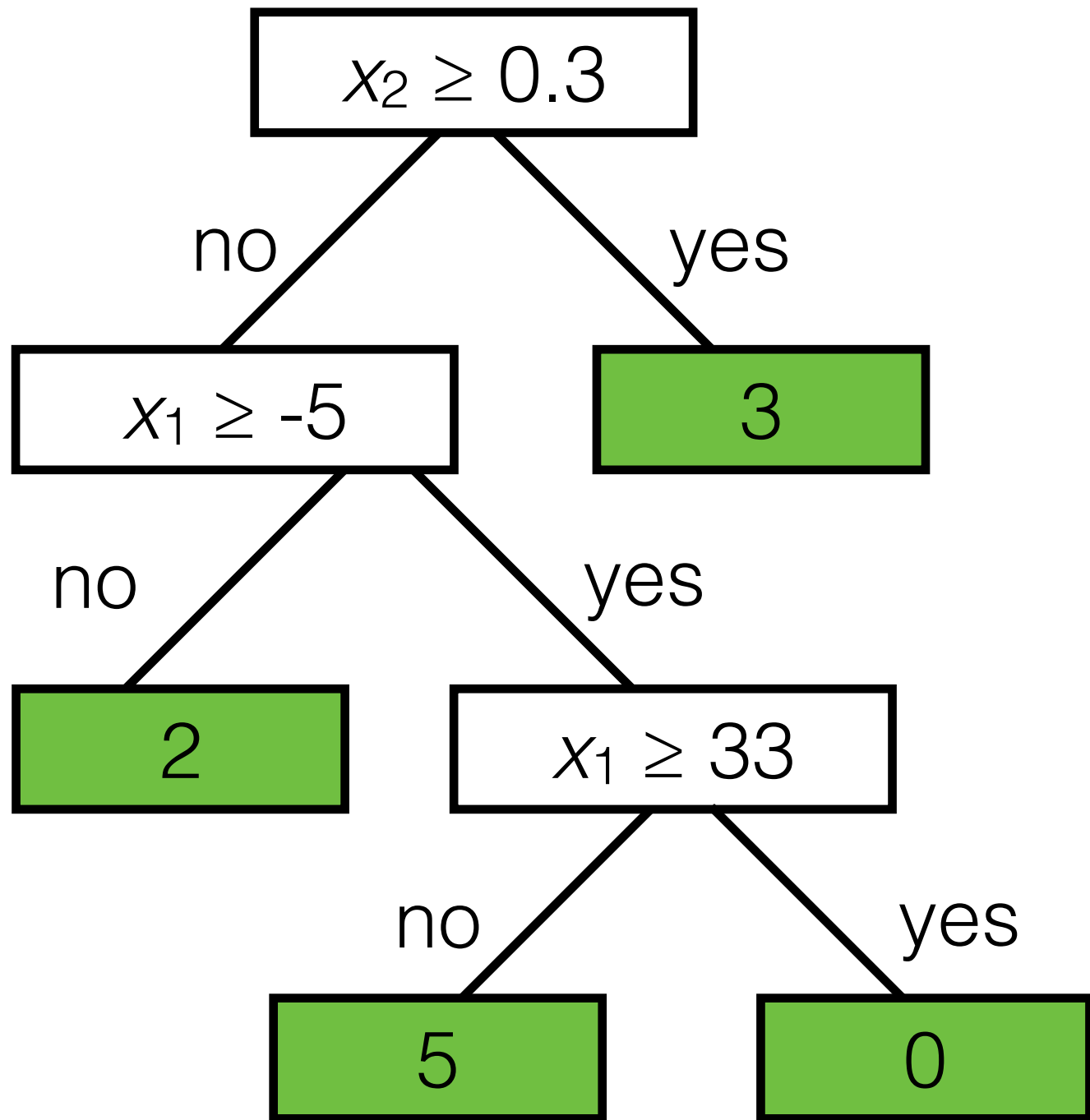


Decision tree



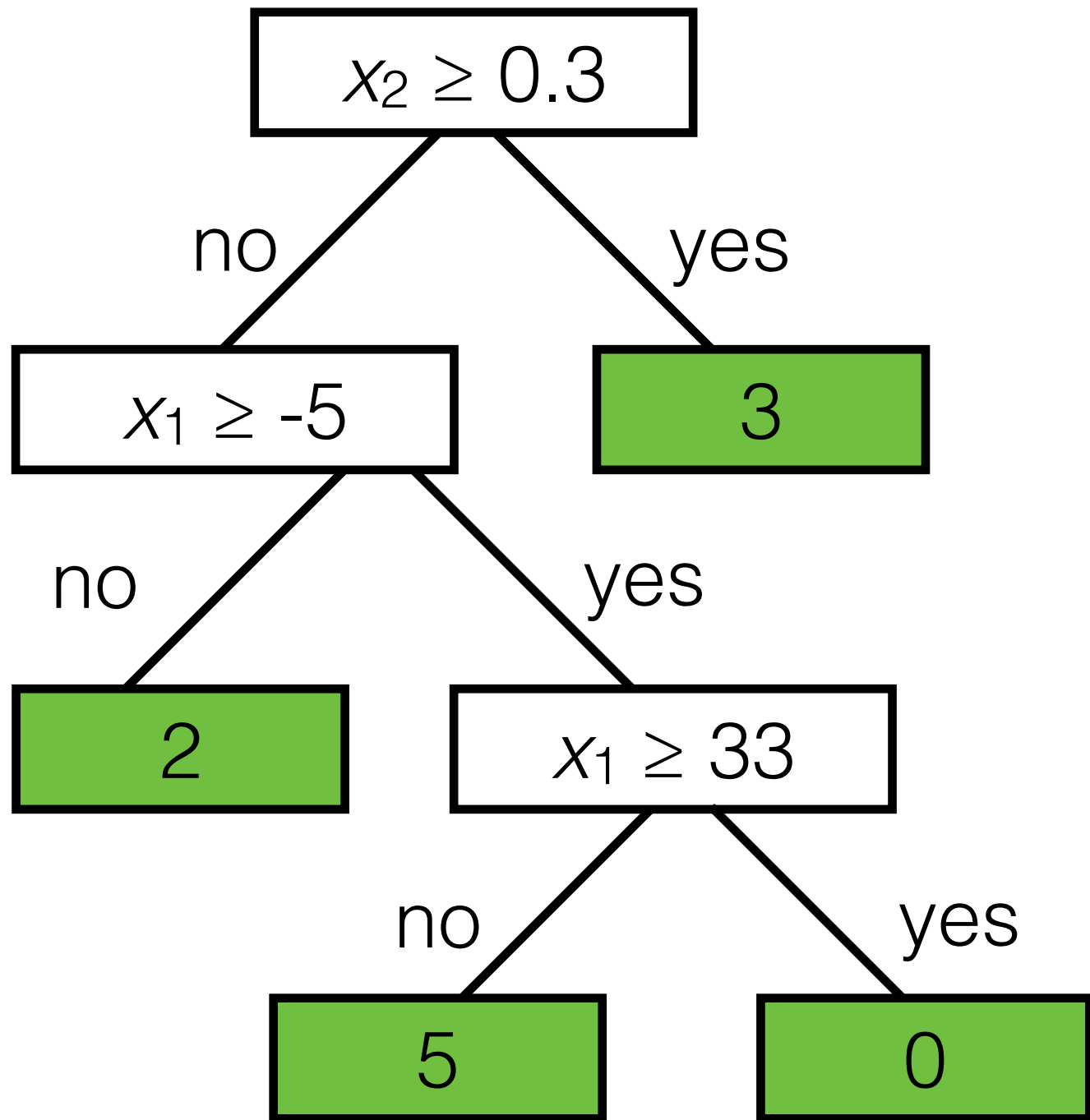
Recall: familiar pattern
1. Choose how to predict label (given features & parameters)

Decision tree



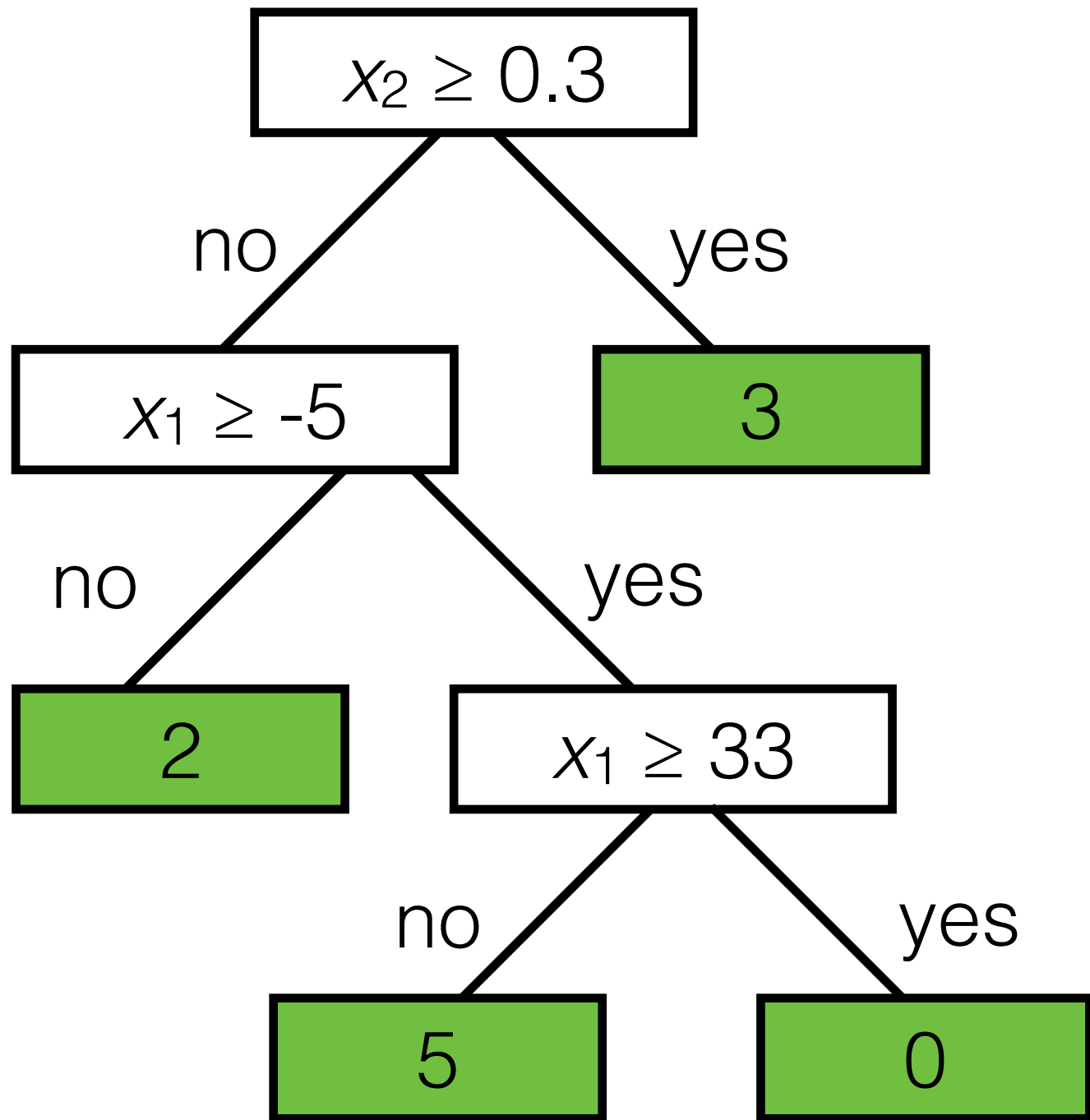
- Recall: familiar pattern
1. Choose how to predict label (given features & parameters)
 2. Choose a loss (between guess & actual label)

Decision tree



- Recall: familiar pattern
1. Choose how to predict label (given features & parameters)
 2. Choose a loss (between guess & actual label)
 3. Choose parameters by trying to minimize the training loss

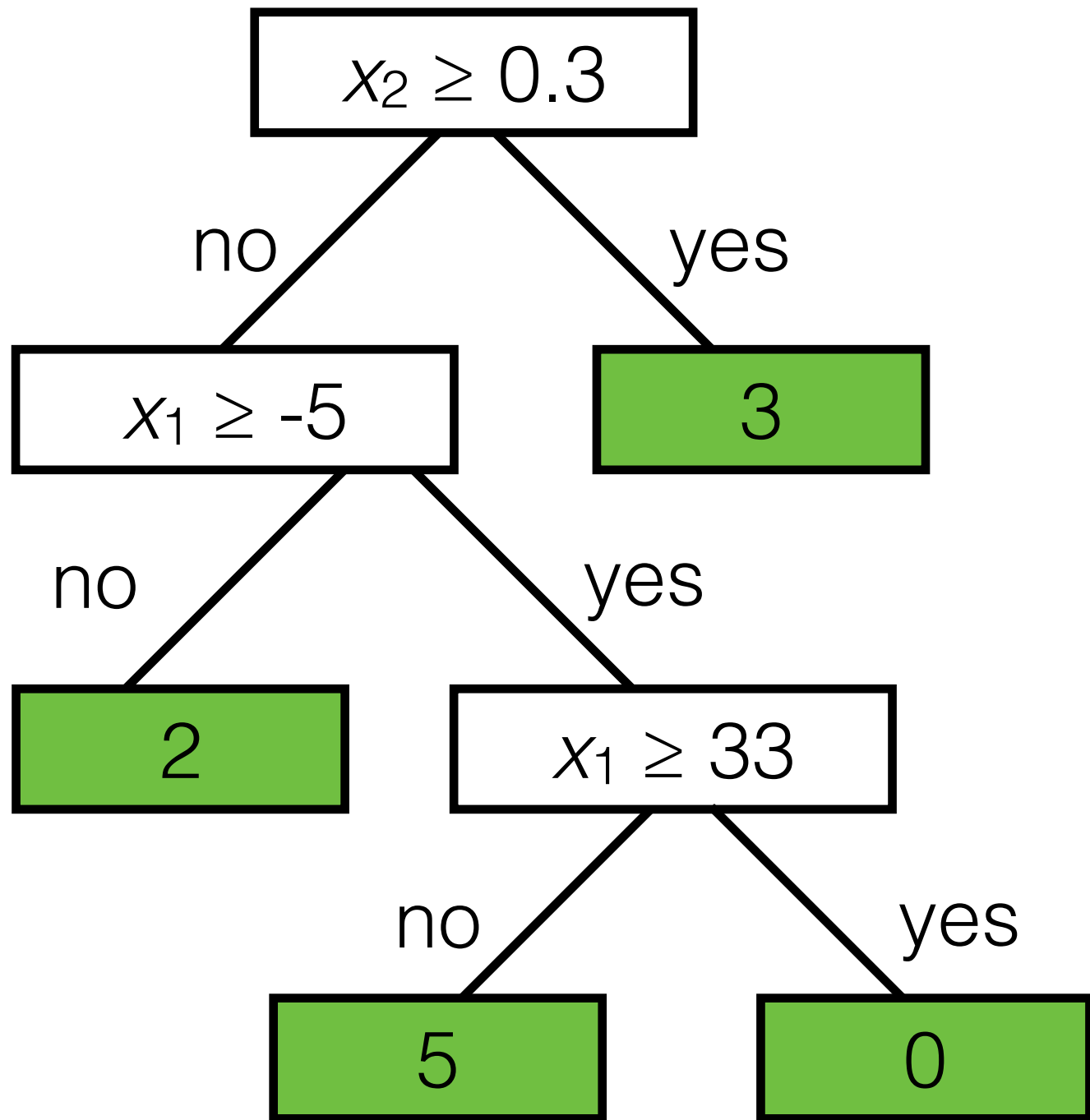
Decision tree



- Recall: familiar pattern
1. Choose how to predict label (given features & parameters)
 2. Choose a loss (between guess & actual label)
 3. Choose parameters by trying to minimize the training loss

- Parameters here:

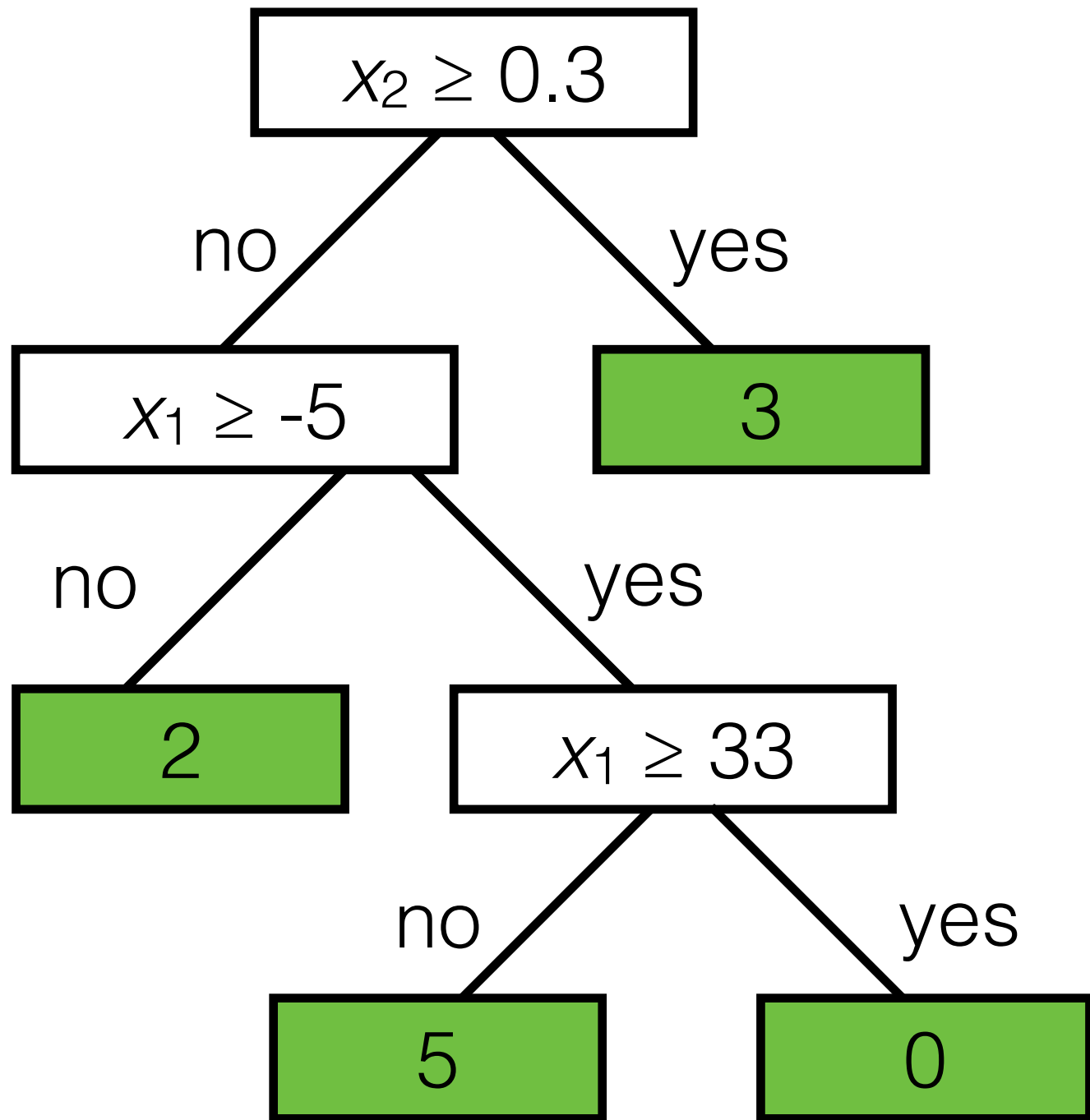
Decision tree



- Recall: familiar pattern
1. Choose how to predict label (given features & parameters)
 2. Choose a loss (between guess & actual label)
 3. Choose parameters by trying to minimize the training loss

- Parameters here:
 - For each internal node:

Decision tree

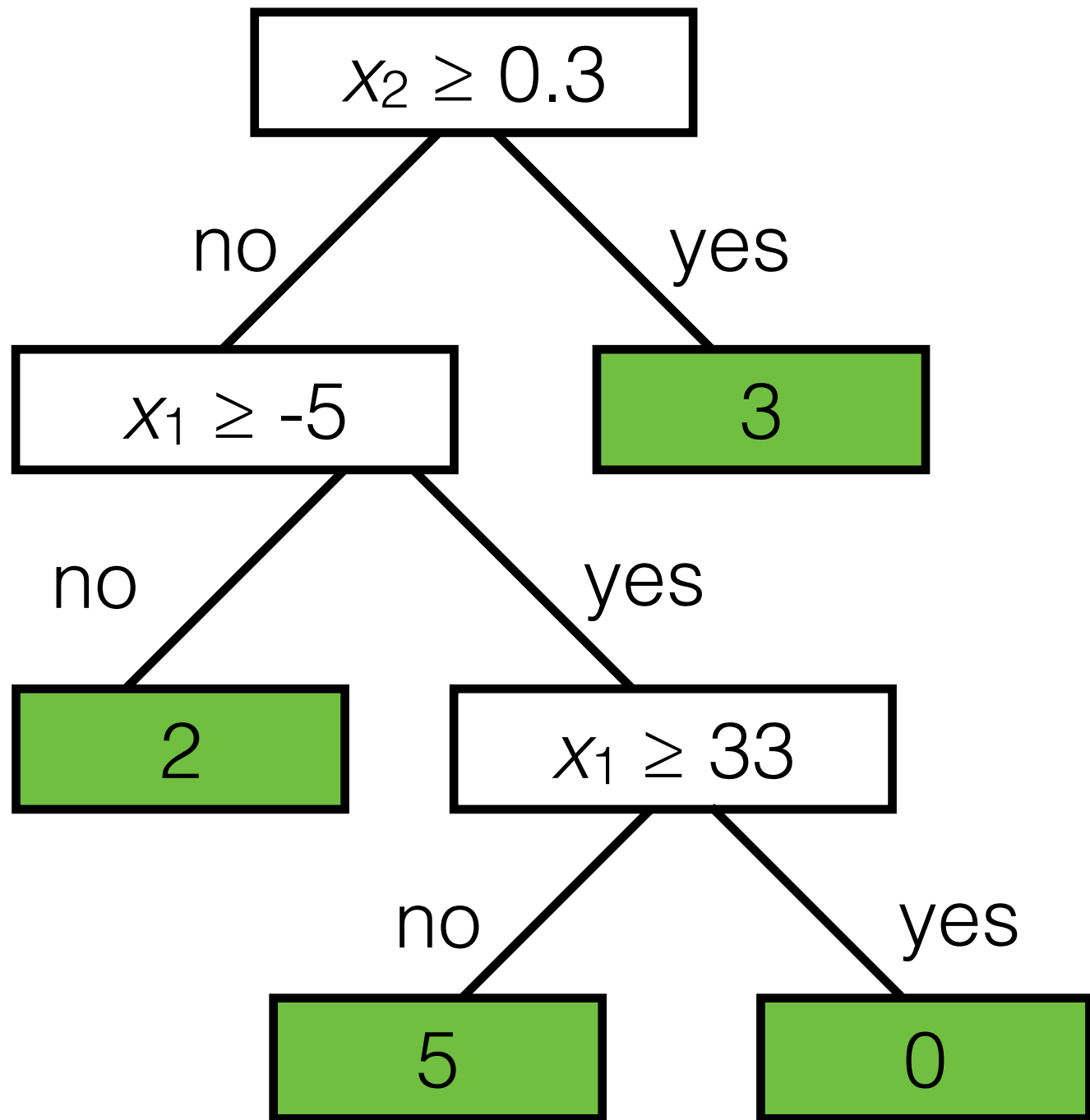


Recall: familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guess & actual label)
3. Choose parameters by trying to minimize the training loss

- Parameters here:
 - For each internal node:
 - split dimension

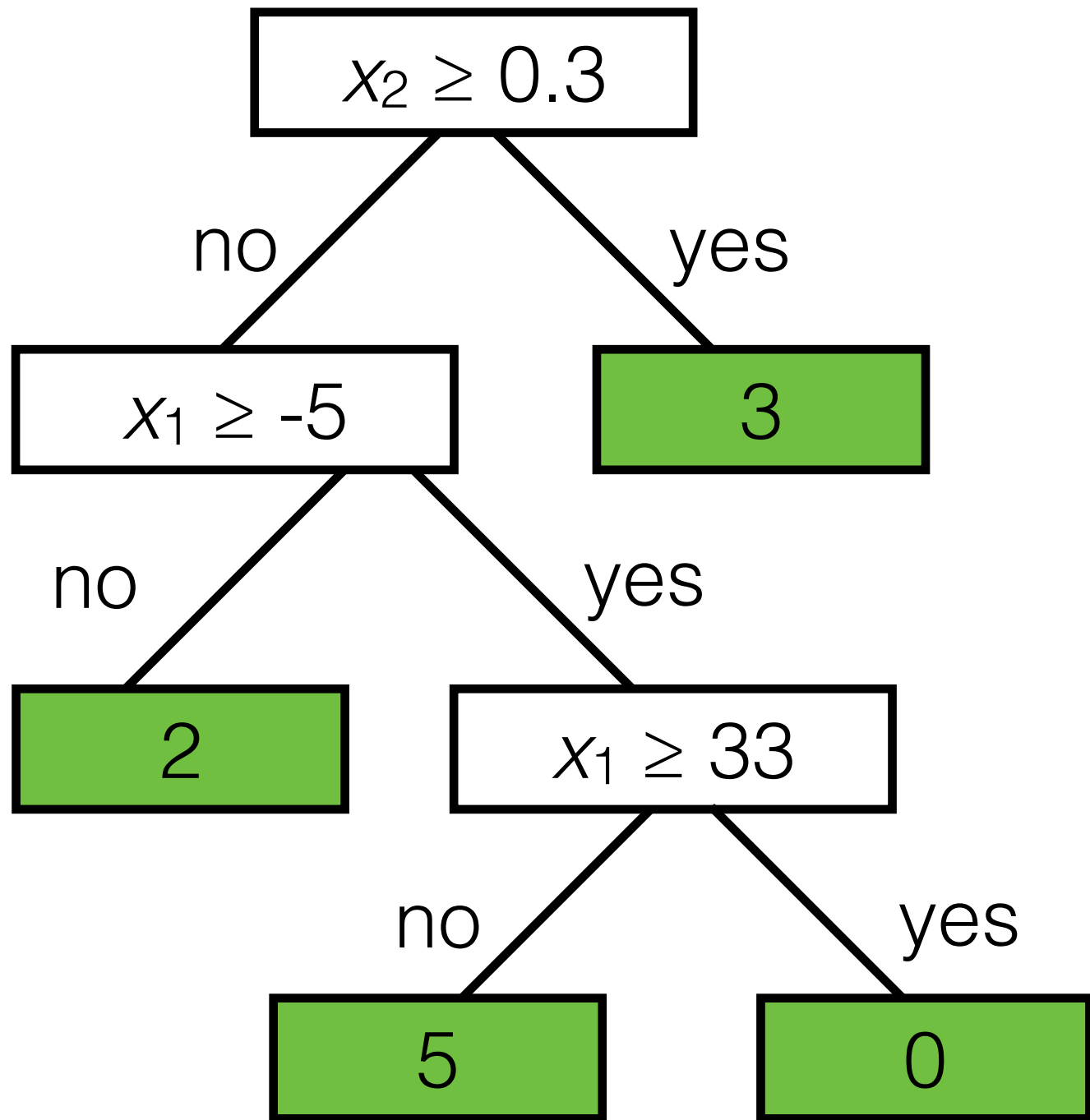
Decision tree



- Recall: familiar pattern
1. Choose how to predict label (given features & parameters)
 2. Choose a loss (between guess & actual label)
 3. Choose parameters by trying to minimize the training loss

- Parameters here:
 - For each internal node:
 - split dimension
 - split value

Decision tree

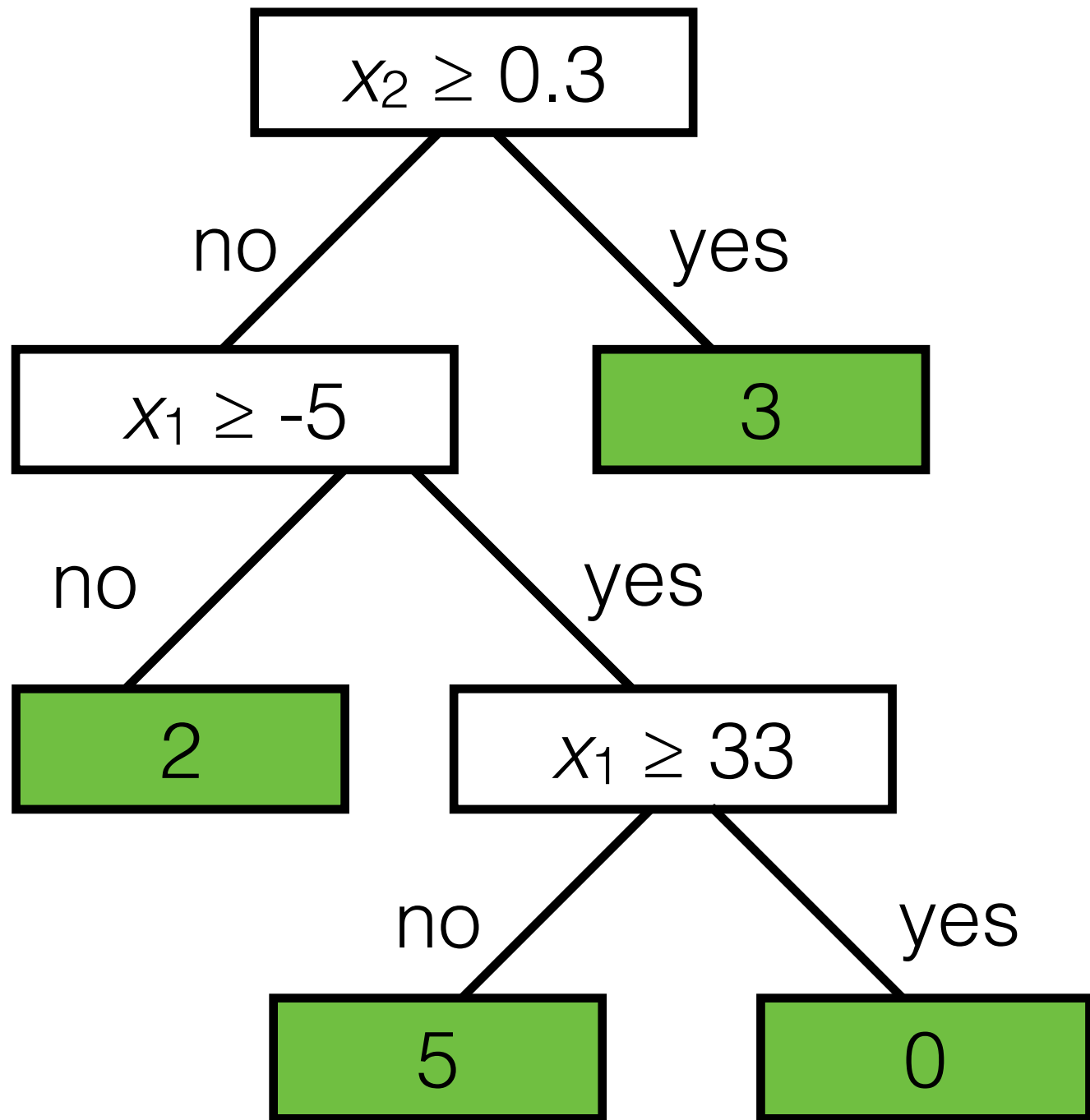


Recall: familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guess & actual label)
3. Choose parameters by trying to minimize the training loss

- Parameters here:
 - For each internal node:
 - split dimension
 - split value
 - child nodes

Decision tree

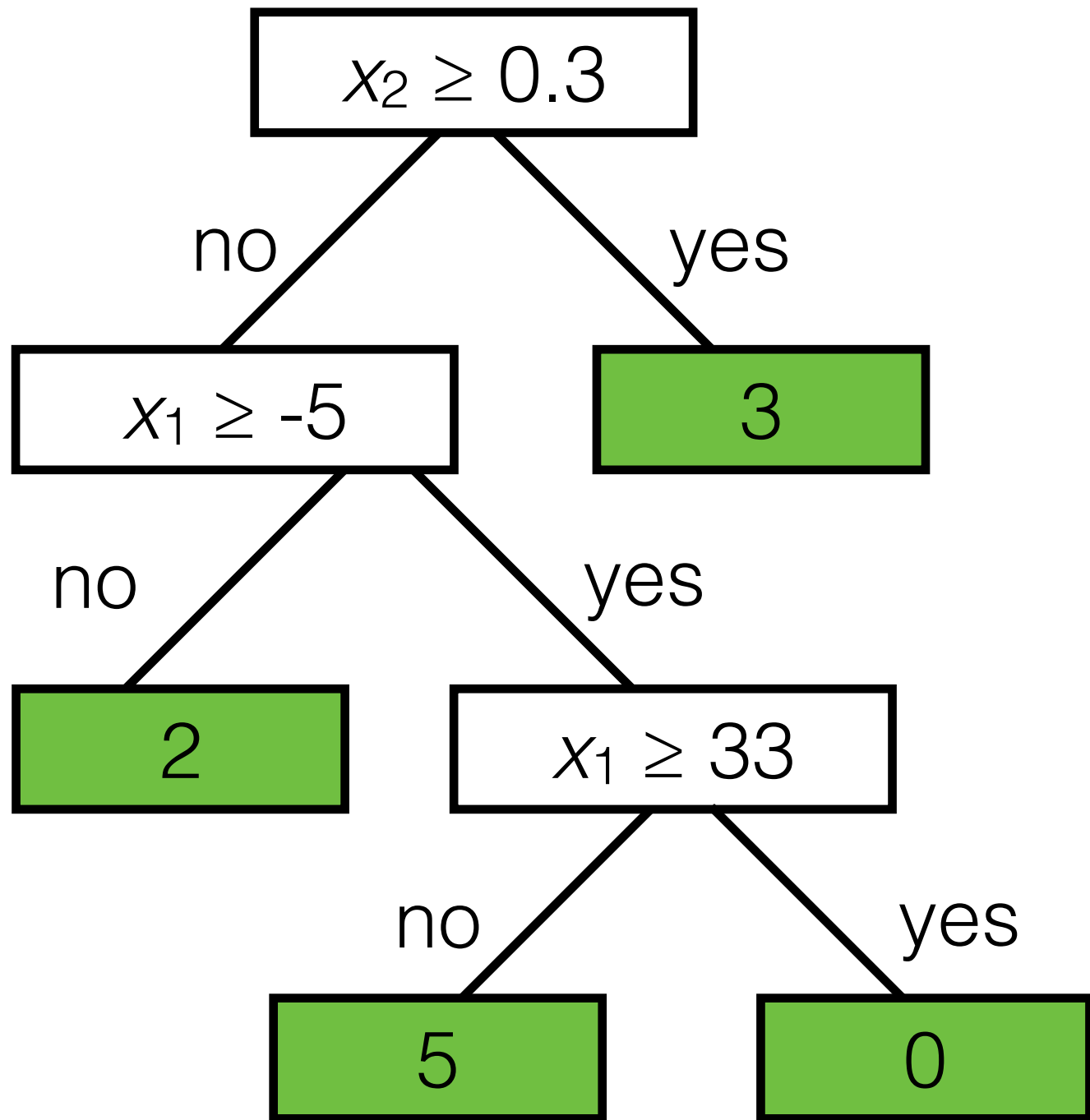


Recall: familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guess & actual label)
3. Choose parameters by trying to minimize the training loss

- Parameters here:
 - For each internal node:
 - split dimension
 - split value
 - child nodes
 - For each leaf node:
 - label

Decision tree

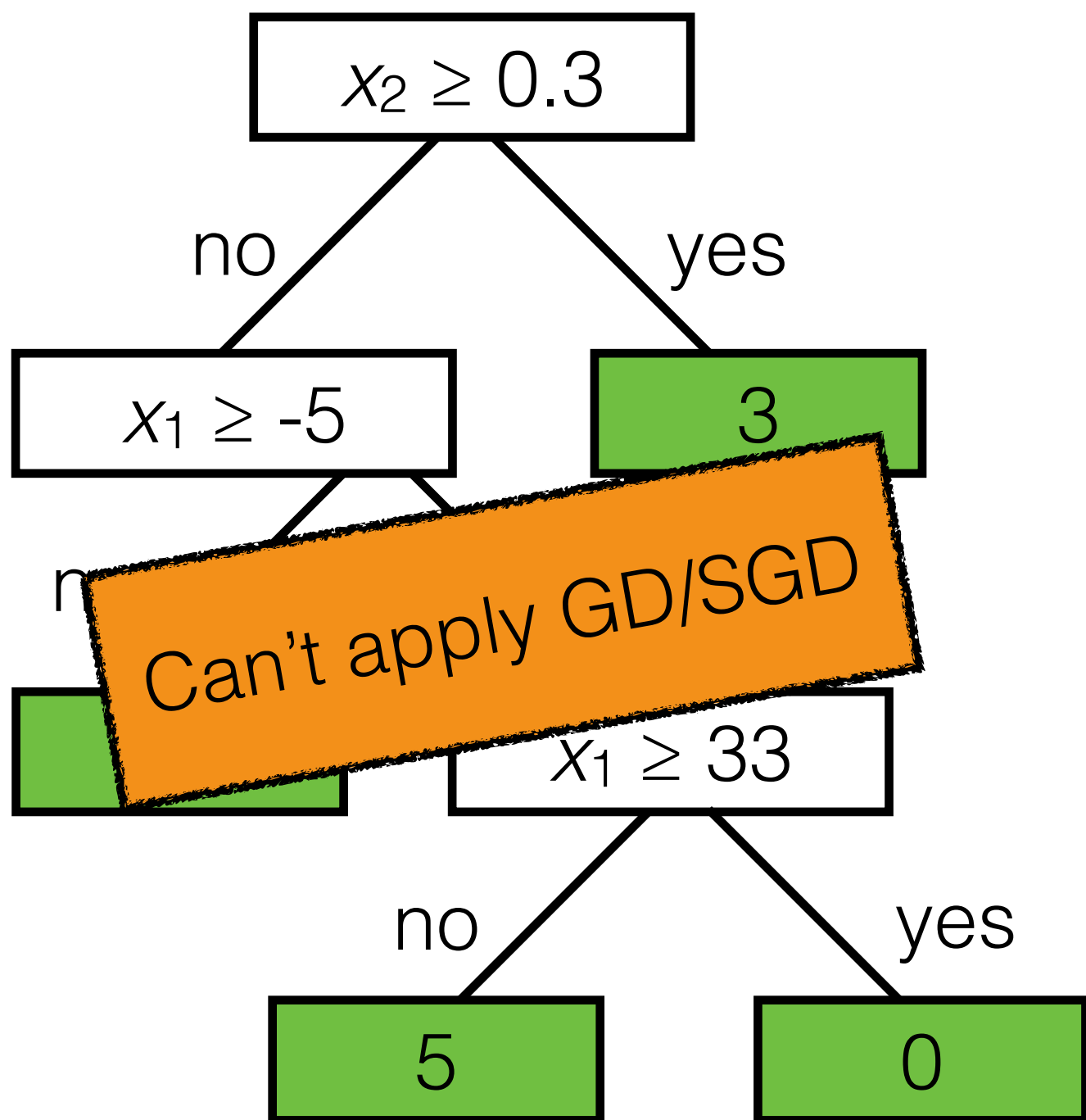


- Note: parameters here don't have a fixed dimension

- Recall: familiar pattern
1. Choose how to predict label (given features & parameters)
 2. Choose a loss (between guess & actual label)
 3. Choose parameters by trying to minimize the training loss

- Parameters here:
 - For each internal node:
 - split dimension
 - split value
 - child nodes
 - For each leaf node:
 - label

Decision tree

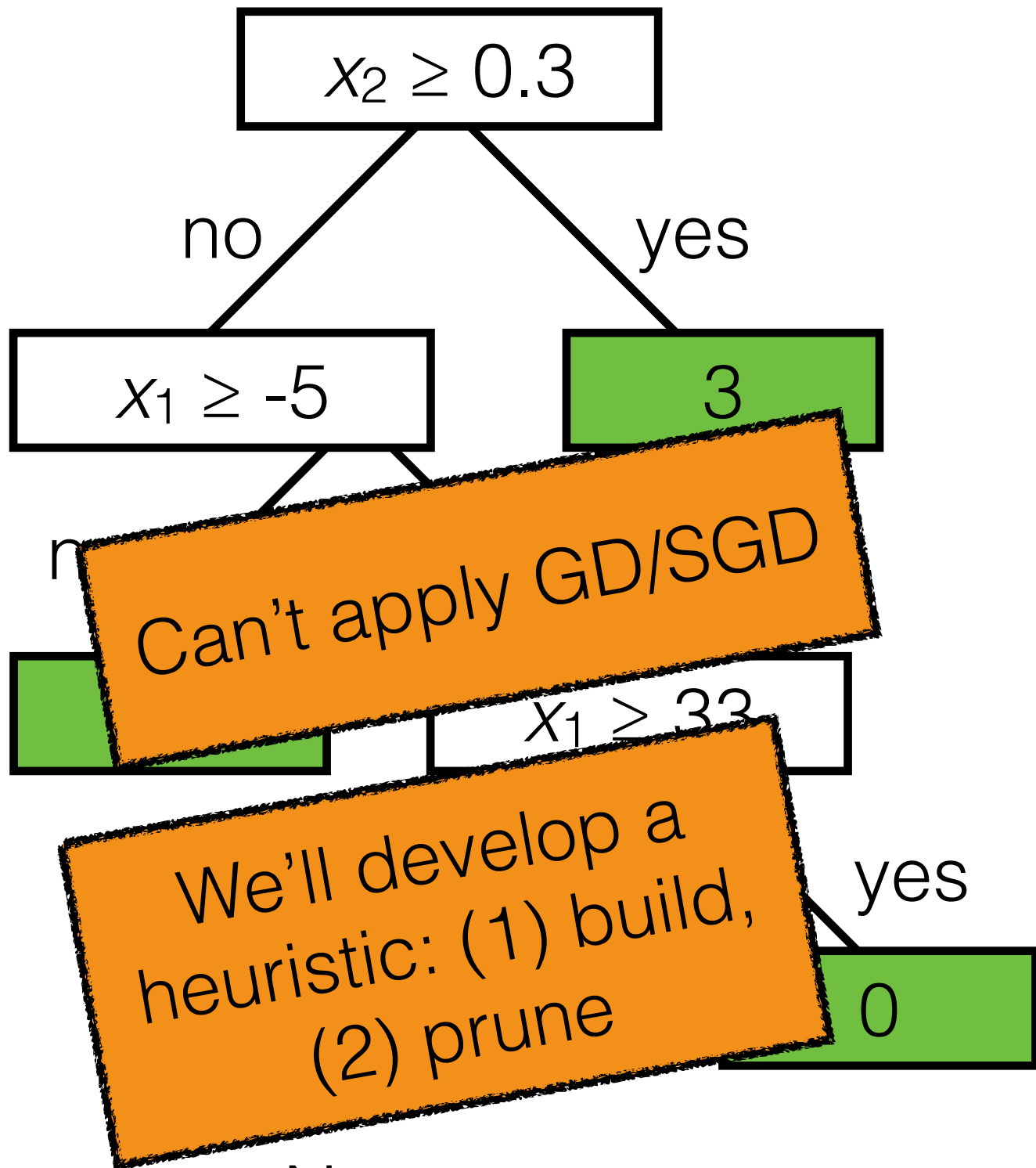


- Note: parameters here don't have a fixed dimension

- Recall: familiar pattern
1. Choose how to predict label (given features & parameters)
 2. Choose a loss (between guess & actual label)
 3. Choose parameters by trying to minimize the training loss

- Parameters here:
 - For each internal node:
 - split dimension
 - split value
 - child nodes
 - For each leaf node:
 - label

Decision tree



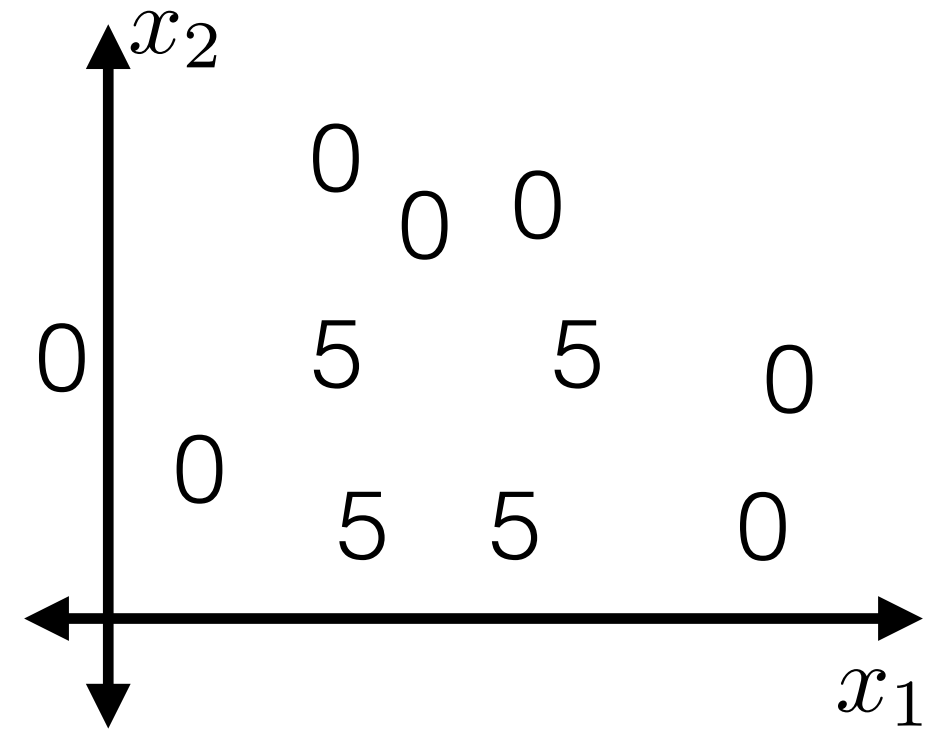
- Note: parameters here don't have a fixed dimension

- Recall: familiar pattern
1. Choose how to predict label (given features & parameters)
 2. Choose a loss (between guess & actual label)
 3. Choose parameters by trying to minimize the training loss

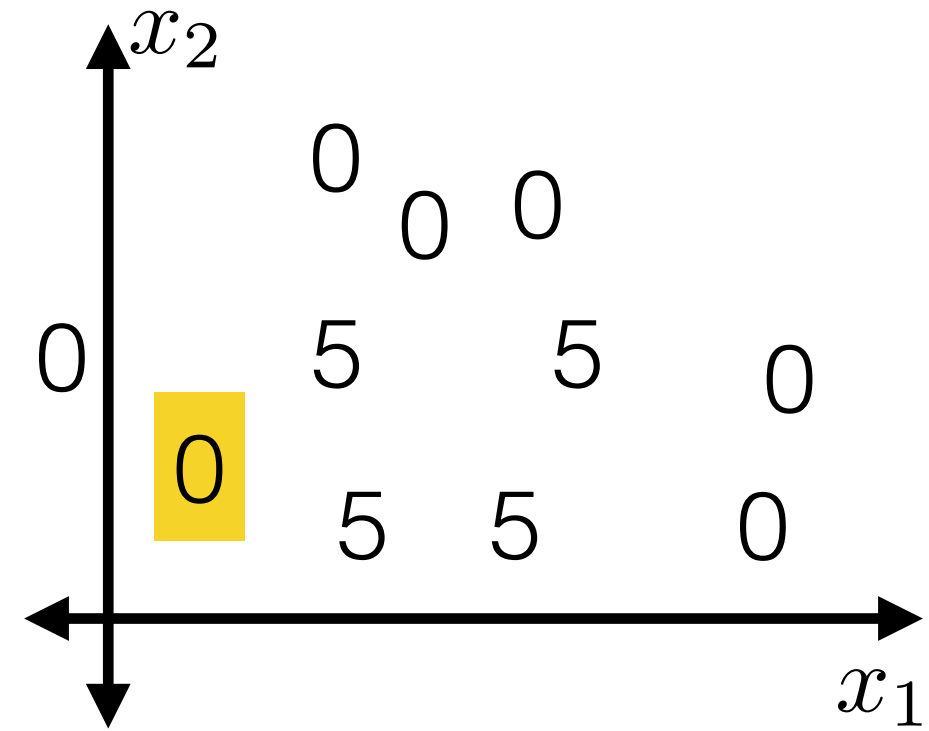
- Parameters here:
 - For each internal node:
 - split dimension
 - split value
 - child nodes
 - For each leaf node:
 - label

