6.036/6.862: Introduction to Machine Learning

Lecture: starts Tuesdays 9:35am (Boston time zone)

Course website: introml.odl.mit.edu

Who's talking? Prof. Tamara Broderick

Questions? discourse.odl.mit.edu ("Lecture 12" category)

Materials: Will all be available at course website

Last Time(s)

I. Neural networks
II. Convolutional neural nets
III. Recurrent 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

[https://www.pbs.org/newshour/show/how-some-election-officials-are-trying-to-verify-the-vote-more-easily]

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

[https://www.pbs.org/newshour/show/how-some-election-officials-are-trying-to-verify-the-vote-more-easily;
Stark, Annals of Statistics, 2008]

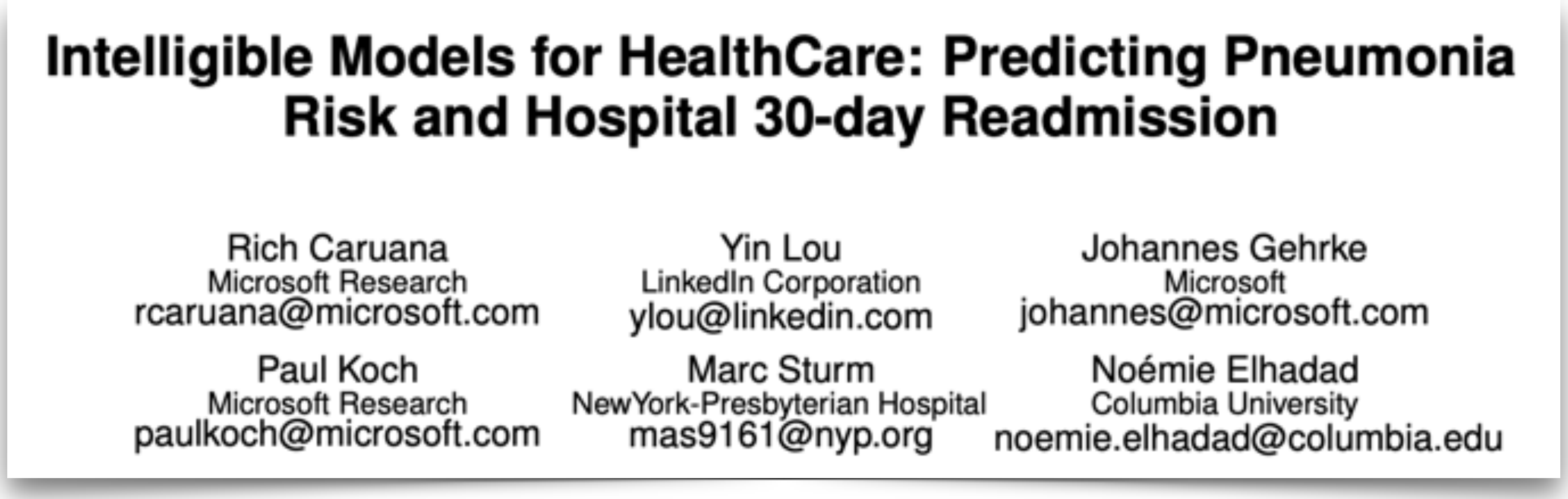
Predictive performance and beyond

- 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

Yin Lou
LinkedIn Corporation
ylou@linkedin.com

Johannes Gehrke
Microsoft
johannes@microsoft.com

Paul Koch
Microsoft Research
paulkoch@microsoft.com

Marc Sturm
NewYork-Presbyterian Hospital
mas9161@nyp.org

Noémie Elhadad
Columbia University
noemie.elhadad@columbia.edu

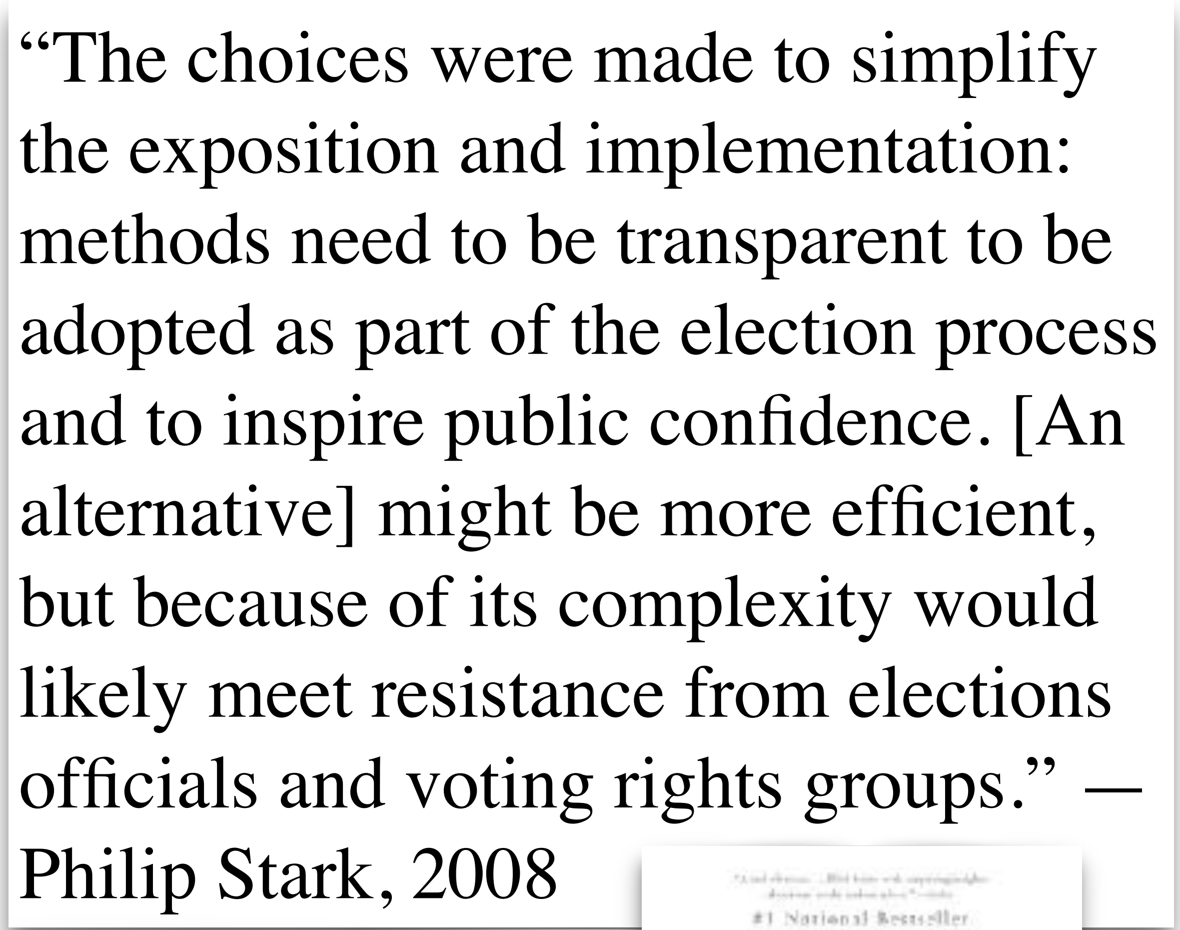
[https://www.pbs.org/newshour/show/how-some-election-officials-are-trying-to-verify-the-vote-more-easily; Stark, Annals of Statistics, 2008; Caruana et al, KDD, 2015]

Predictive performance and beyond