Building a decision tree

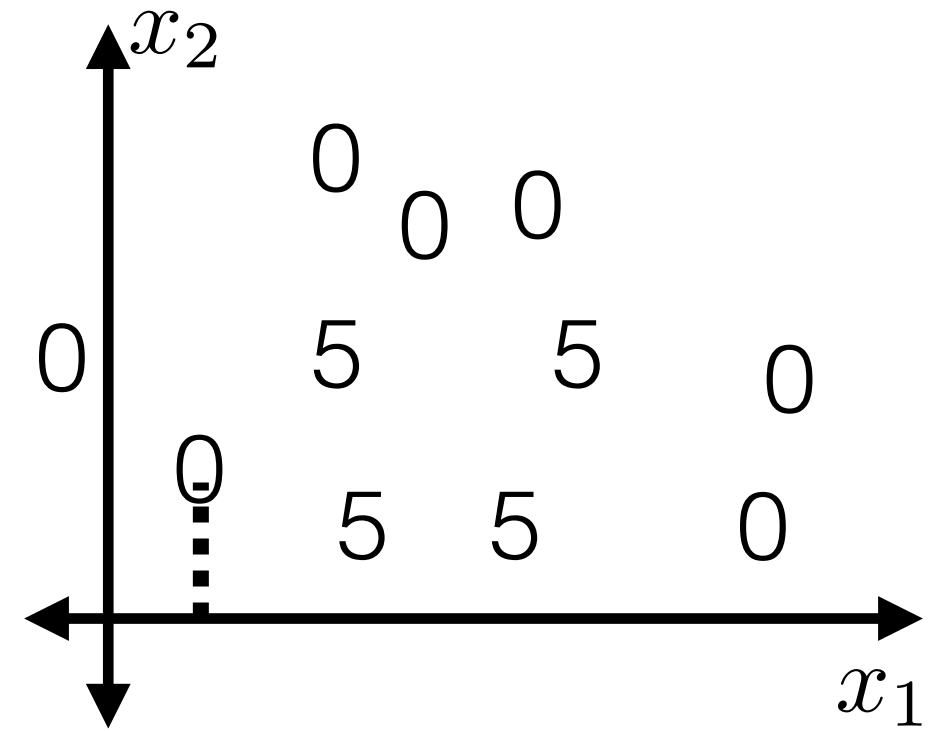
Building a decision tree



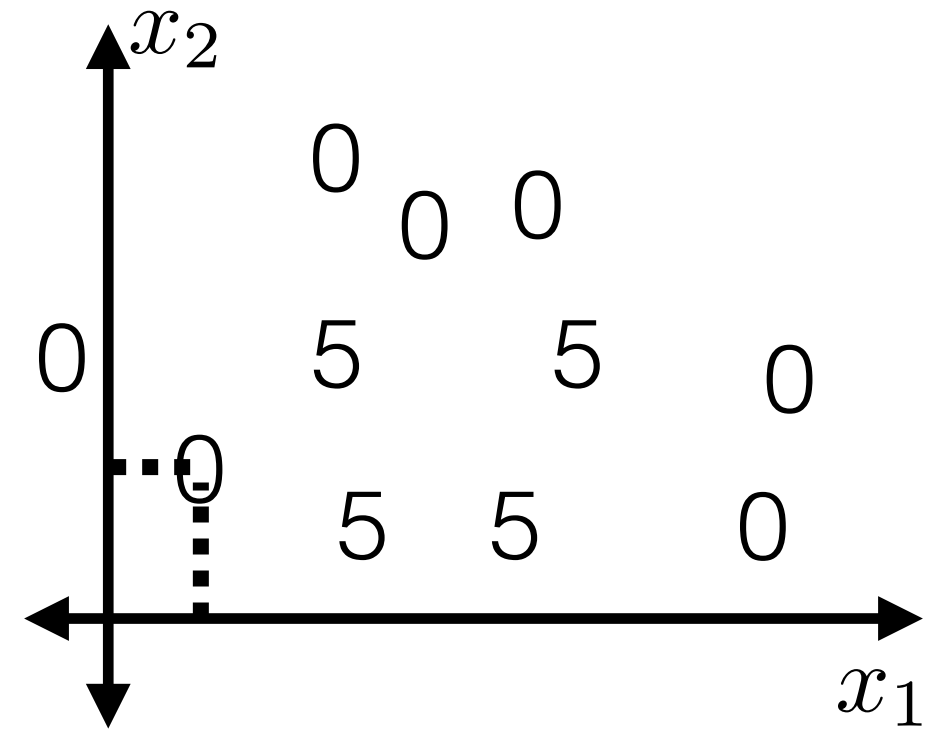
Building a decision tree



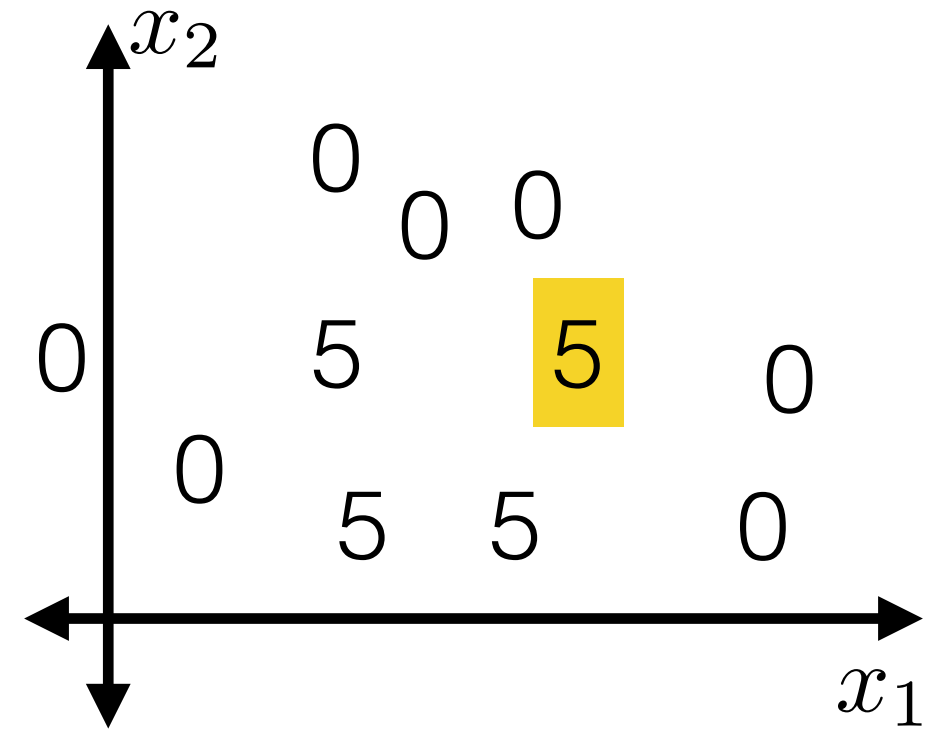
Building a decision tree



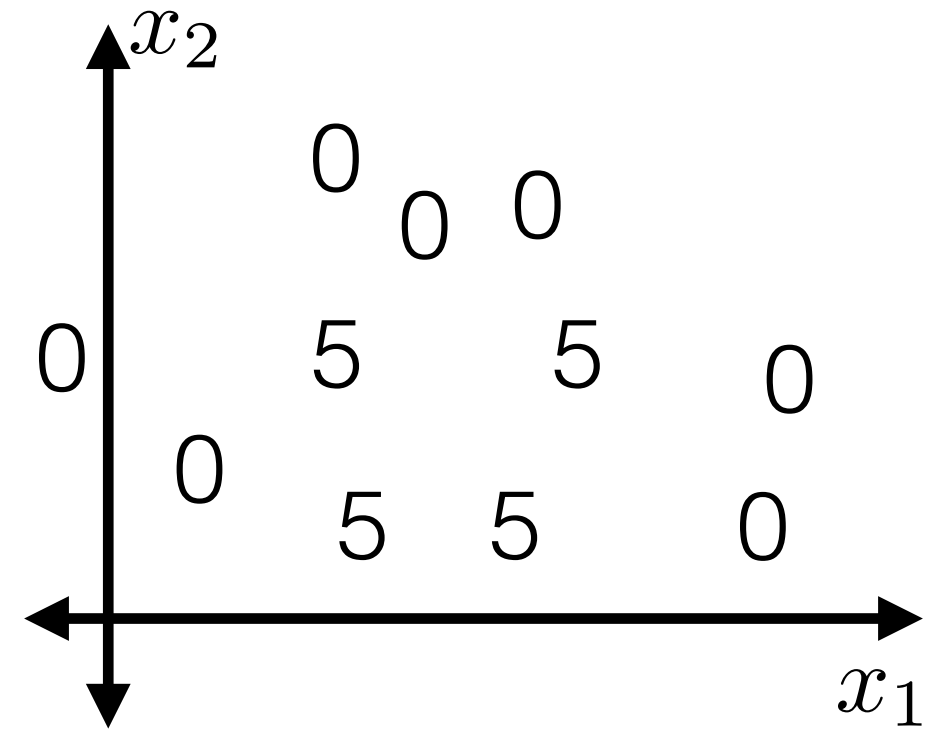
Building a decision tree



Building a decision tree

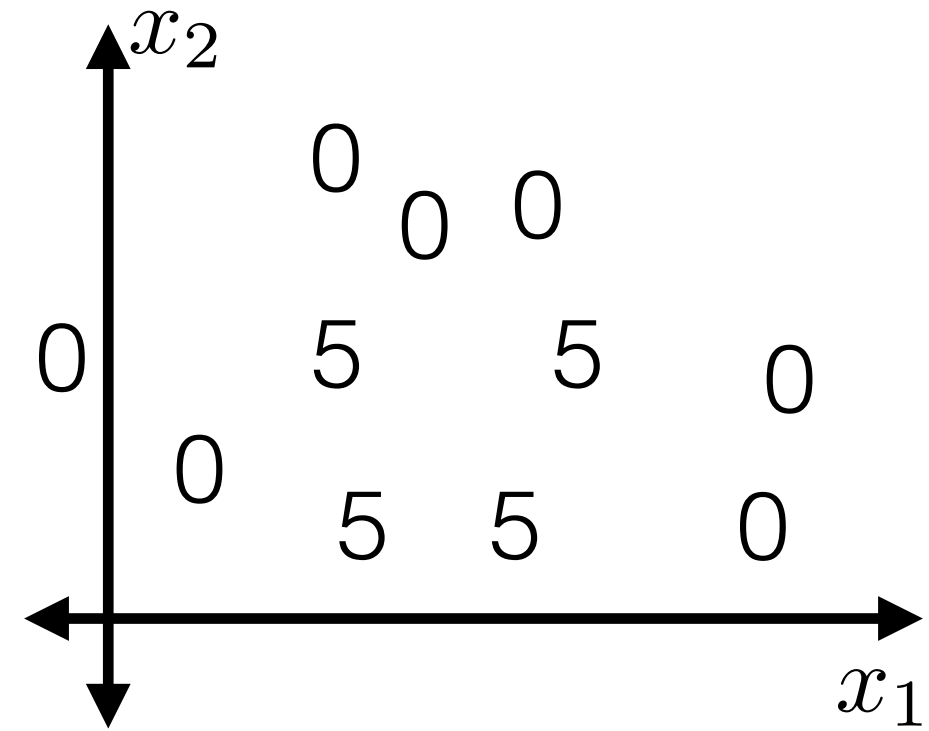


Building a decision tree



Building a decision tree

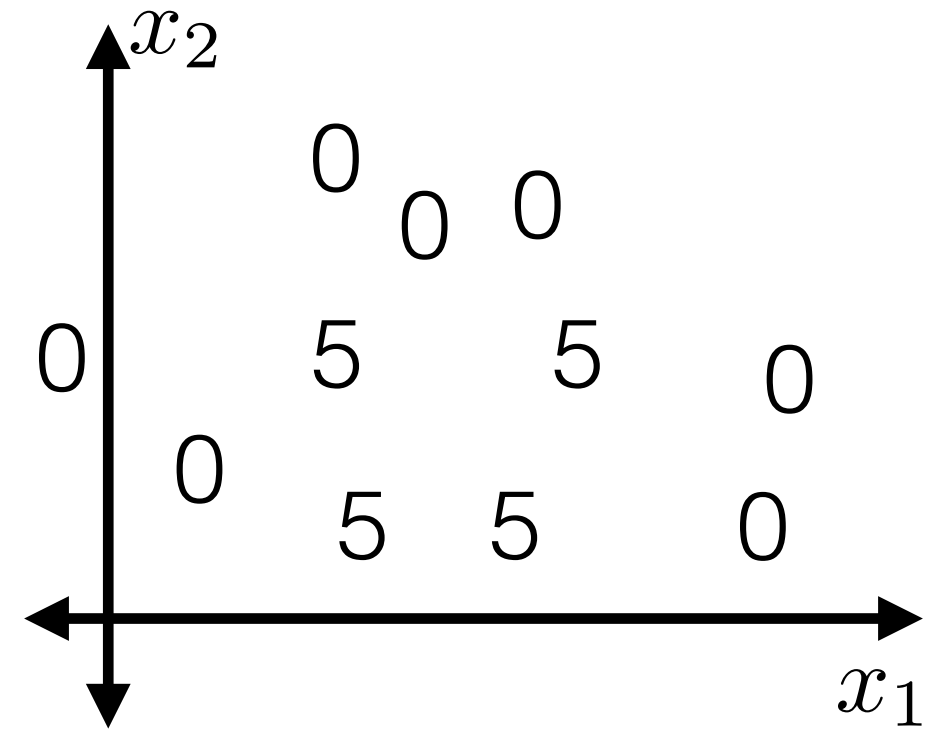
- Regression tree with squared error loss



Building a decision tree

- Regression tree with squared error loss

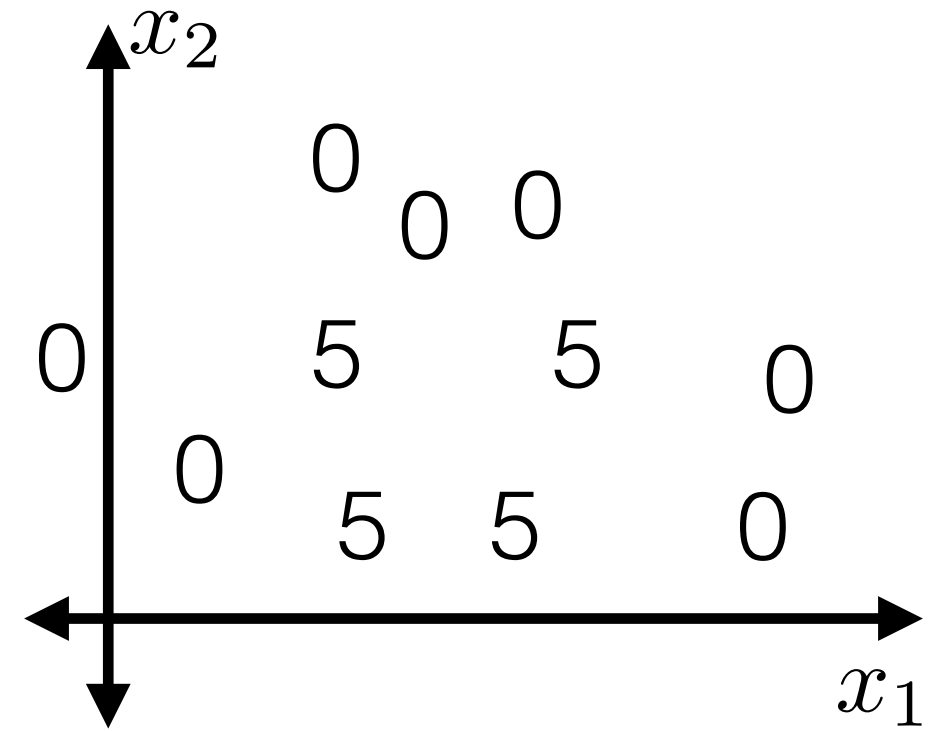
BuildTree



Building a decision tree

- Regression tree with squared error loss

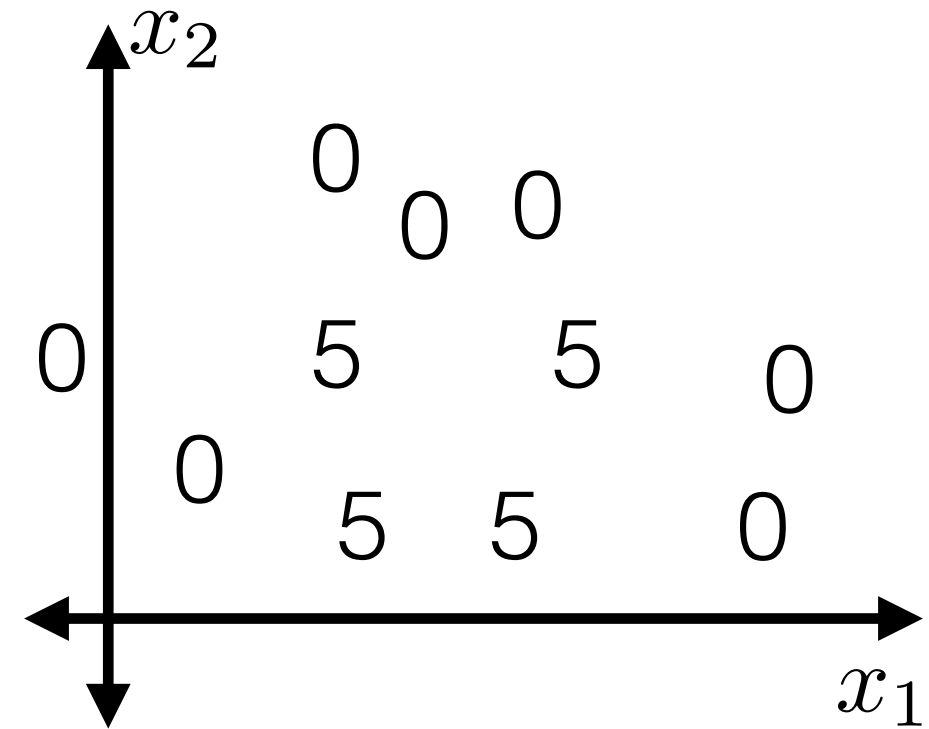
`BuildTree($I; k$)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

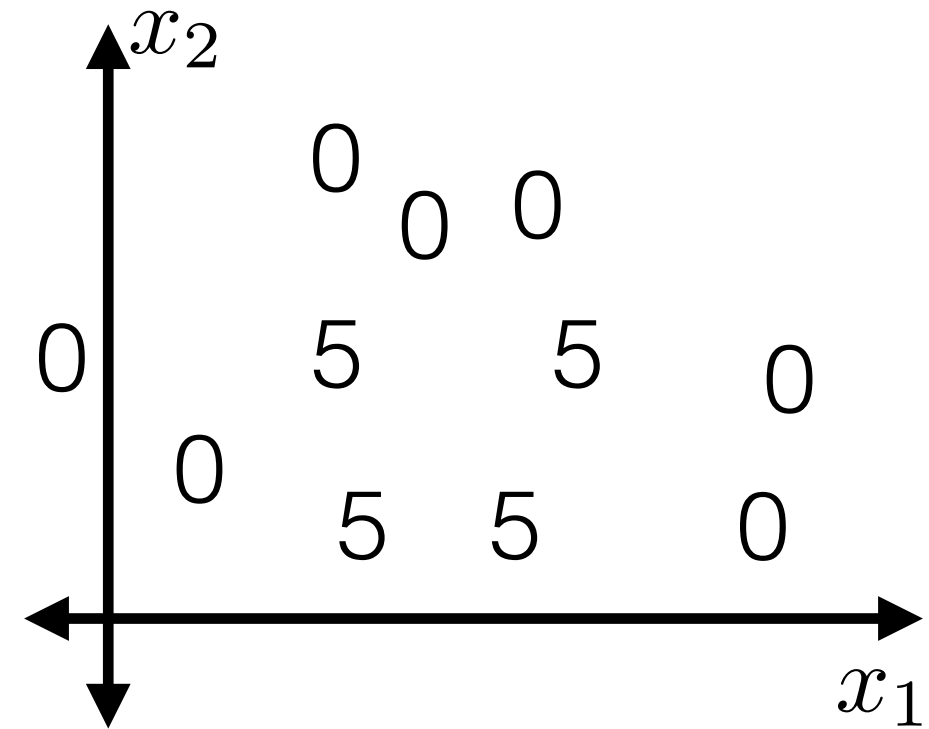


`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

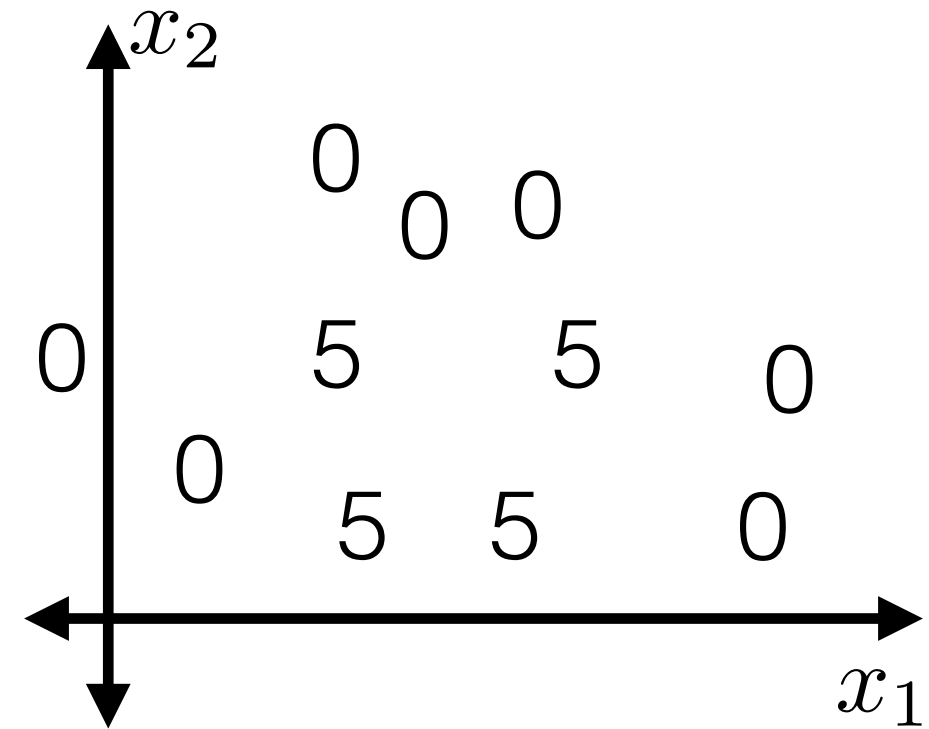


`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

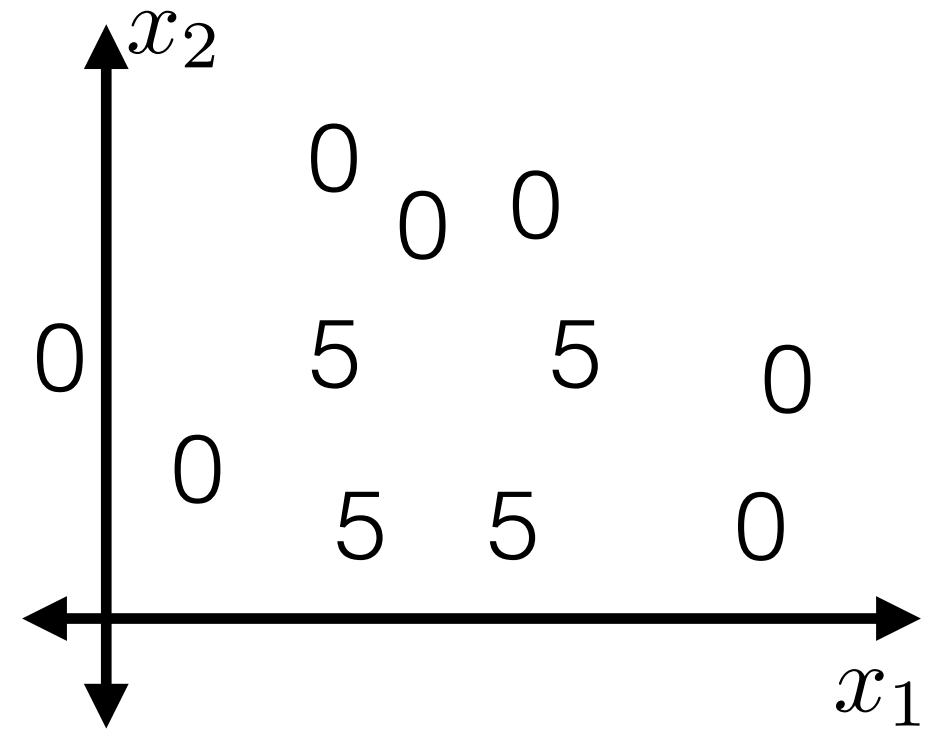


`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`



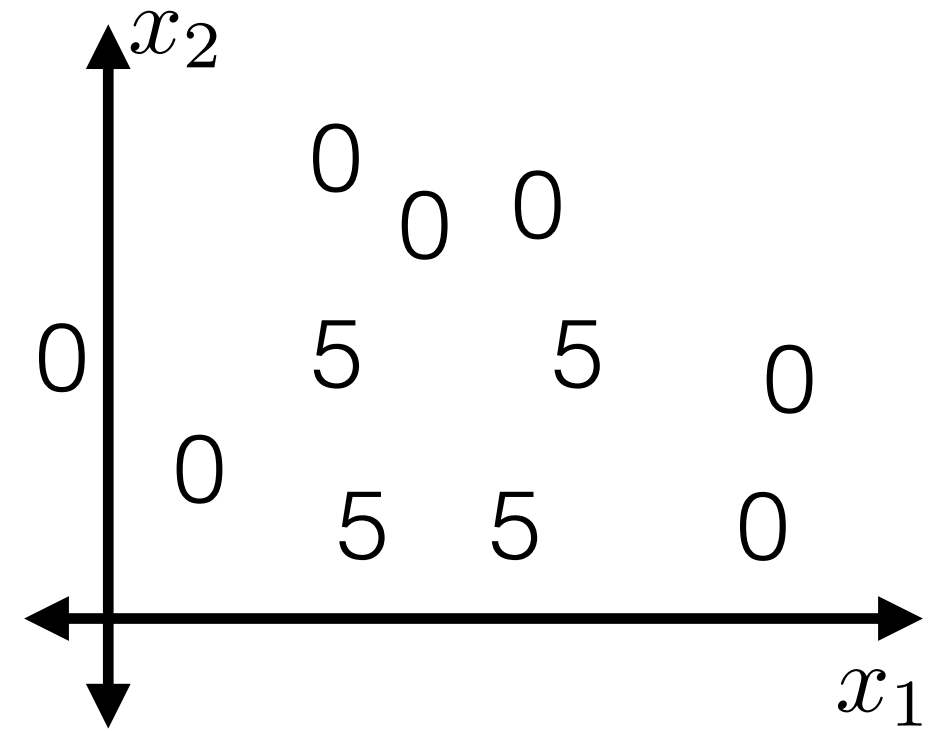
`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

`BuildTree($I; k$)`

if $|I| \leq k$



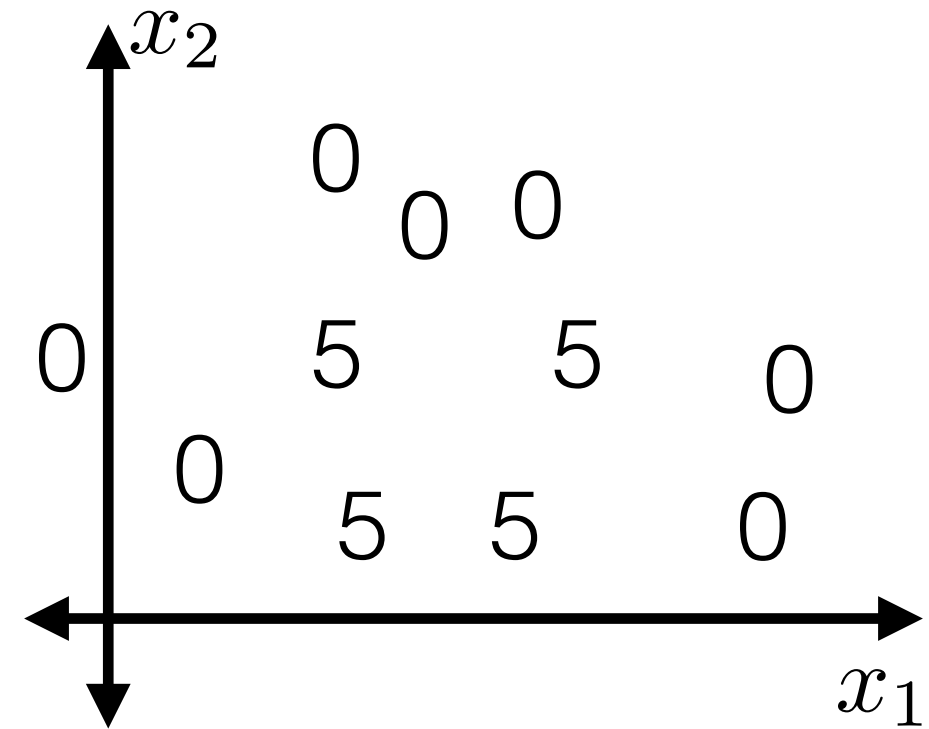
`BuildTree($\{1, \dots, n\}; 2$)`

Building a decision tree

- Regression tree with squared error loss

`BuildTree($I; k$)`

if `| I |` \leq `k`



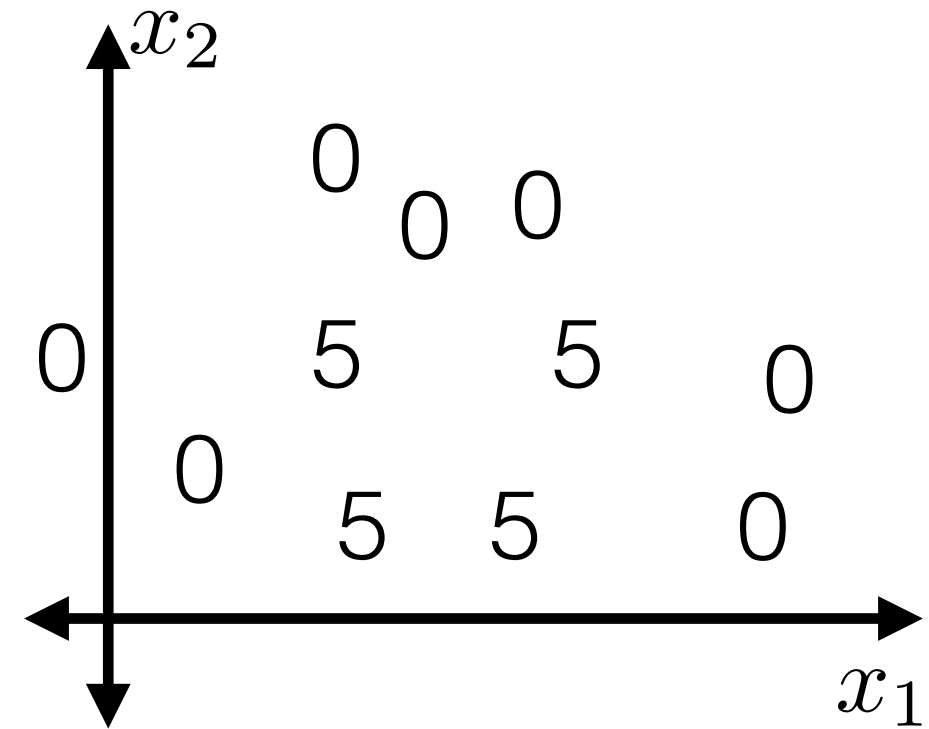
`BuildTree($\{1, \dots, n\}; 2$)`

Building a decision tree

- Regression tree with squared error loss

```
BuildTree( $I; k$ )
```

```
  if  $|I| \leq k$ 
```



```
BuildTree( $\{1, \dots, n\}; 2$ )
```

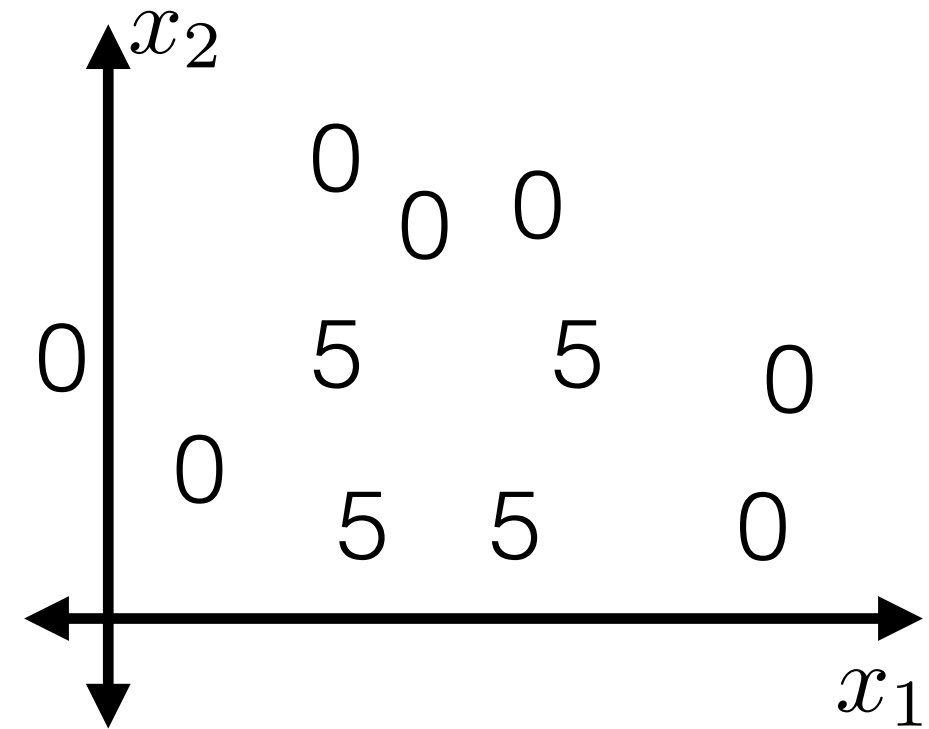
Building a decision tree

- Regression tree with squared error loss

```
BuildTree( $I; k$ )
```

```
if  $|I| \leq k$ 
```

```
  Set  $\hat{y} =$ 
```



```
BuildTree( $\{1, \dots, n\}; 2$ )
```

Building a decision tree

- Regression tree with squared error loss

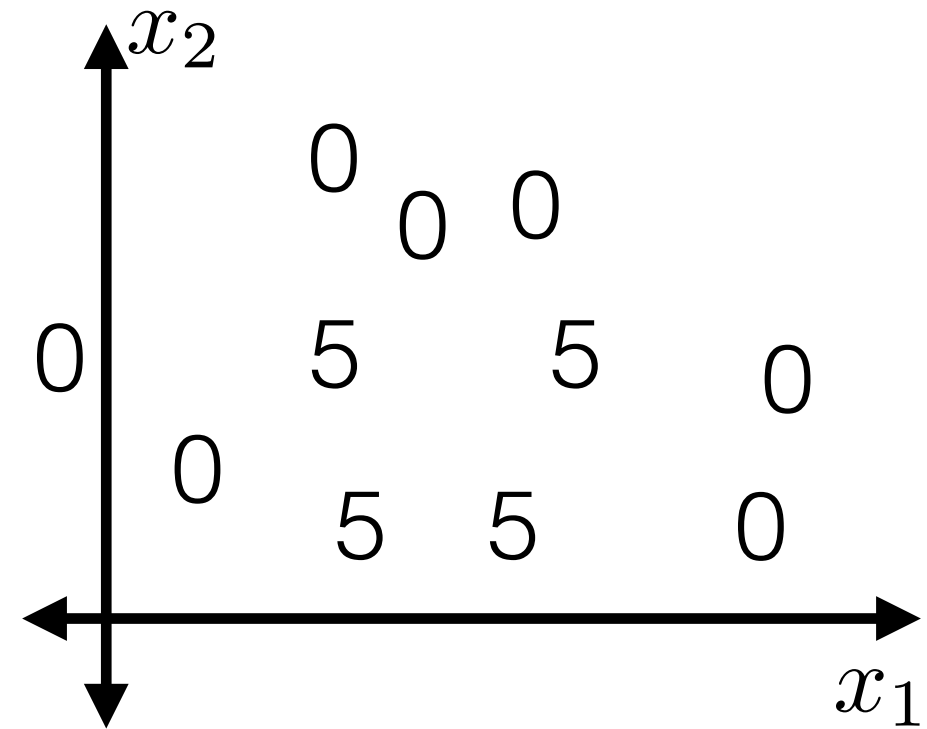
`BuildTree(I; k)`

if $|I| \leq k$

Set $\hat{y} =$

Want to minimize
loss/error:

$$E = \sum_{i \in I} (y^{(i)} - \hat{y})^2$$



`BuildTree({1, ..., n}; 2)`

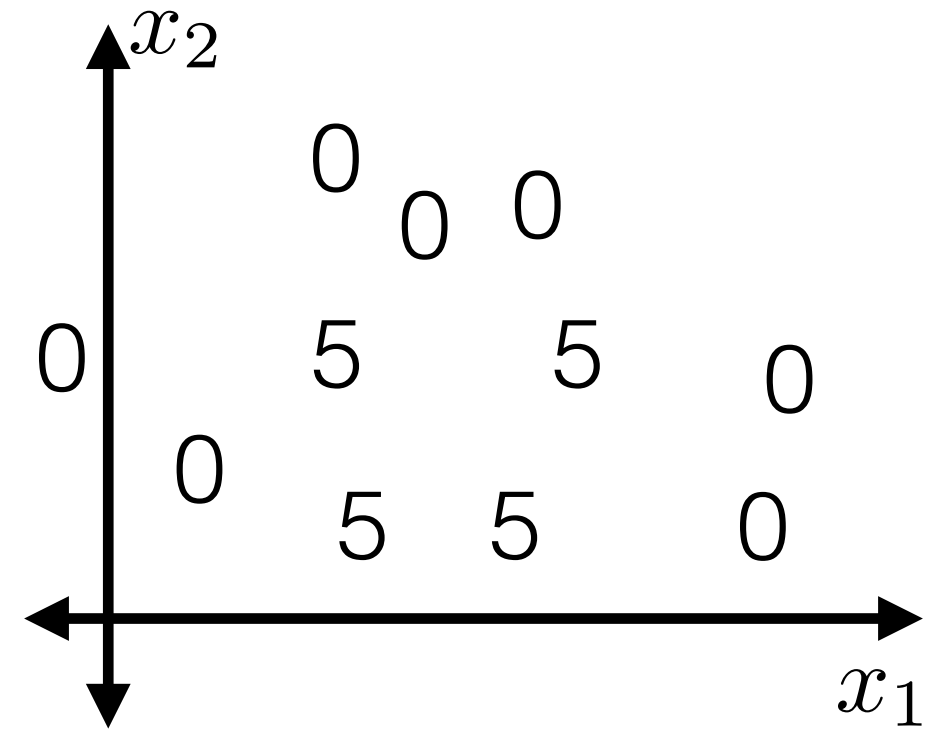
Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

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



`BuildTree({1, ..., n}; 2)`

Want to minimize
loss/error:

$$E = \sum_{i \in I} (y^{(i)} - \hat{y})^2$$

Building a decision tree

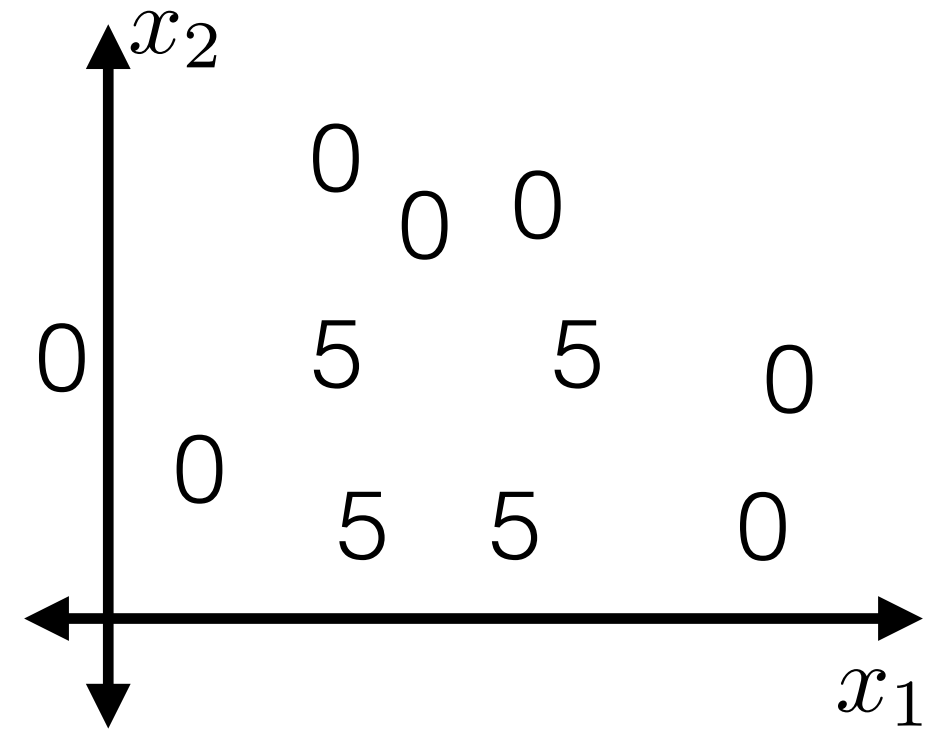
- Regression tree with squared error loss

```
BuildTree( $I; k$ )
```

```
if  $|I| \leq k$ 
```

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

```
return Leaf(label =  $\hat{y}$ )
```



```
BuildTree( $\{1, \dots, n\}; 2$ )
```

Building a decision tree

- Regression tree with squared error loss

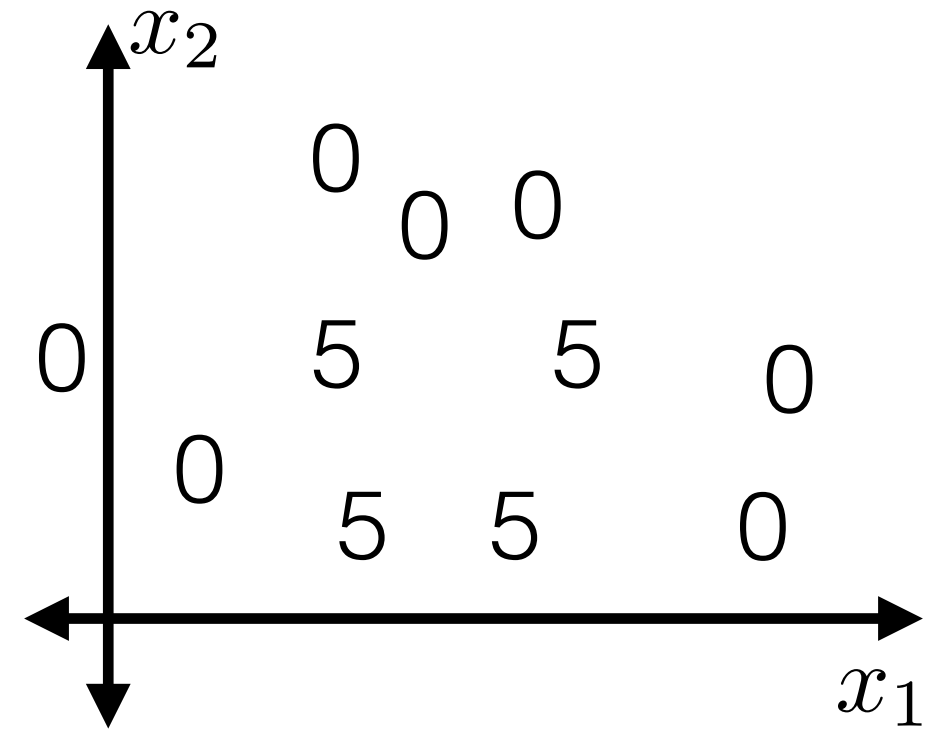
```
BuildTree( $I; k$ )
```

```
if  $|I| \leq k$ 
```

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

```
    return Leaf(label =  $\hat{y}$ )
```

```
else
```



```
BuildTree( $\{1, \dots, n\}; 2$ )
```


Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

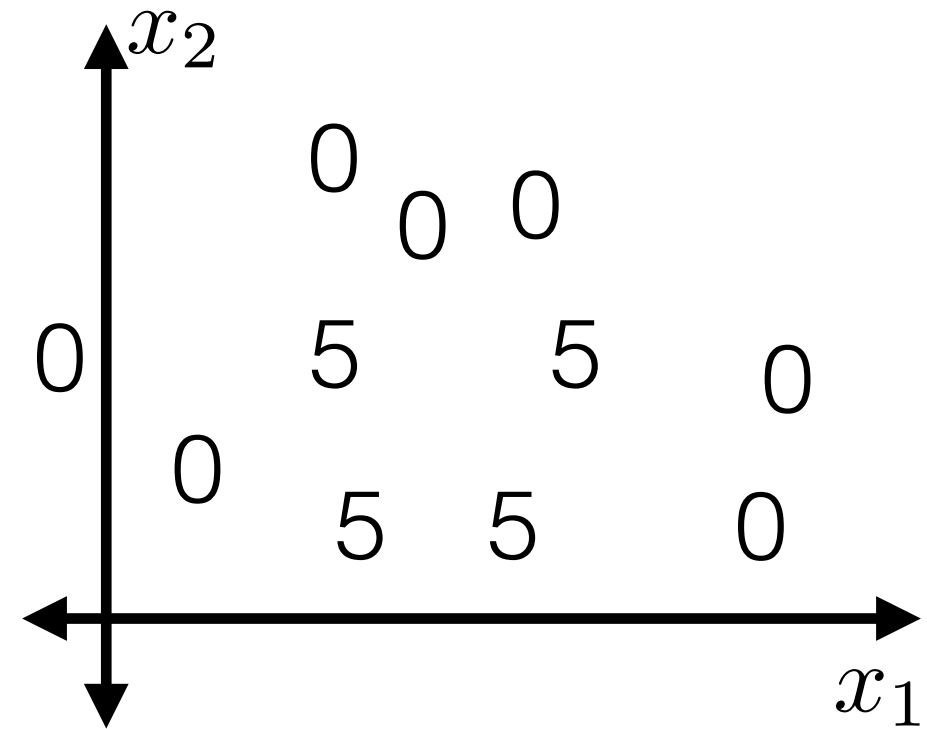
Set $\hat{y} = \text{average}_{i \in I} y^{(i)}$

return Leaf(label = \hat{y})

else

for each split dim j & value s

`BuildTree({1, ..., n}; 2)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

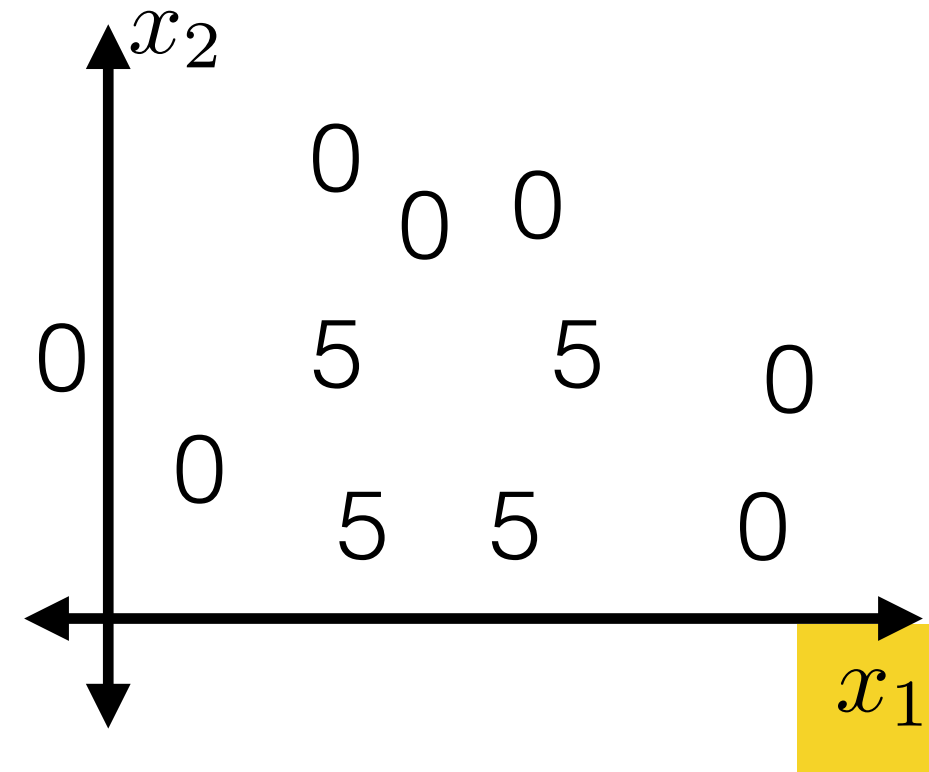
Set $\hat{y} = \text{average}_{i \in I} y^{(i)}$

return Leaf(label = \hat{y})

else

for each split dim j & value s

`BuildTree({1, ..., n}; 2)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

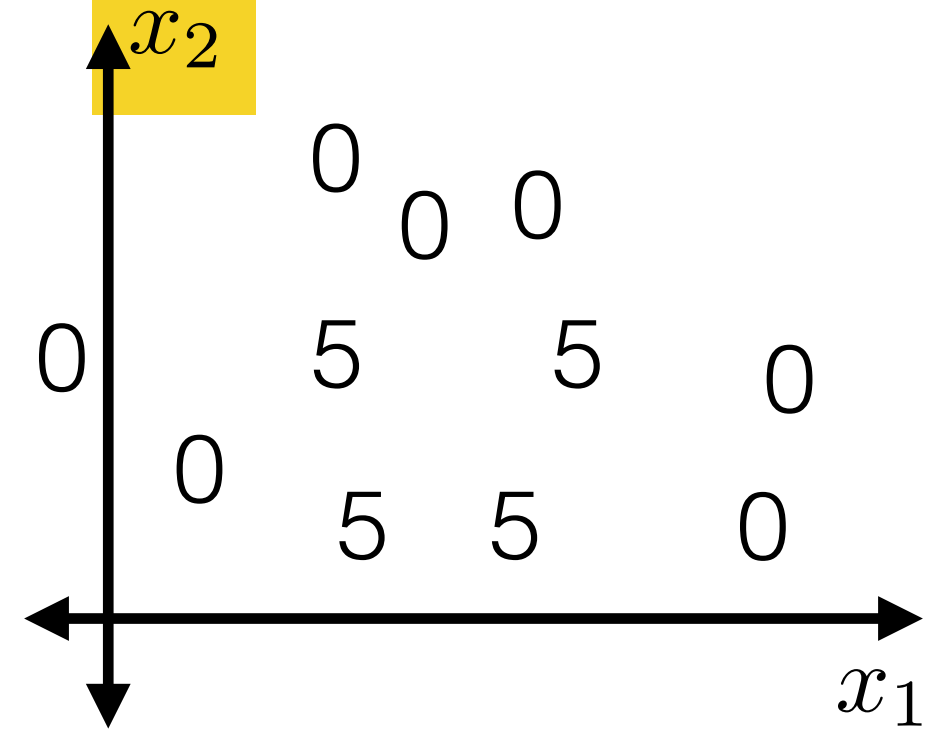
Set $\hat{y} = \text{average}_{i \in I} y^{(i)}$

return Leaf(label = \hat{y})

else

for each split dim j & value s

`BuildTree({1, ..., n}; 2)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

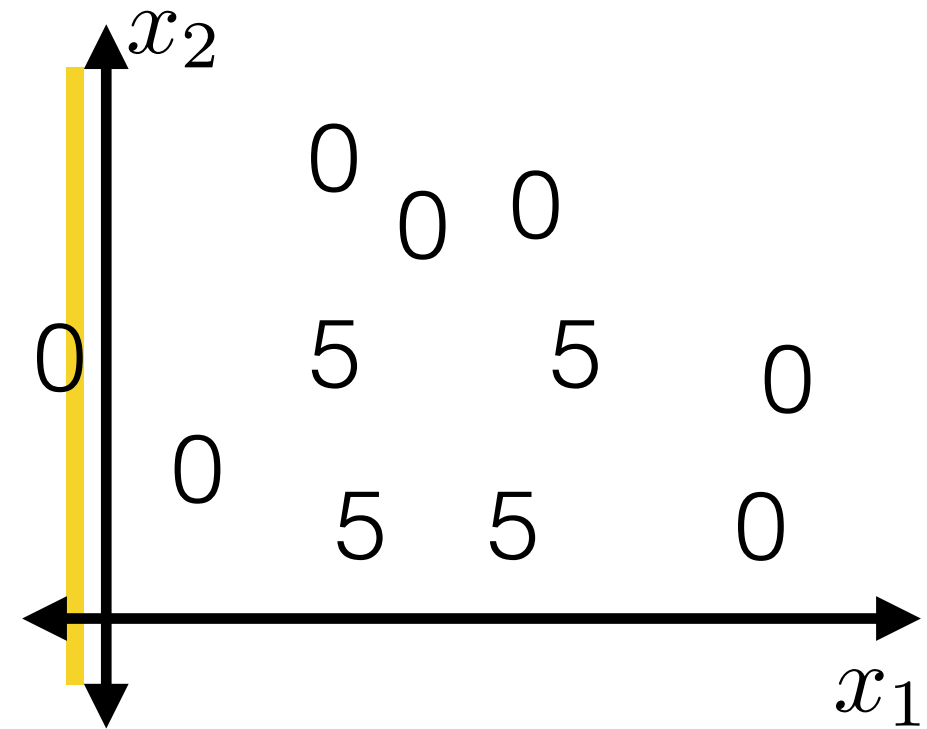
Set $\hat{y} = \text{average}_{i \in I} y^{(i)}$

return Leaf(label = \hat{y})

else

for each split dim j & value s

`BuildTree({1, ..., n}; 2)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

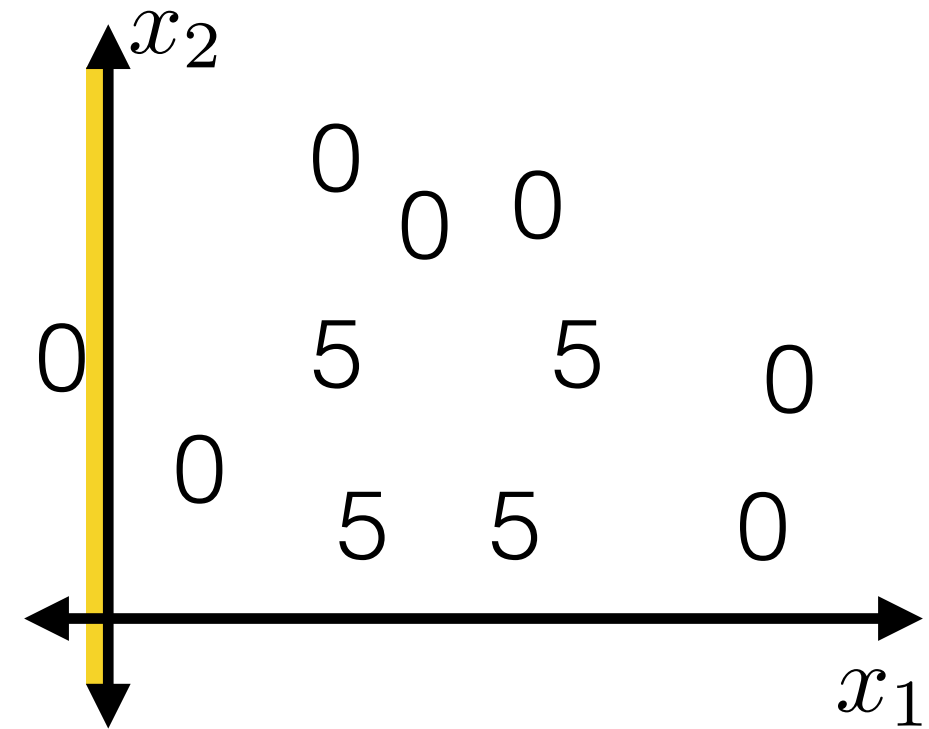
if $|I| \leq k$

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

return Leaf(label = \hat{y}) `BuildTree({1, ..., n}; 2)`

else

for each split dim j & value s



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

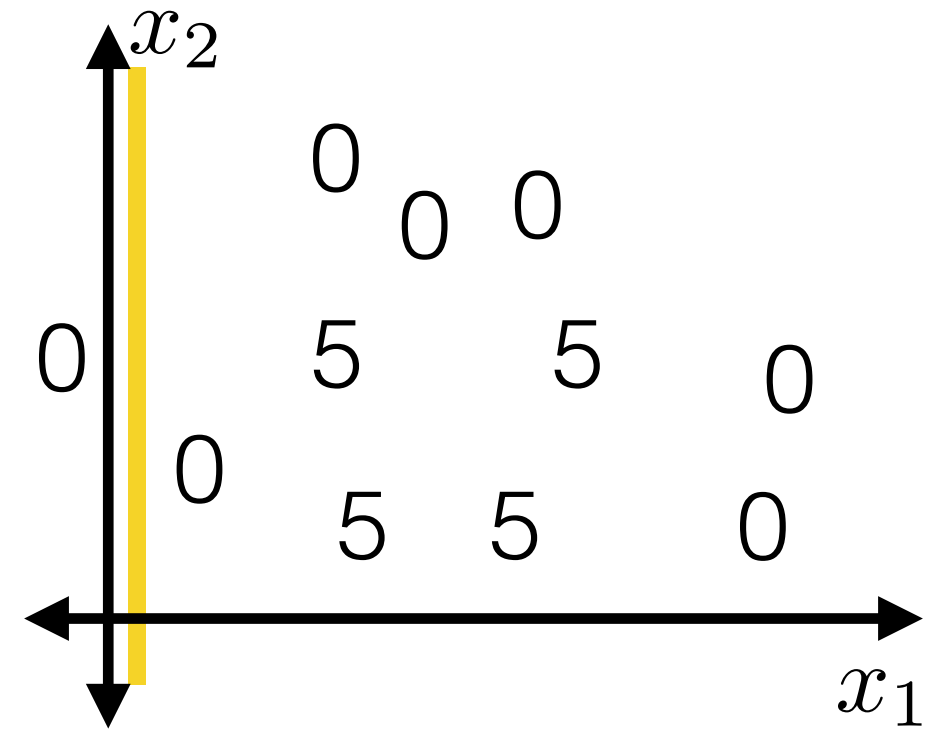
if $|I| \leq k$

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

return Leaf(label = \hat{y}) `BuildTree({1, ..., n}; 2)`

else

for each split dim j & value s



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

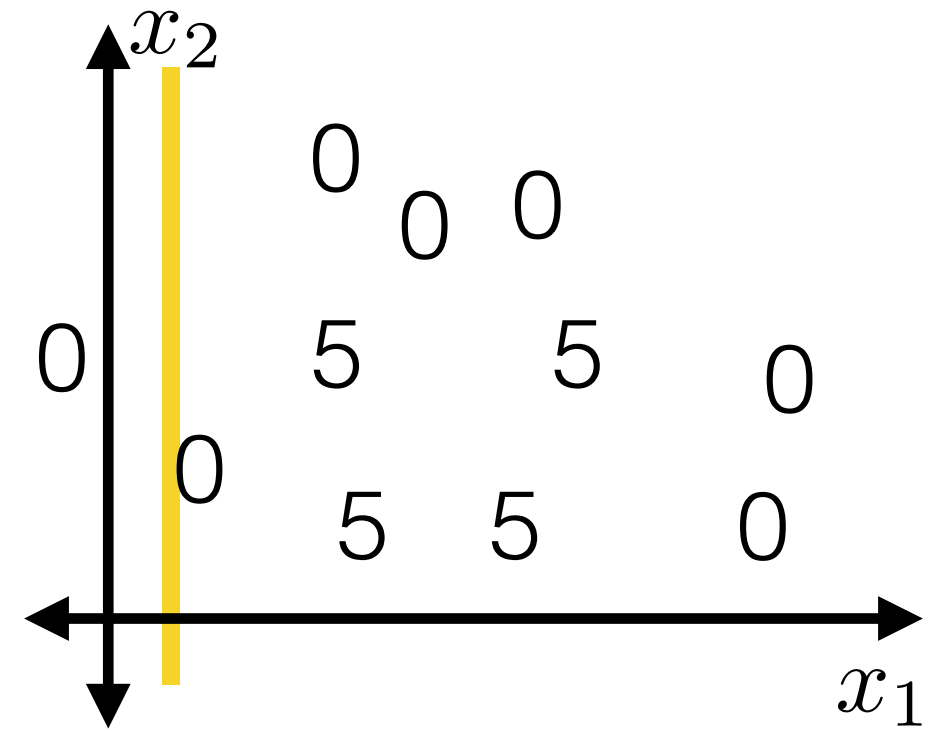
Set $\hat{y} = \text{average}_{i \in I} y^{(i)}$

return Leaf(label = \hat{y})

else

for each split dim j & value s

`BuildTree({1, ..., n}; 2)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

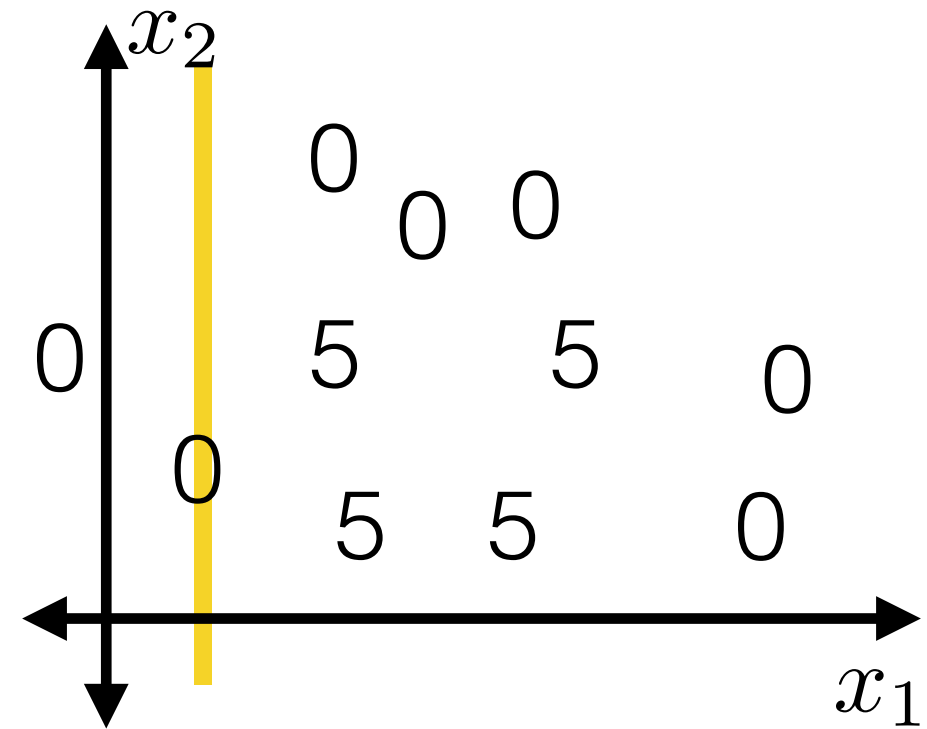
Set $\hat{y} = \text{average}_{i \in I} y^{(i)}$

return Leaf(label = \hat{y})

else

for each split dim j & value s

`BuildTree({1, ..., n}; 2)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

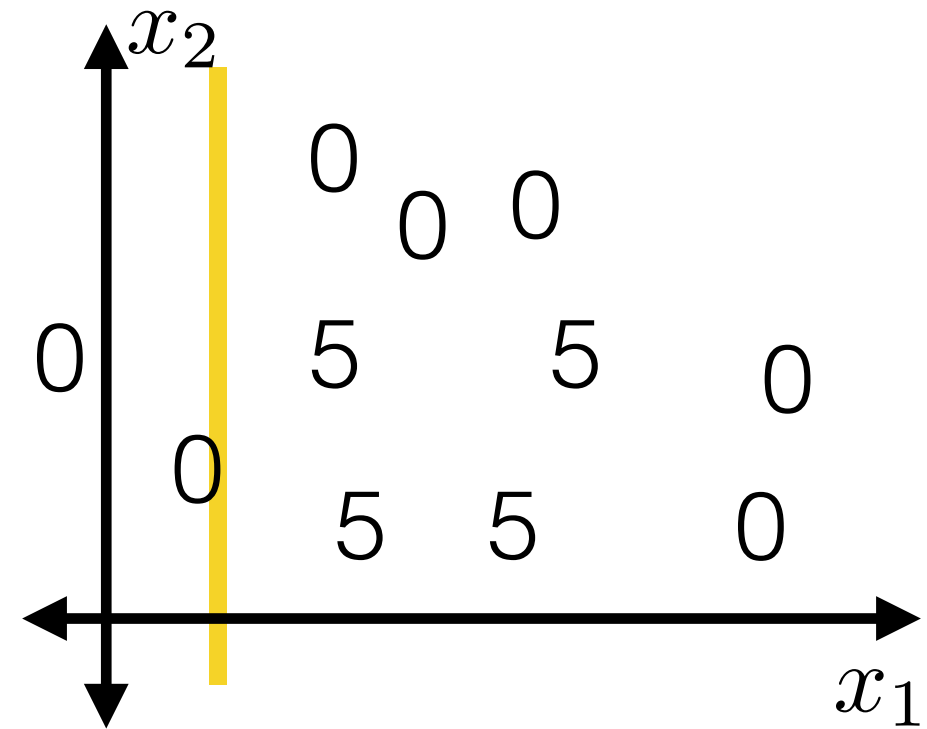
Set $\hat{y} = \text{average}_{i \in I} y^{(i)}$

return Leaf(label = \hat{y})

else

for each split dim j & value s

`BuildTree({1, ..., n}; 2)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

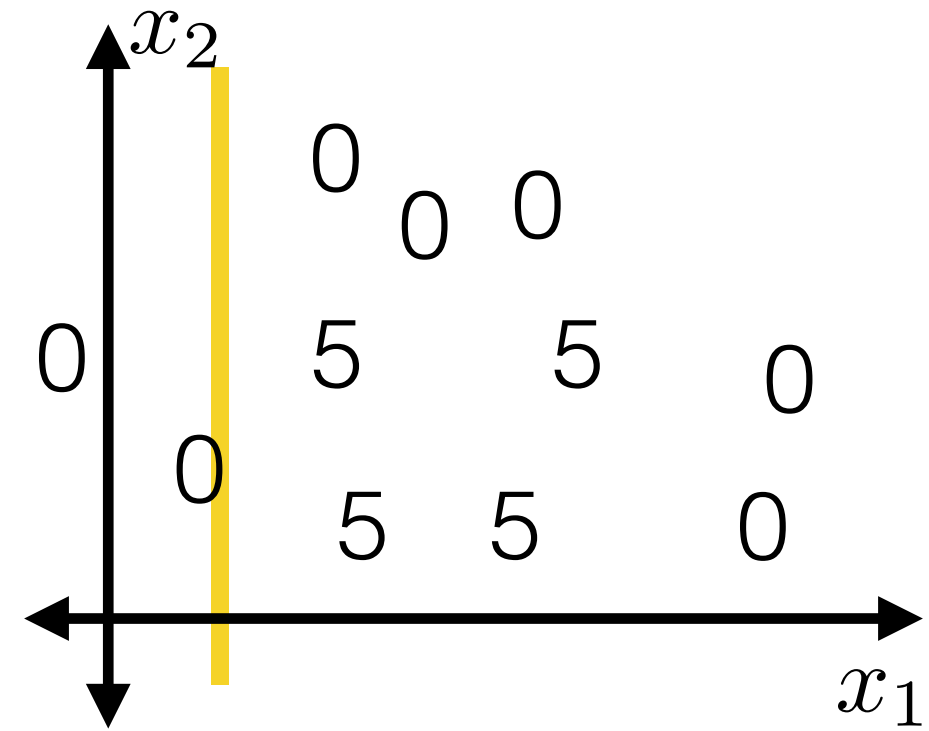
Set $\hat{y} = \text{average}_{i \in I} y^{(i)}$

return Leaf(label = \hat{y})

else

for each split dim j & value s

`BuildTree({1, ..., n}; 2)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

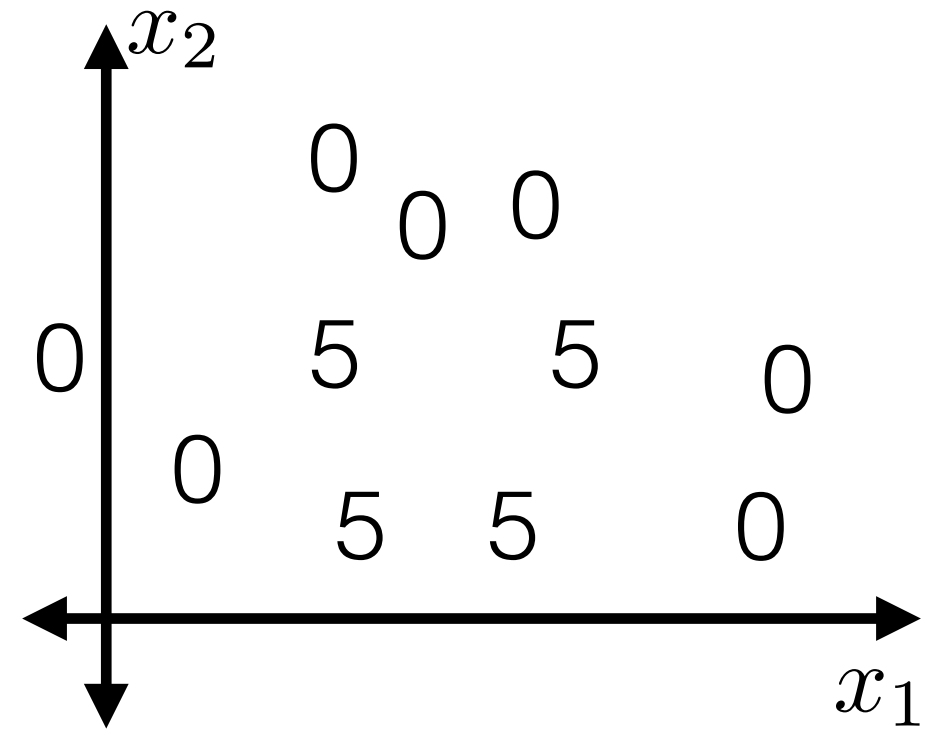
Set $\hat{y} = \text{average}_{i \in I} y^{(i)}$

return Leaf(label = \hat{y})

else

for each split dim j & value s

`BuildTree({1, ..., n}; 2)`



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

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

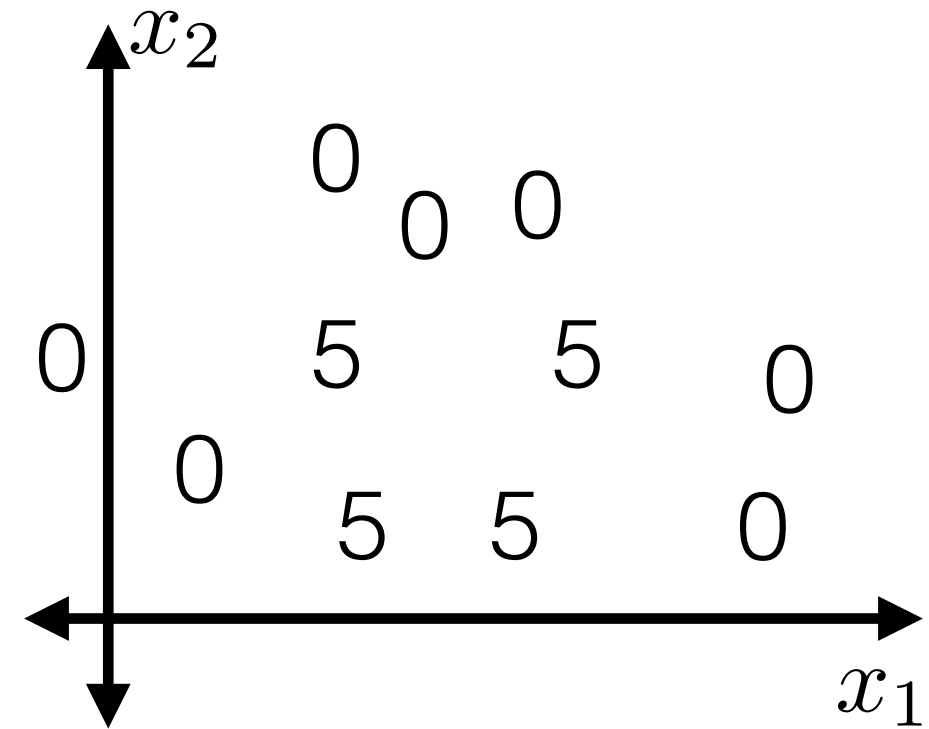
return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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



`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

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

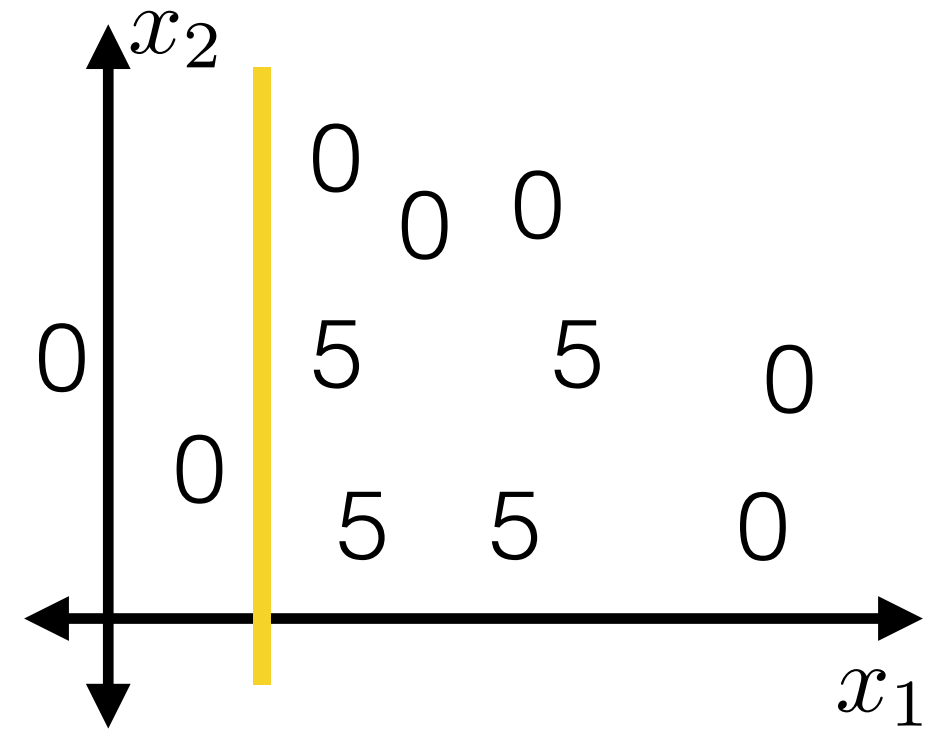
return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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



`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

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

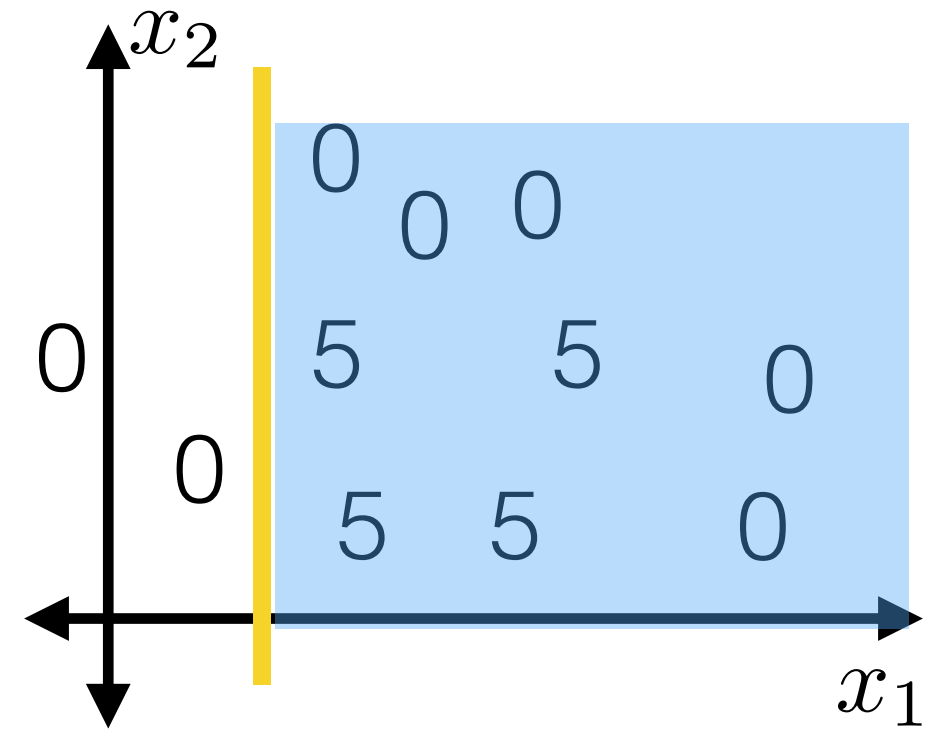
return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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



`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

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

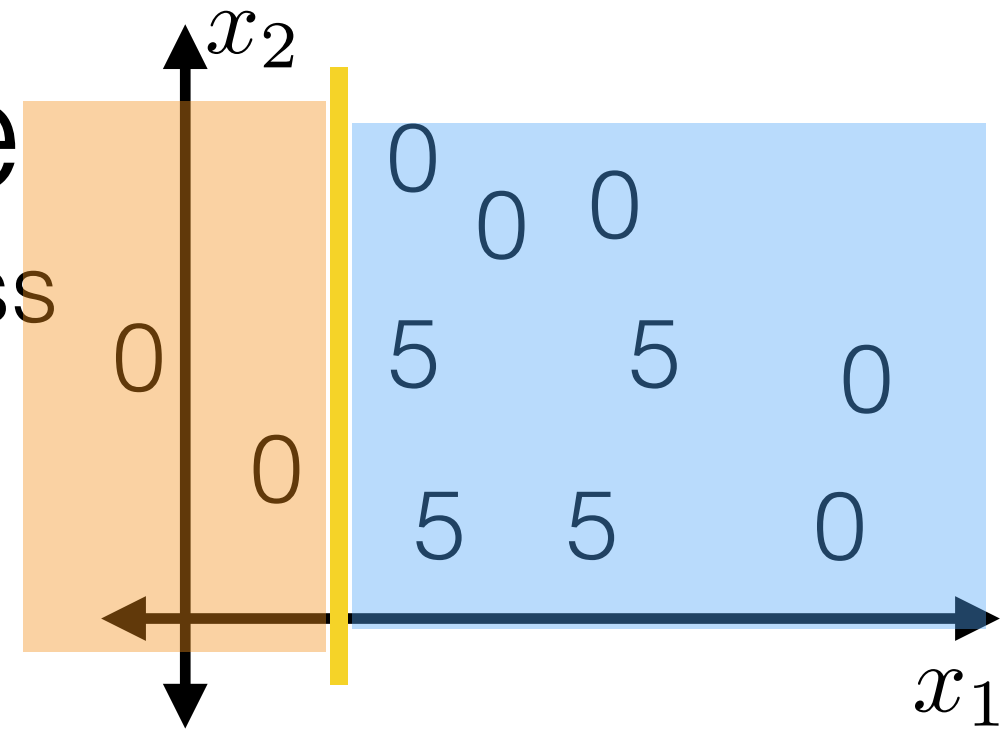
return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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



`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

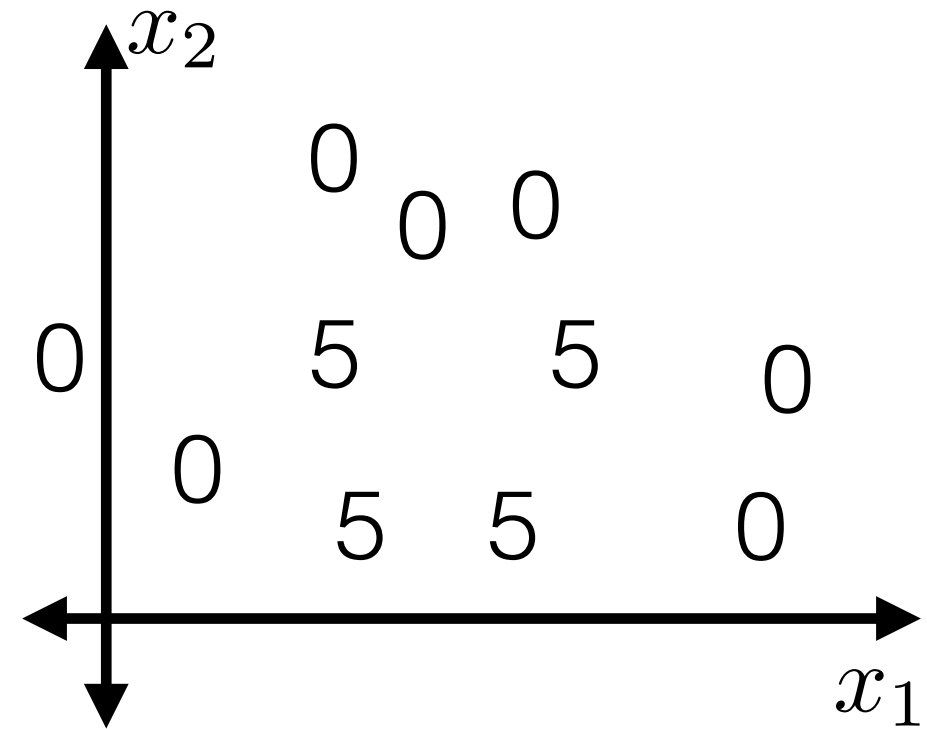
for each split dim j & value s

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

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

$\hat{y}_{j,s}^+$

$\hat{y}_{j,s}^-$



`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y}) BuildTree($\{1, \dots, n\}; 2$)

else

for each split dim j & value s

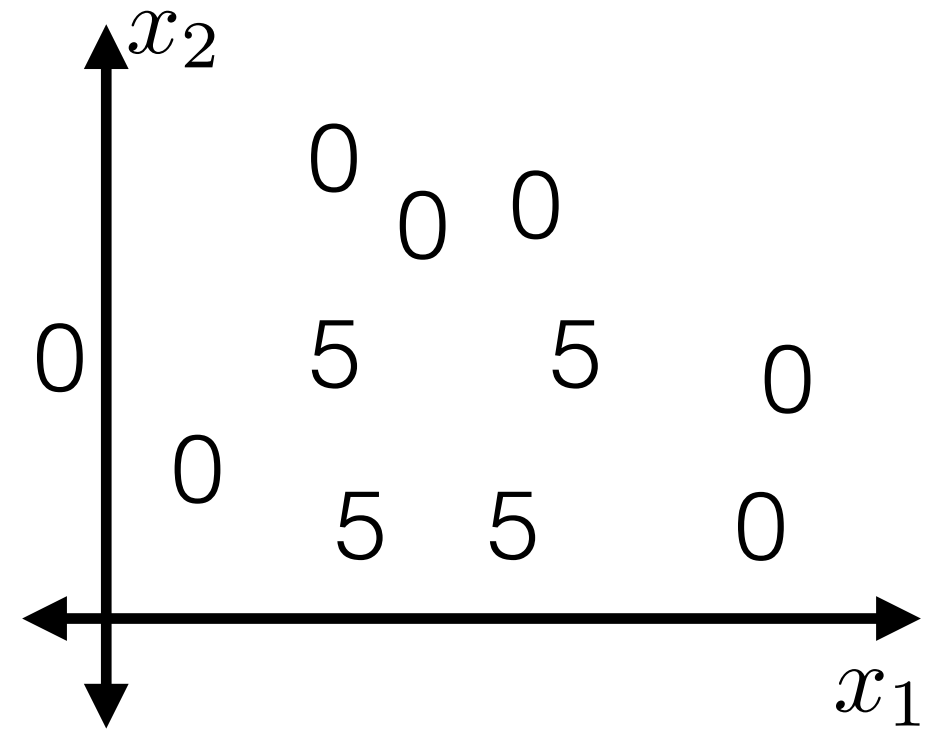
Set $I_{j,s}^+ = \{i \in I \mid x_j^{(i)} \geq s\}$

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

$\hat{y}_{j,s}^+$

$\hat{y}_{j,s}^-$

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y}) BuildTree($\{1, \dots, n\}; 2$)

else

for each split dim j & value s

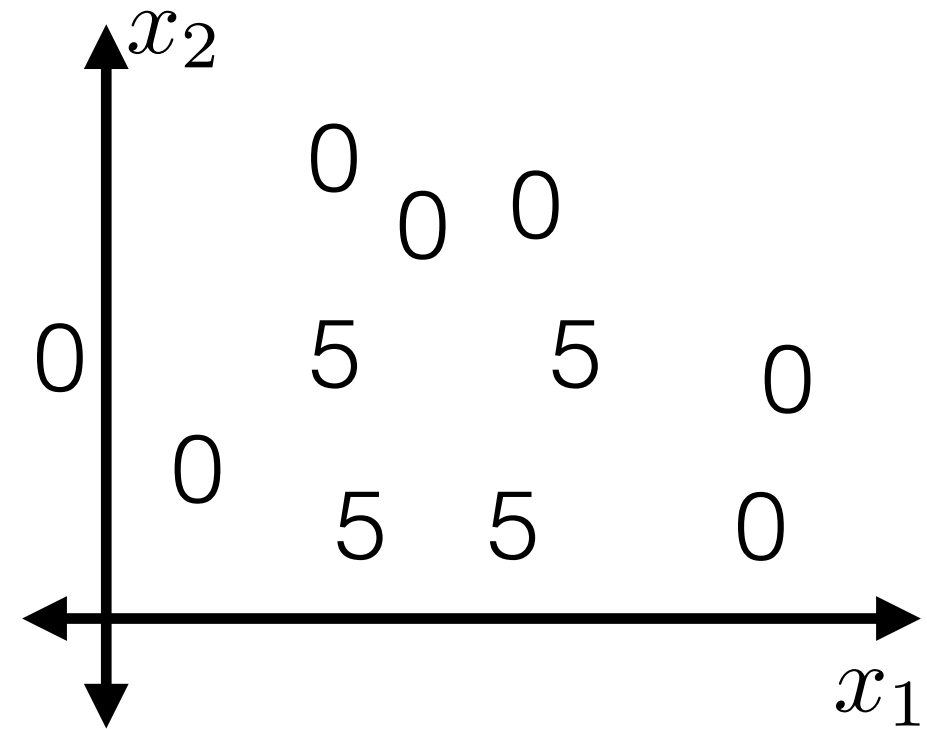
Set $I_{j,s}^+ = \{i \in I \mid x_j^{(i)} \geq s\}$

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

$\hat{y}_{j,s}^+$

$\hat{y}_{j,s}^-$

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y}) BuildTree($\{1, \dots, n\}; 2$)

else

for each split dim j & value s

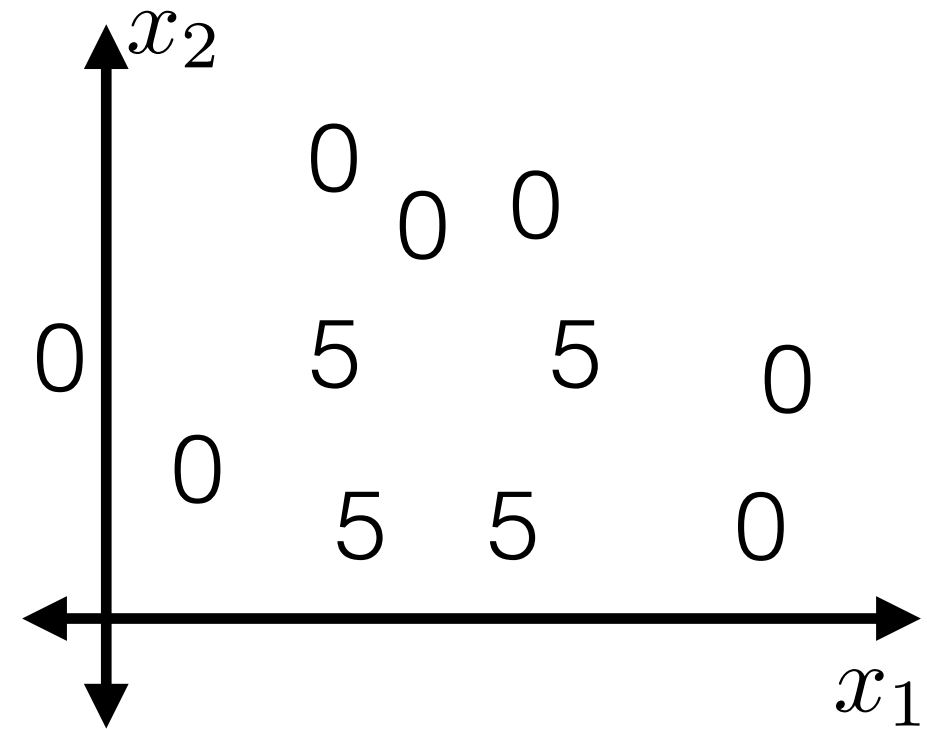
Set $I_{j,s}^+ = \{i \in I \mid x_j^{(i)} \geq s\}$

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

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

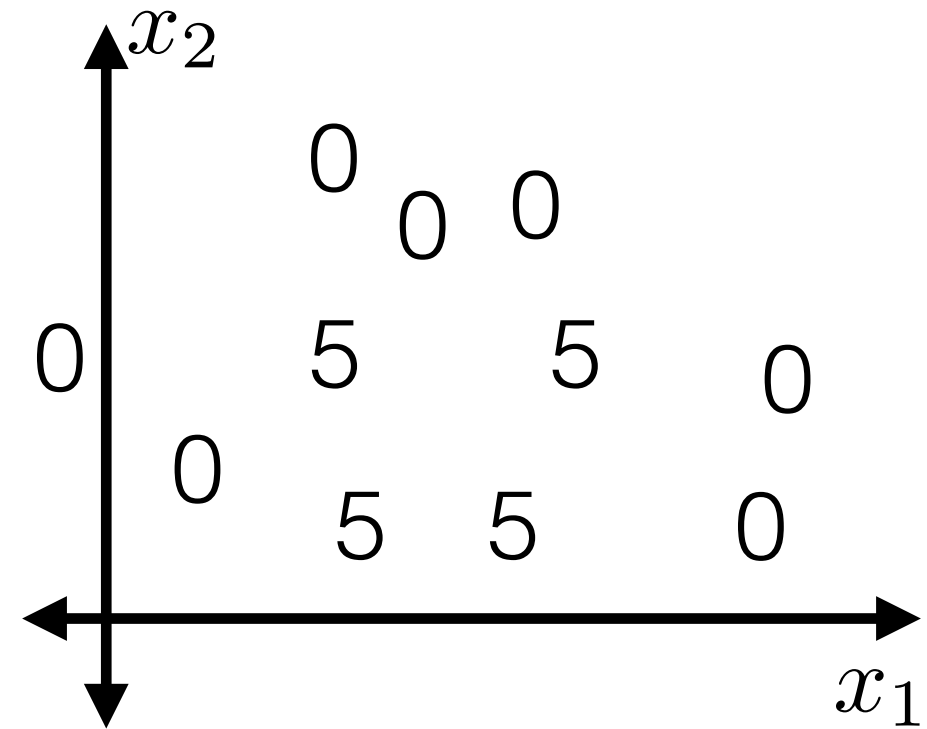
Set $I_{j,s}^+ = \{i \in I \mid x_j^{(i)} \geq s\}$

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

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



BuildTree($\{1, \dots, n\}; 2$)

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y}) BuildTree($\{1, \dots, n\}; 2$)

else

for each split dim j & value s

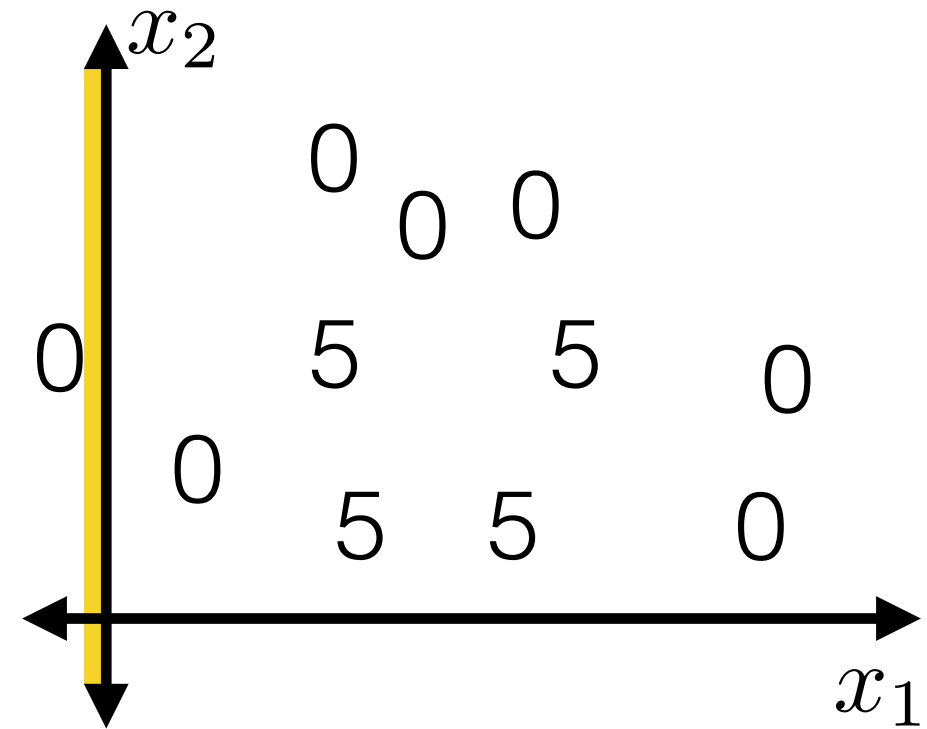
Set $I_{j,s}^+ = \{i \in I \mid x_j^{(i)} \geq s\}$

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

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y}) BuildTree($\{1, \dots, n\}; 2$)

else

for each split dim j & value s

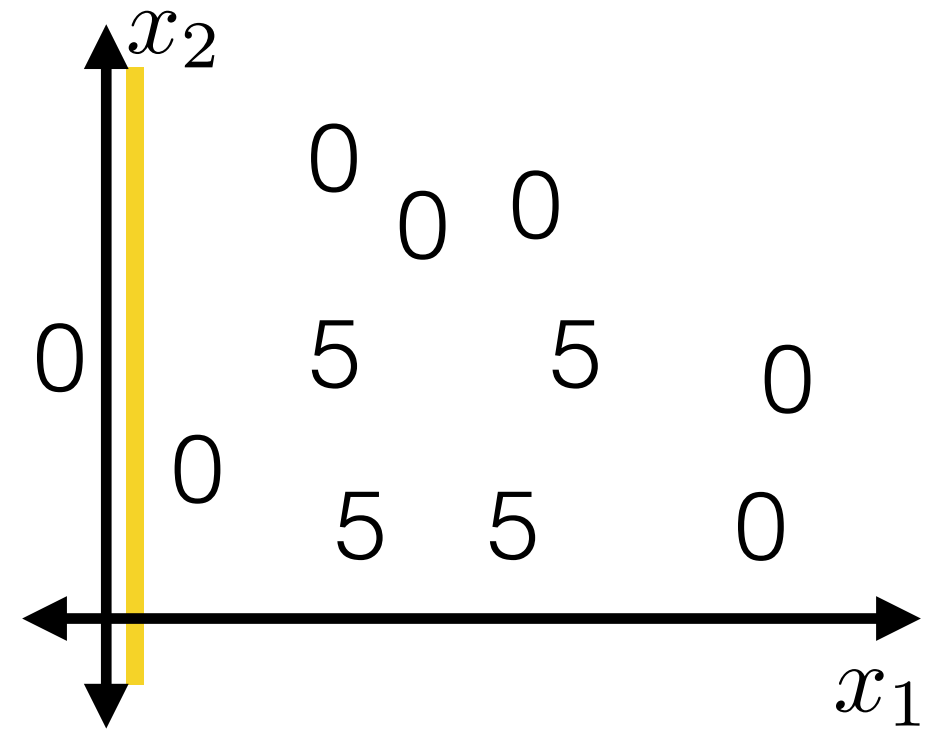
Set $I_{j,s}^+ = \{i \in I \mid x_j^{(i)} \geq s\}$

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

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y}) BuildTree($\{1, \dots, n\}; 2$)

else

for each split dim j & value s

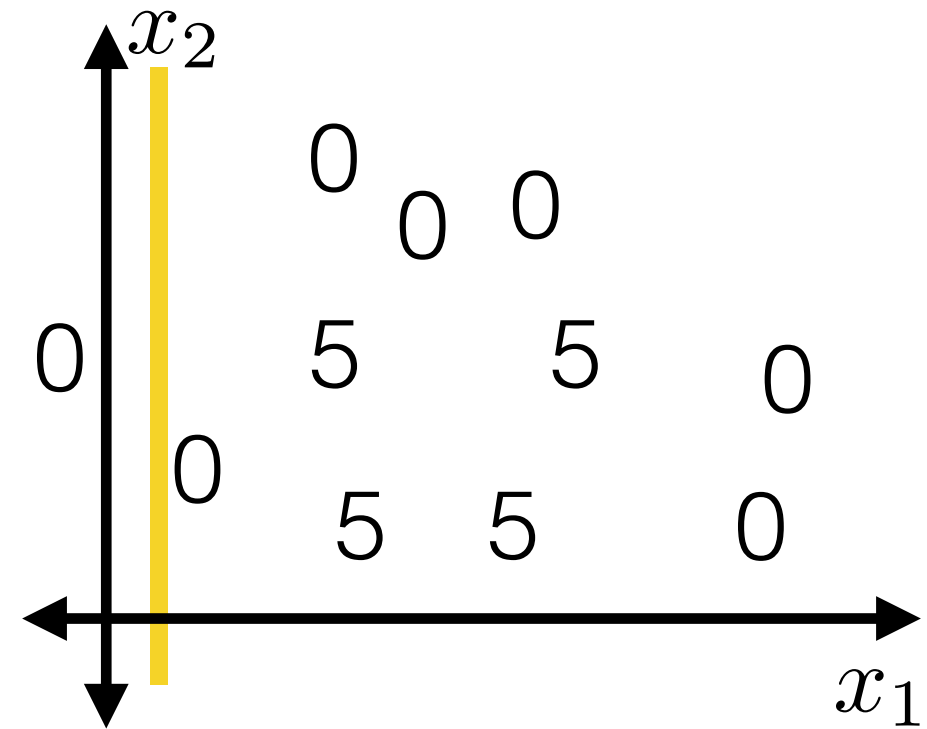
Set $I_{j,s}^+ = \{i \in I \mid x_j^{(i)} \geq s\}$

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

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y}) BuildTree($\{1, \dots, n\}; 2$)

else

for each split dim j & value s

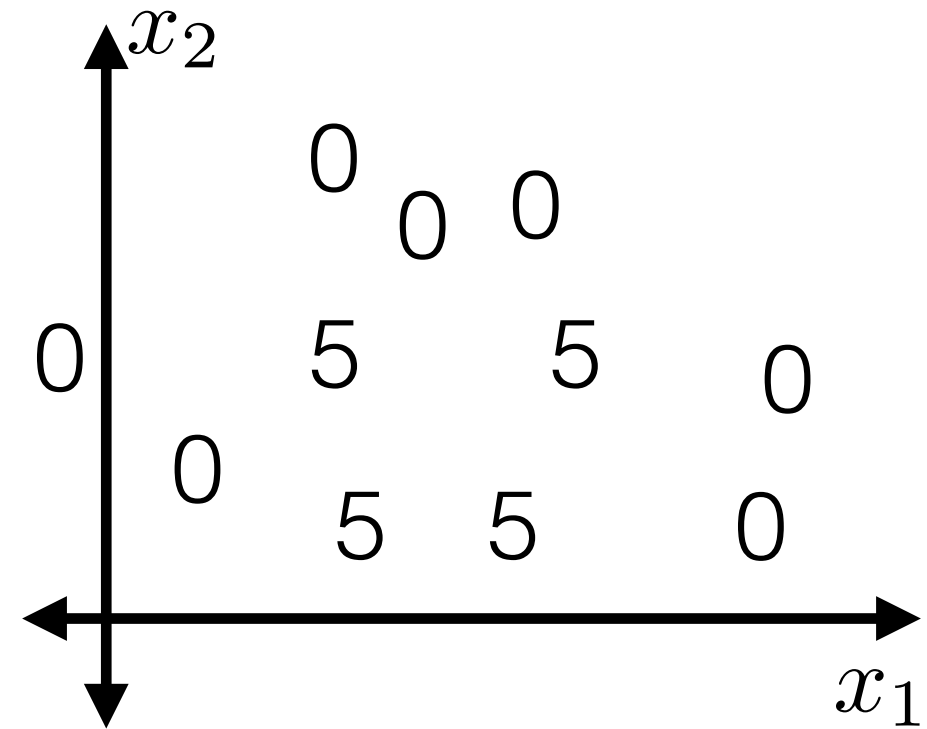
Set $I_{j,s}^+ = \{i \in I \mid x_j^{(i)} \geq s\}$

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

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y}) BuildTree($\{1, \dots, n\}; 2$)

else

for each split dim j & value s

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

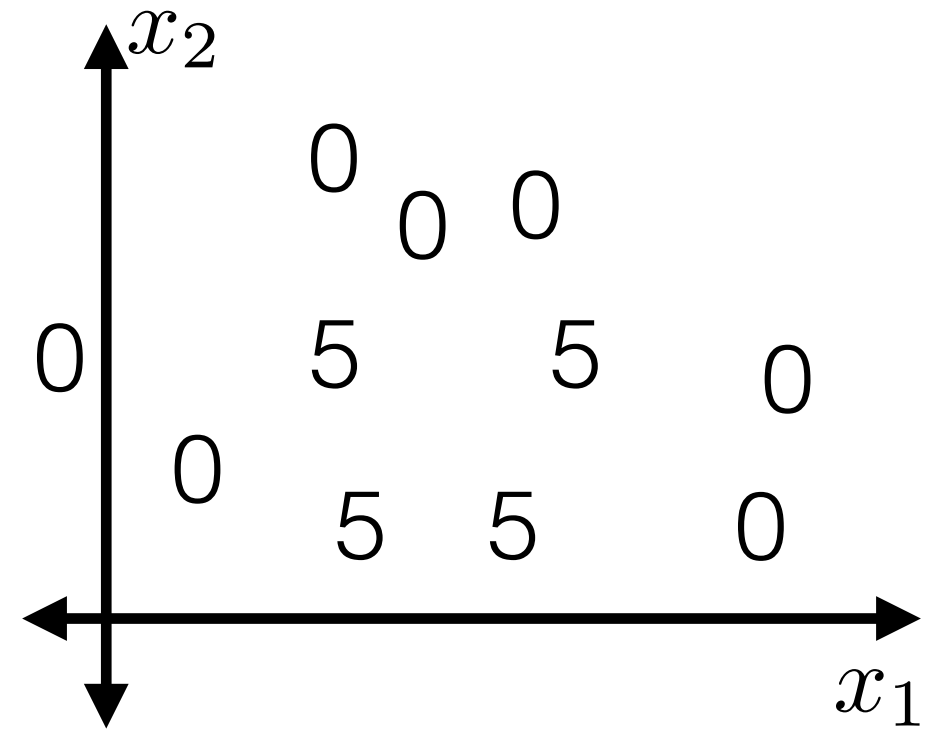
Set $I_{j,s}^- = \{i \in I \mid x_j^{(i)} < s\}$

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

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

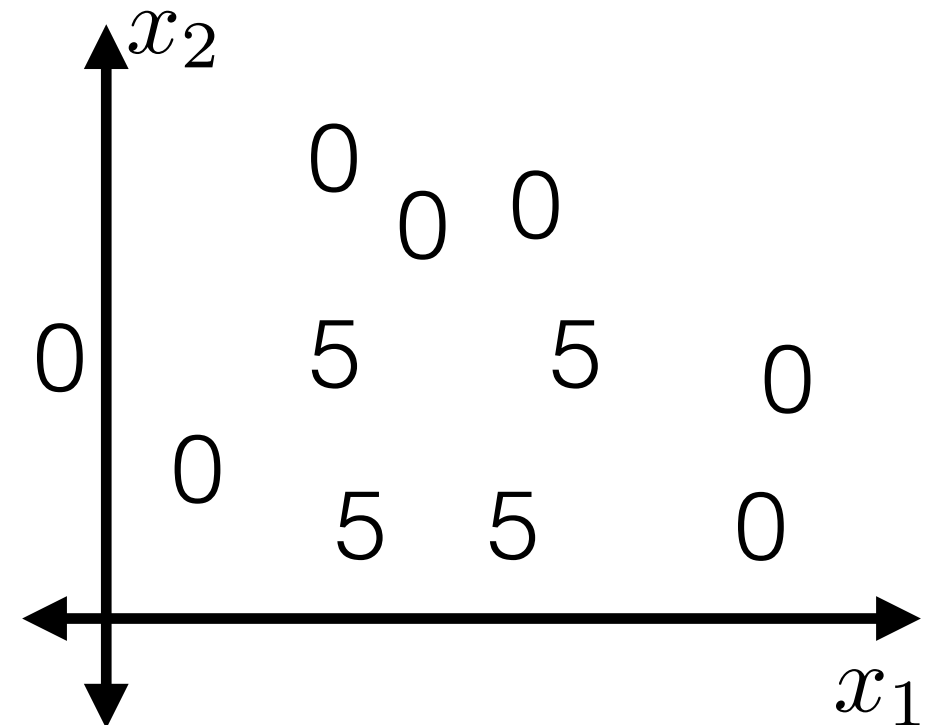
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node



BuildTree($\{1, \dots, n\}; 2$)

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

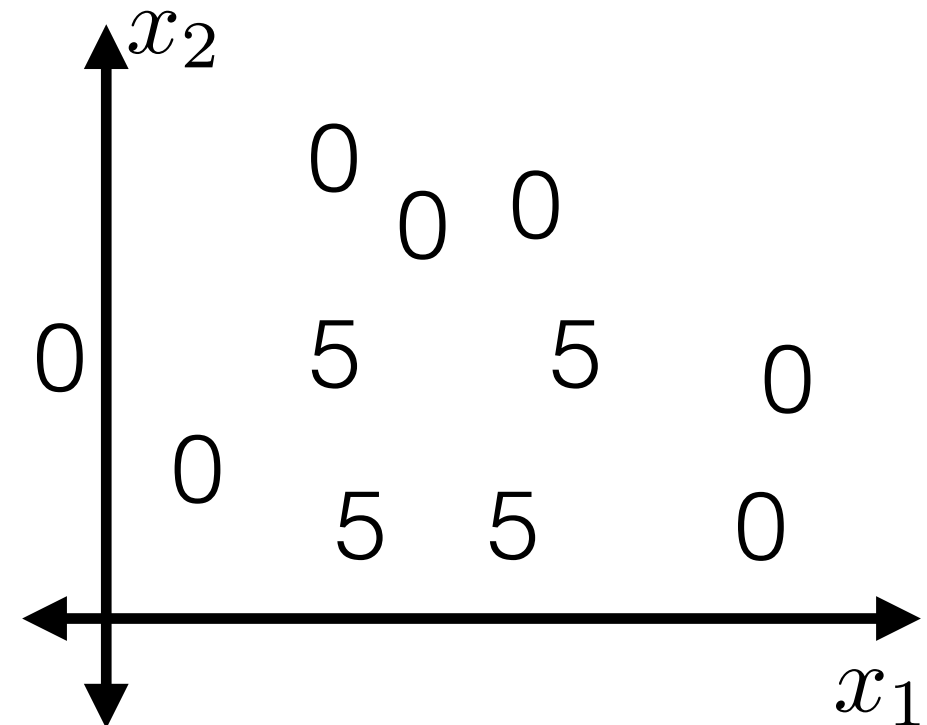
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(dim = j^* , val = s^* , left-child, right-child)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

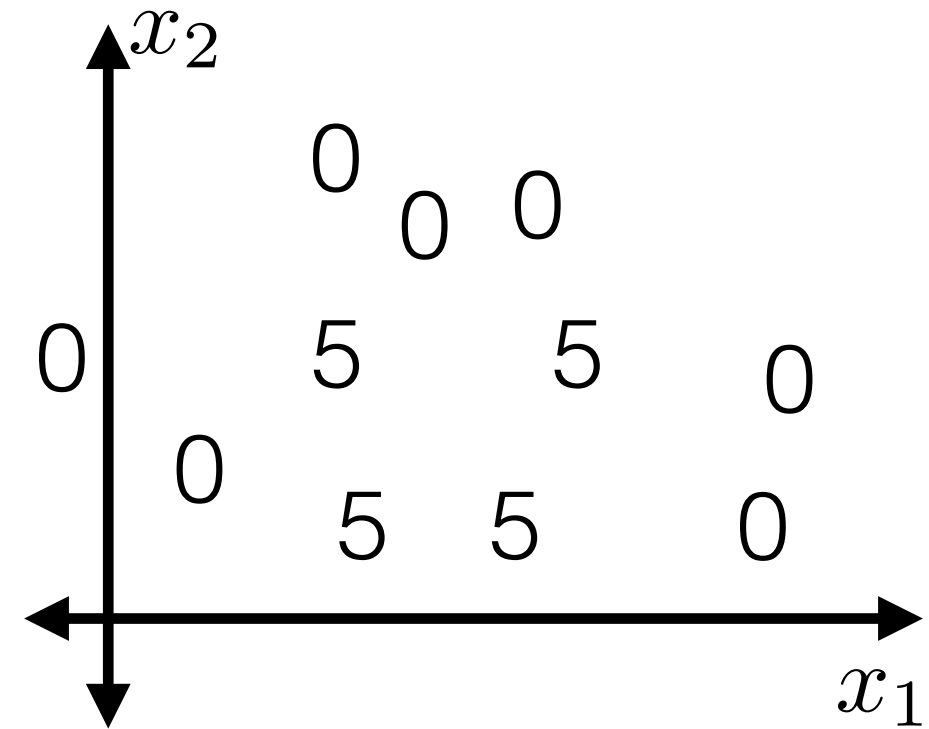
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , val, left-child, right-child)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

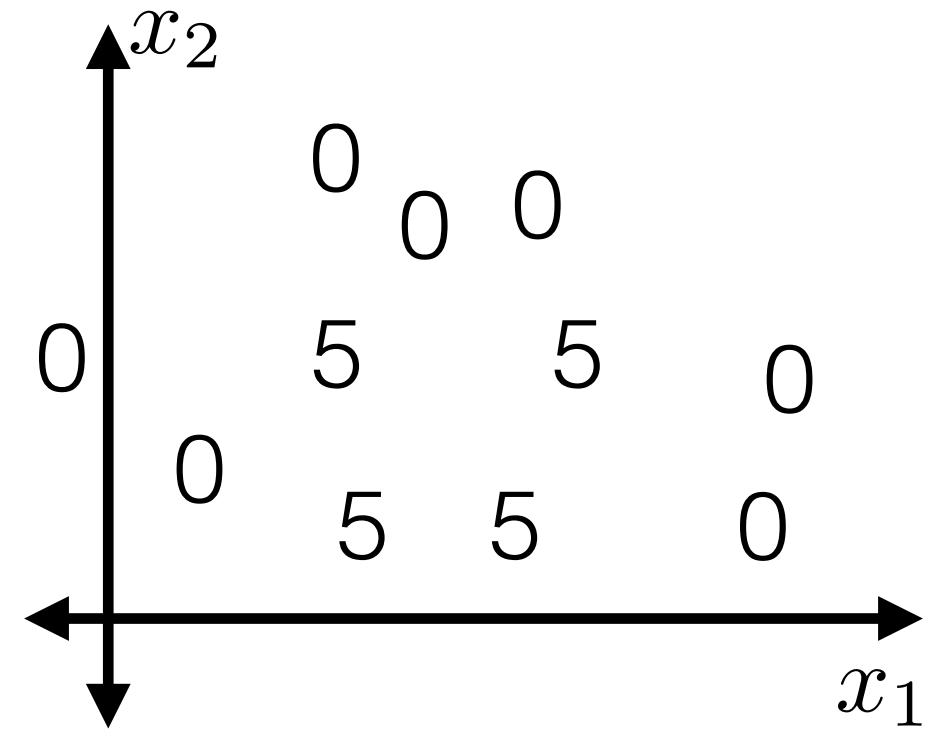
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^* , val, left-child, right-child)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

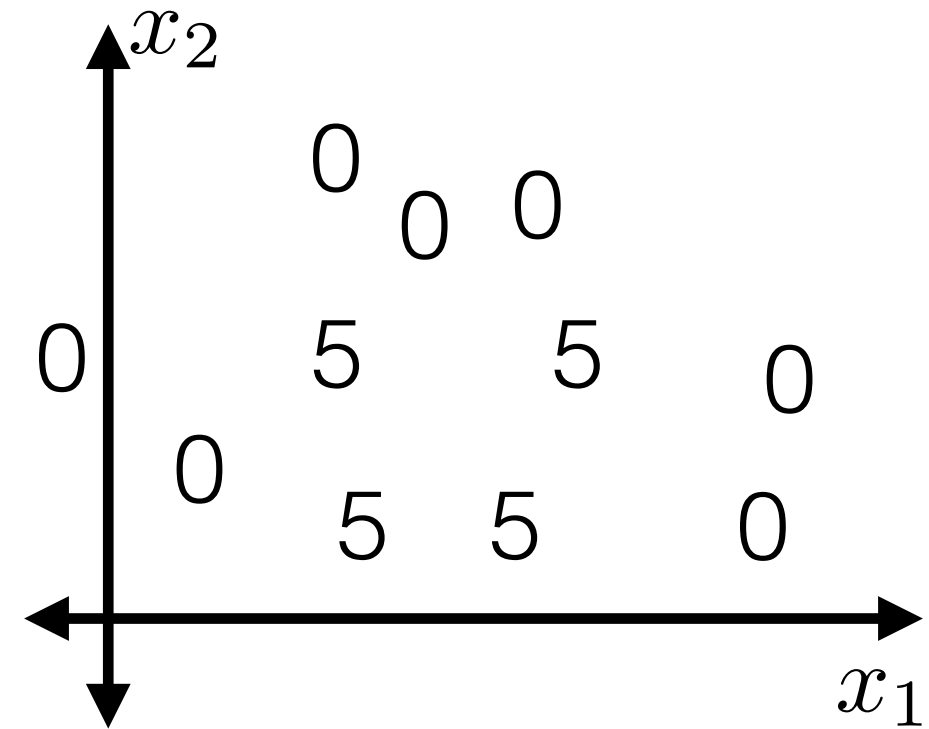
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , left-child, right-child)



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

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

return `Leaf(label = \hat{y})` `BuildTree({1, ..., n}; 2)`

else

for each split dim j & value s

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

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

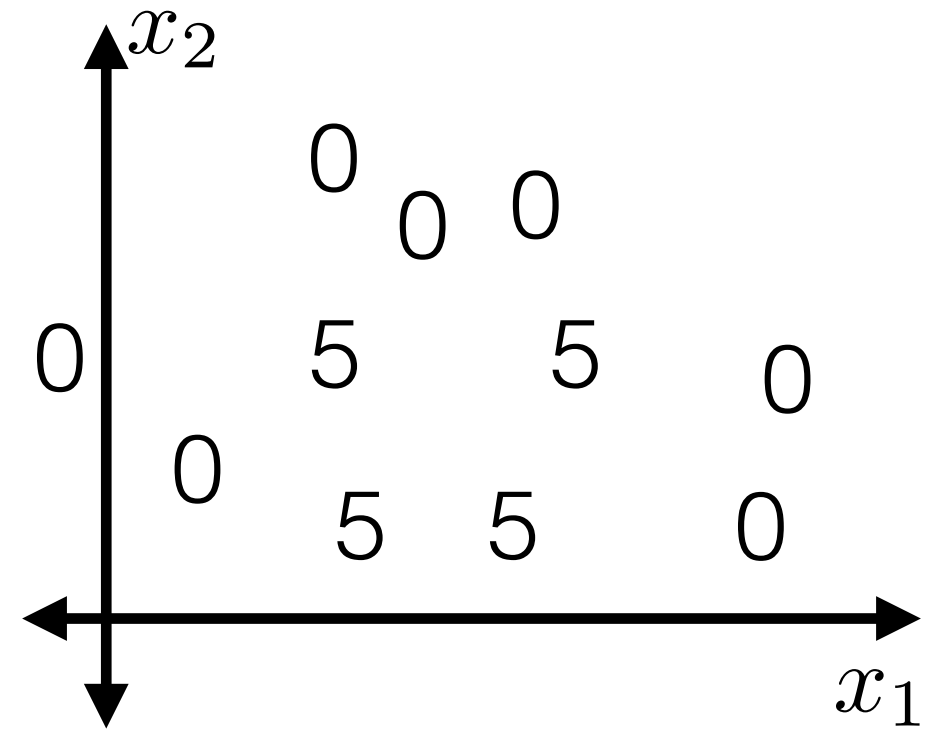
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return `Node(j^*, s^* , left-child, right-child)`



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

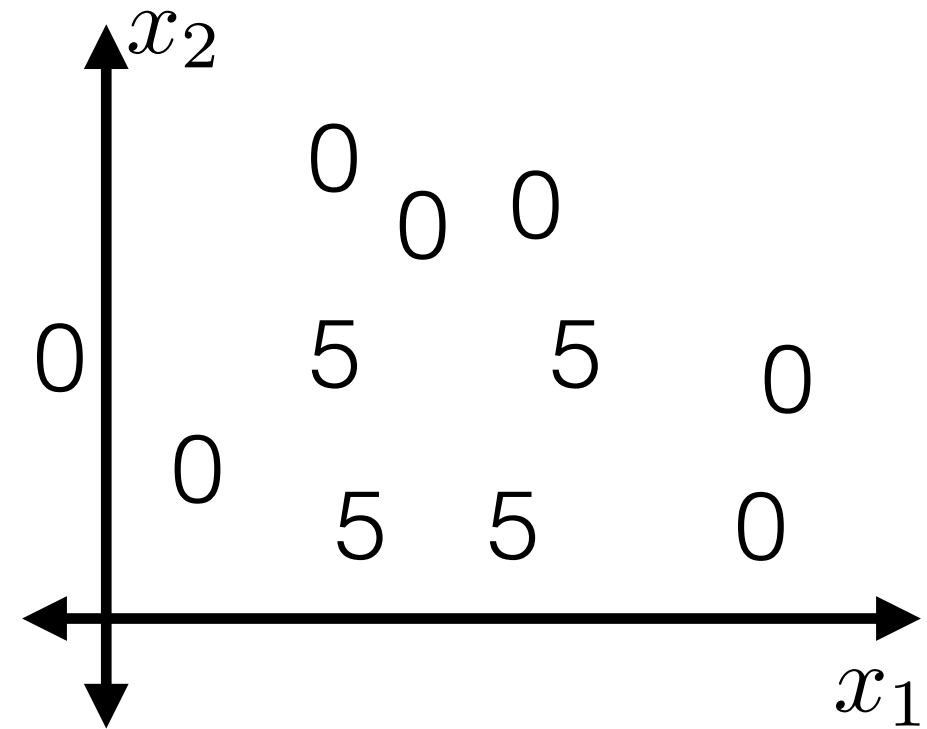
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

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



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

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

return `Leaf(label = \hat{y})` `BuildTree({1, ..., n}; 2)`

else

for each split dim j & value s

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

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

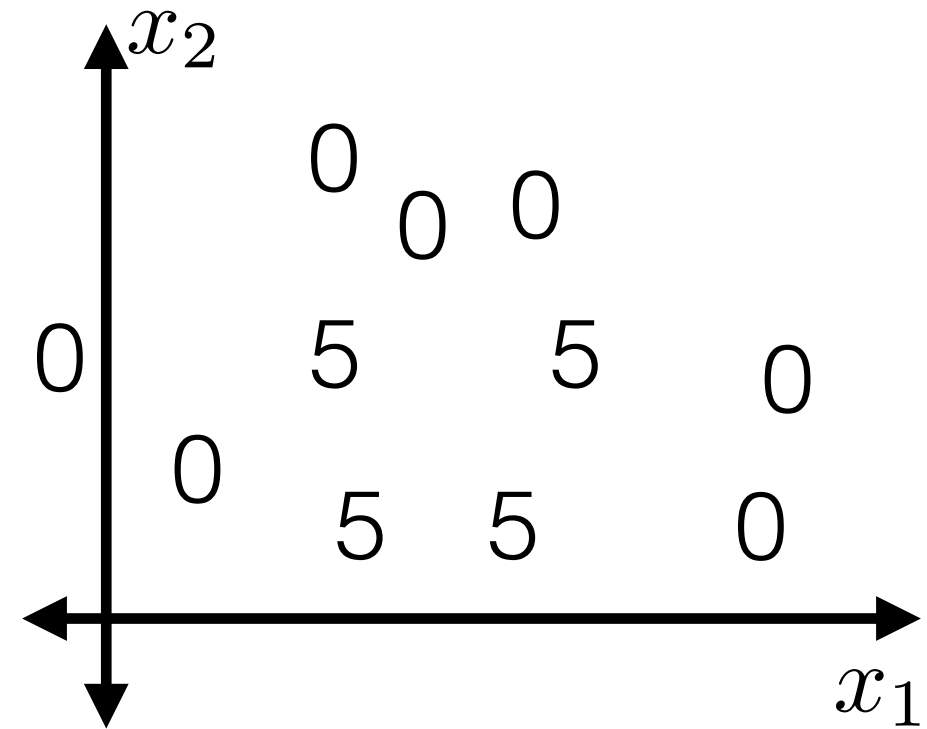
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return `Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), right-child)`



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

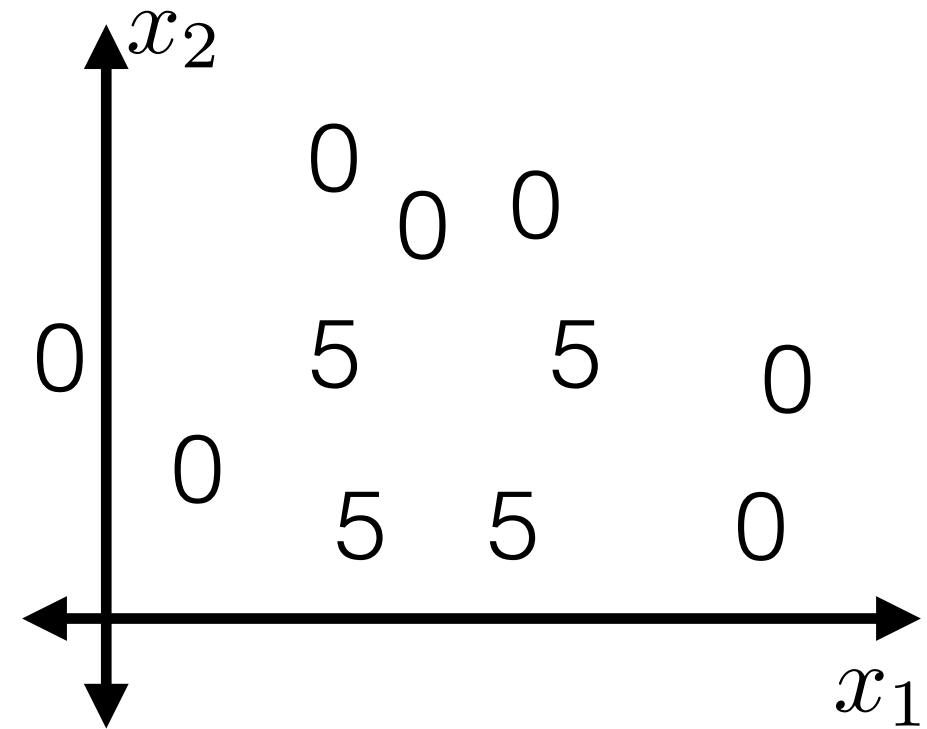
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



Building a decision tree

- Regression tree with squared error loss

`BuildTree(I; k)`

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

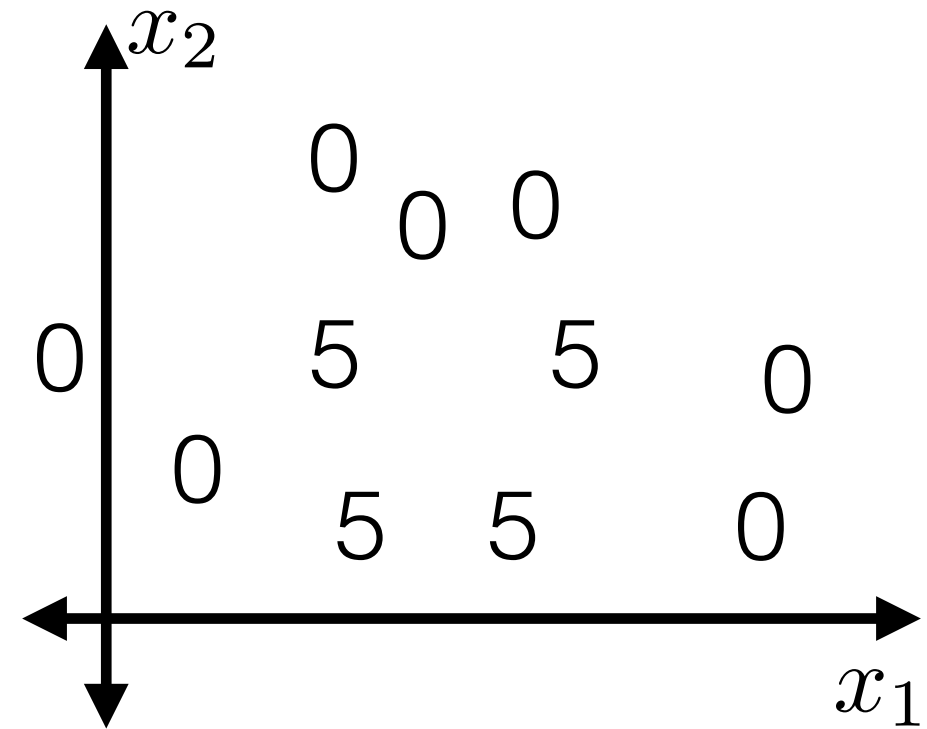
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



`BuildTree({1, ..., n}; 2)`

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

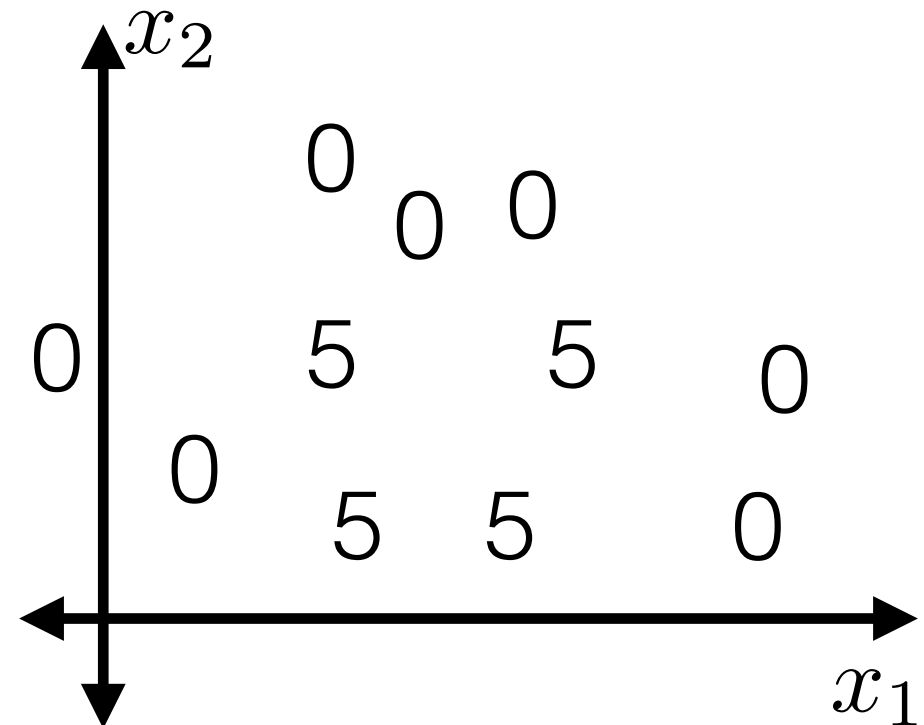
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

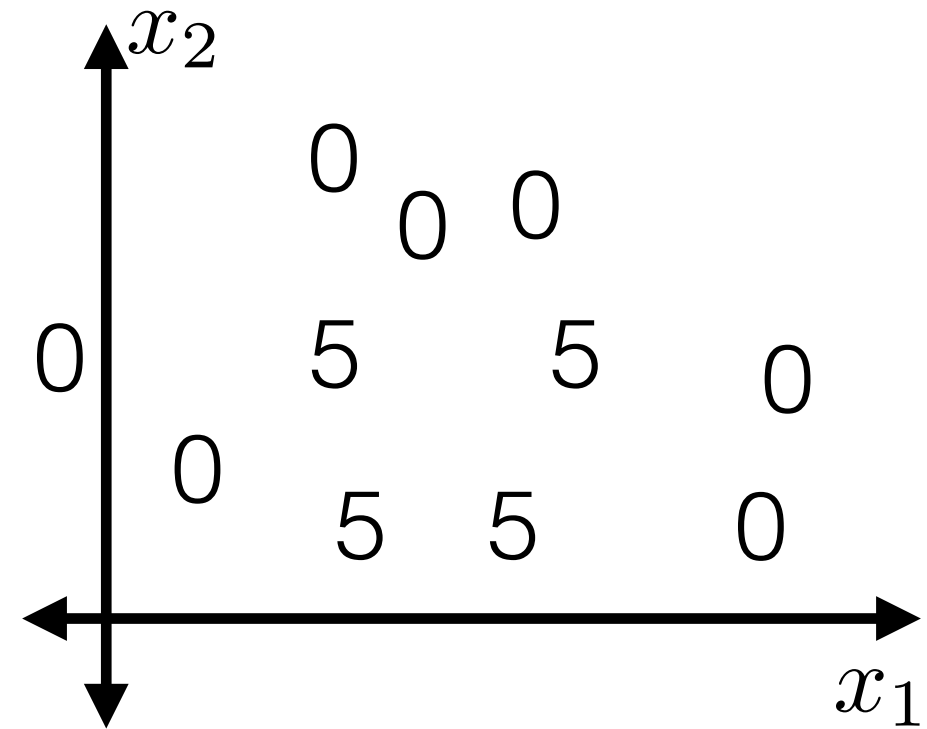
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

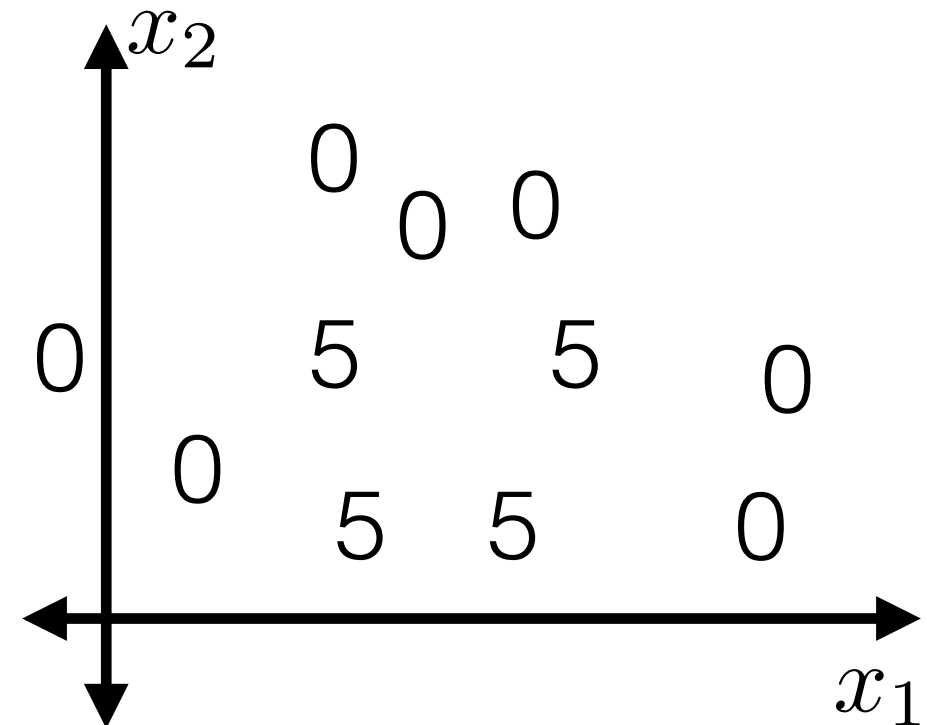
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

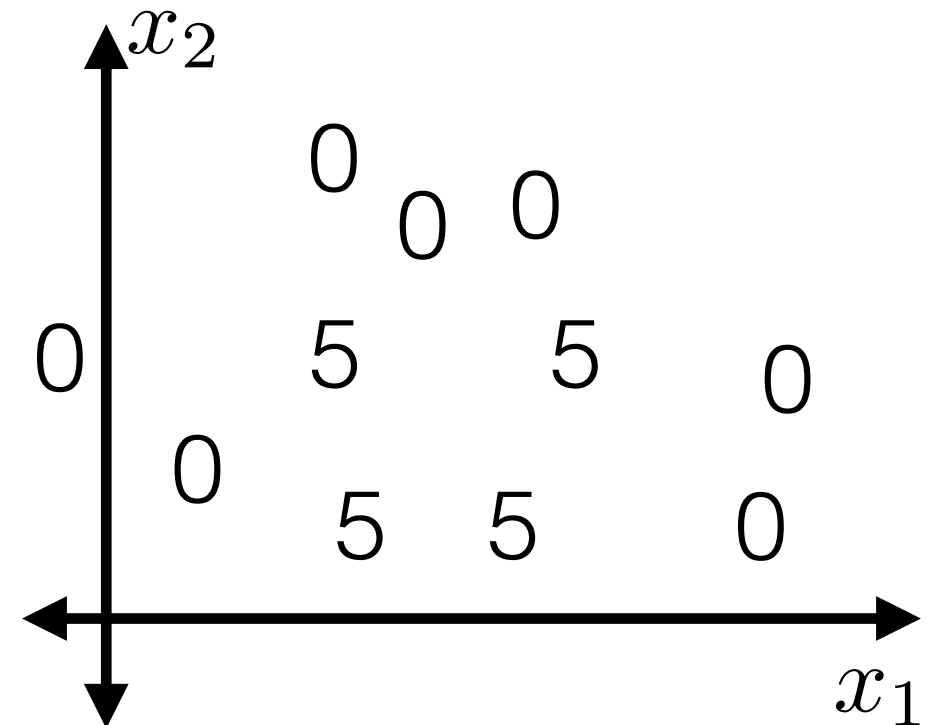
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

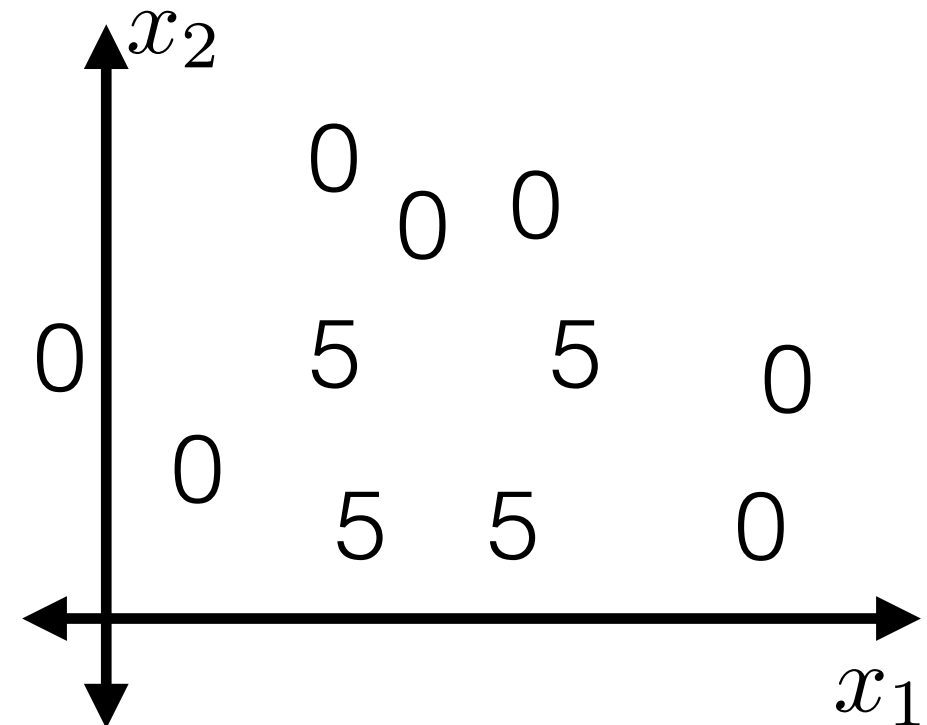
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

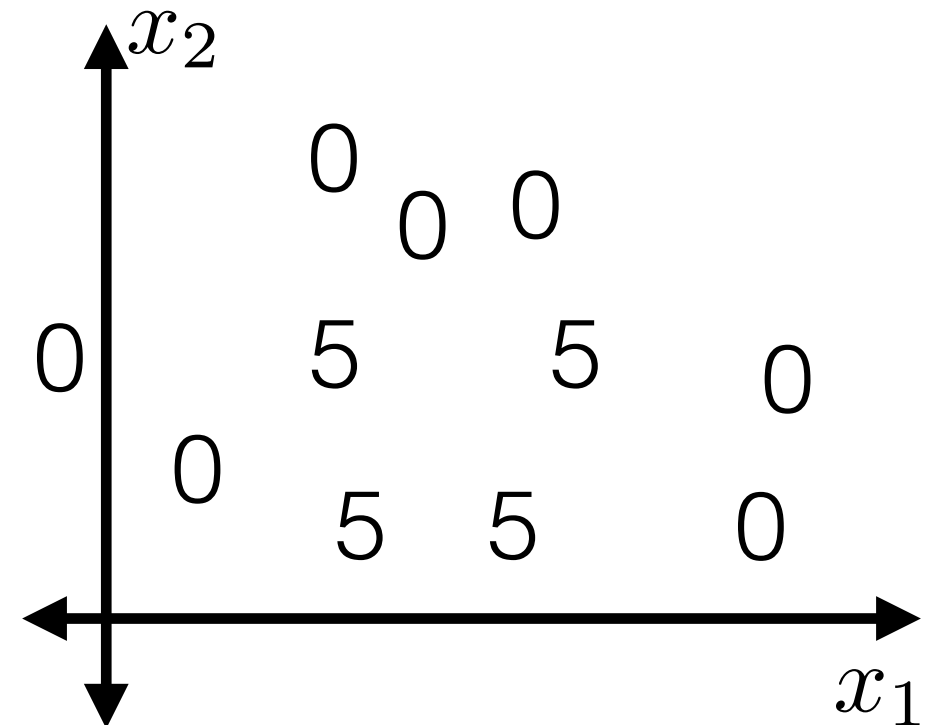
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

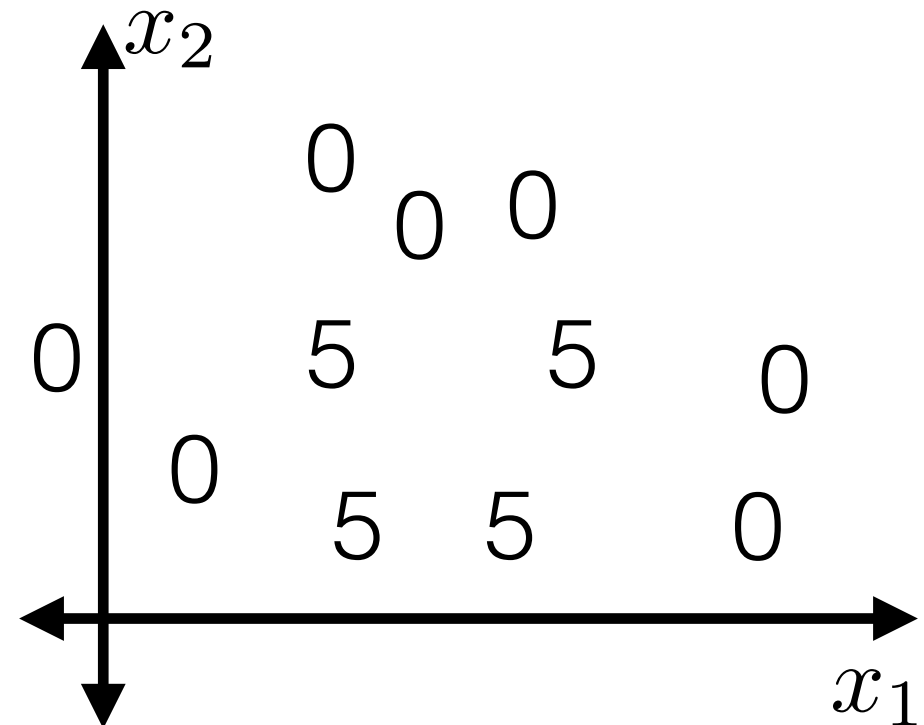
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

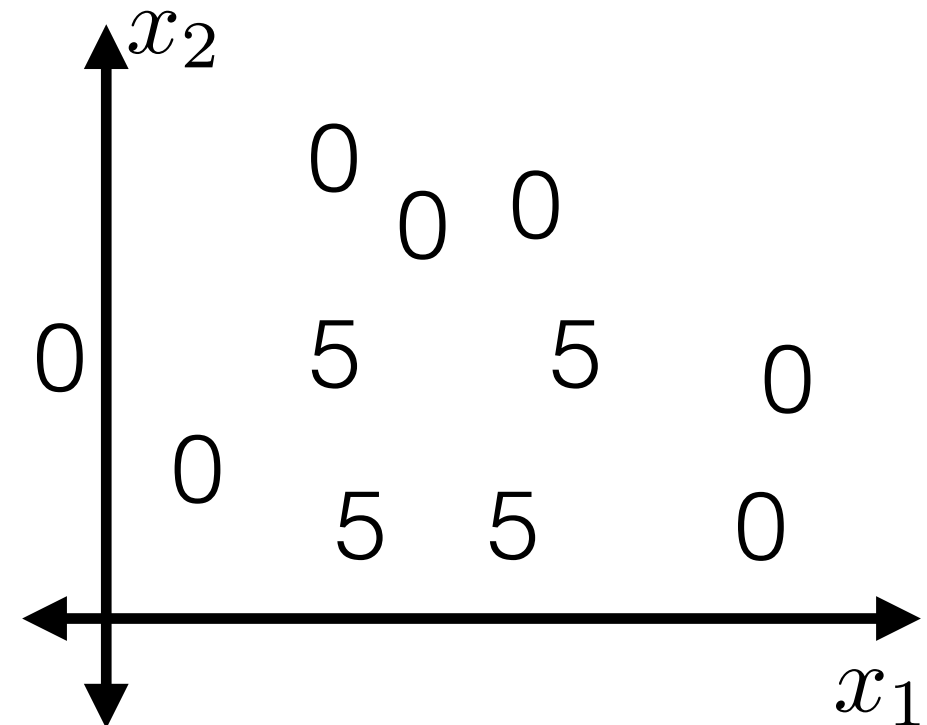
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



$x_2 \geq 0.28$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

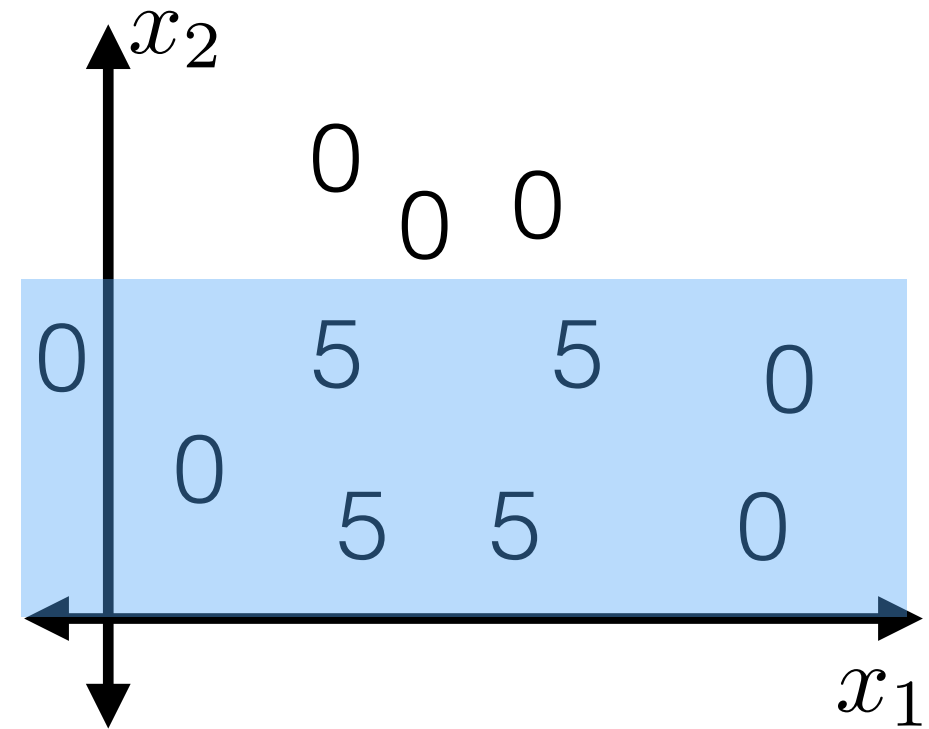
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



$x_2 \geq 0.28$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

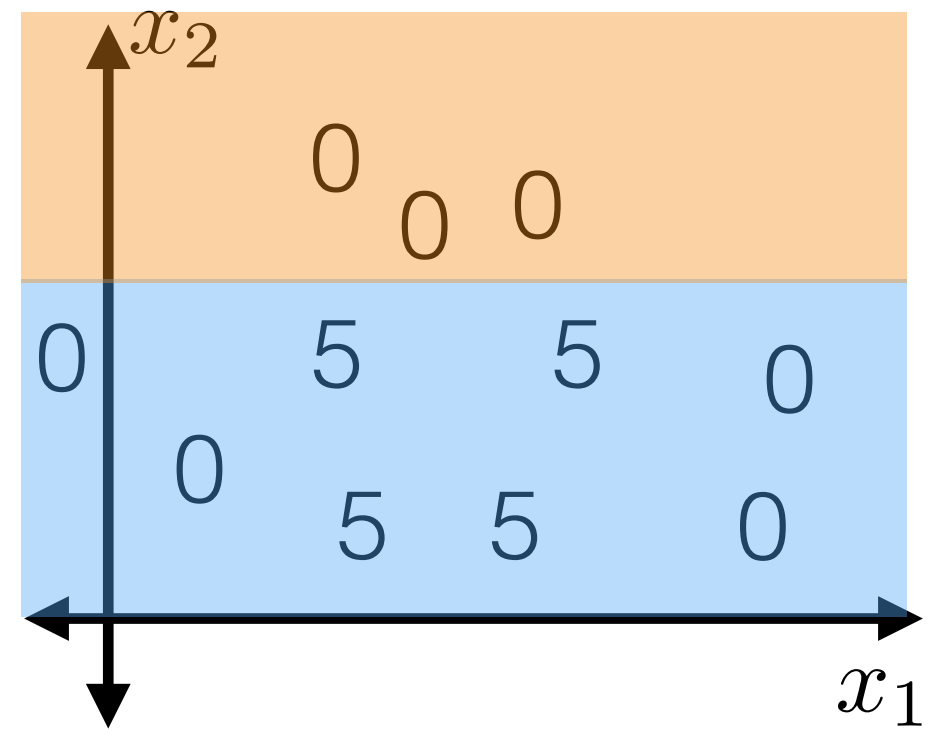
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



$$x_2 \geq 0.28$$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

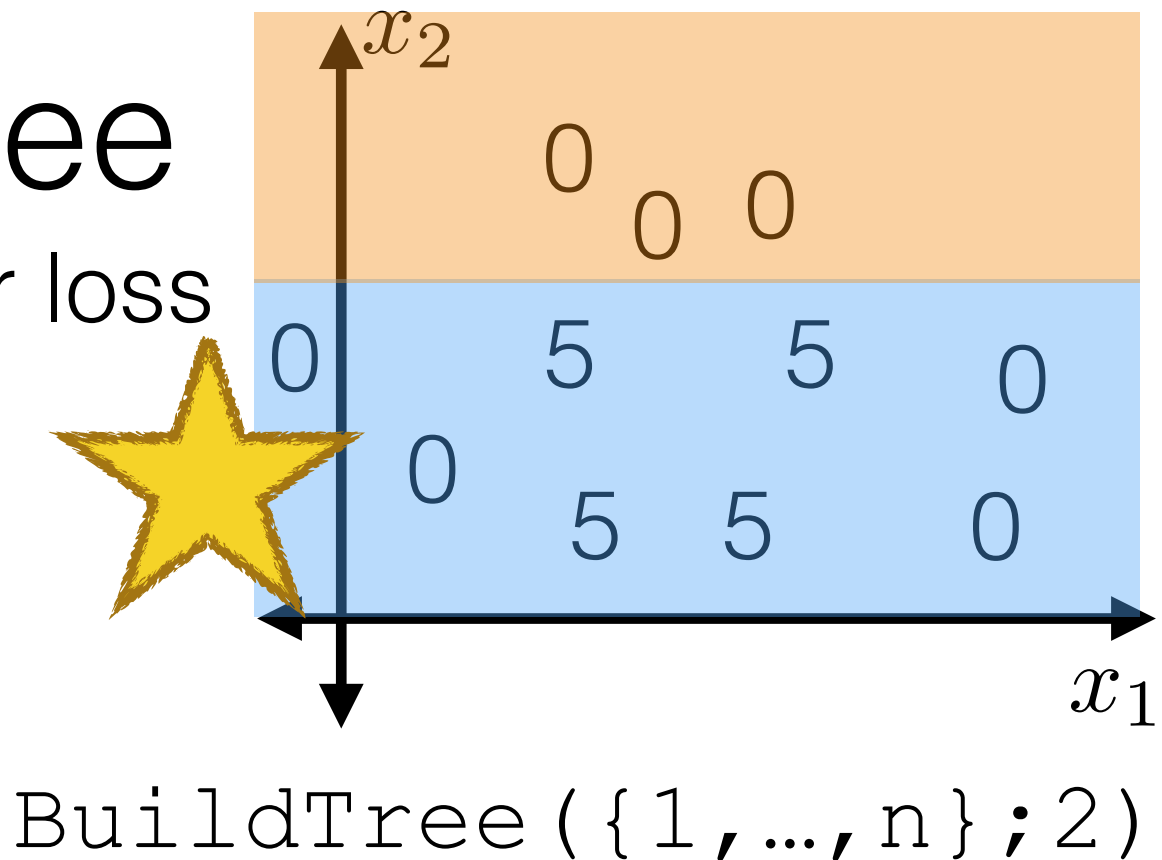
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

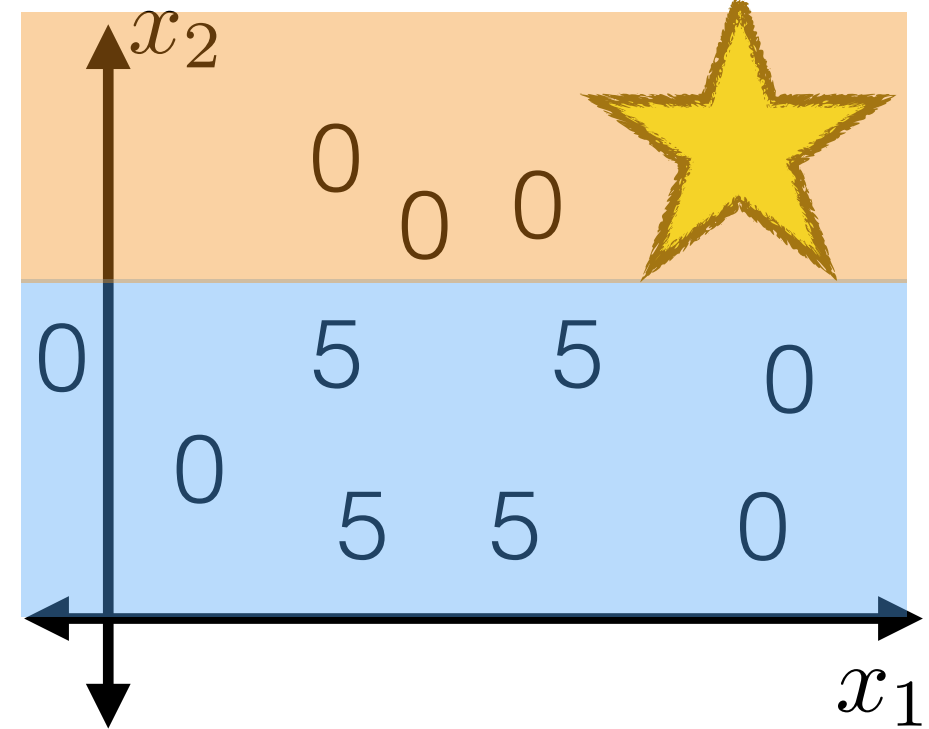
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

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



BuildTree($\{1, \dots, n\}; 2$)

$x_2 \geq 0.28$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

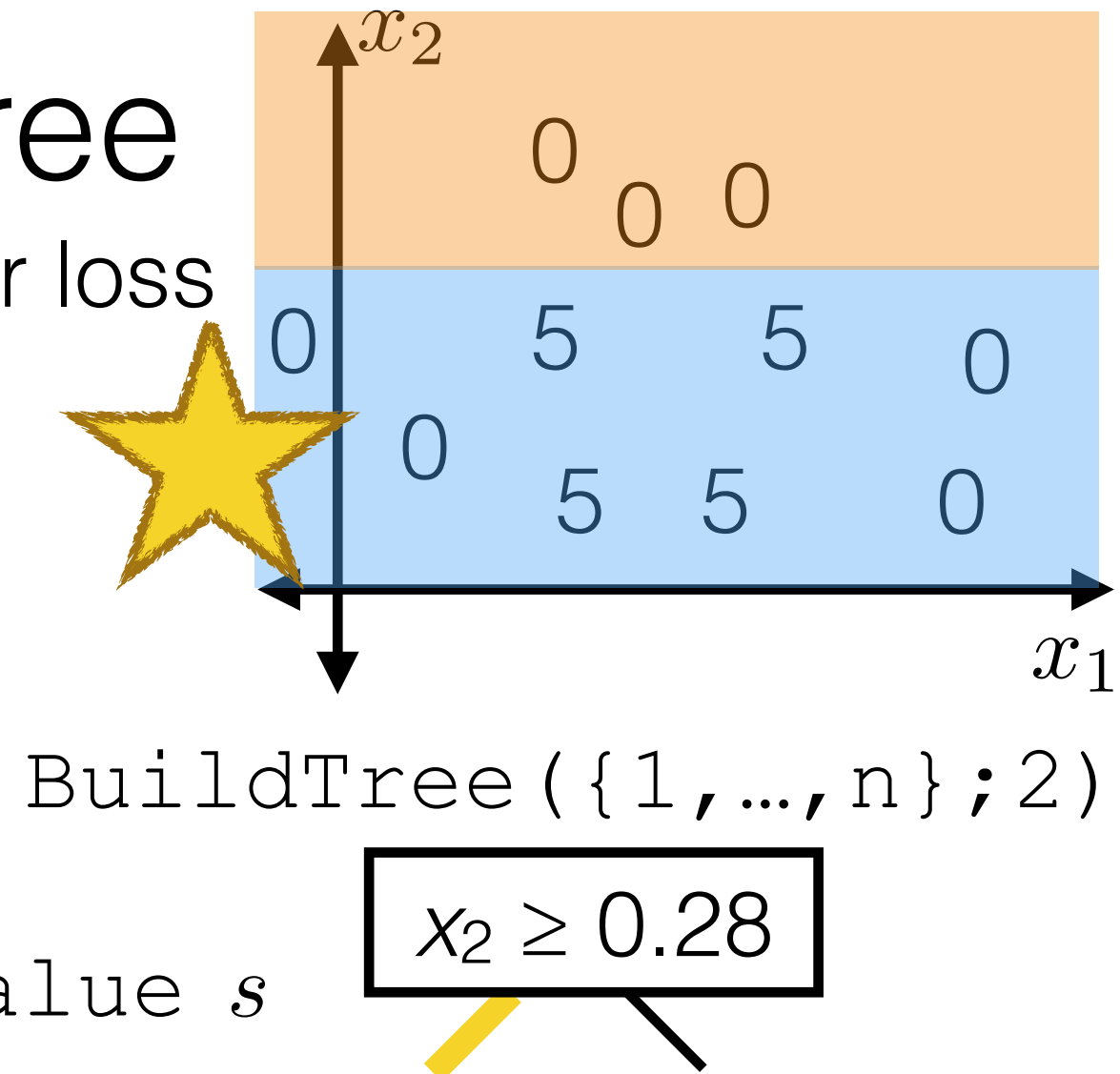
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

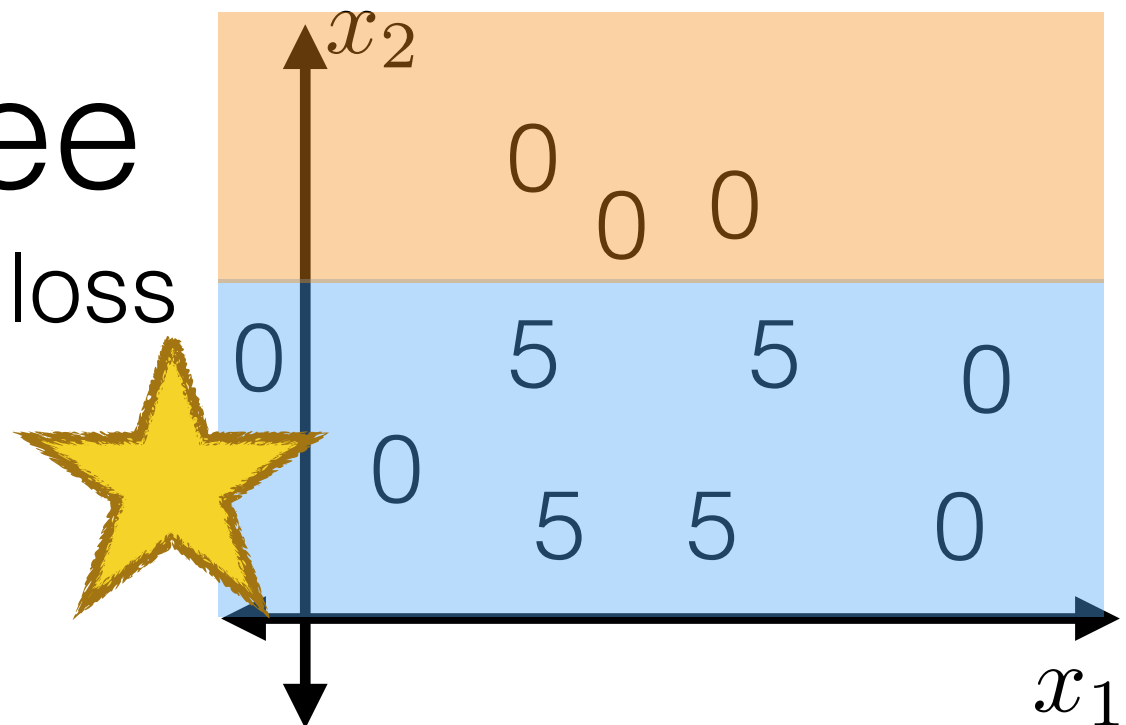
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)

$x_2 \geq 0.28$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

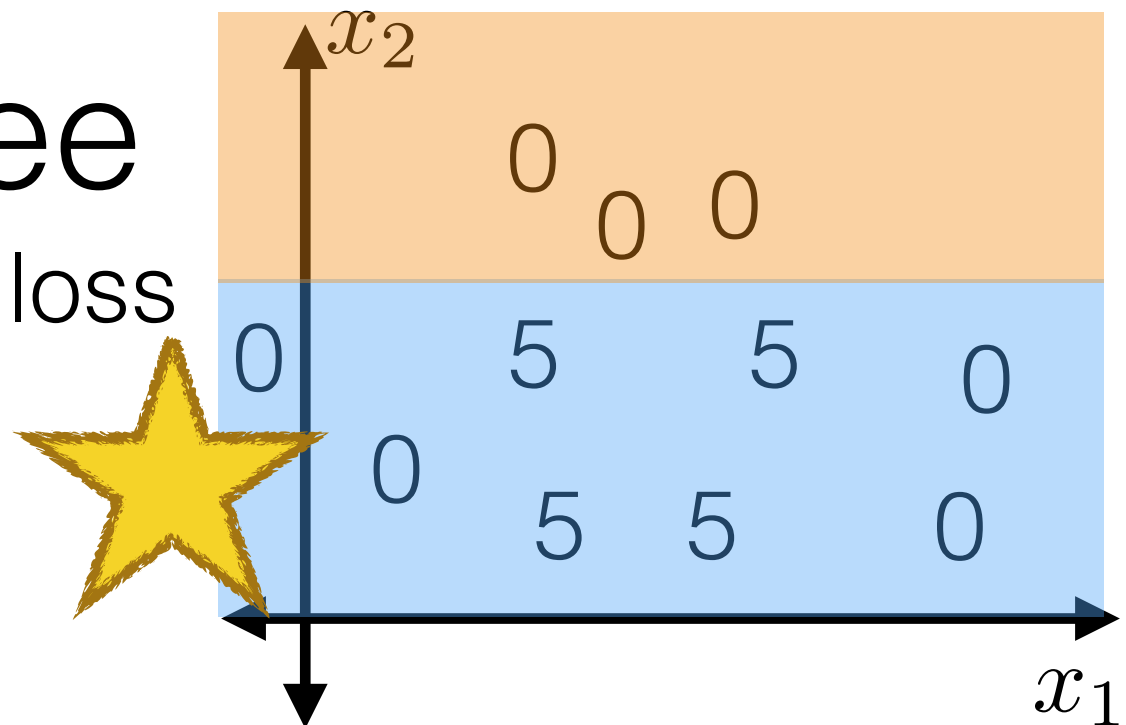
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

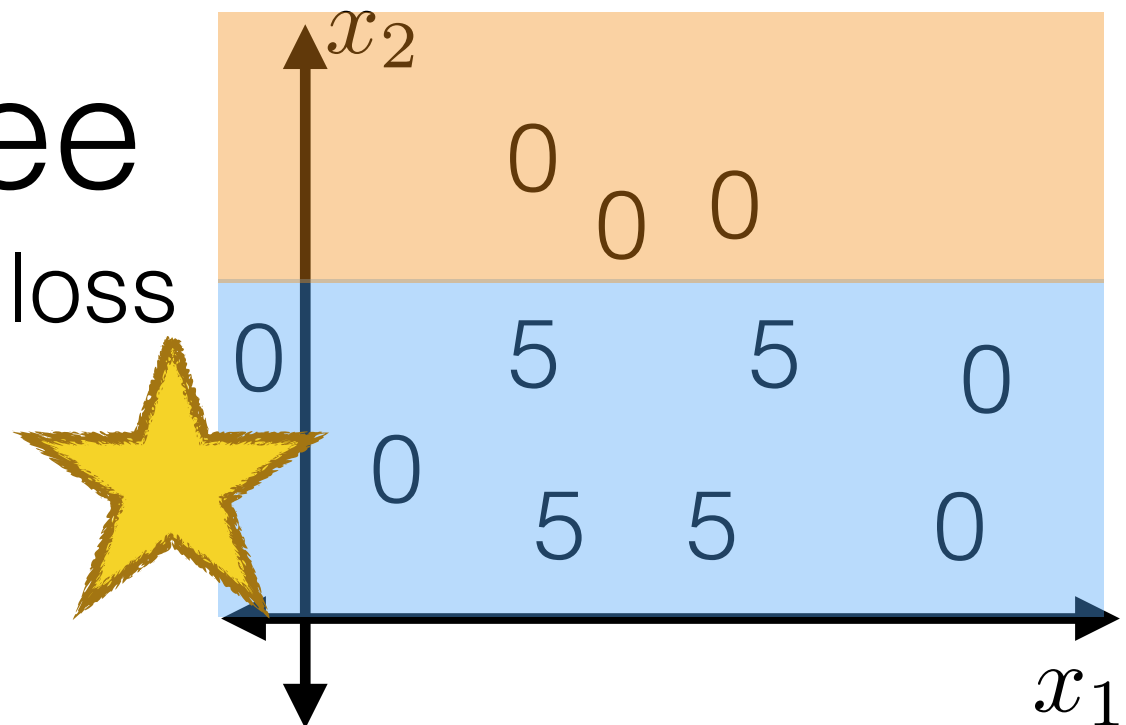
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)

$x_2 \geq 0.28$

$x_1 \geq 7.2$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

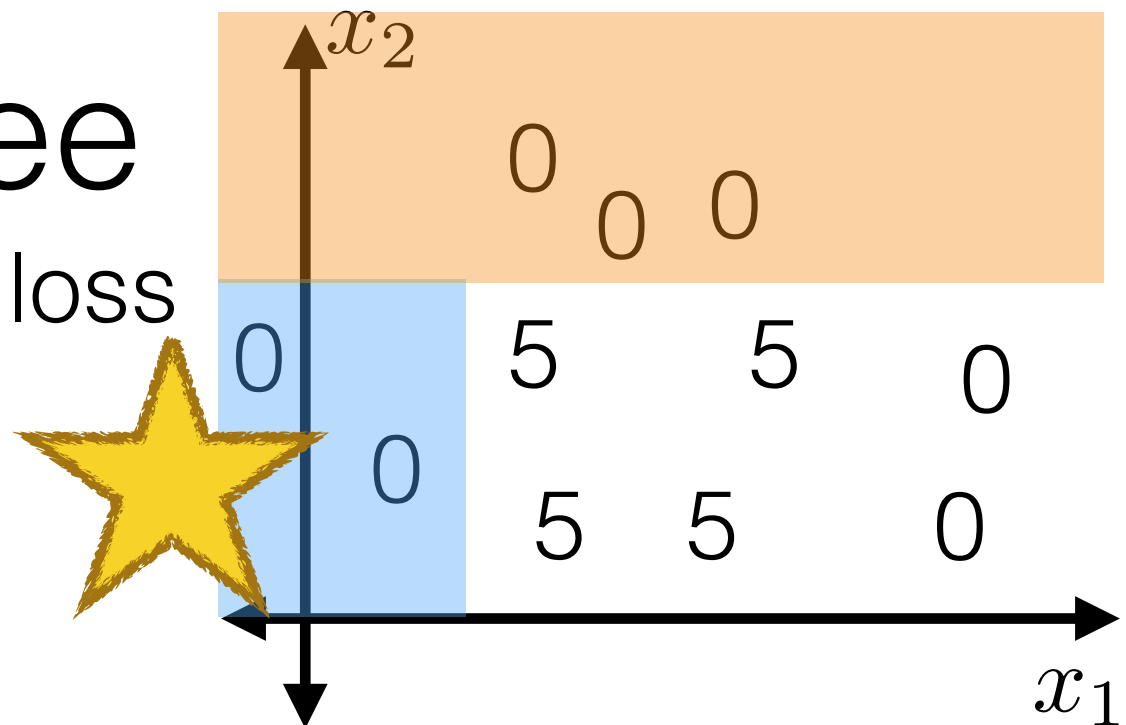
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)

$x_2 \geq 0.28$

$x_1 \geq 7.2$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

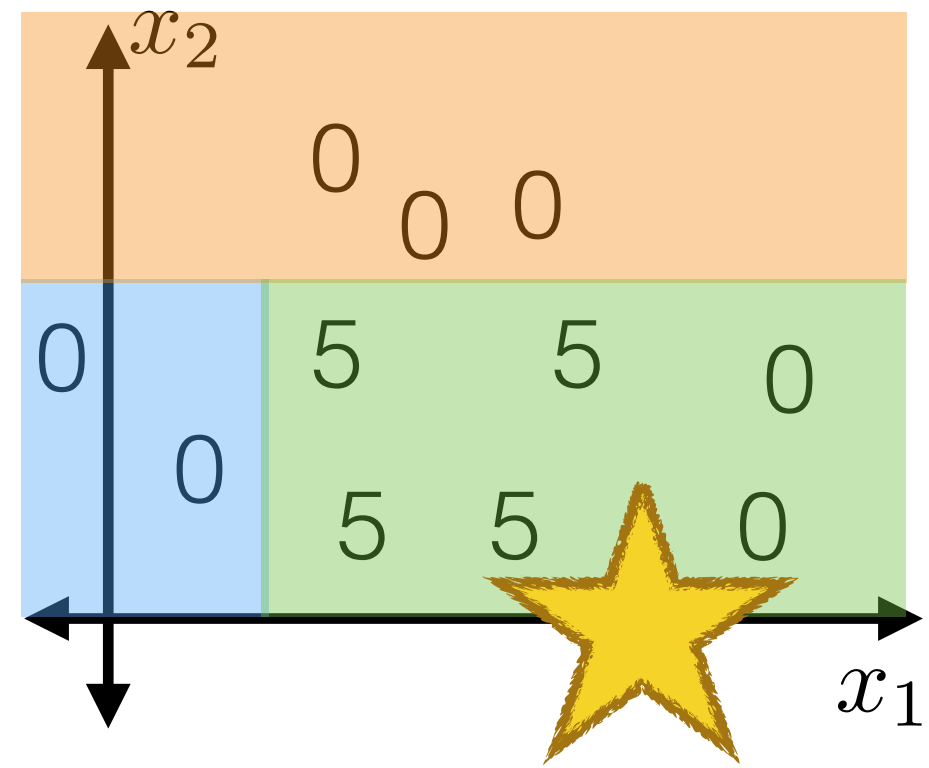
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)

$x_2 \geq 0.28$

$x_1 \geq 7.2$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

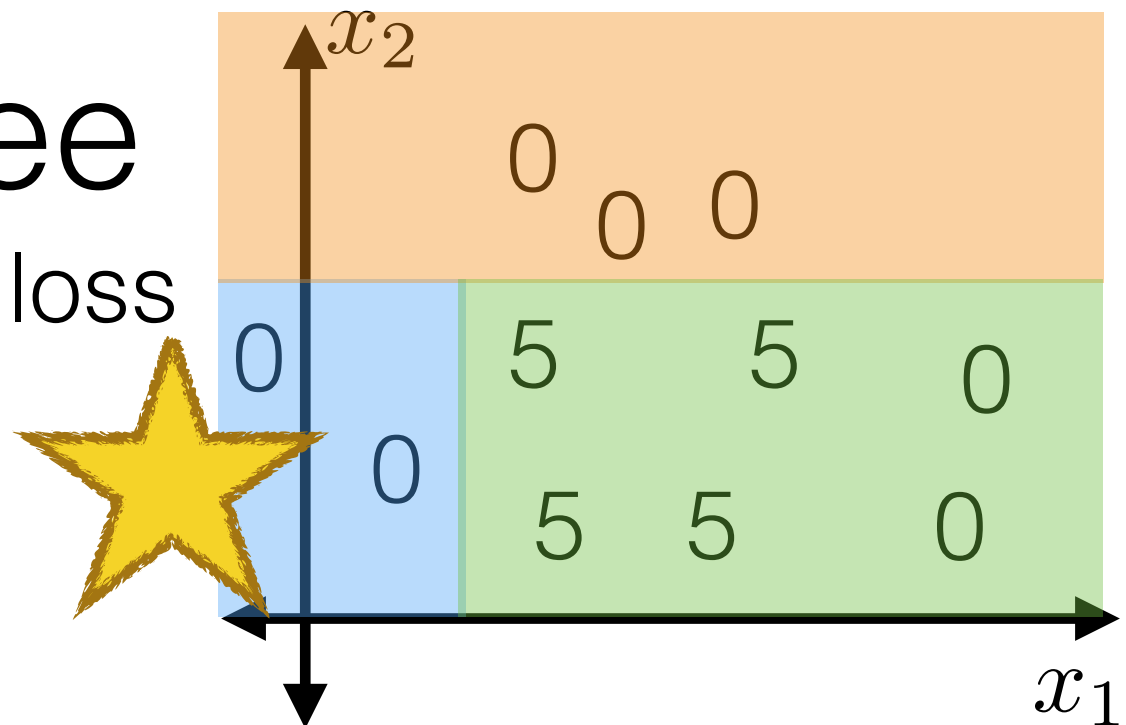
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)

$x_2 \geq 0.28$

$x_1 \geq 7.2$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

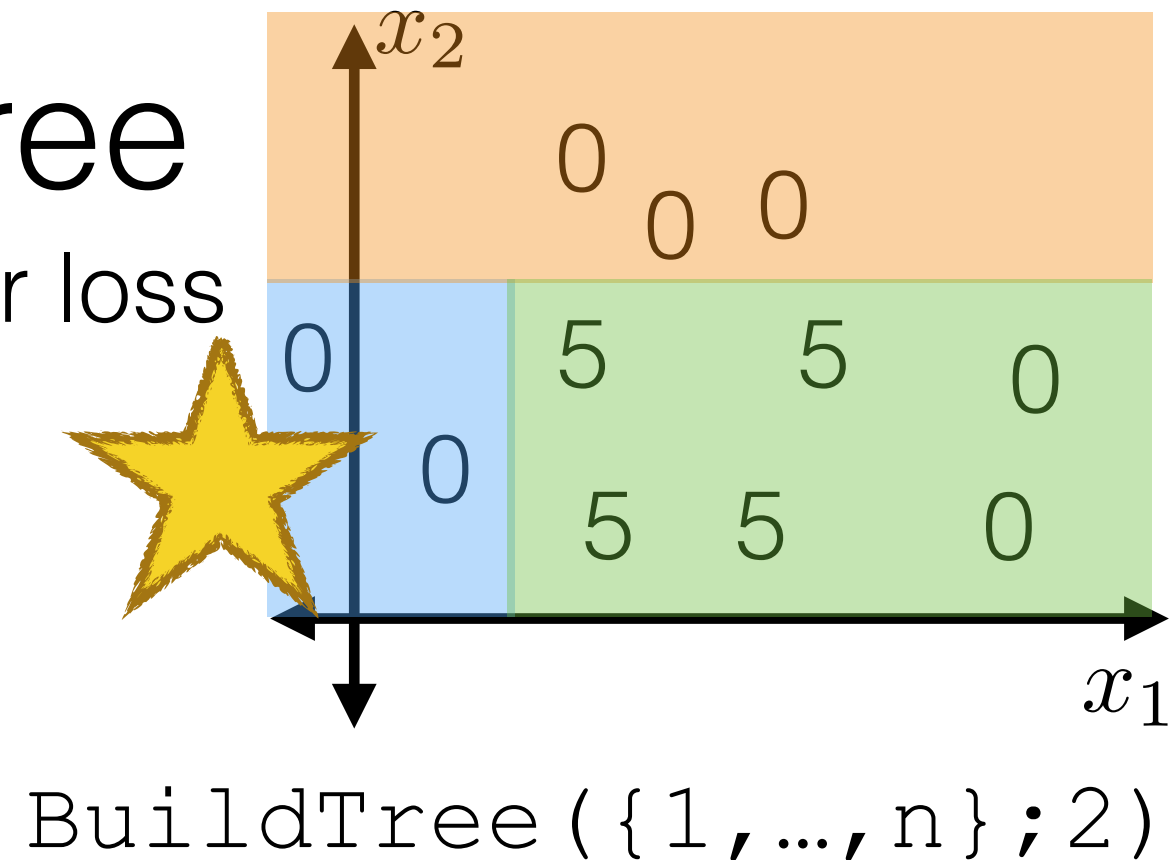
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

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



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

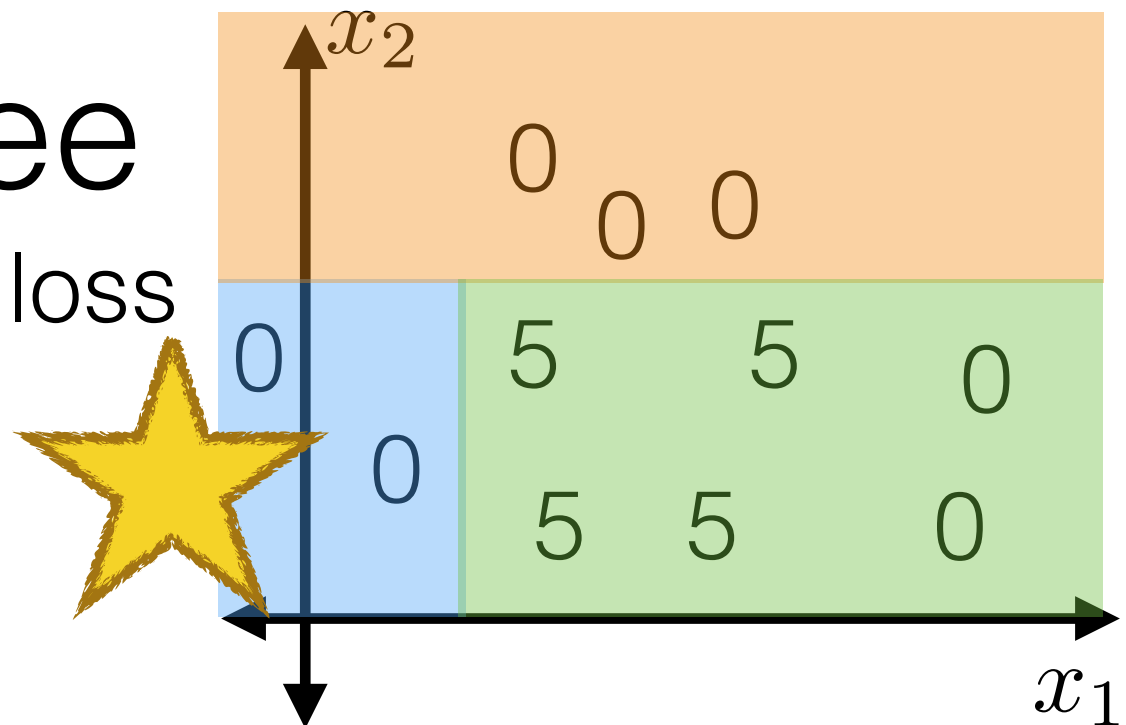
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)

$x_2 \geq 0.28$

$x_1 \geq 7.2$

Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

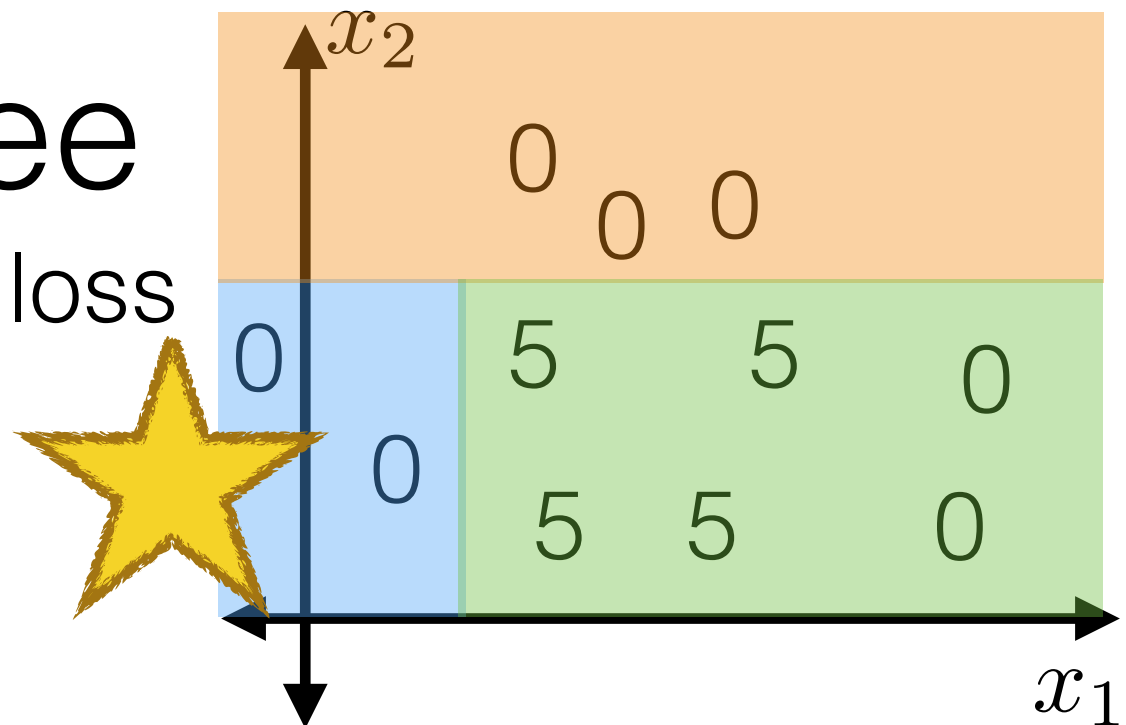
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

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

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

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



BuildTree($\{1, \dots, n\}; 2$)

$x_2 \geq 0.28$

$x_1 \geq 7.2$



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

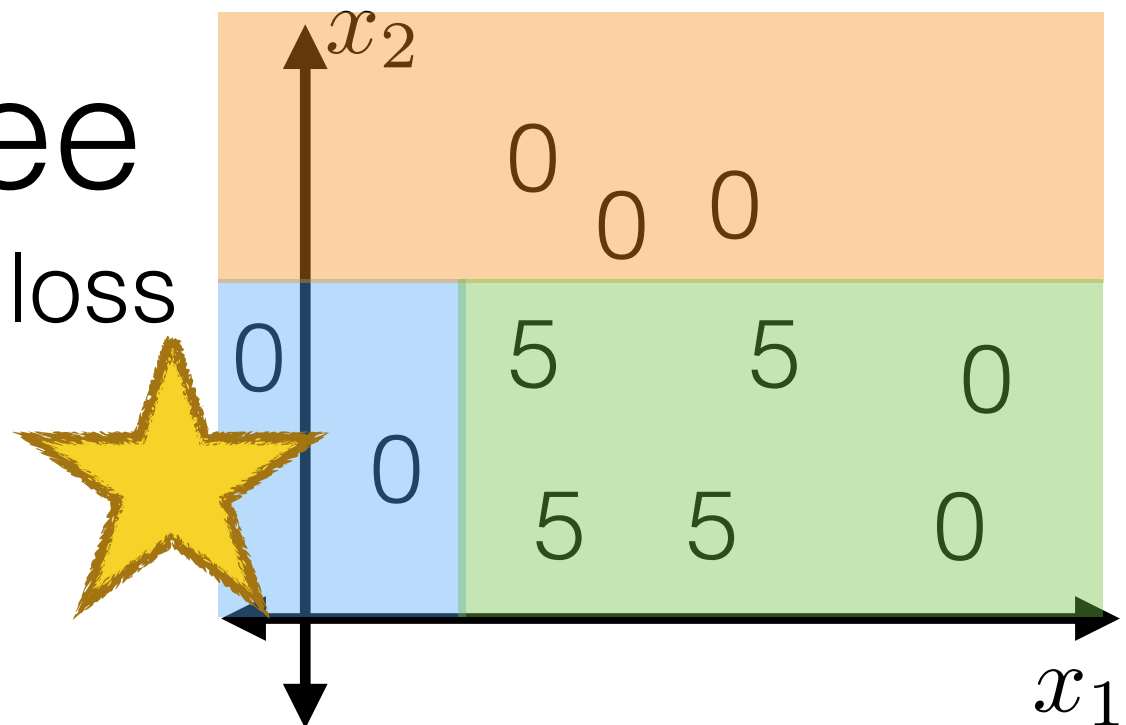
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

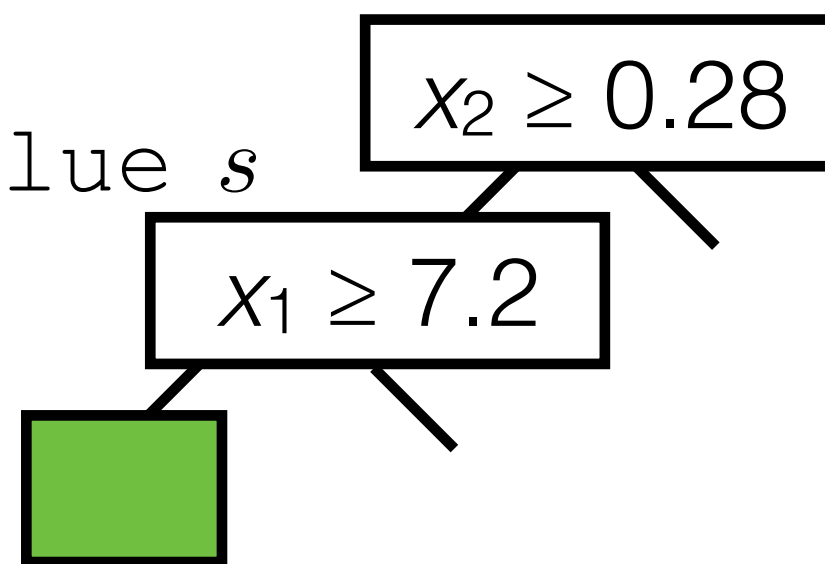
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

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



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

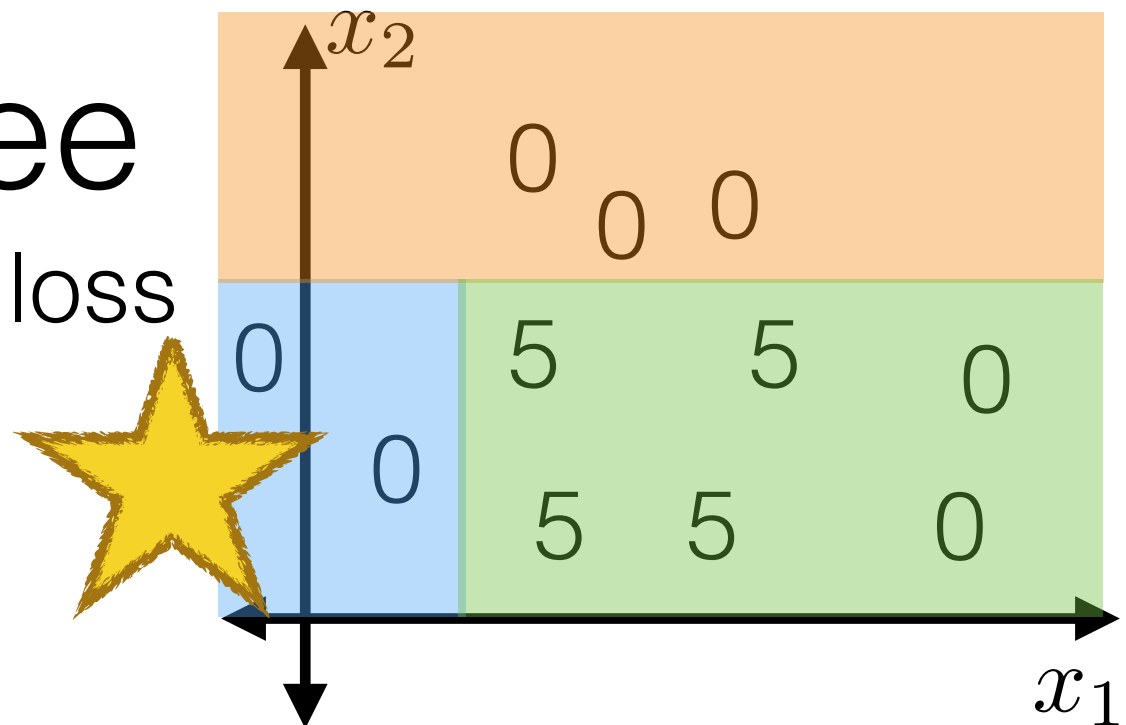
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

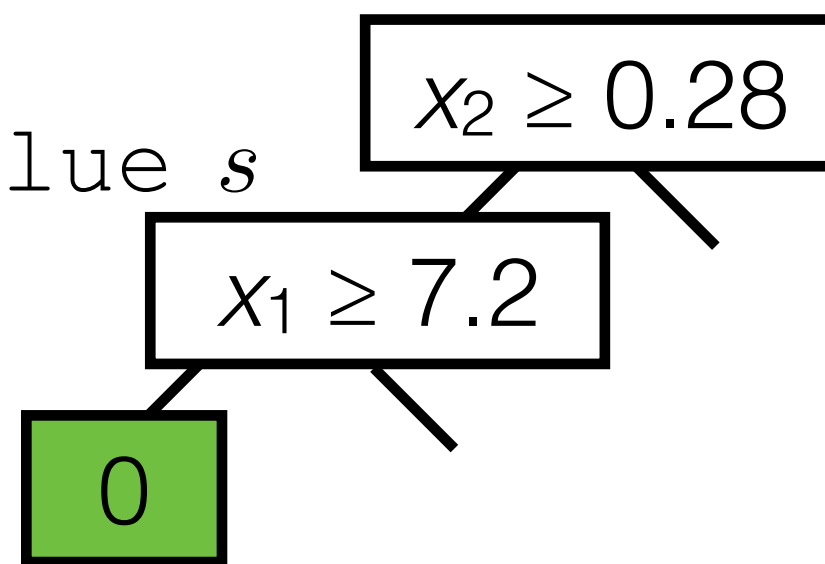
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

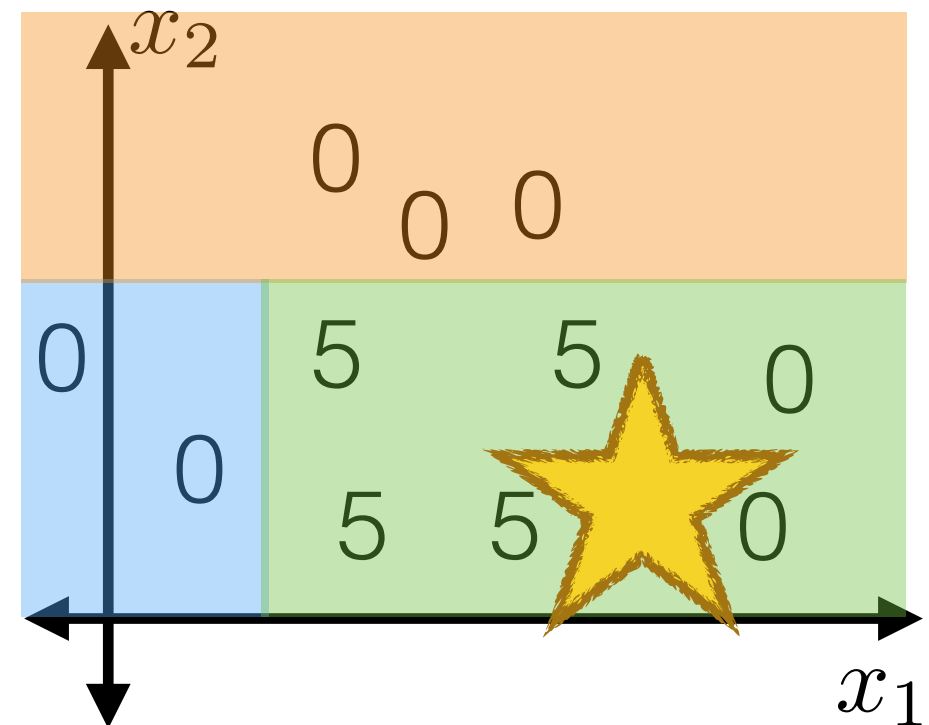
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

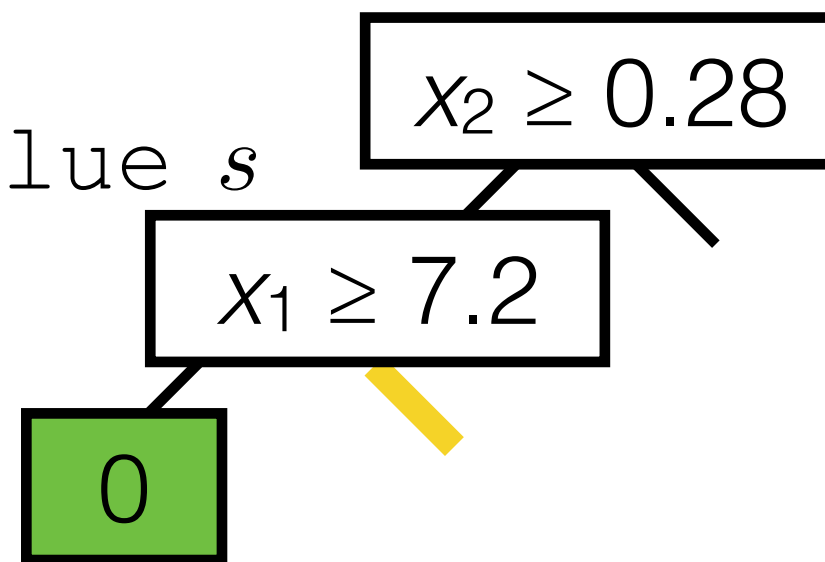
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

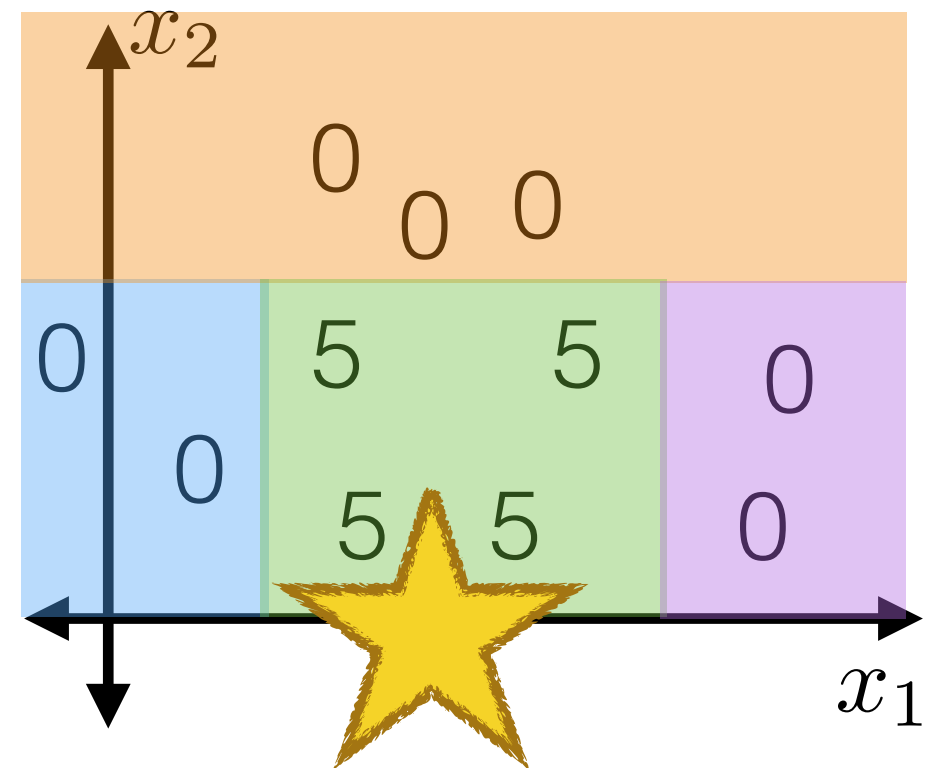
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

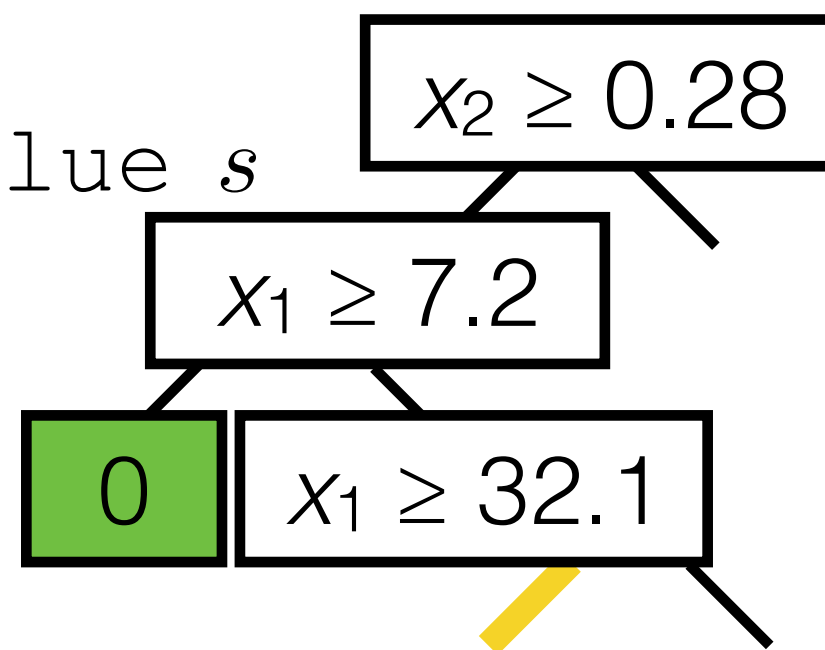
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

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



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

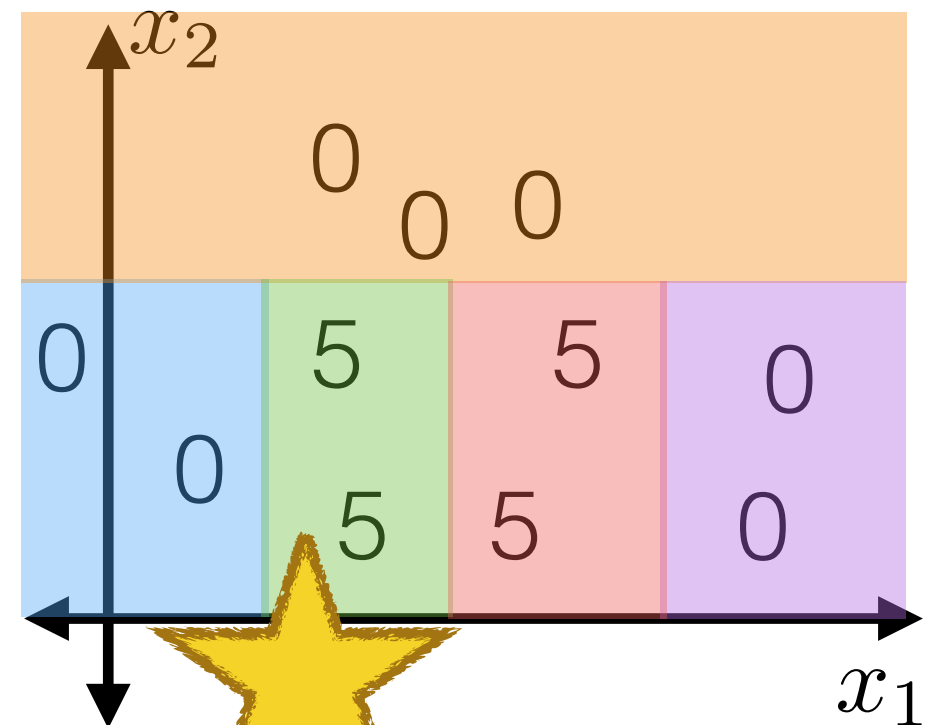
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

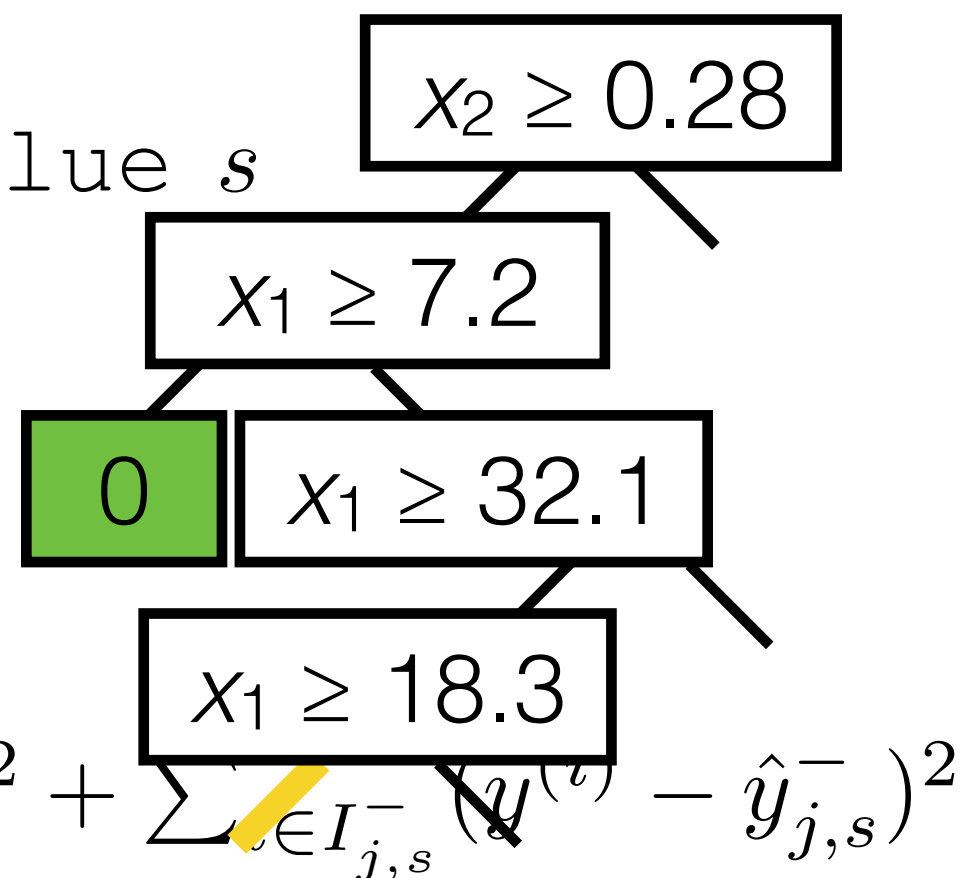
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

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



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

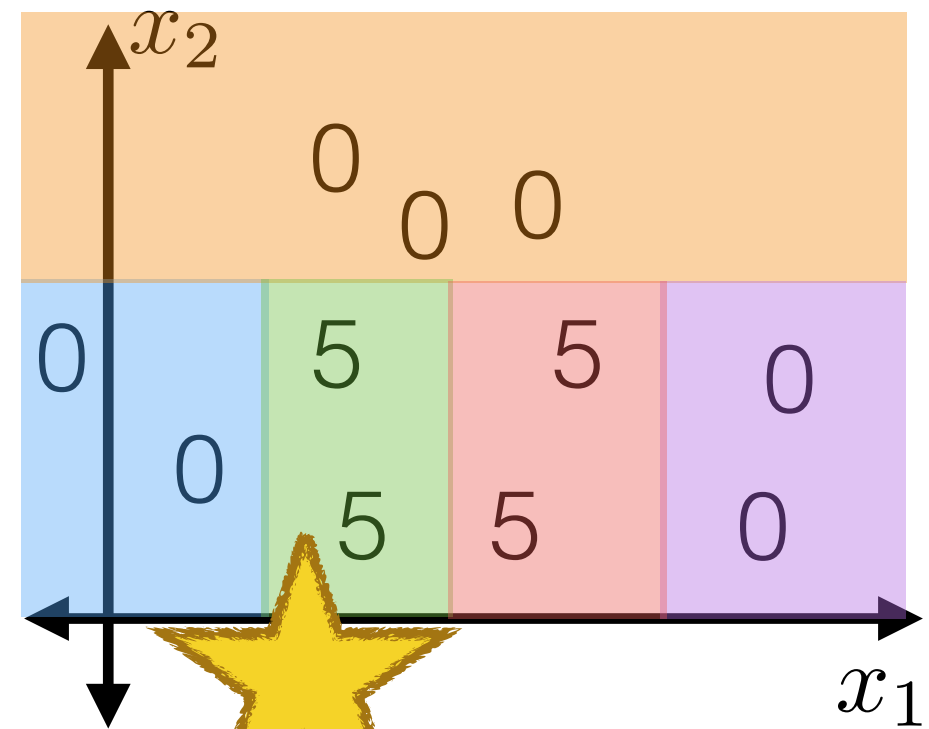
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

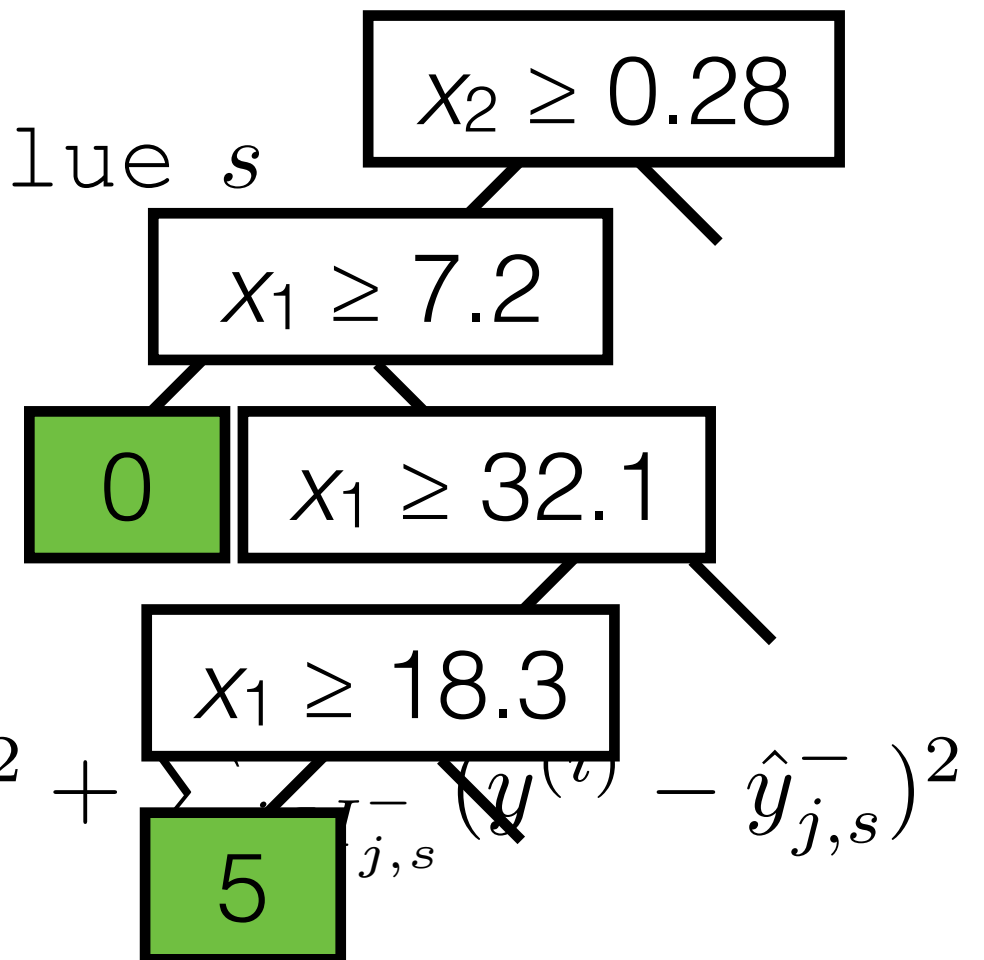
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

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



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

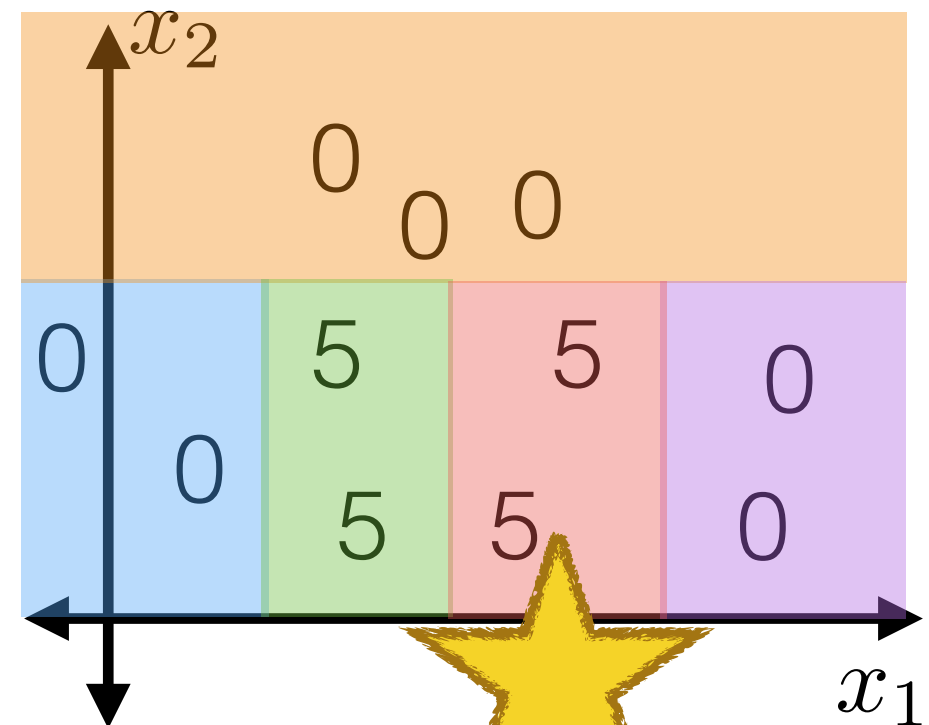
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

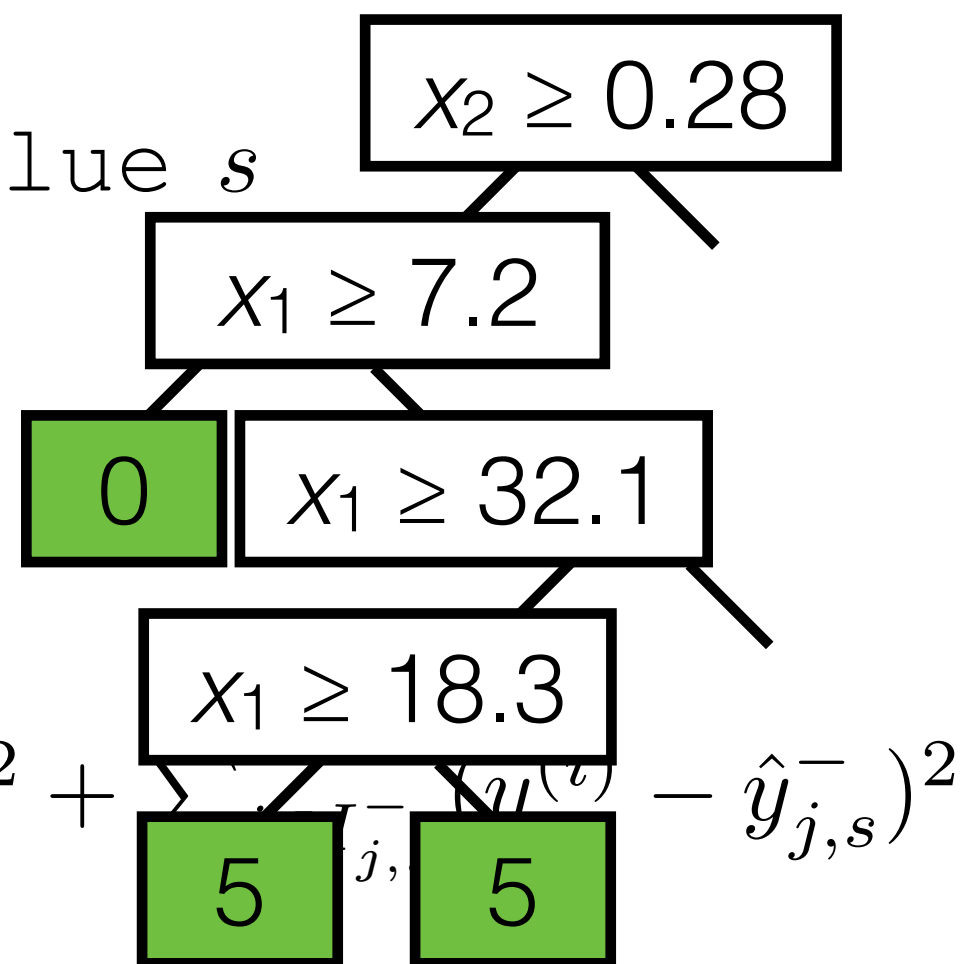
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

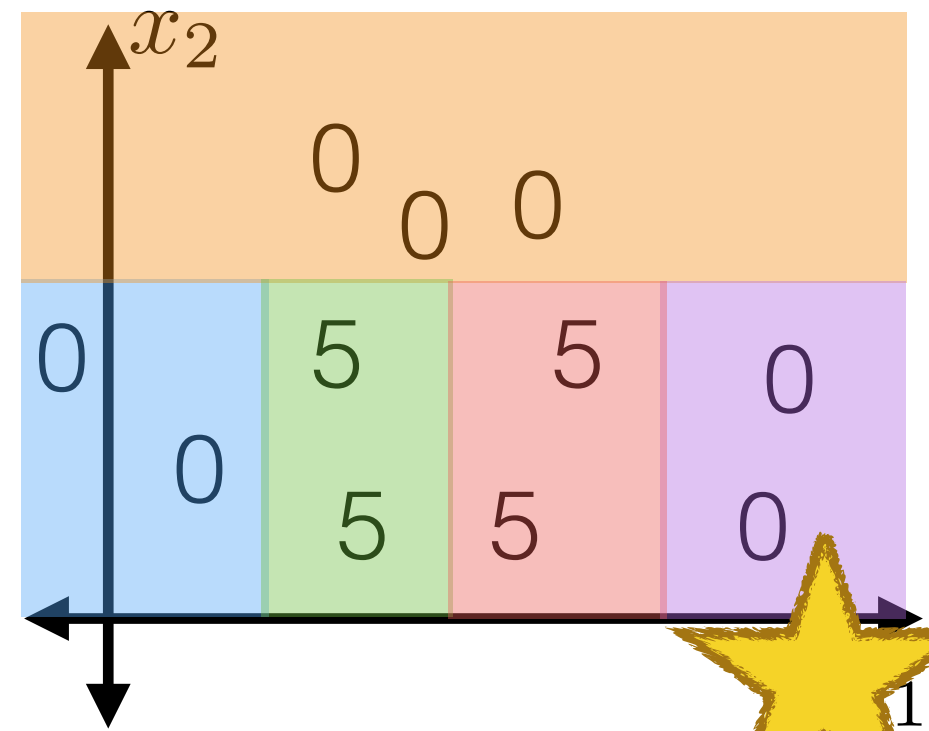
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

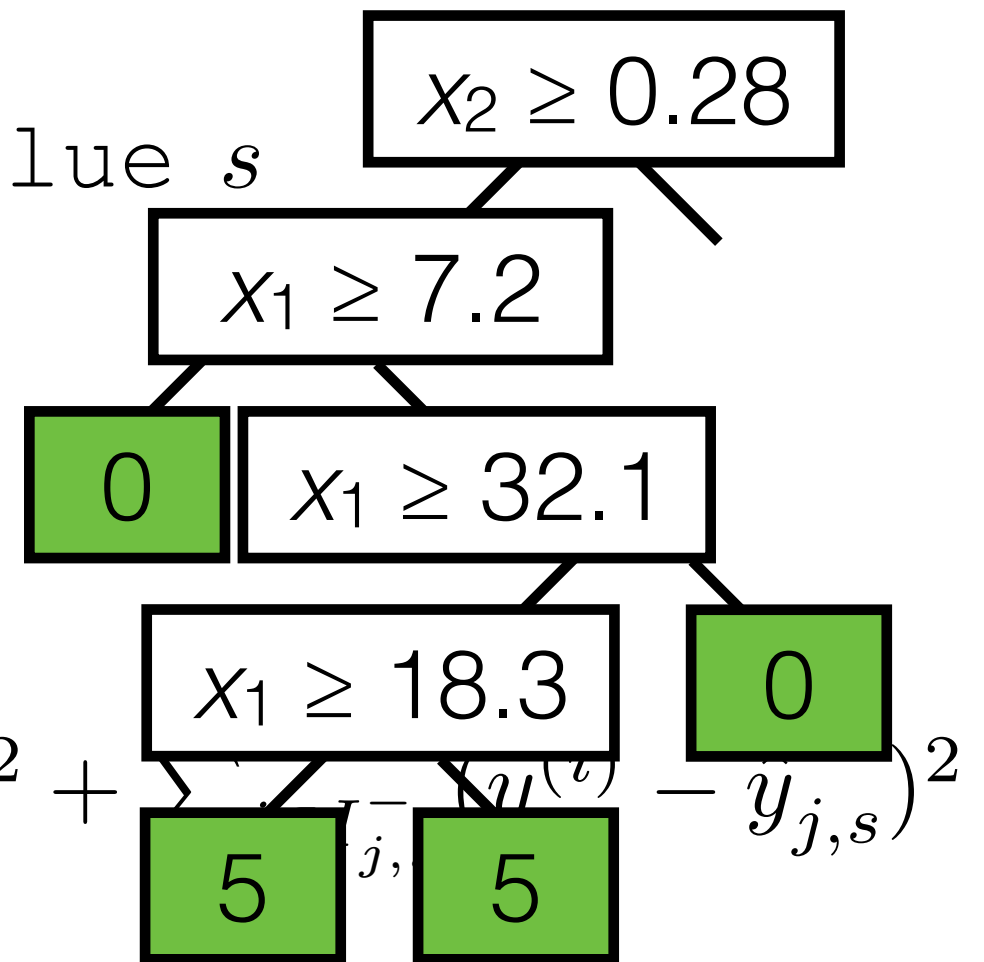
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

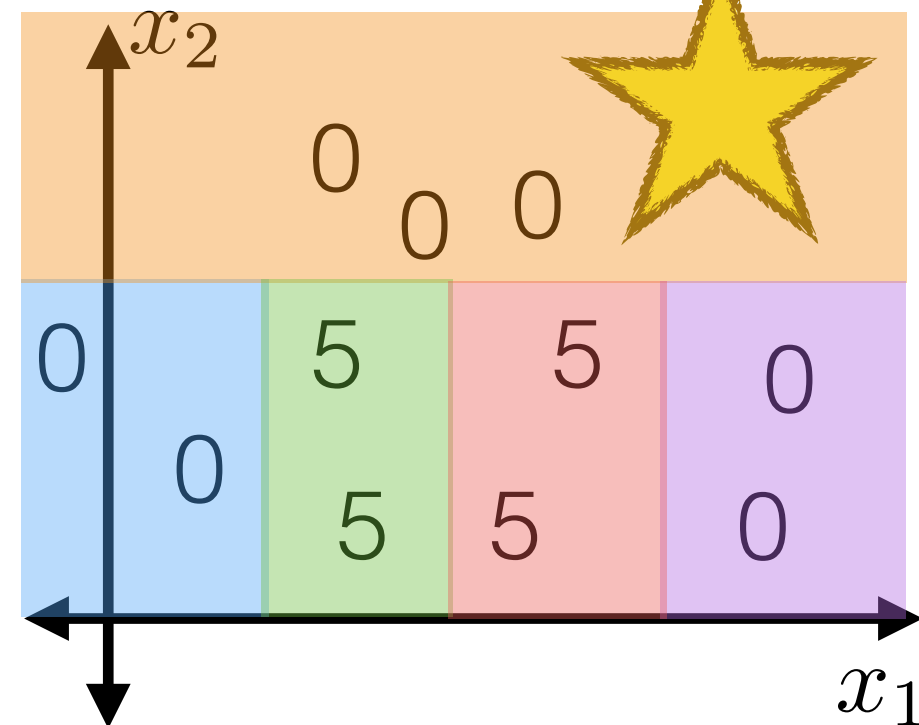
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

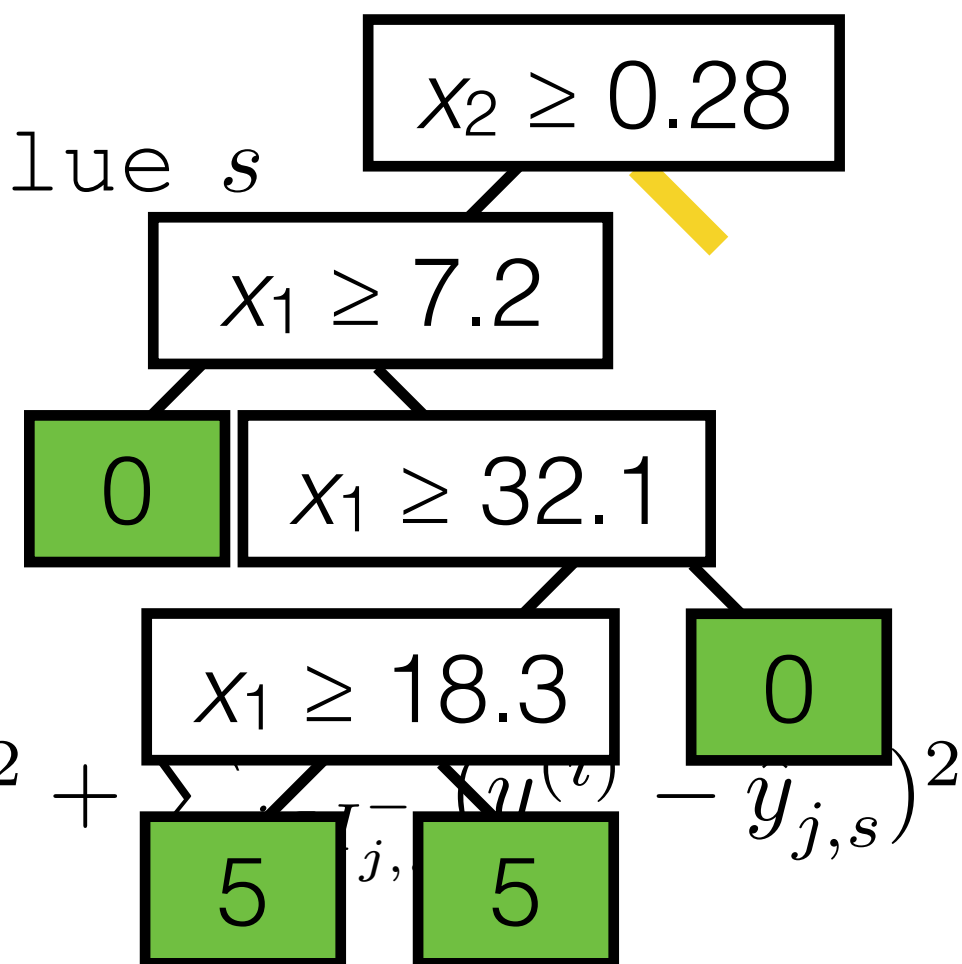
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

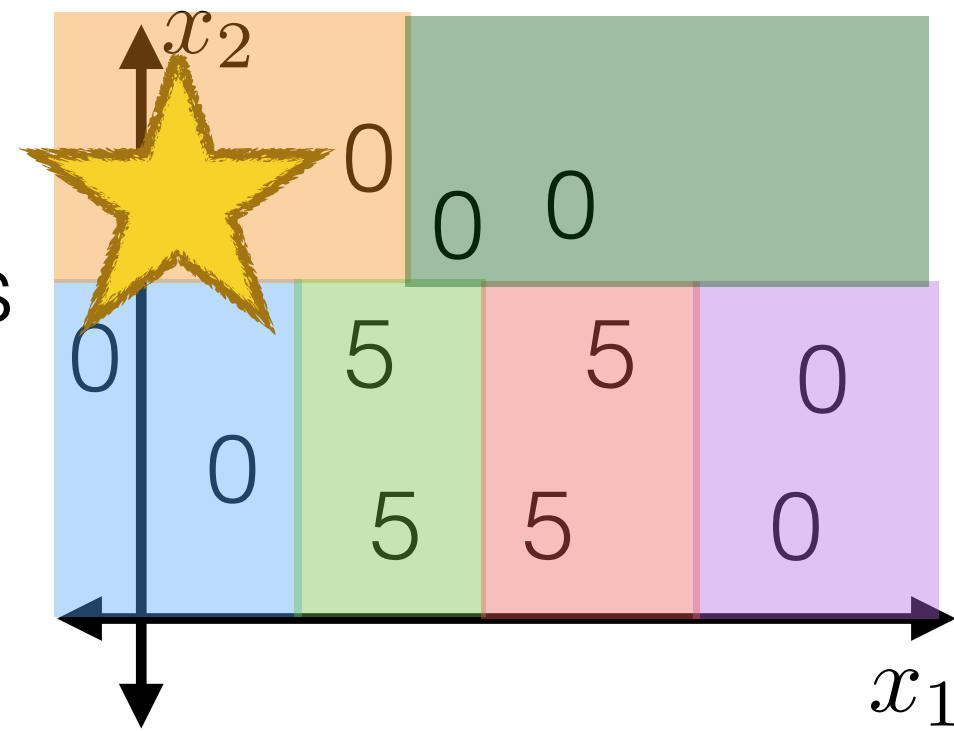
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

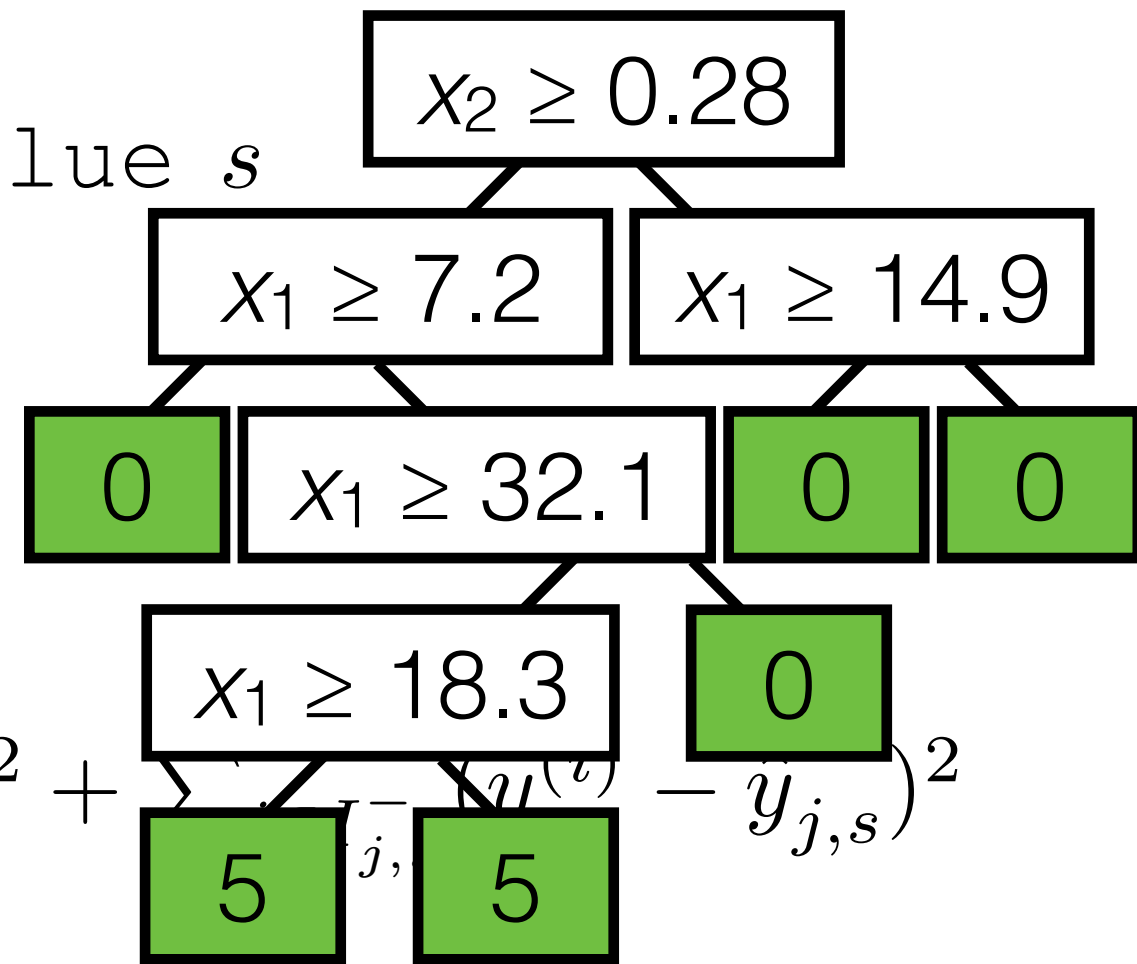
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

if $|I| \leq k$

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

return Leaf(label = \hat{y})

else

for each split dim j & value s

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

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

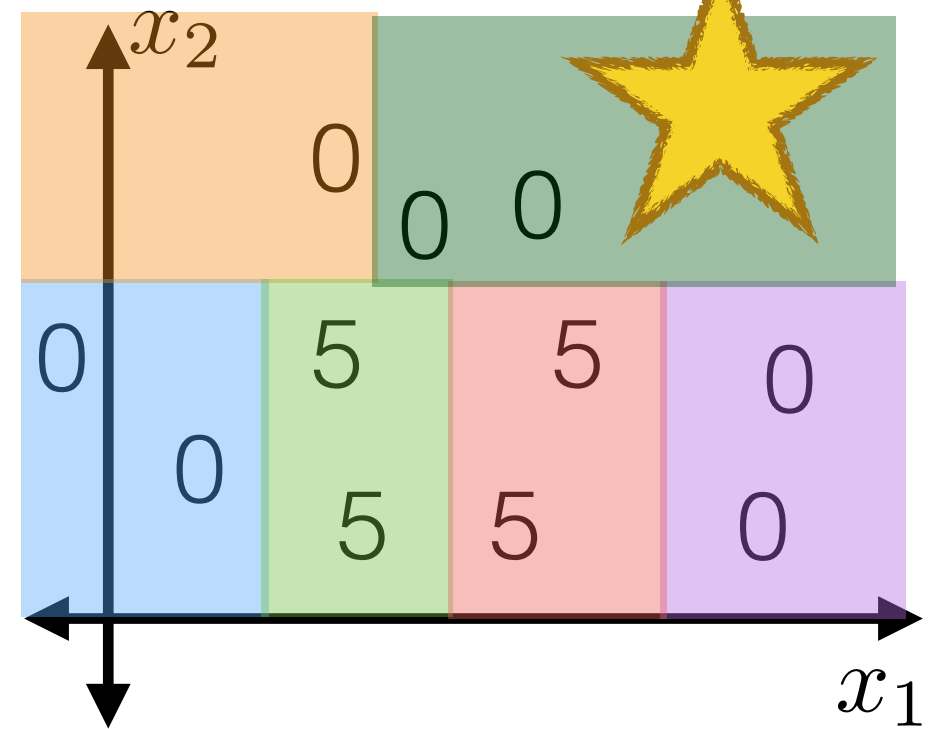
Set $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$

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

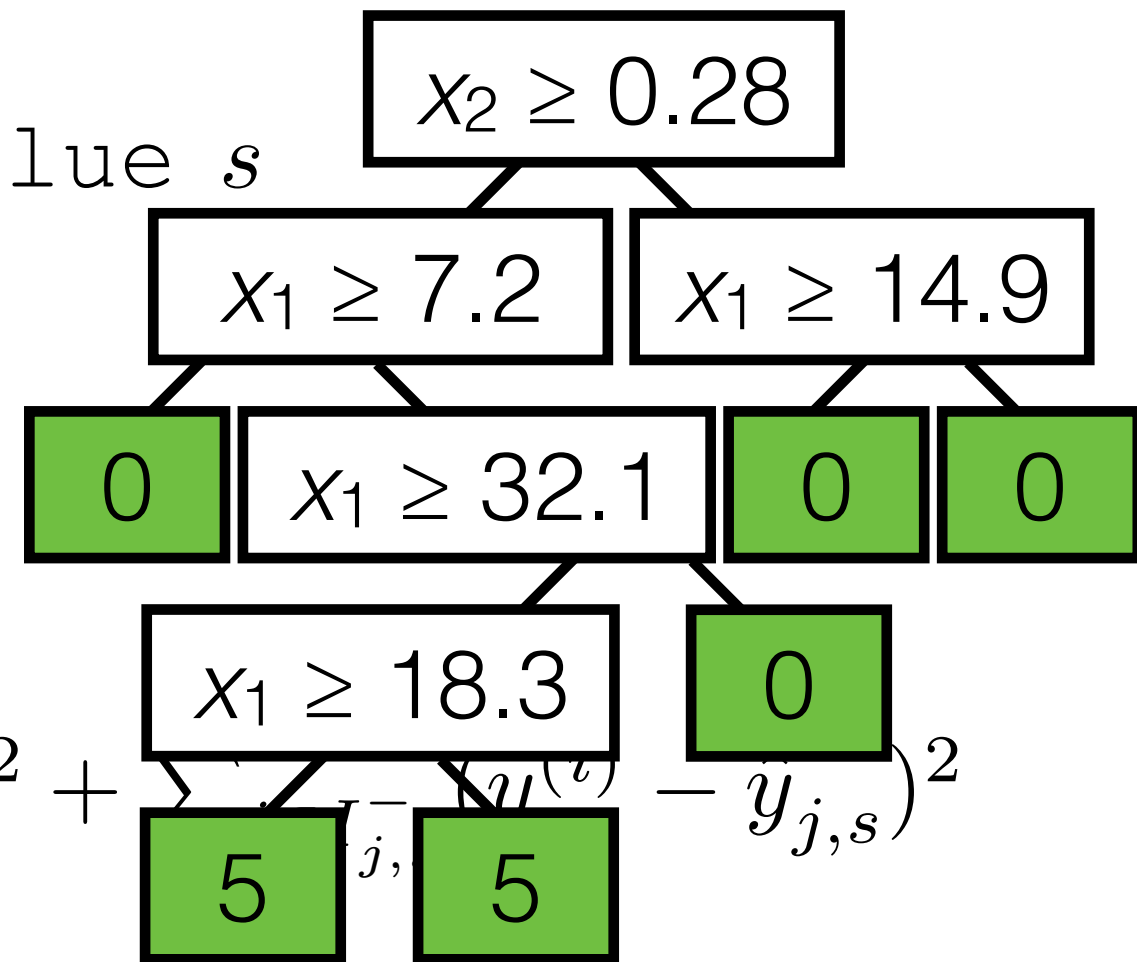
Set $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

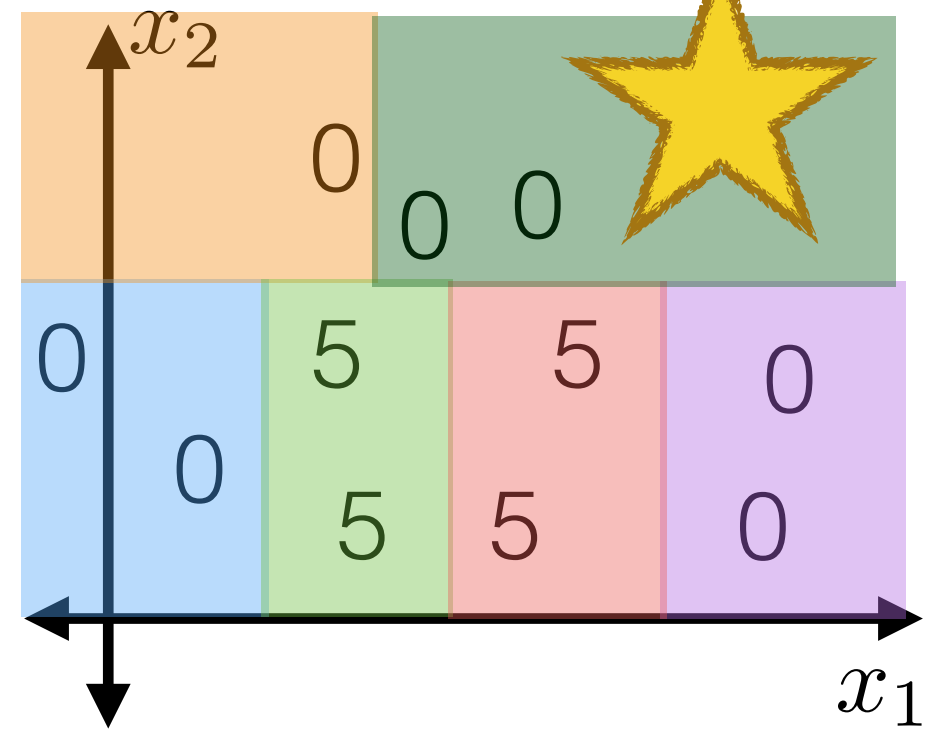
BuildTree($I; k$)

```

if |I| < k
    return Node(label =  $\hat{y}$ )
else

```

Why keep splitting after our error is low?



BuildTree($\{1, \dots, n\}; 2$)

for each split dim j & value s

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

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

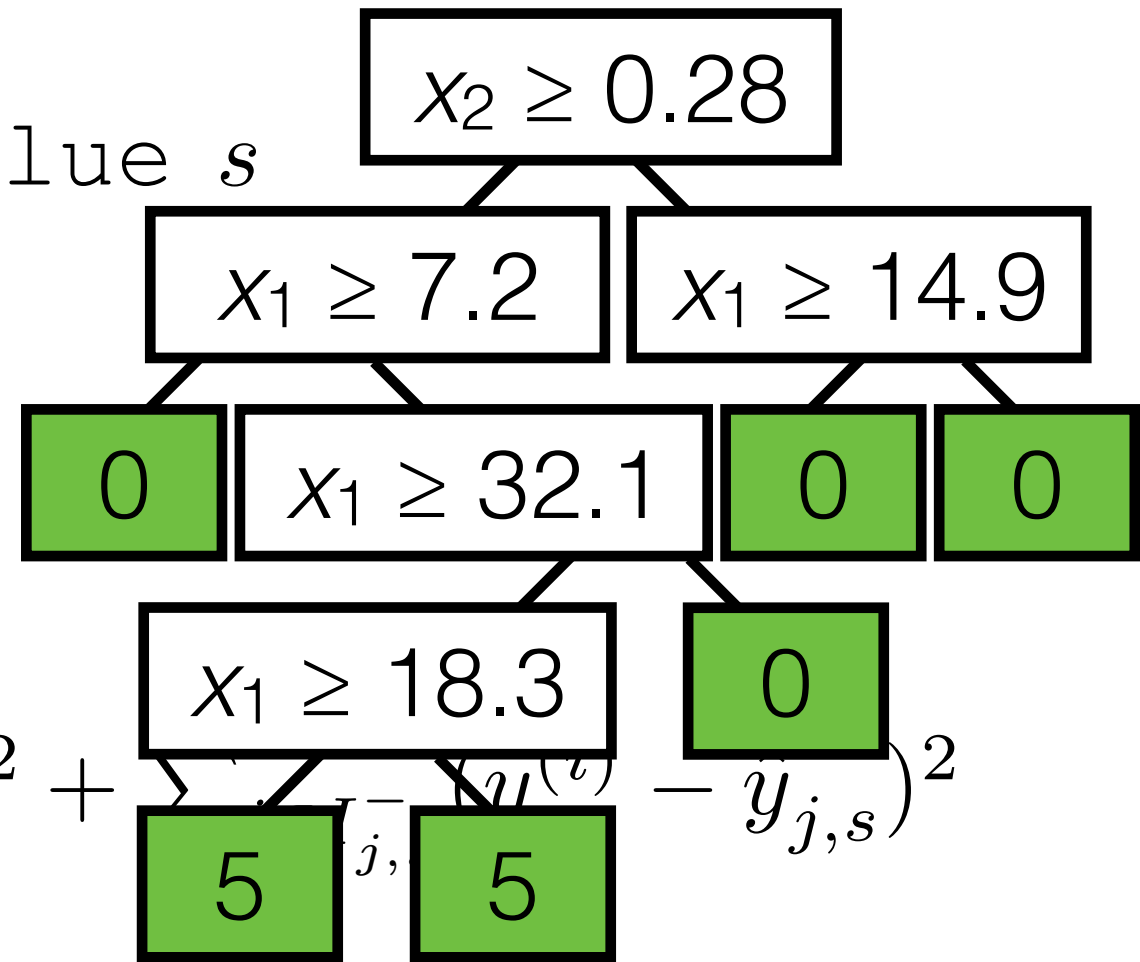
$$\text{Set } \hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$$

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

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

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

return Node(j^*, s^* , BuildTree(I_{j^*,s^*}^- , k), BuildTree(I_{j^*,s^*}^+ , k))



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

```

if  $|I| < k$ 
     $\hat{y} = \text{average}_{i \in I} y^{(i)}$ 
else

```

Why keep splitting after our error is low?

Is overfitting an issue?

```

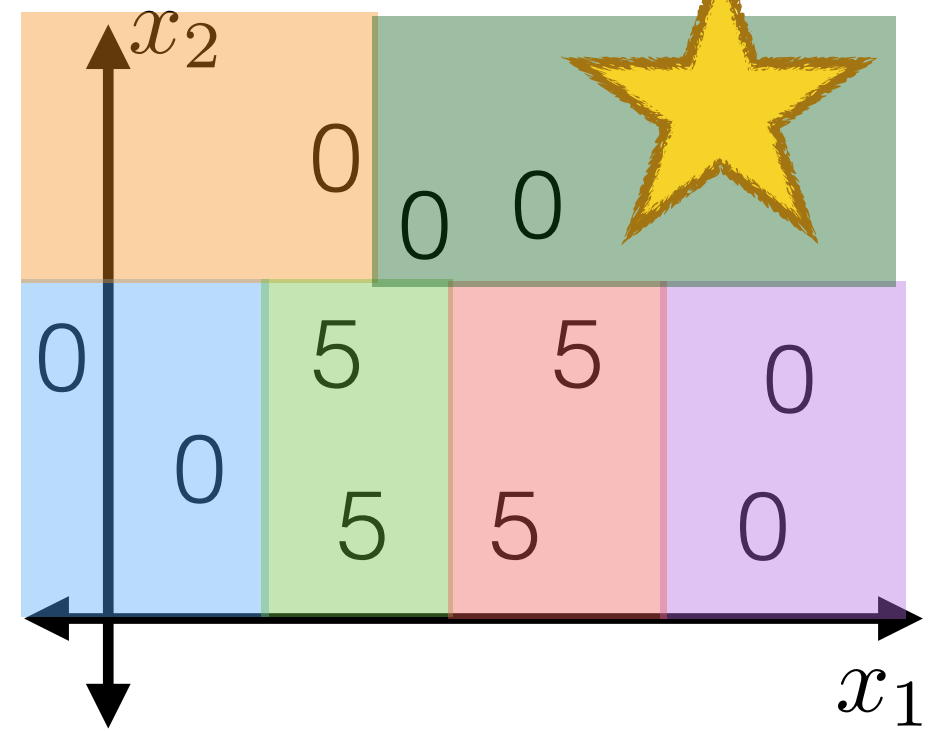
for  $j \in \{1, \dots, k\}$  & value  $s$ 
    Set  $I_{j,s}^- = \{i \in I \mid x_j^{(i)} < s\}$ 
    Set  $I_{j,s}^+ = \{i \in I \mid x_j^{(i)} \geq s\}$ 
    Set  $\hat{y}_{j,s}^+ = \text{average}_{i \in I_{j,s}^+} y^{(i)}$ 
    Set  $\hat{y}_{j,s}^- = \text{average}_{i \in I_{j,s}^-} y^{(i)}$ 
    Set  $E_{j,s} = \sum_{i \in I_{j,s}^+} (y^{(i)} - \hat{y}_{j,s}^+)^2 + \sum_{i \in I_{j,s}^-} (y^{(i)} - \hat{y}_{j,s}^-)^2$ 
    Set  $(j^*, s^*) = \arg \min E_{j,s}$ 

```

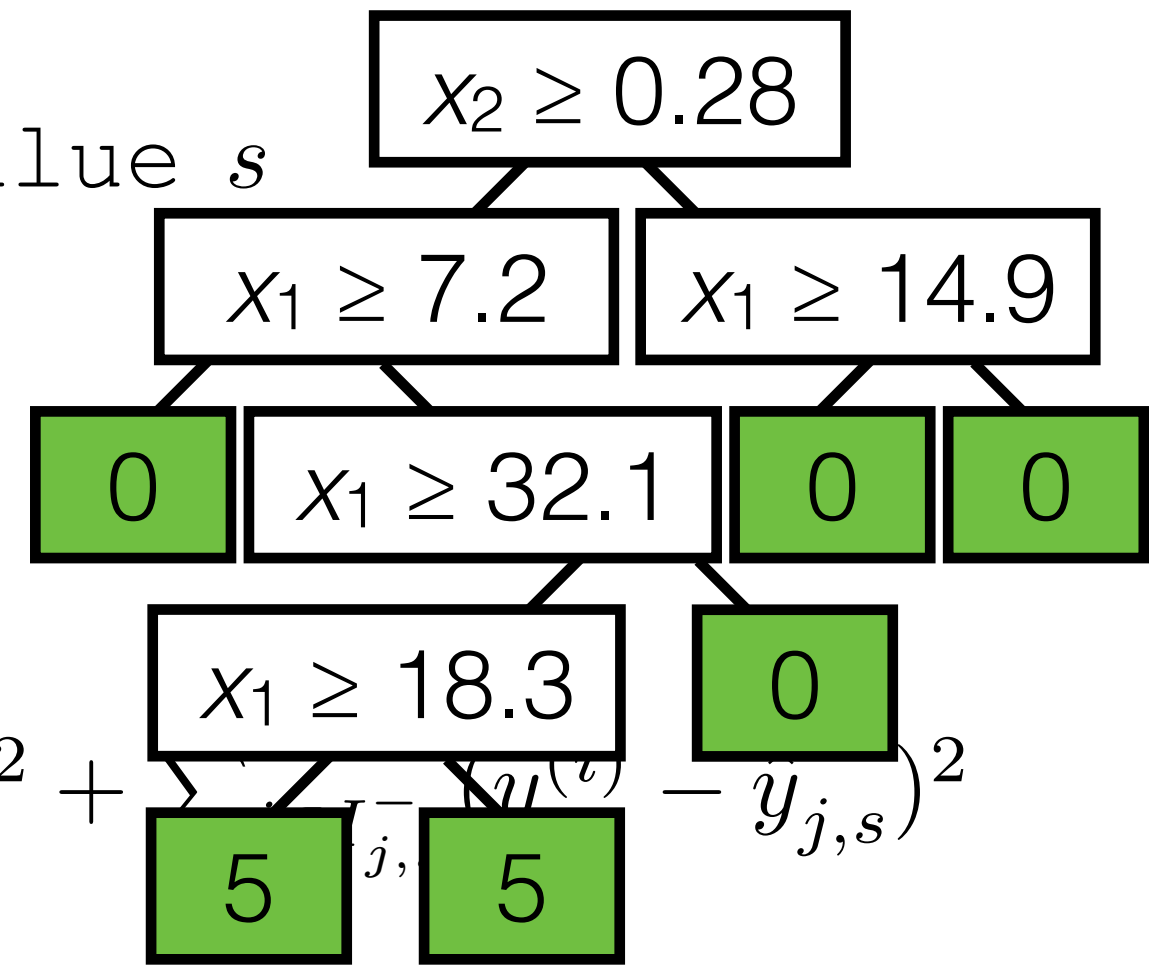
```

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

```



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

```

if |I| < k
    return Node(I, label =  $\hat{y}$ )
else

```

Why keep splitting after our error is low?

```

for  $j \in \{1, \dots, k-1\}$  & value  $s$ 
    Set  $I_{j,s}^+ = \{i \in I \mid x^{(j)} \geq s\}$ 
    Set  $I_{j,s}^- = \{i \in I \mid x^{(j)} < s\}$ 

```

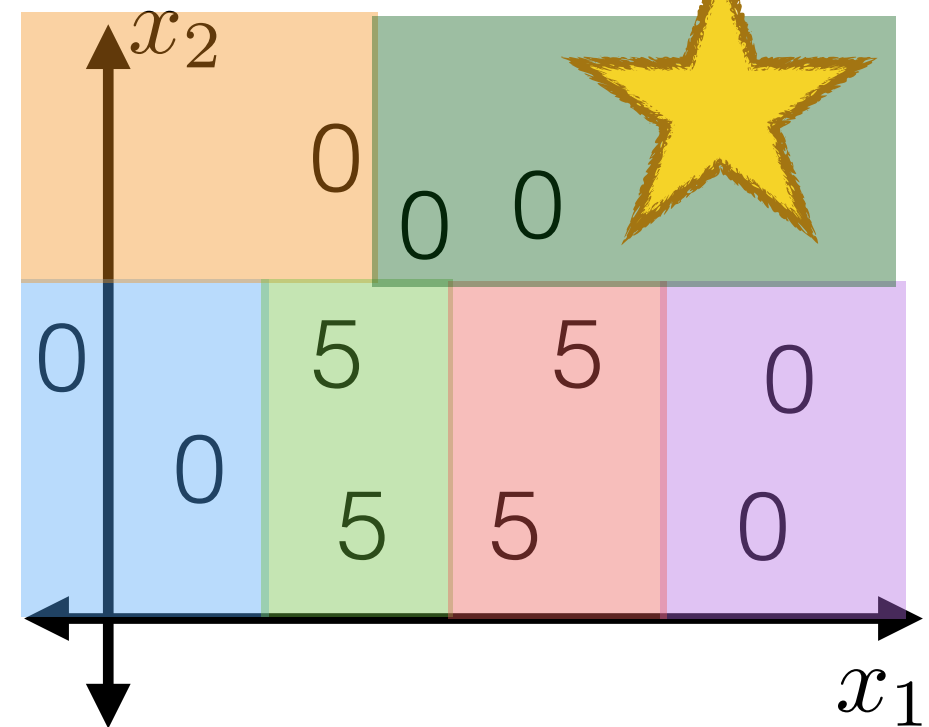
Idea: stop when no/ small change in loss

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

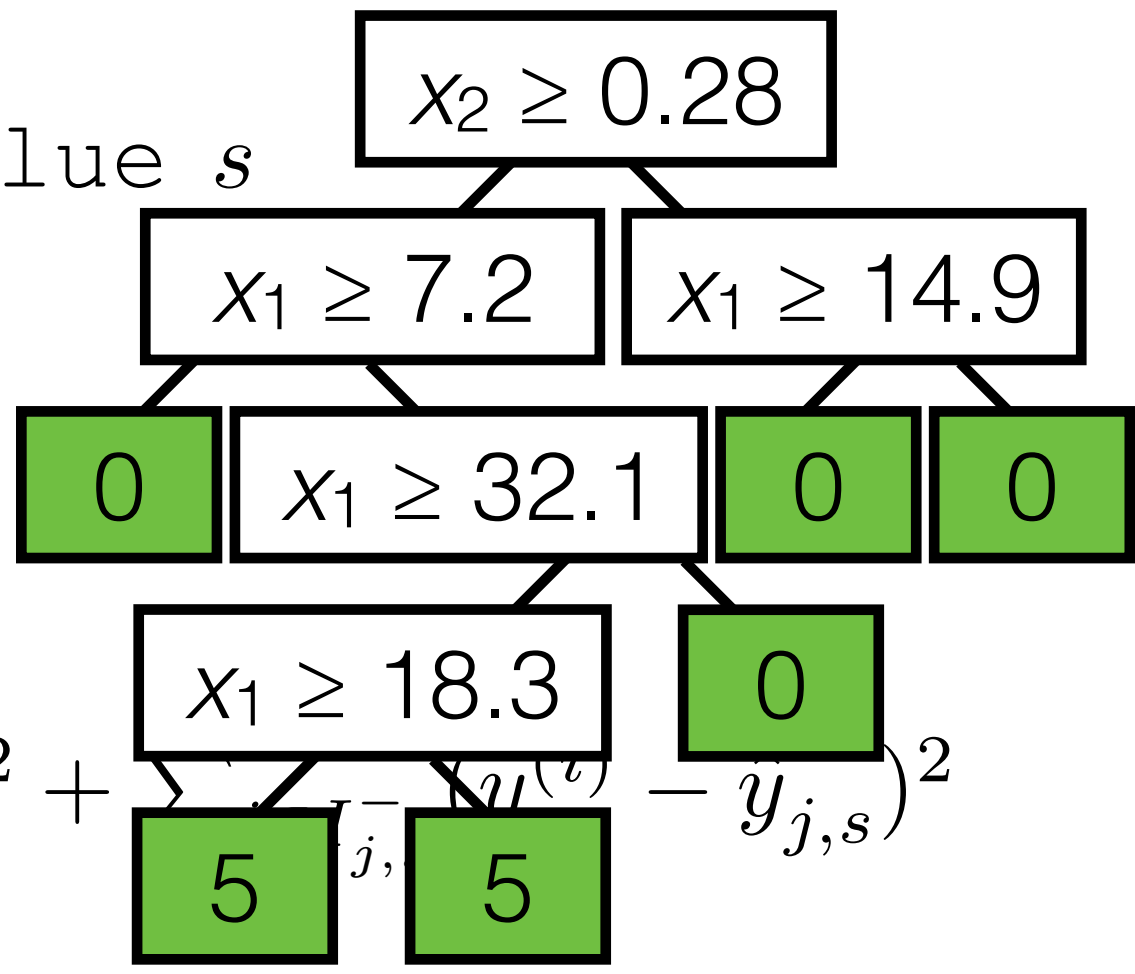
```

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

```



BuildTree($\{1, \dots, n\}; 2$)



Building a decision tree

- Regression tree with squared error loss

BuildTree($I; k$)

```

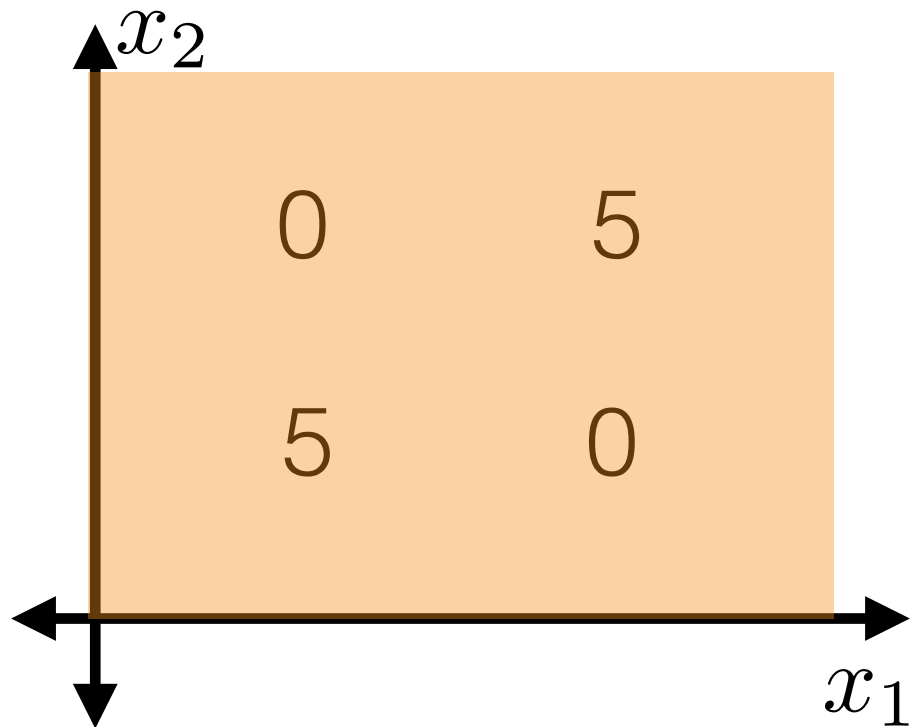
if  $|I| < k$ 
     $\text{label} = \hat{y}$ 
else

```

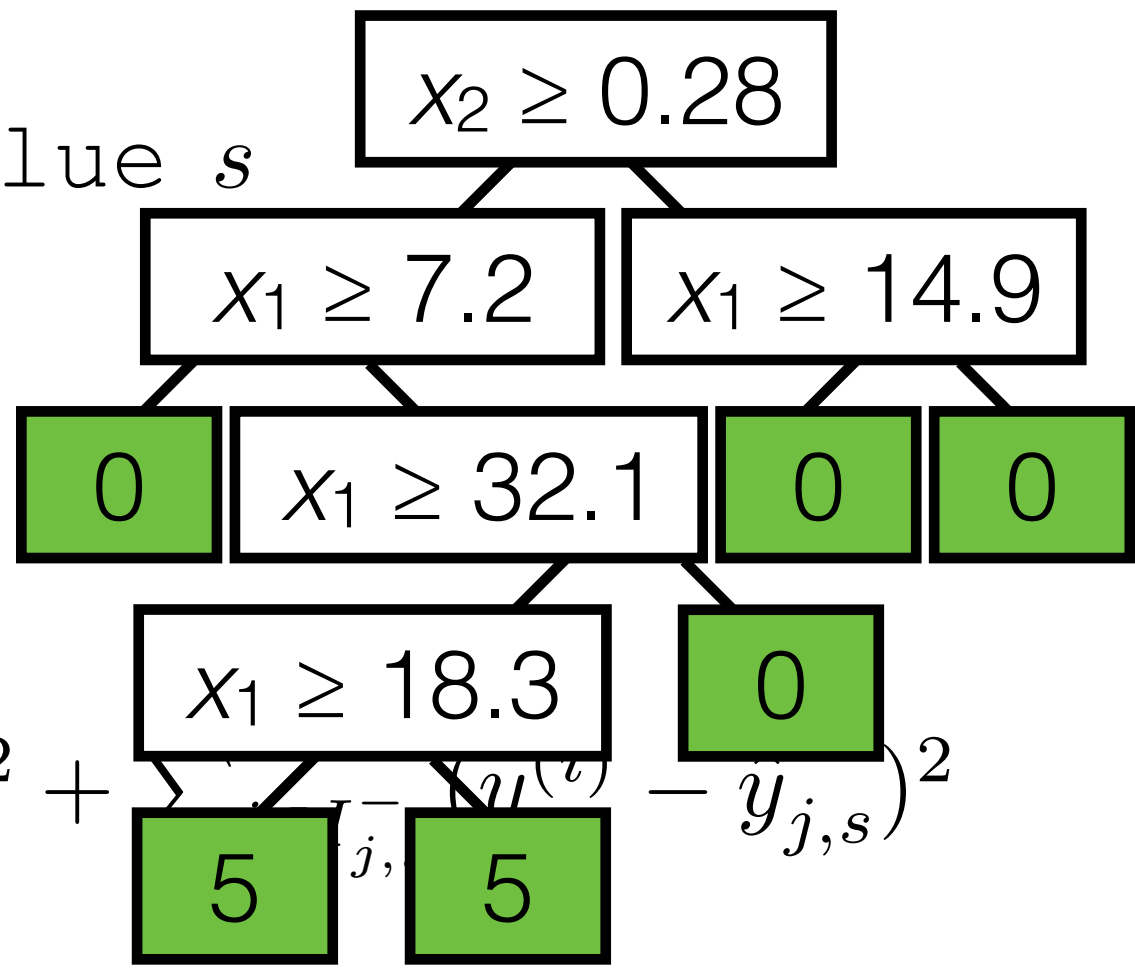
Why keep splitting after our error is low?

Is overfitting an issue?

Idea: stop when no/ small change in loss



BuildTree($\{1, \dots, n\}; 2$)

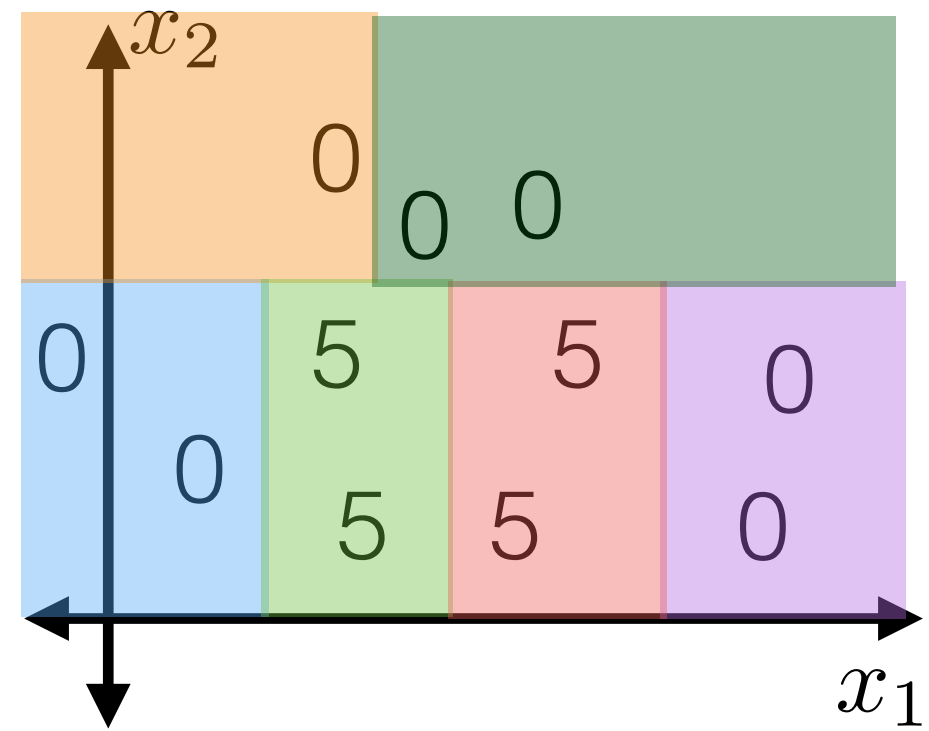


```

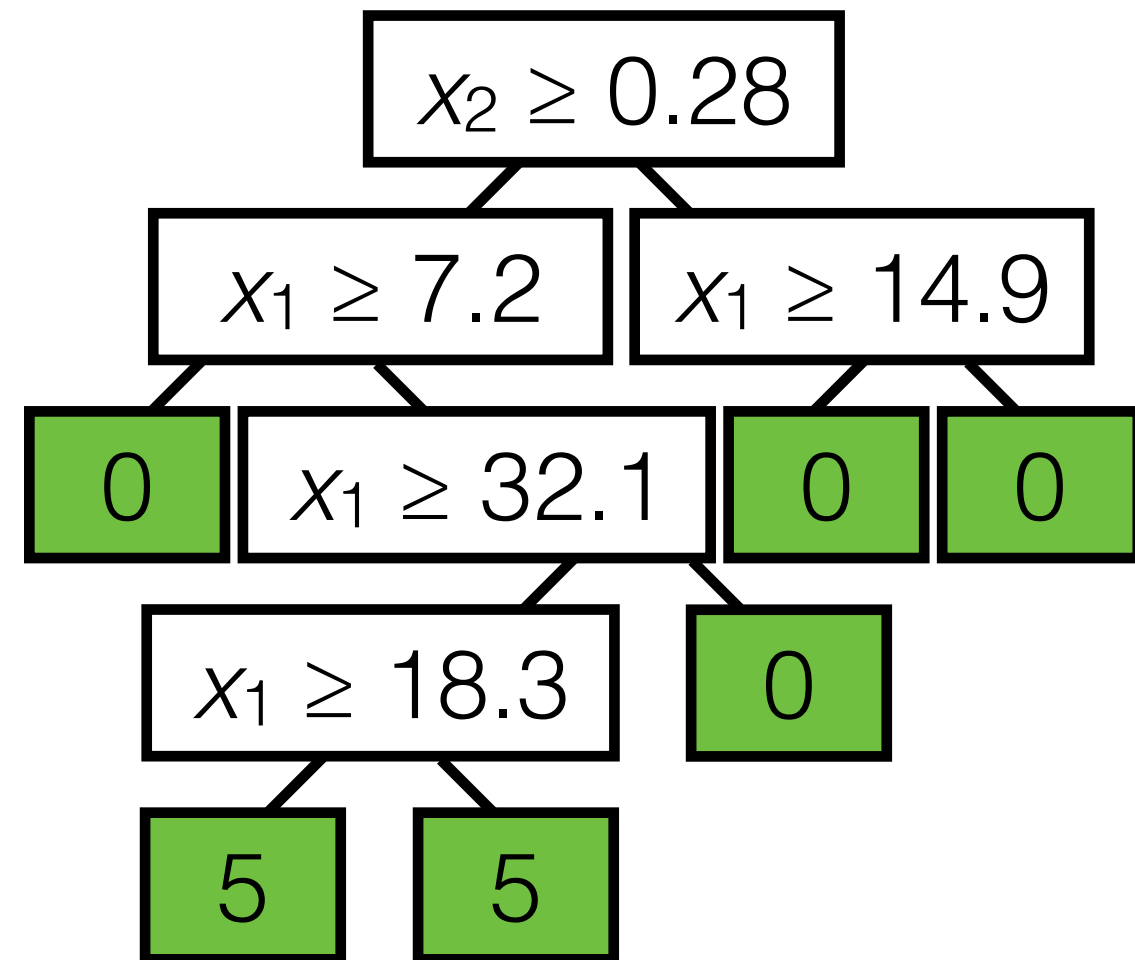
Set  $(j^*, s^*) = \arg \min E_{j,s}$ 
return Node  $(j^*, s^*, \text{BuildTree}(I_{j^*, s^*}^-, k), \text{BuildTree}(I_{j^*, s^*}^+, k))$ 

```


How to regularize?

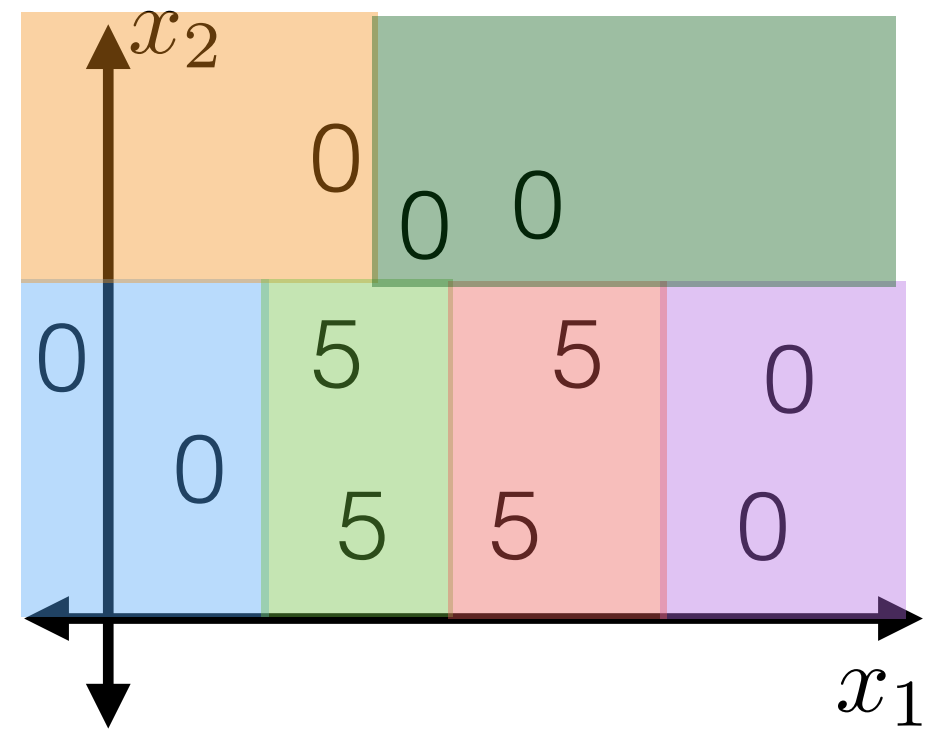


BuildTree($\{1, \dots, n\}; 2$)

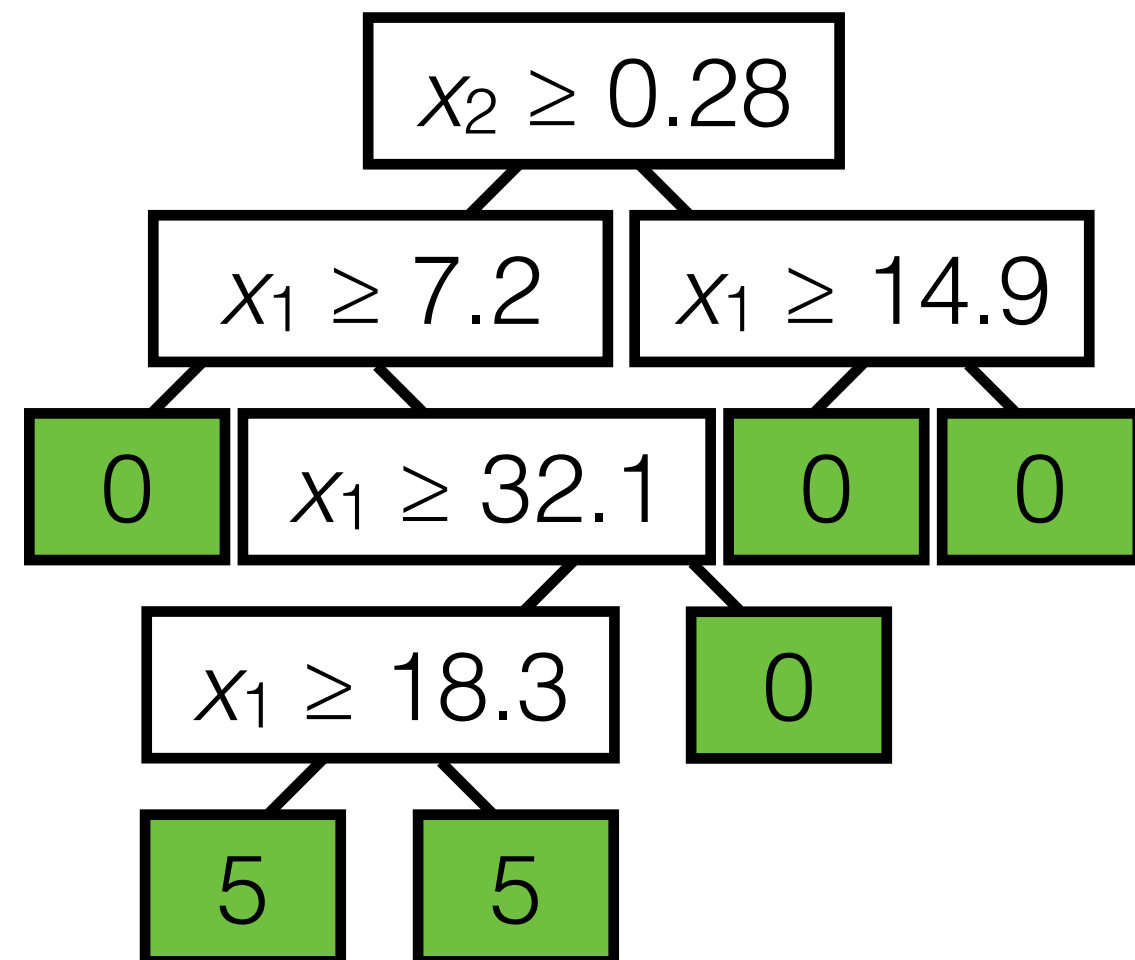


How to regularize?

- Objective = training loss + constant * regularizer

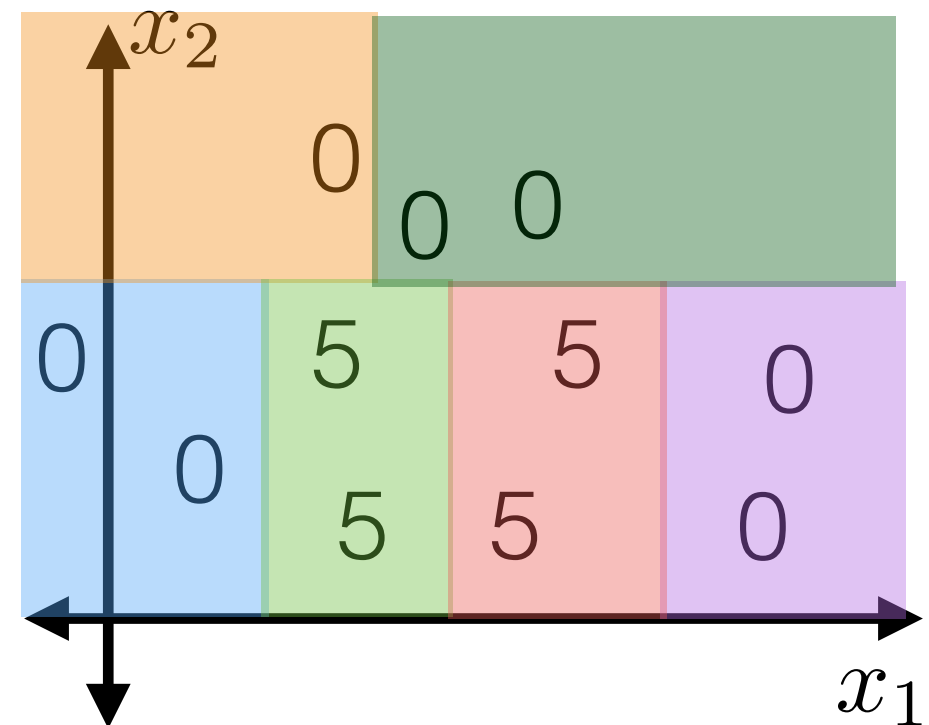


BuildTree($\{1, \dots, n\}; 2$)

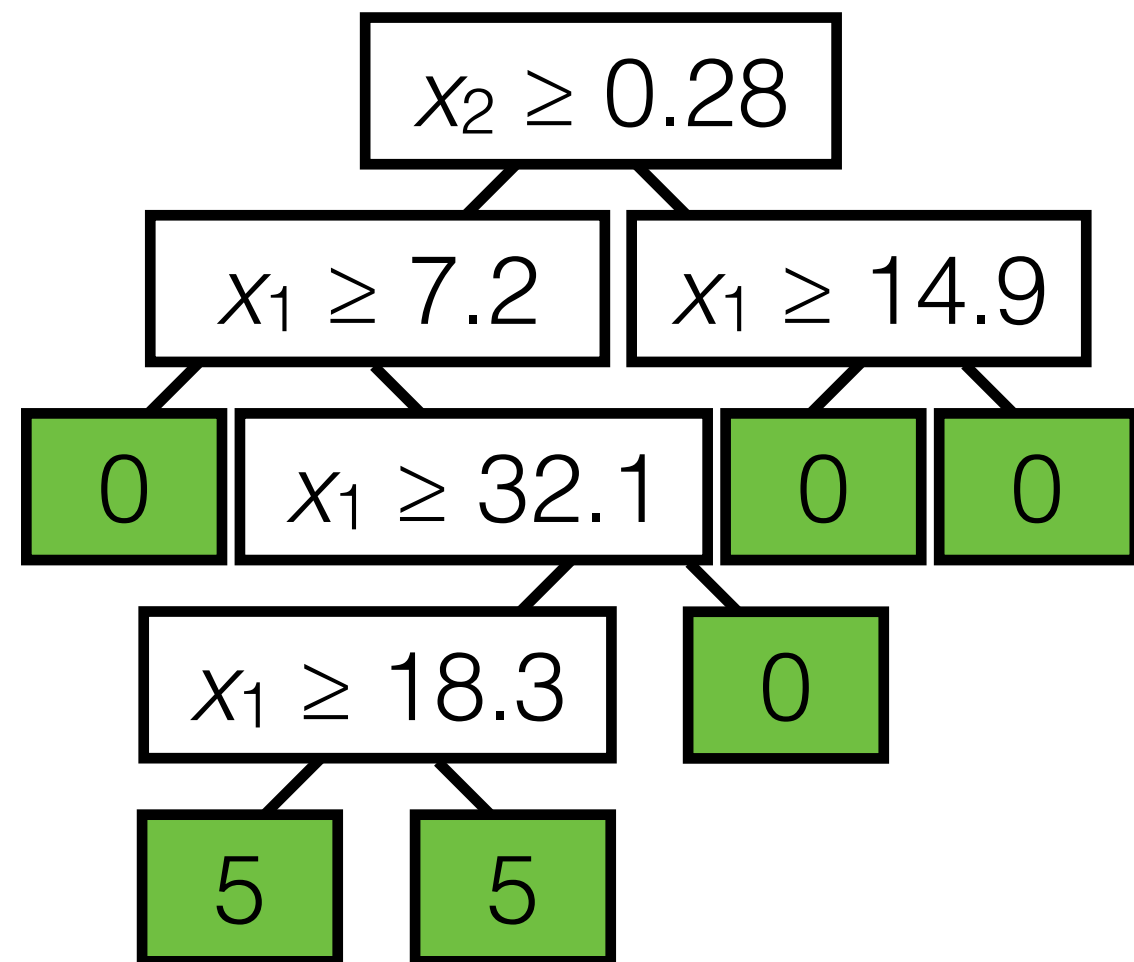


How to regularize?

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$

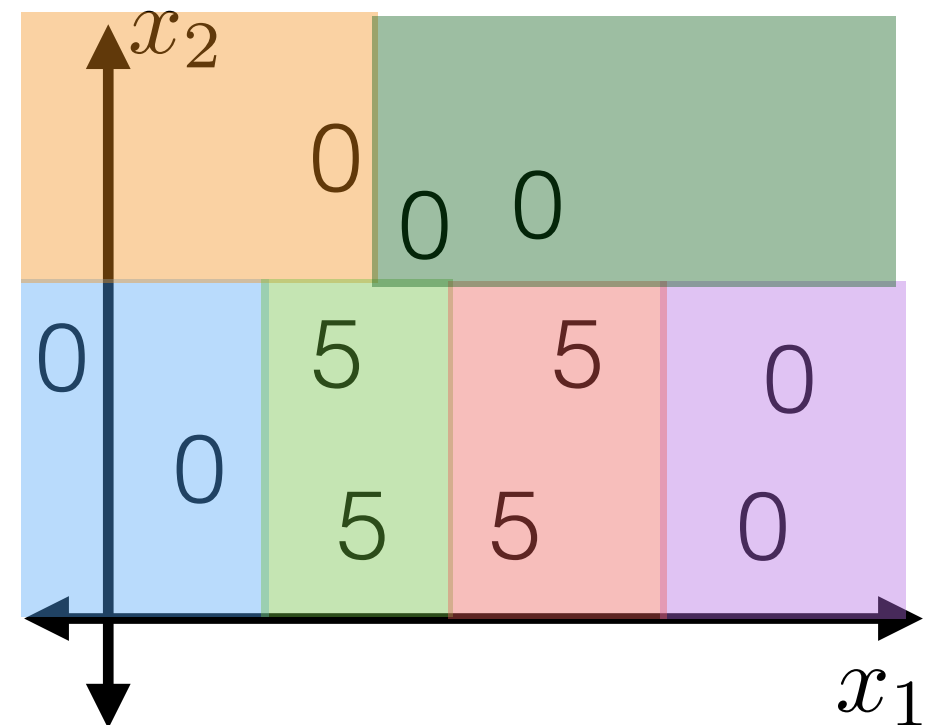


BuildTree({1, ..., n}; 2)

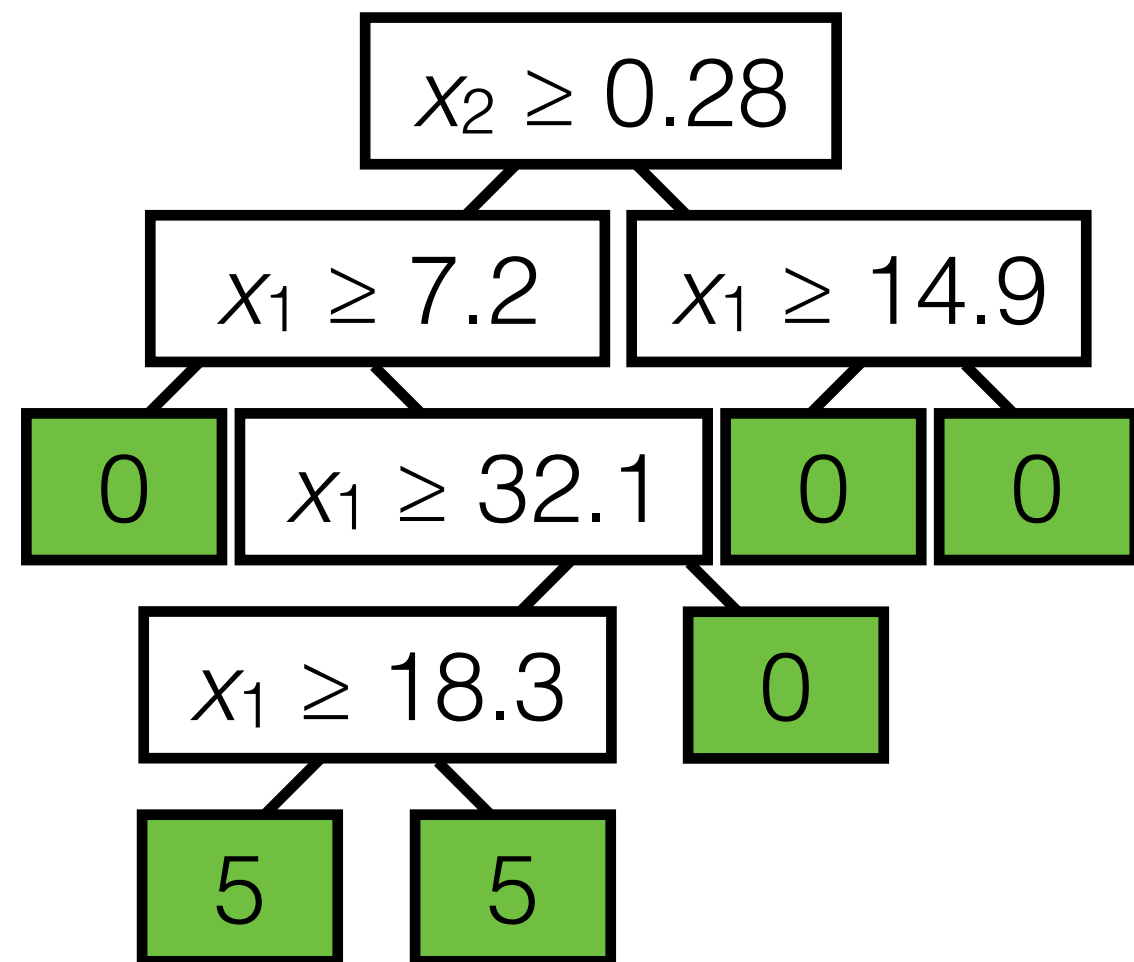


How to regularize?

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$

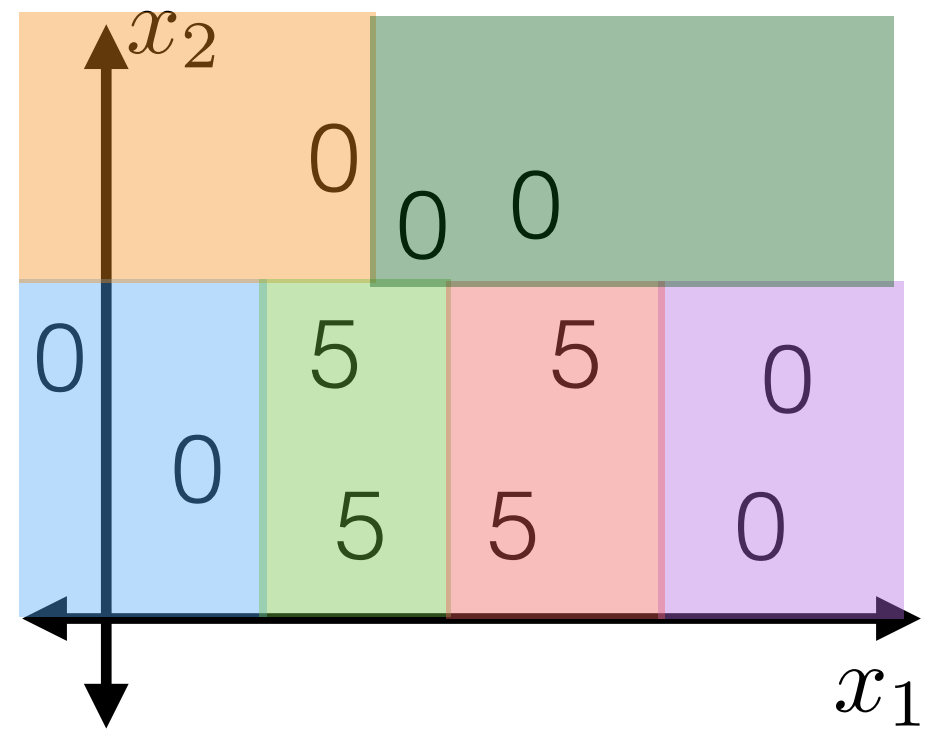


BuildTree({1, ..., n}; 2)

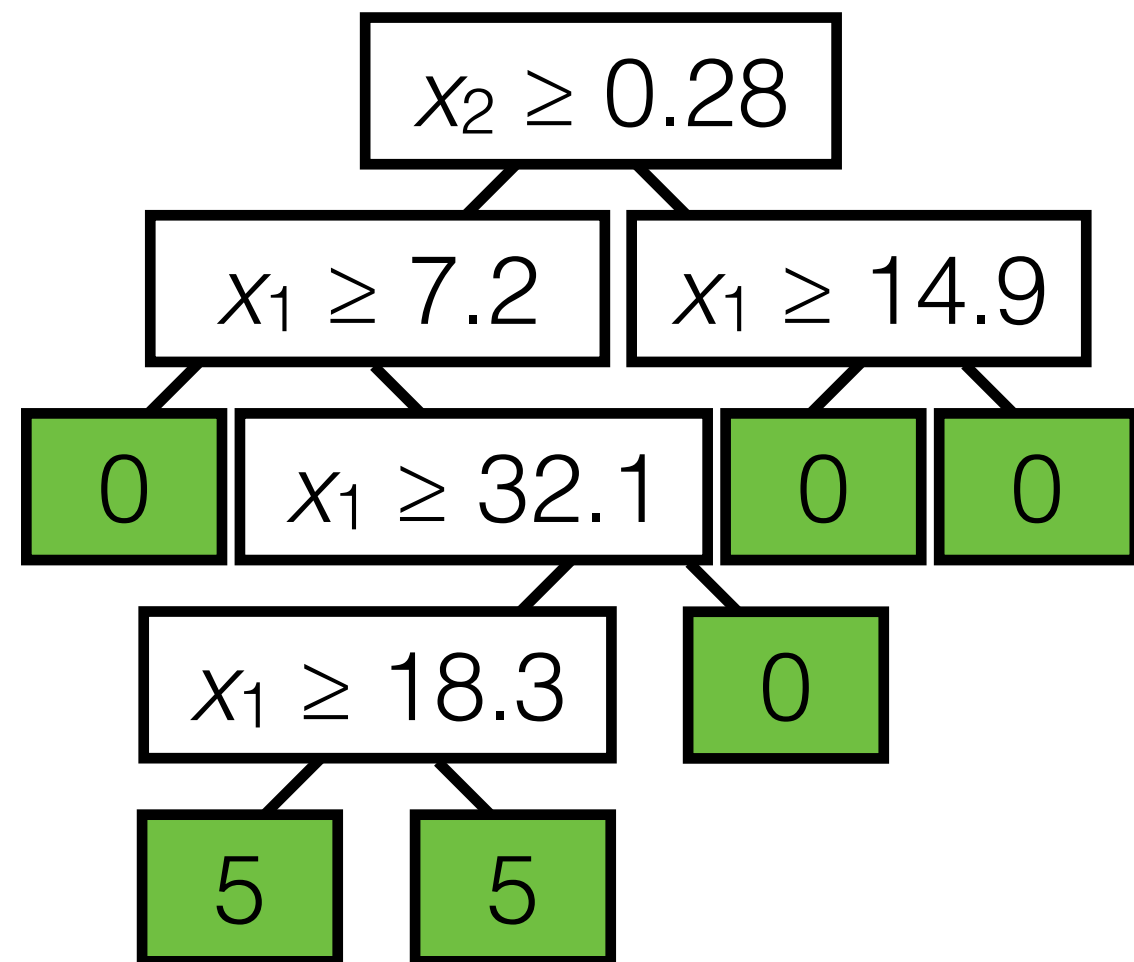


How to regularize?

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$

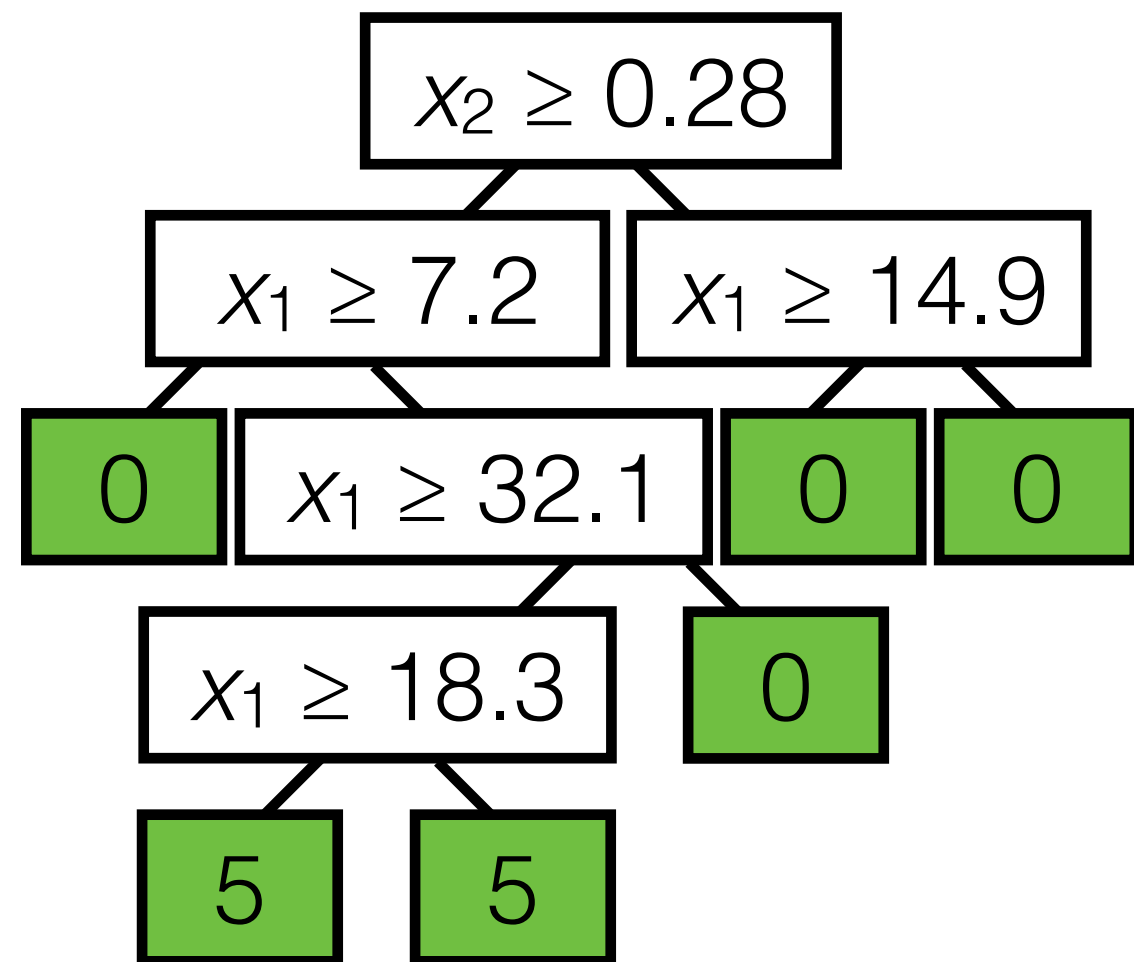
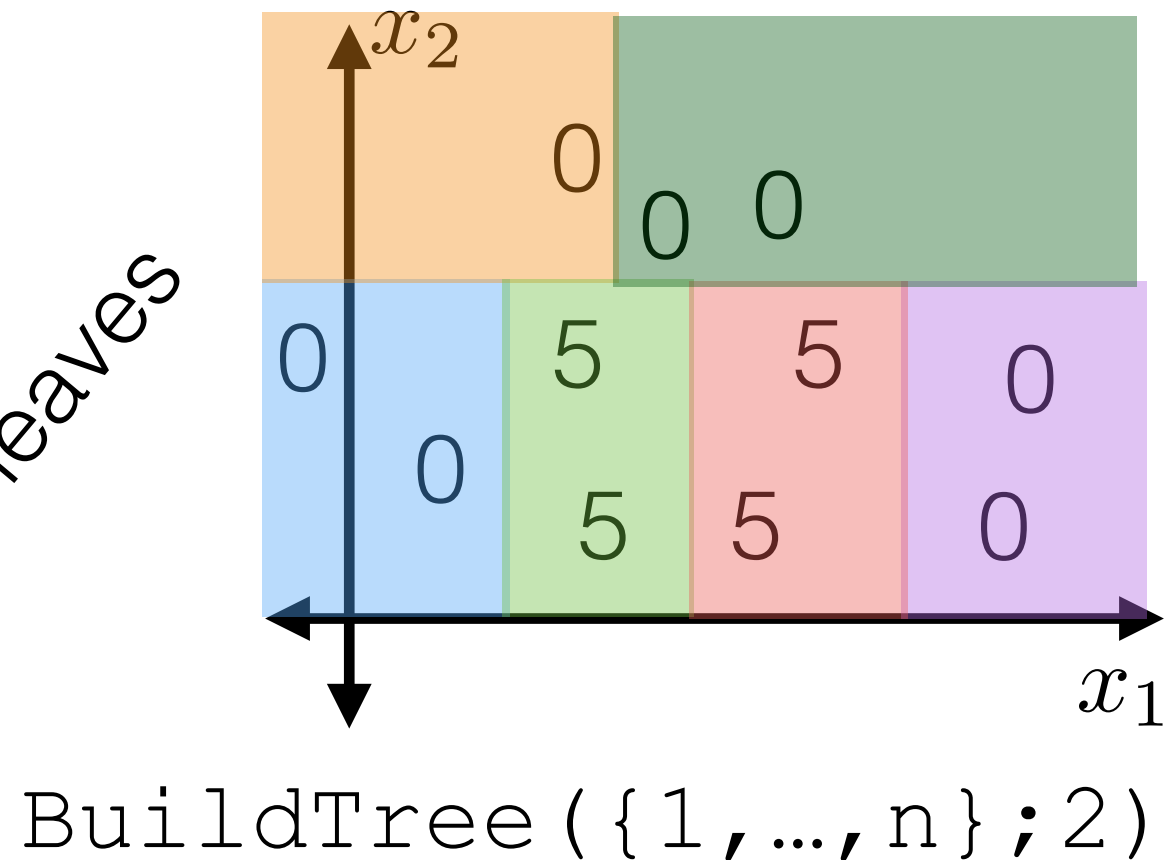


BuildTree({1, ..., n}; 2)



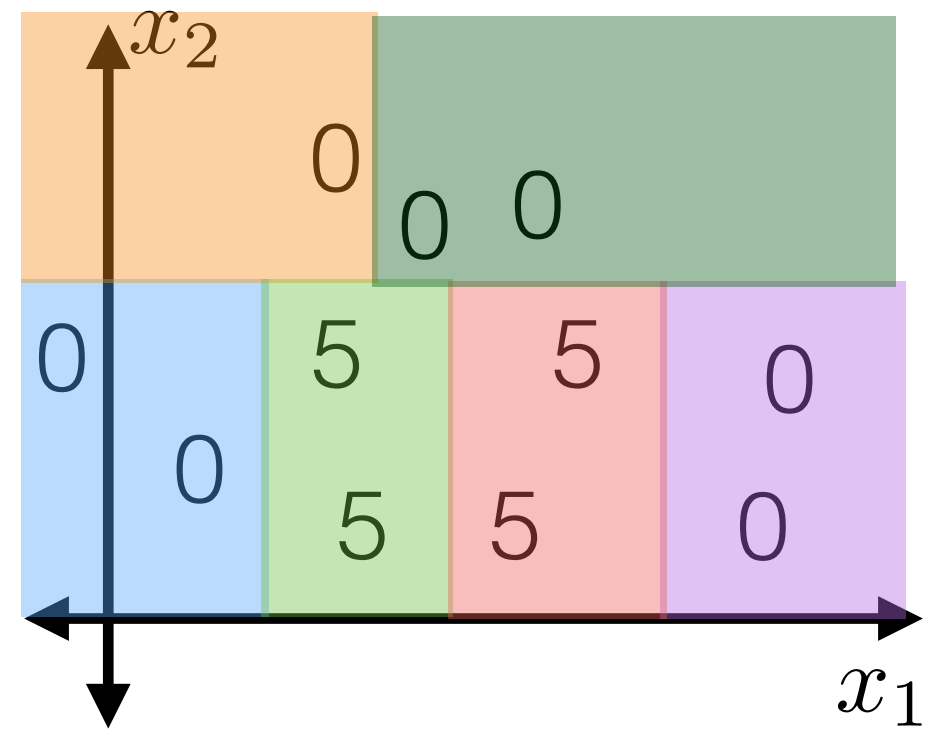
How to regularize?

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$

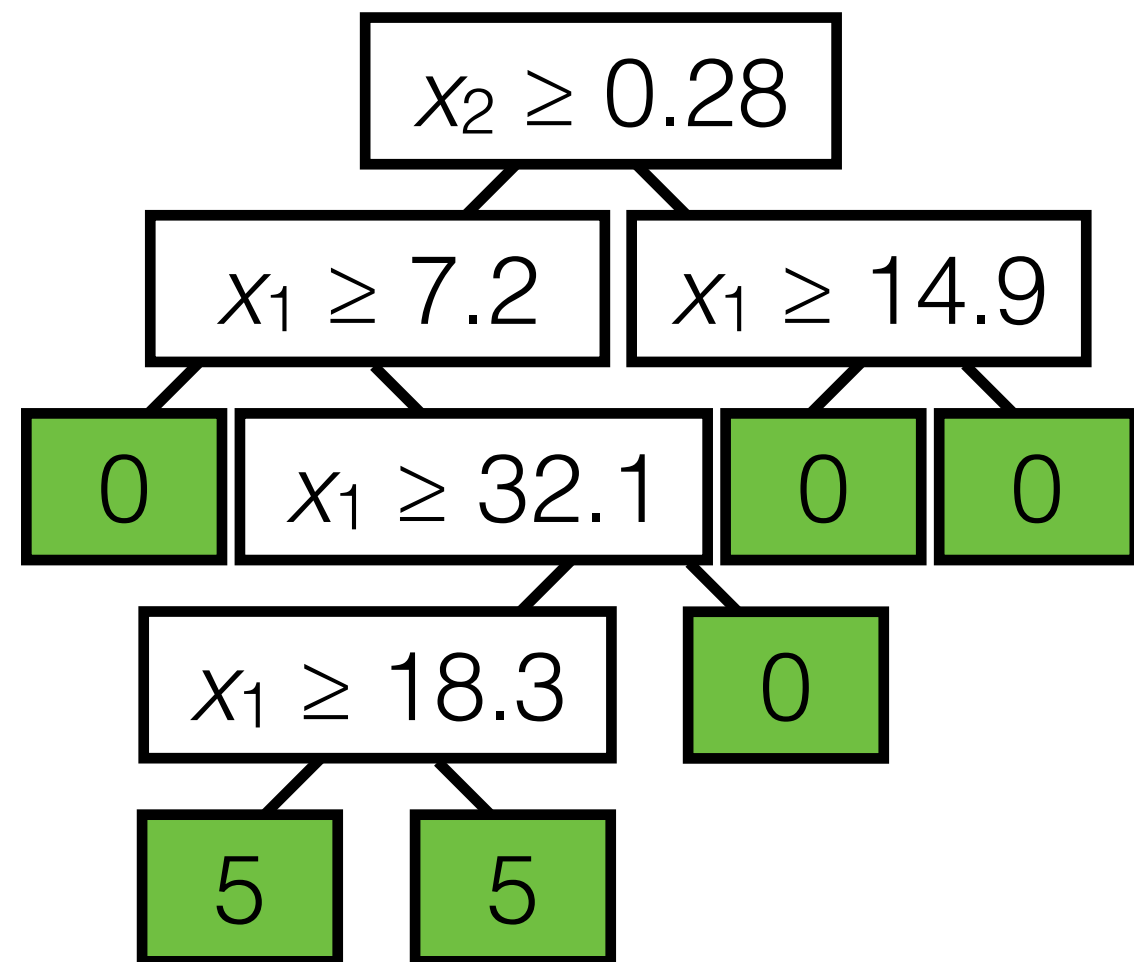


How to regularize?

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$



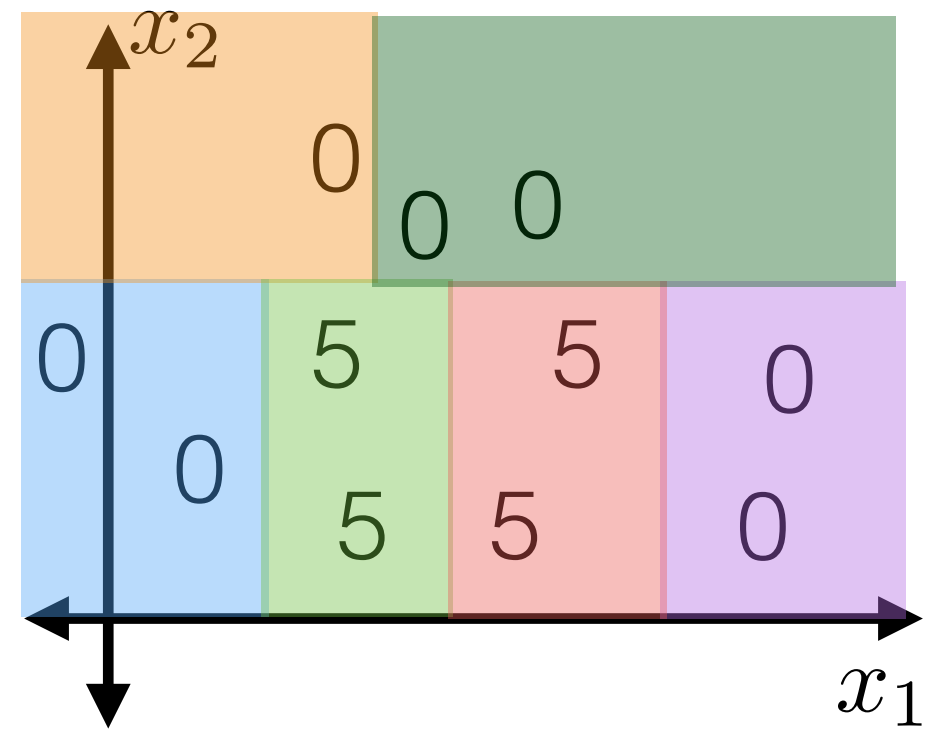
BuildTree({1, ..., n}; 2)



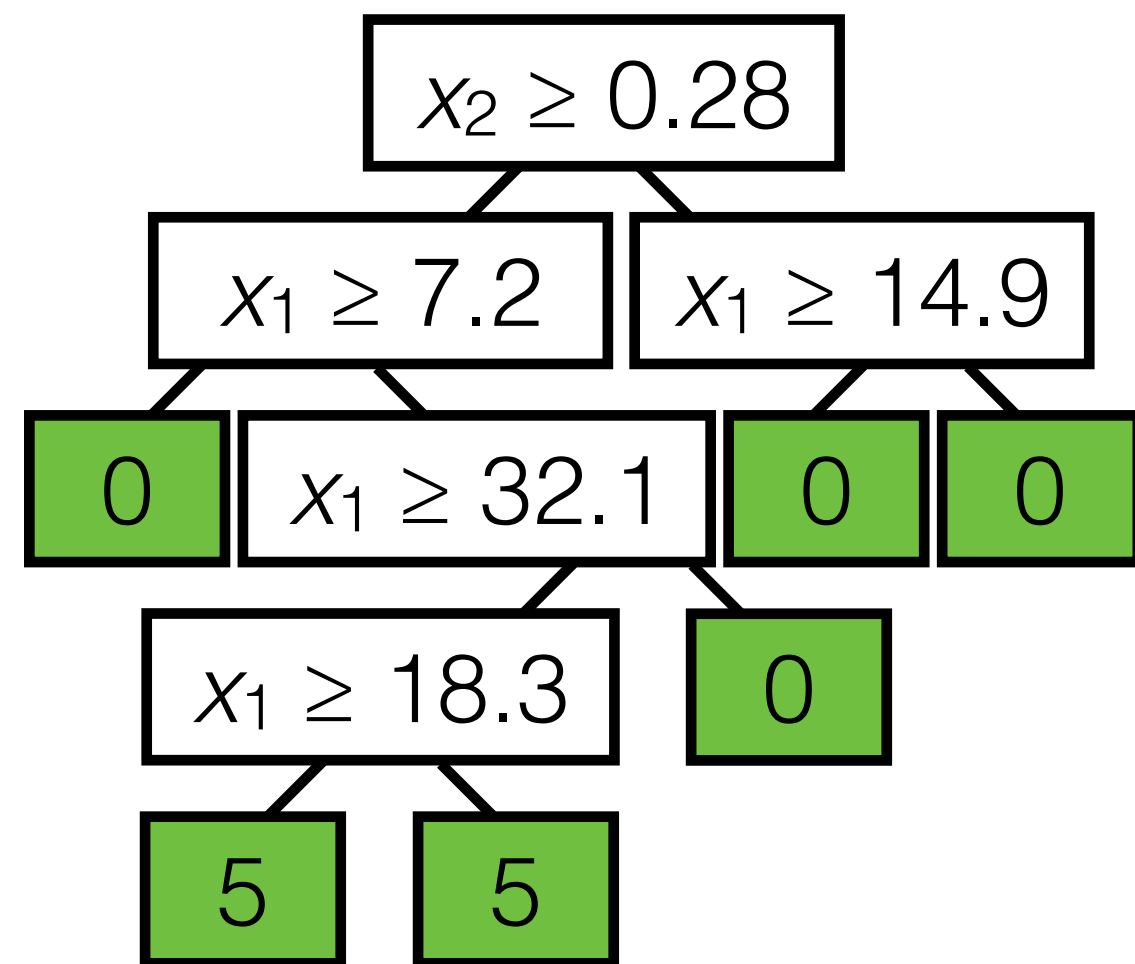
How to regularize?

- “Cost complexity” of a tree T

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$



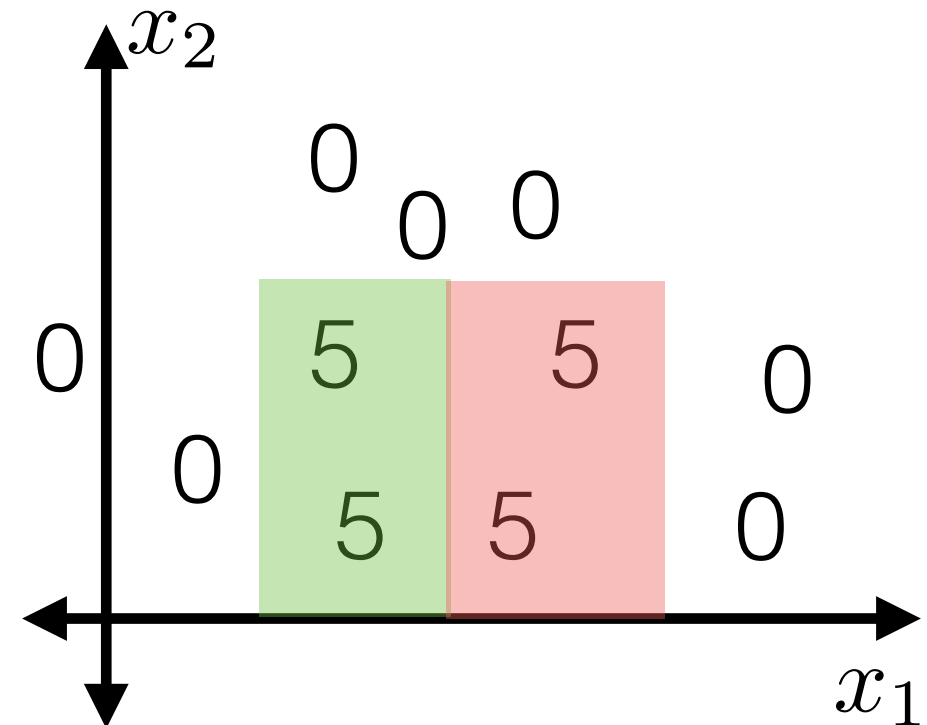
BuildTree({1, ..., n}; 2)



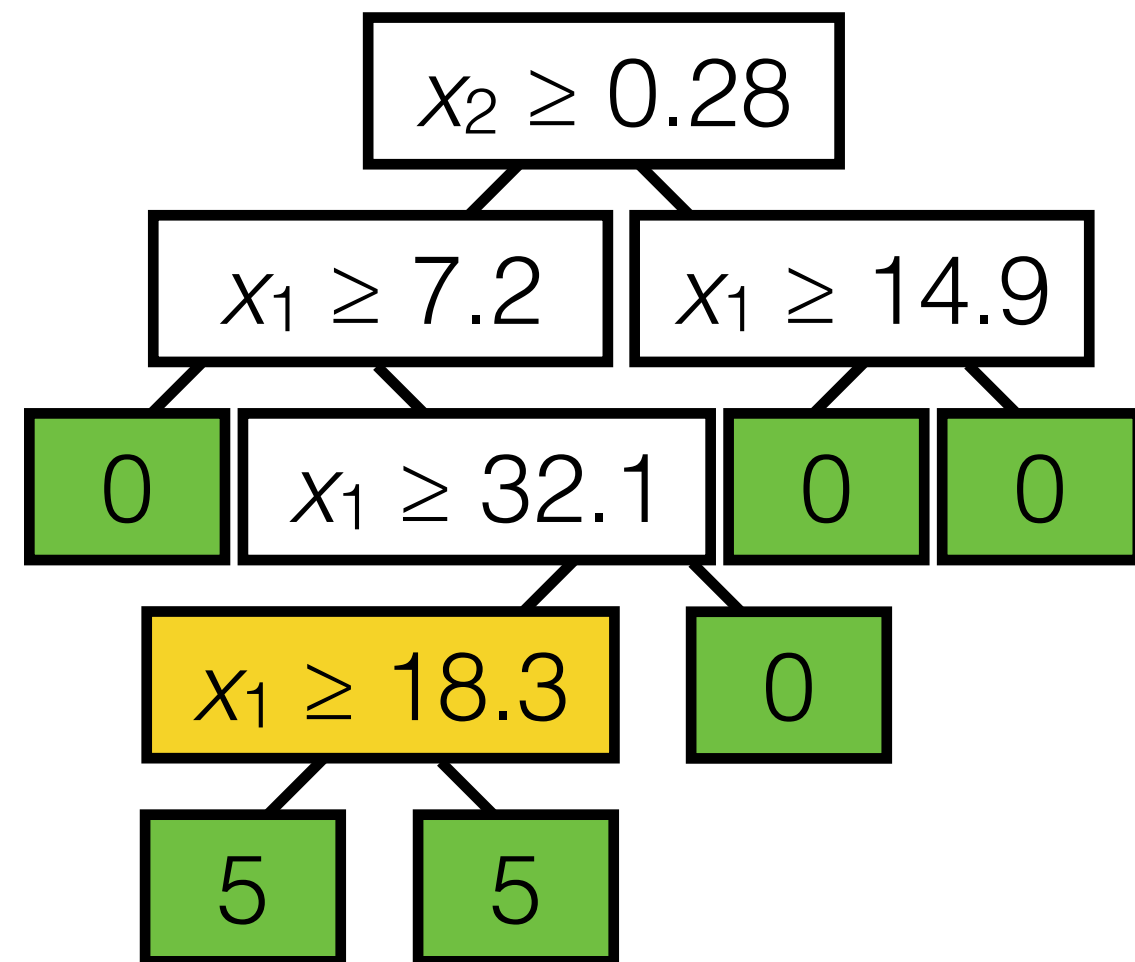
How to regularize?

- “Cost complexity” of a tree T

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$



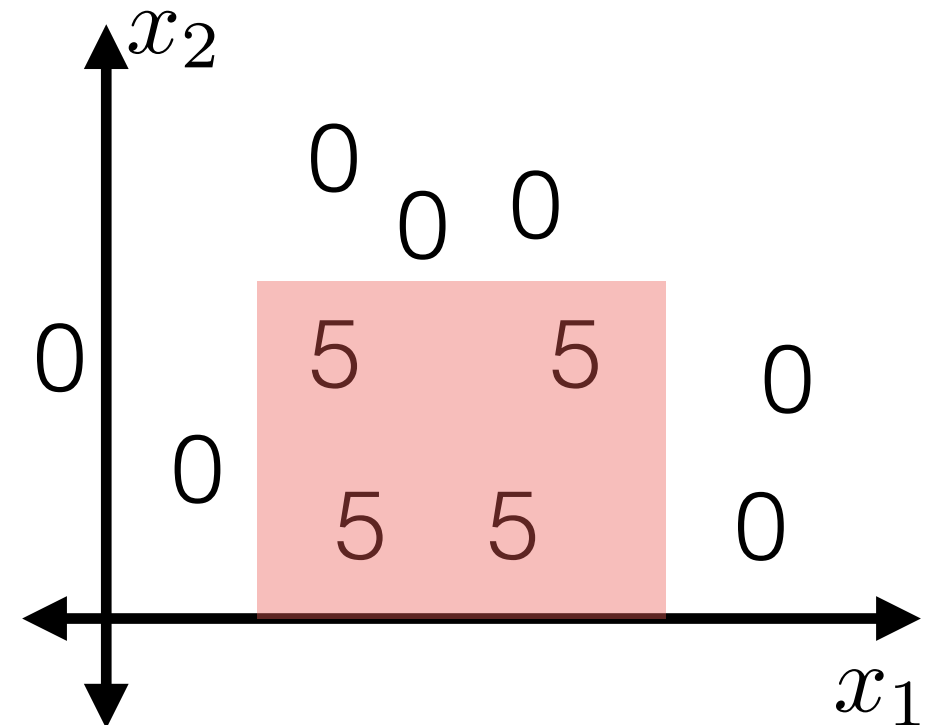
BuildTree({1, ..., n}; 2)



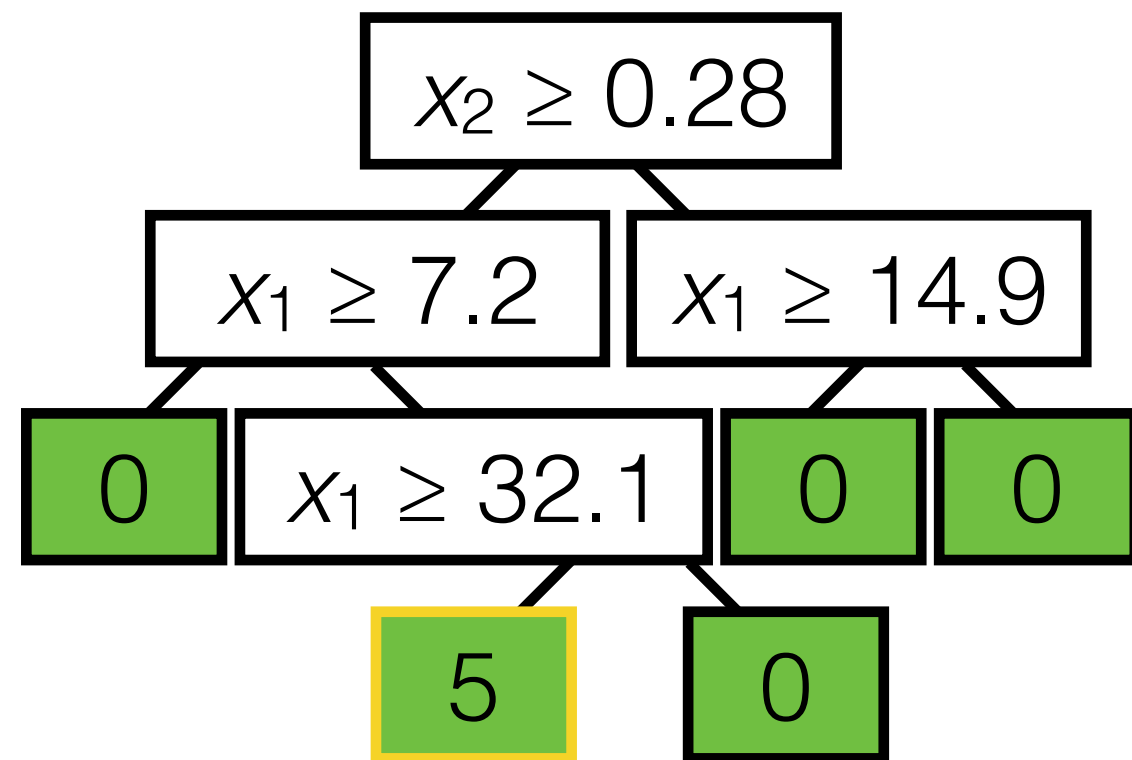
How to regularize?

- “Cost complexity” of a tree T

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$



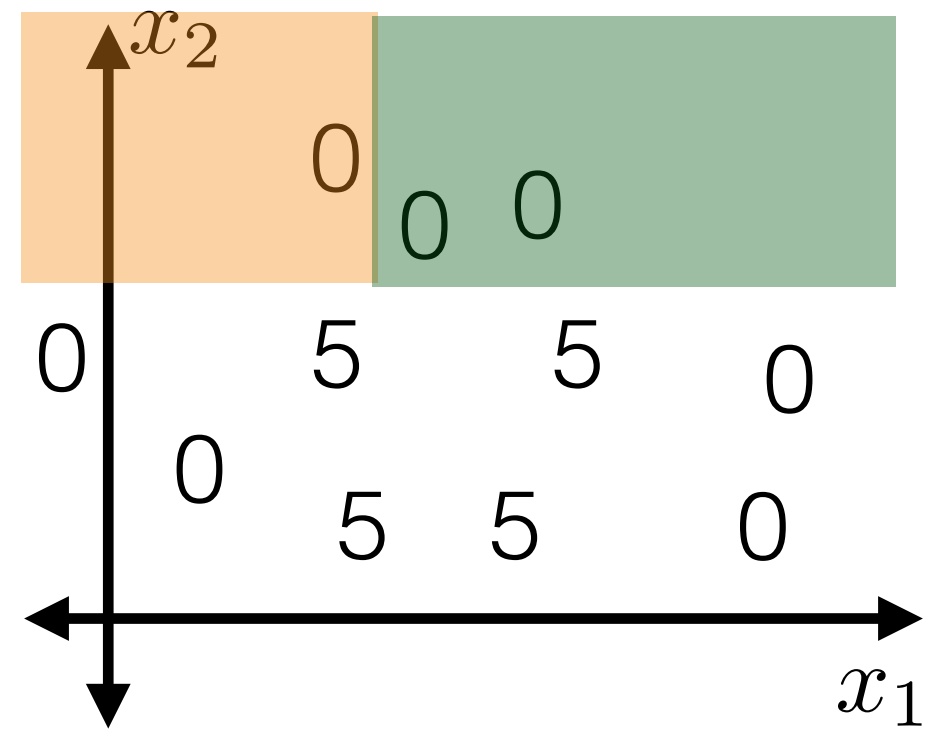
BuildTree({1, ..., n}; 2)



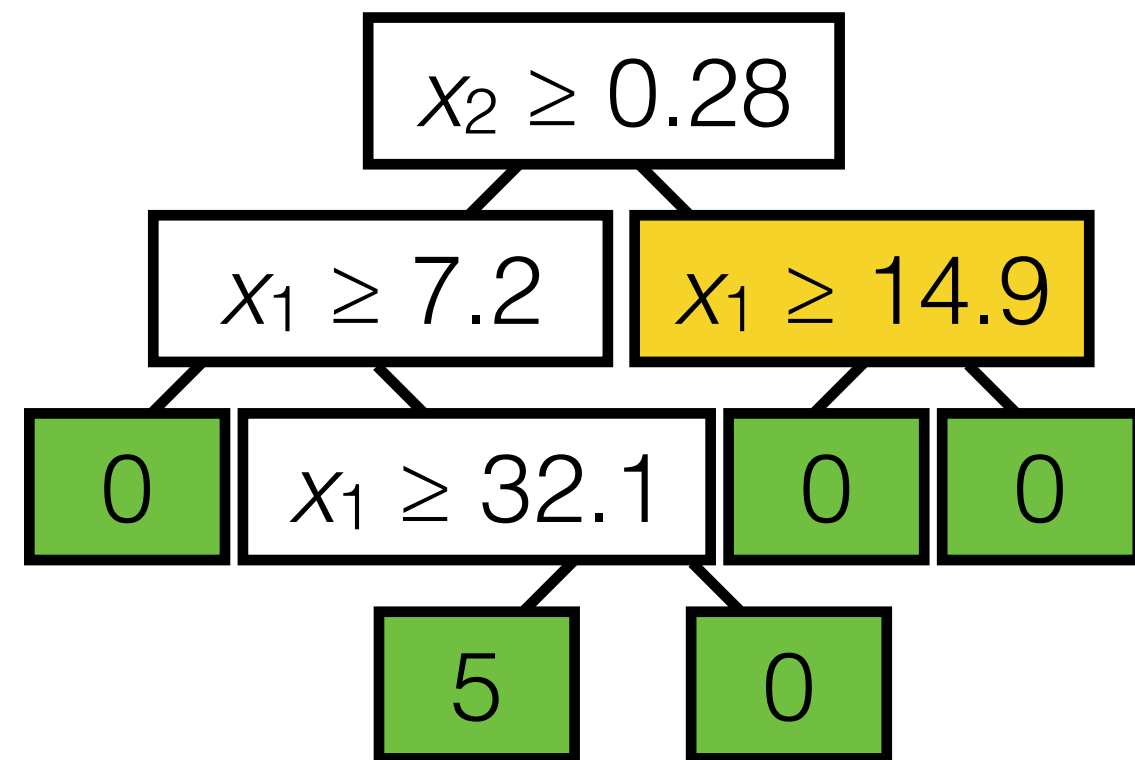
How to regularize?

- “Cost complexity” of a tree T

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$



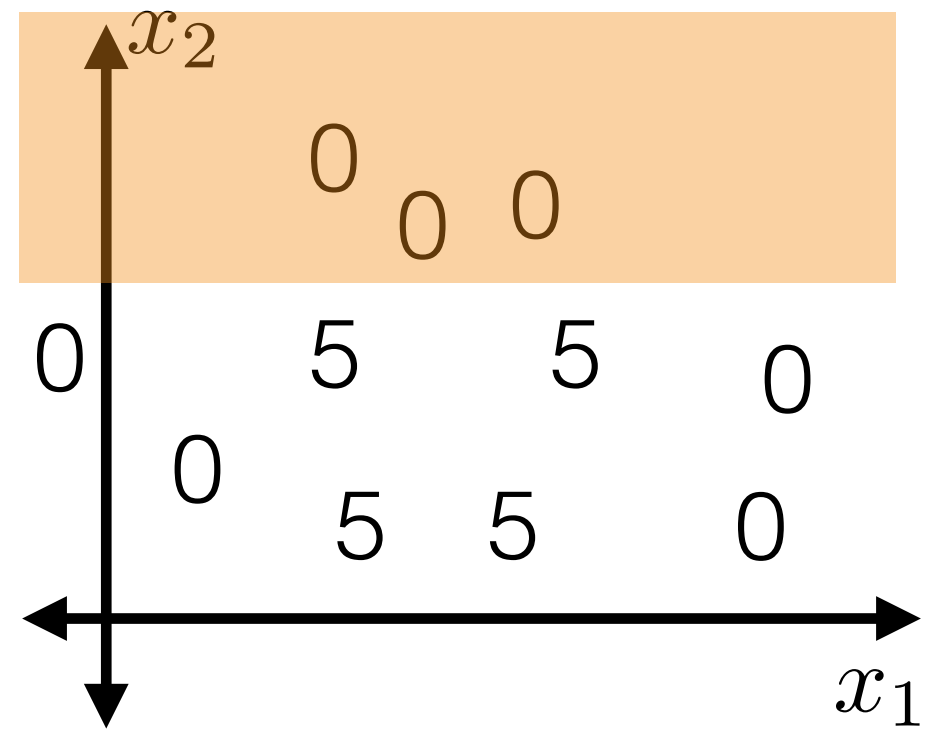
BuildTree({1, ..., n}; 2)



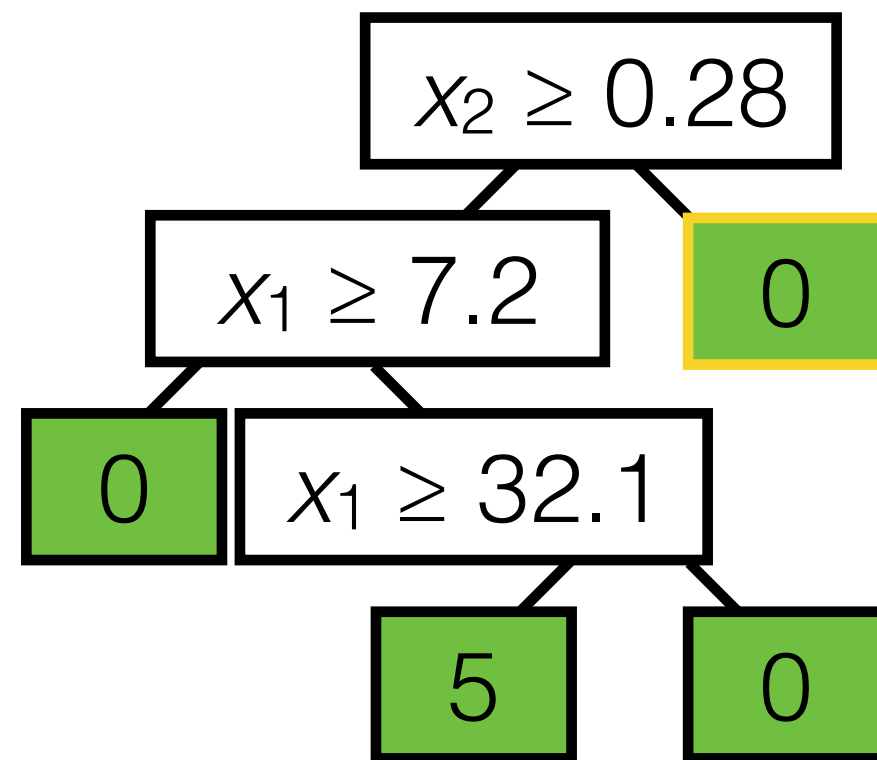
How to regularize?

- “Cost complexity” of a tree T

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$



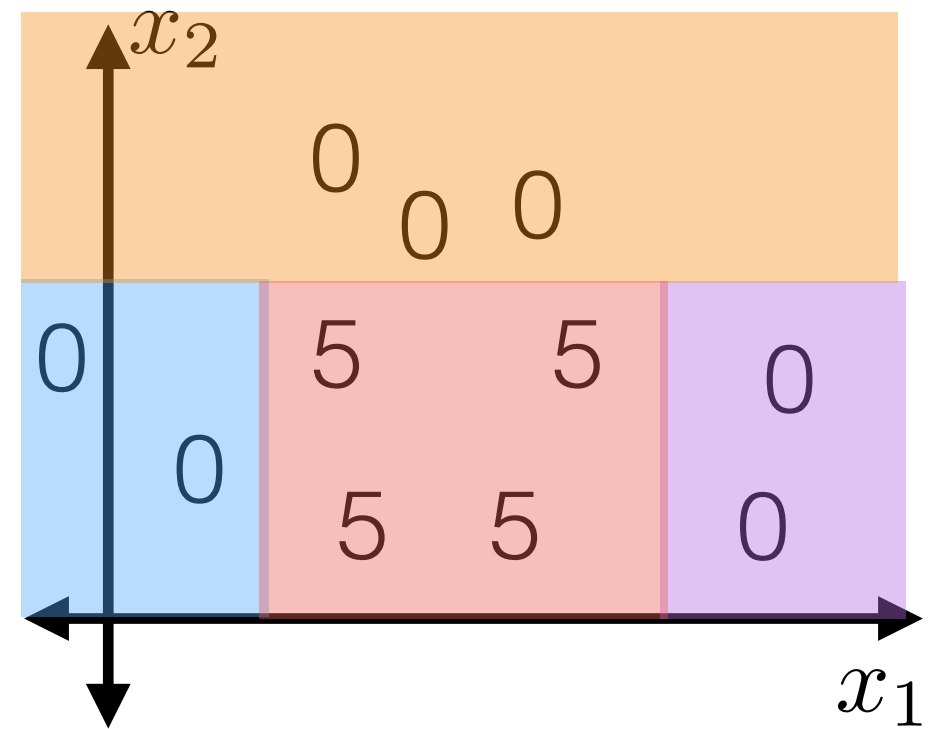
BuildTree({1, ..., n}; 2)



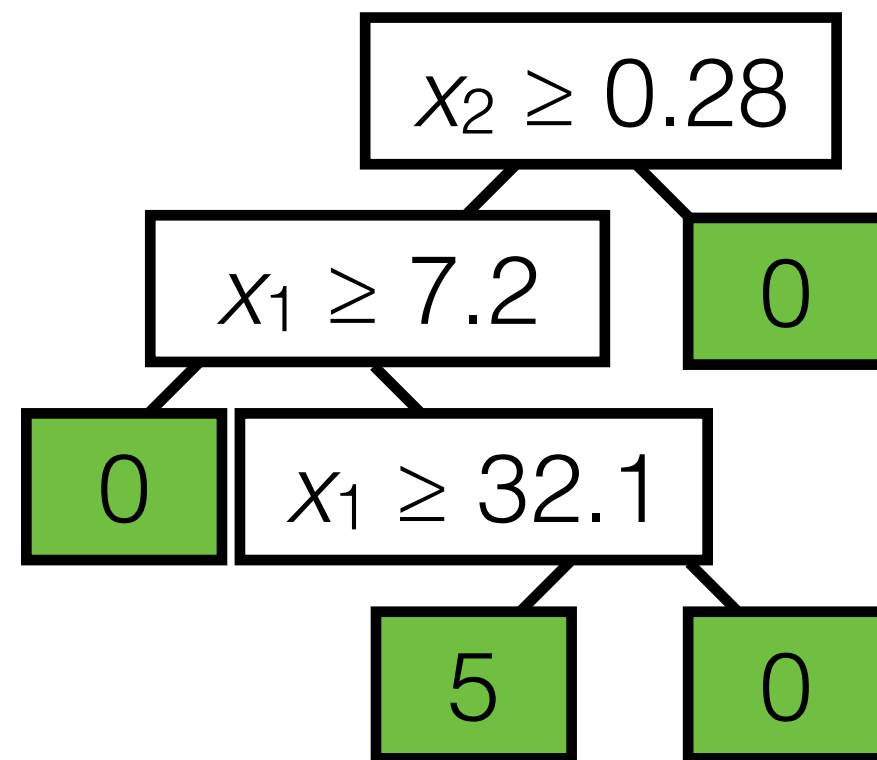
How to regularize?

- “Cost complexity” of a tree T

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$



BuildTree($\{1, \dots, n\}; 2$)

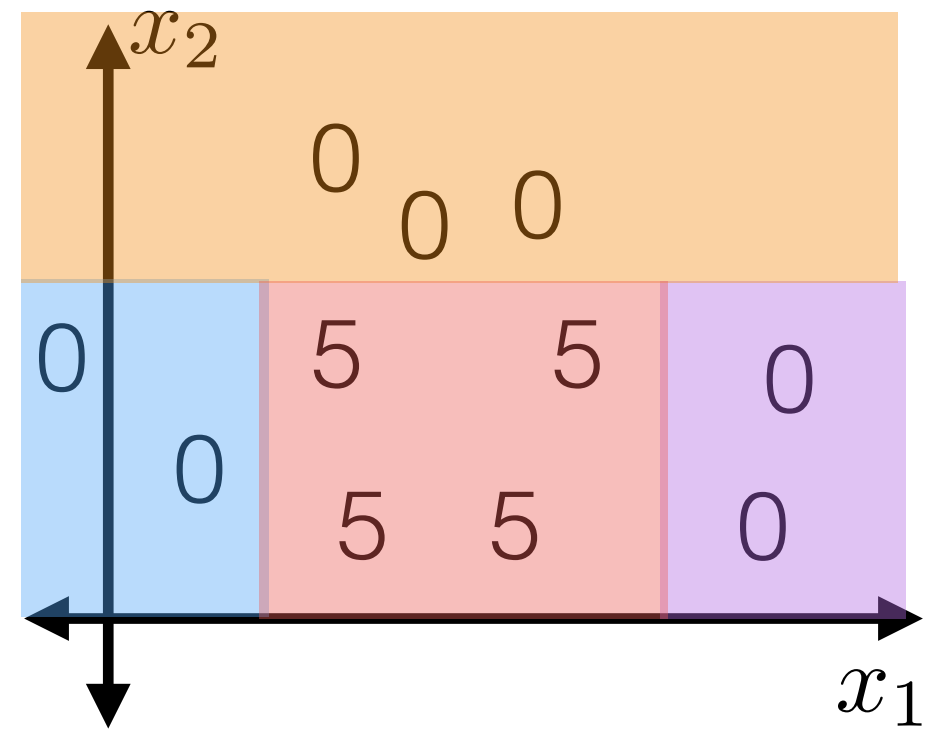


How to regularize?

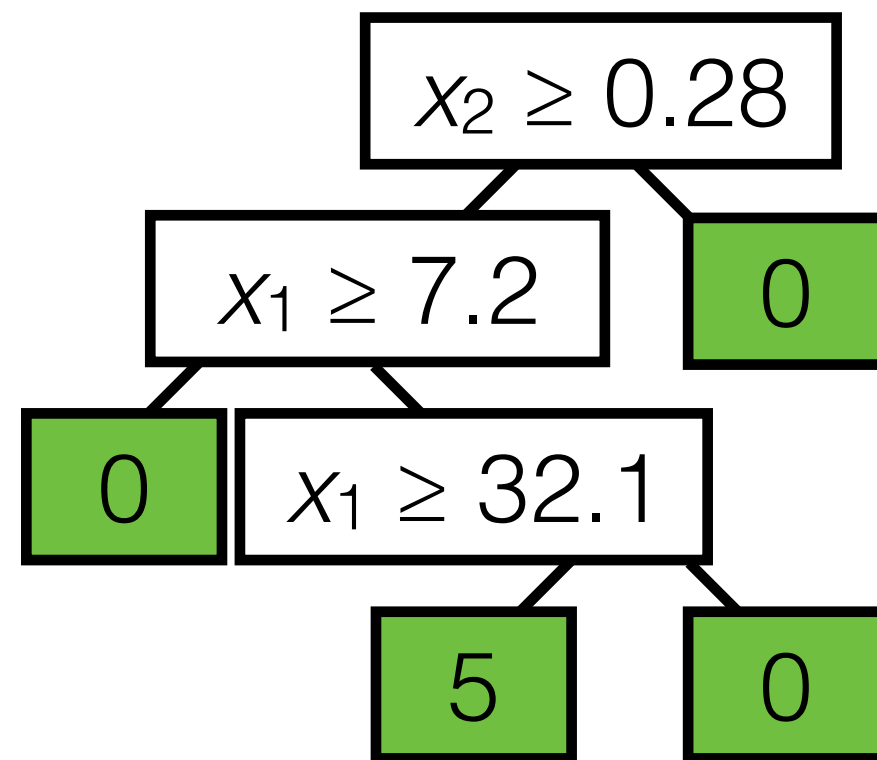
- “Cost complexity” of a tree T

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$

- Pruning



BuildTree($\{1, \dots, n\}; 2$)



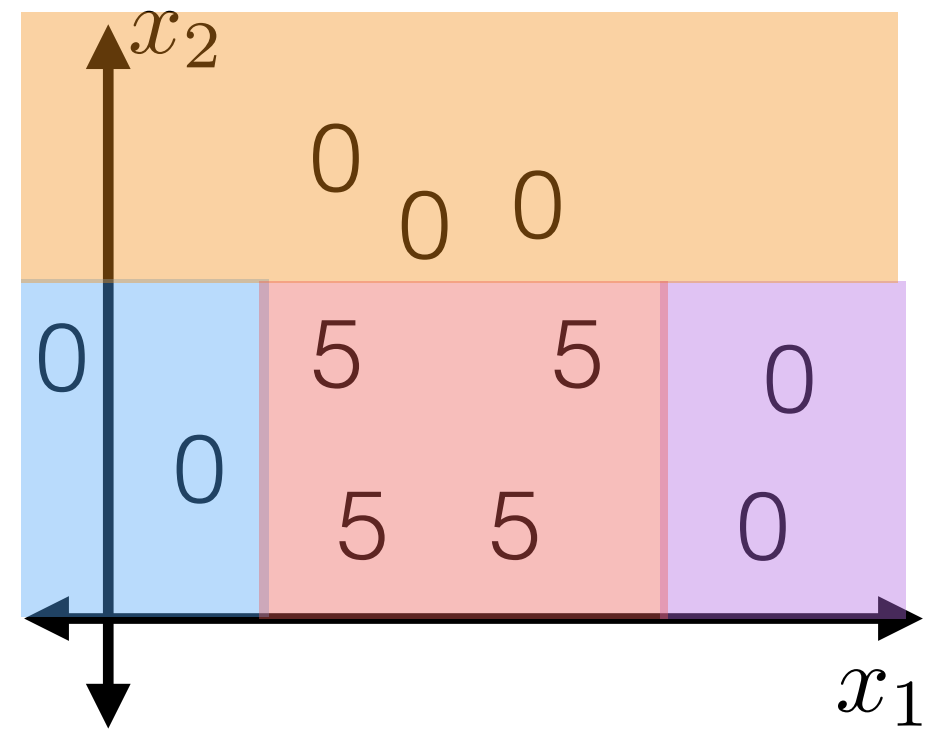
How to regularize?

- “Cost complexity” of a tree T

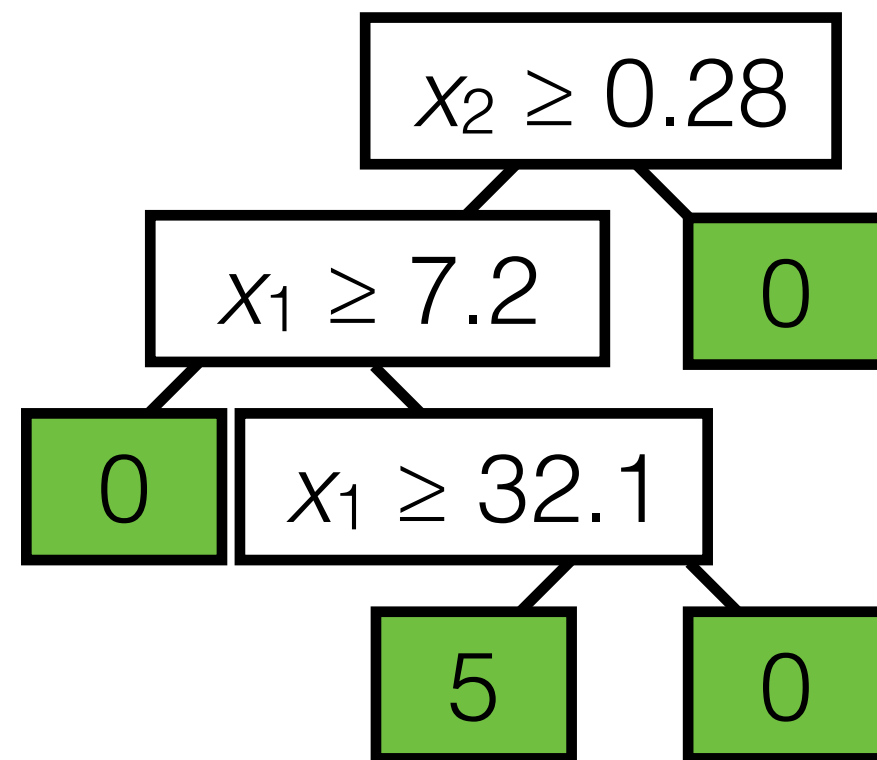
$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$

- Pruning

- For each α , choose T_α by pruning subtrees until it's not worthwhile



BuildTree($\{1, \dots, n\}; 2$)



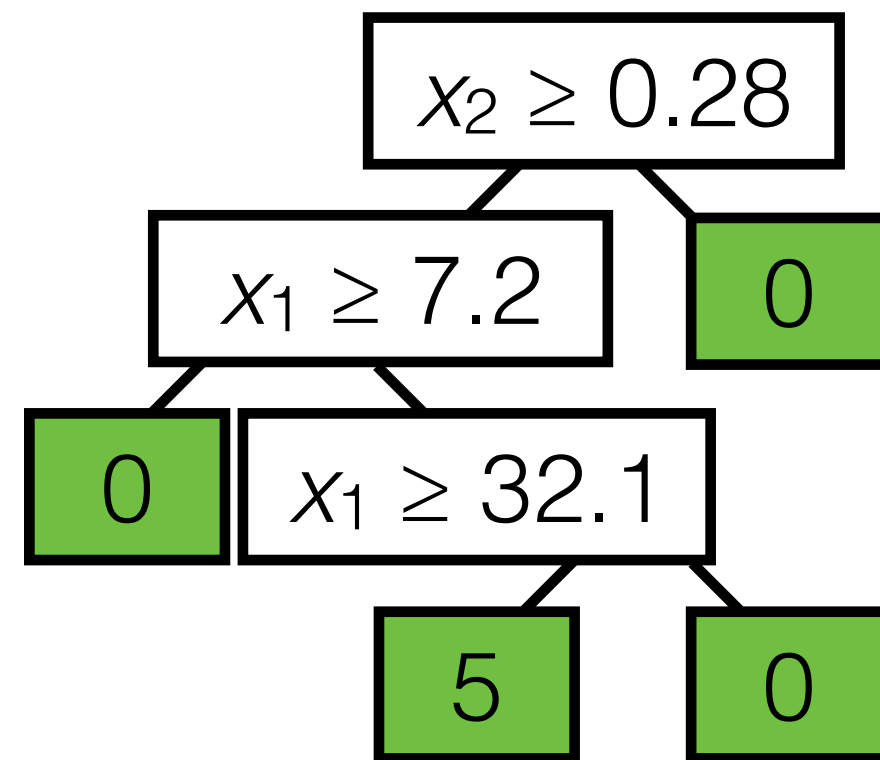
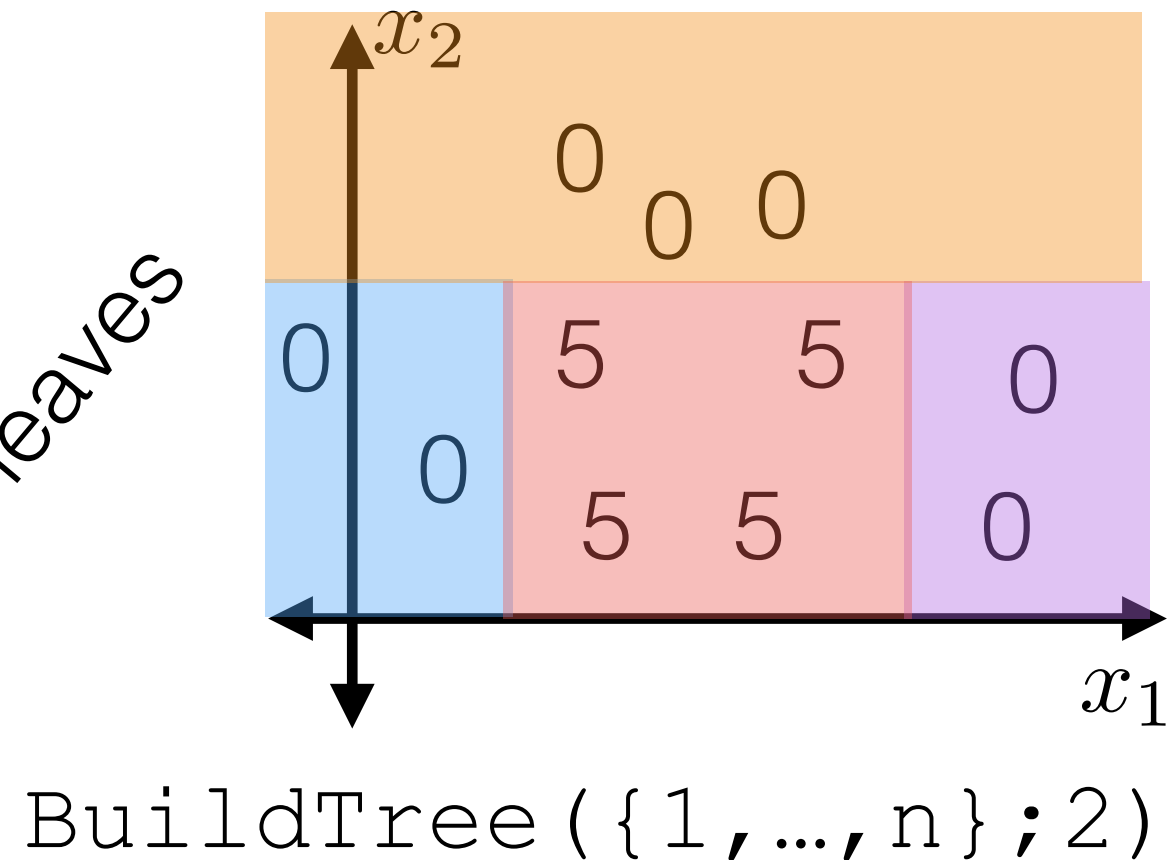
How to regularize?

- “Cost complexity” of a tree T

$$C_\alpha(T) = \sum_{i=1}^n L(T(x^{(i)}), y^{(i)}) + \alpha \underbrace{|T|}_{= \# \text{ leaves}}$$

- Pruning

- For each α , choose T_α by pruning subtrees until it's not worthwhile
- Choose a final tree by cross validation



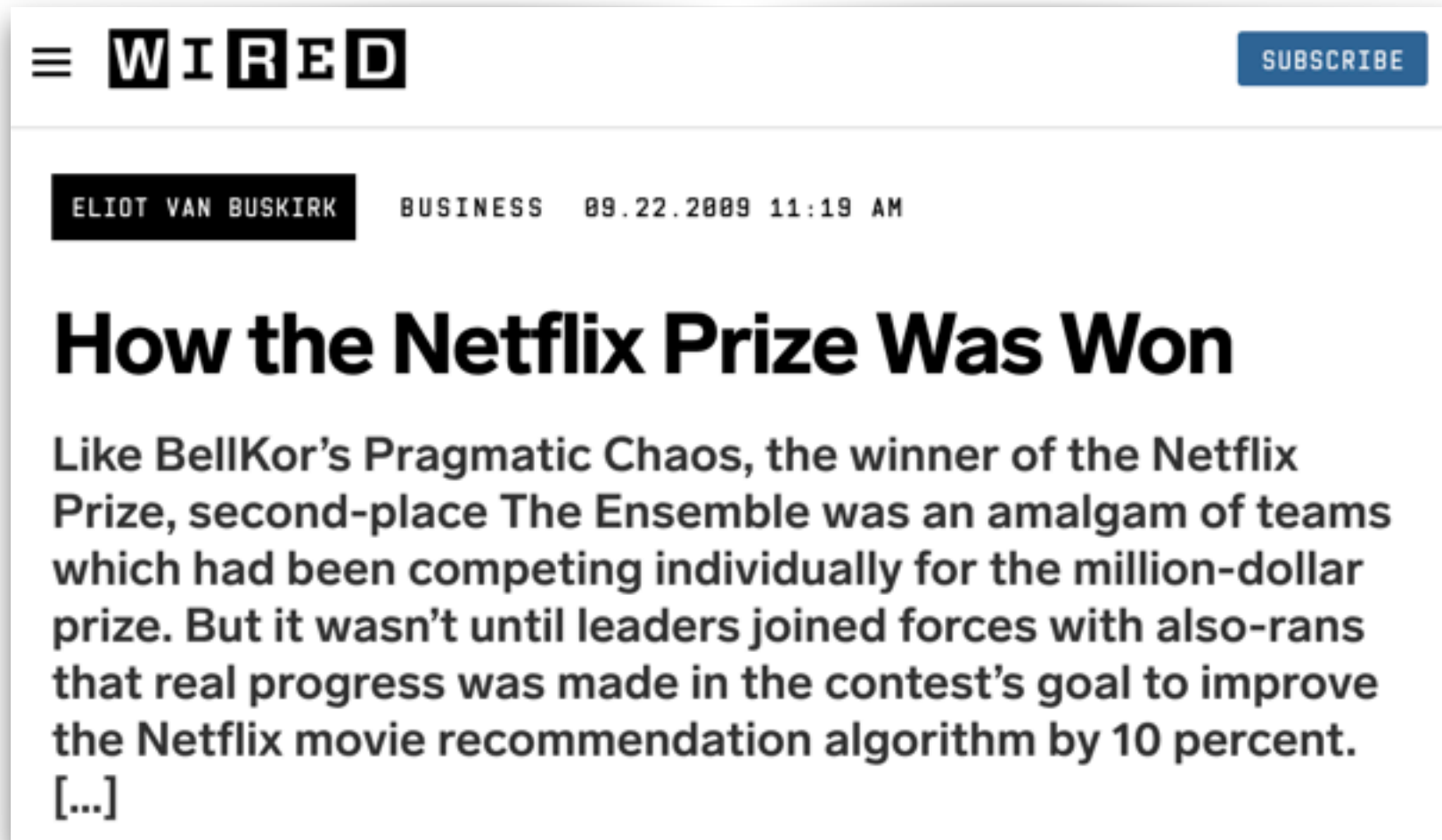
Ensembling

Ensembling

- Using multiple machine learning predictors to make one (ideally way-better) predictor

Ensembling

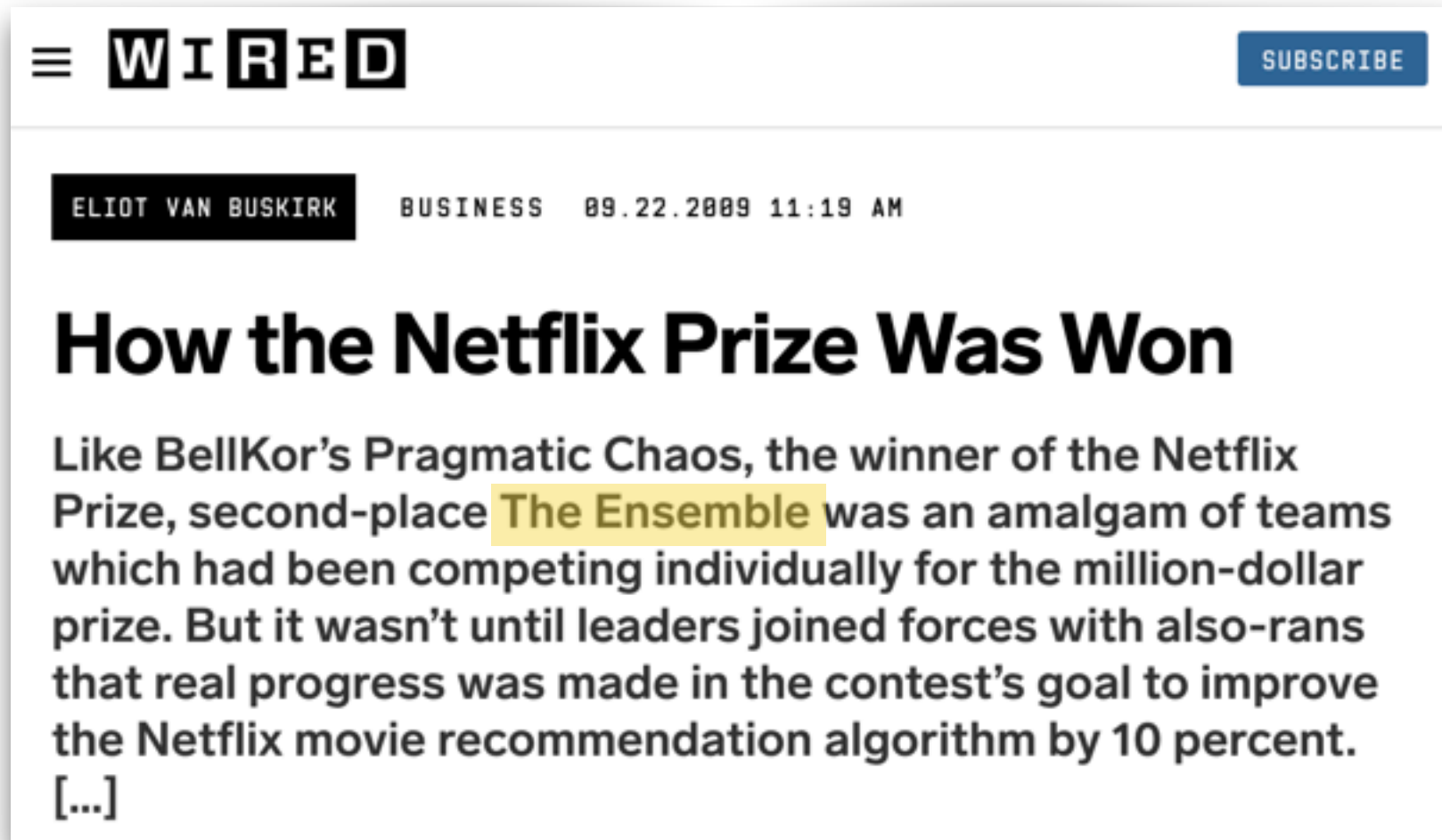
- Using multiple machine learning predictors to make one (ideally way-better) predictor



The image is a screenshot of a Wired article. At the top left is the Wired logo with a hamburger menu icon. To the right is a blue 'SUBSCRIBE' button. Below the logo is the author's name 'ELIOT VAN BUSKIRK', the category 'BUSINESS', and the date '09.22.2009 11:19 AM'. The main title is 'How the Netflix Prize Was Won'. The text below the title reads: '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 contest's goal to improve the Netflix movie recommendation algorithm by 10 percent. [...]'

Ensembling

- Using multiple machine learning predictors to make one (ideally way-better) predictor



The image is a screenshot of a Wired article. At the top left is the Wired logo, and at the top right is a blue 'SUBSCRIBE' button. Below the logo is the author's name 'ELIOT VAN BUSKIRK', the category 'BUSINESS', and the date '09.22.2009 11:19 AM'. The main title of the article is 'How the Netflix Prize Was Won'. The first paragraph of the article reads: '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 contest's goal to improve the Netflix movie recommendation algorithm by 10 percent. [...]'

Ensembling

- Using multiple machine learning predictors to make one (ideally way-better) predictor



The image shows a screenshot of a Wired article titled "How the Netflix Prize Was Won" by Eliot Van Buskirk, dated 09.22.2009. The article text states: "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". Below this, there is a snippet of a journal cover for the "International Journal of Forecasting" from Elsevier, Volume 36, Issue 1, January-March 2020, Pages 54-74. The journal title is partially obscured by another overlapping snippet that says "so-rans improve percent." Below the journal information, the title of a paper is visible: "The M4 Competition: 100,000 time series and 61 forecasting methods" by Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos.

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

so-rans improve percent.

International Journal of Forecasting

ELSEVIER 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

Ensembling

- Using multiple machine learning predictors to make one (ideally way-better) predictor

WIRED

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 models which had been competing individually for the million-dollar prize.



International Journal of
Forecasting

ELSEVIER 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

SUBSCRIBE

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.

[<https://www.wired.com/2009/09/how-the-netflix-prize-was-won/>;

11 <https://www.sciencedirect.com/science/article/pii/S0169207019301128>; <https://www.cdc.gov/coronavirus/2019-ncov/covid-data/forecasting-us.html>]

Ensembling

- Using multiple machine learning predictors to make one (ideally way-better) predictor

WIRED

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 Prize, second-place The Ensemble was an amalgam which had been competing individually for the million-dollar prize.



International Journal of
Forecasting

ELSEVIER 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

SUBSCRIBE

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.

[<https://www.wired.com/2009/09/how-the-netflix-prize-was-won/>;

11 <https://www.sciencedirect.com/science/article/pii/S0169207019301128>; <https://www.cdc.gov/coronavirus/2019-ncov/covid-data/forecasting-us.html>]

Bagging

Bagging

- One of multiple ways to make and use an ensemble

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating

Bagging

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

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from \mathcal{D}_n

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from \mathcal{D}_n

- Sampling with replacement

$(x^{(1)}, y^{(1)})$

$(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$

$(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from \mathcal{D}_n

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(x^{(2)}, y^{(2)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from \mathcal{D}_n

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(x^{(2)}, y^{(2)})$ $(x^{(3)}, y^{(3)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from \mathcal{D}_n

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(x^{(2)}, y^{(2)})$ $(x^{(3)}, y^{(3)})$

$(x^{(5)}, y^{(5)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from \mathcal{D}_n

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(x^{(2)}, y^{(2)})$ $(x^{(3)}, y^{(3)})$

$(x^{(5)}, y^{(5)})$ $(x^{(2)}, y^{(2)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from \mathcal{D}_n

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(x^{(2)}, y^{(2)})$ $(x^{(3)}, y^{(3)})$

$(x^{(5)}, y^{(5)})$ $(x^{(2)}, y^{(2)})$

$(x^{(2)}, y^{(2)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from \mathcal{D}_n

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$

$(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$

$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - 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 $\tilde{\mathcal{D}}_n^{(b)}$

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$

$(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$

$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - 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

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$

$(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$

$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - 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:

- Sampling with replacement

$$(x^{(1)}, y^{(1)}) \quad (x^{(2)}, y^{(2)})$$

$$(x^{(3)}, y^{(3)}) \quad (x^{(4)}, y^{(4)})$$

$$(x^{(5)}, y^{(5)})$$

$$(\tilde{x}^{(1)}, \tilde{y}^{(1)}) \quad (\tilde{x}^{(2)}, \tilde{y}^{(2)})$$

$$(\tilde{x}^{(3)}, \tilde{y}^{(3)}) \quad (\tilde{x}^{(4)}, \tilde{y}^{(4)})$$

$$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - 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: the predictor

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$

$(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$

$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - 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: the predictor $\hat{f}_{\text{bag}}(x) =$

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$

$(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$

$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - 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: the predictor $\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{(b)}(x)$

- Sampling with replacement

$$(x^{(1)}, y^{(1)}) \quad (x^{(2)}, y^{(2)})$$

$$(x^{(3)}, y^{(3)}) \quad (x^{(4)}, y^{(4)})$$

$$(x^{(5)}, y^{(5)})$$

$$(\tilde{x}^{(1)}, \tilde{y}^{(1)}) \quad (\tilde{x}^{(2)}, \tilde{y}^{(2)})$$

$$(\tilde{x}^{(3)}, \tilde{y}^{(3)}) \quad (\tilde{x}^{(4)}, \tilde{y}^{(4)})$$

$$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - 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: the predictor $\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{(b)}(x)$
 - Classification: predictor at a point is class with highest vote count at that point

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$

$(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$

$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Bagging

- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regating
 - Training data \mathcal{D}_n
 - For $b = 1, \dots, B$
 - 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 $\tilde{\mathcal{D}}_n^{(b)}$

aggregating

- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$

$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

$(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$

$(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$

$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

- For regression: the predictor $\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{(b)}(x)$
- Classification: predictor at a point is class with highest vote count at that point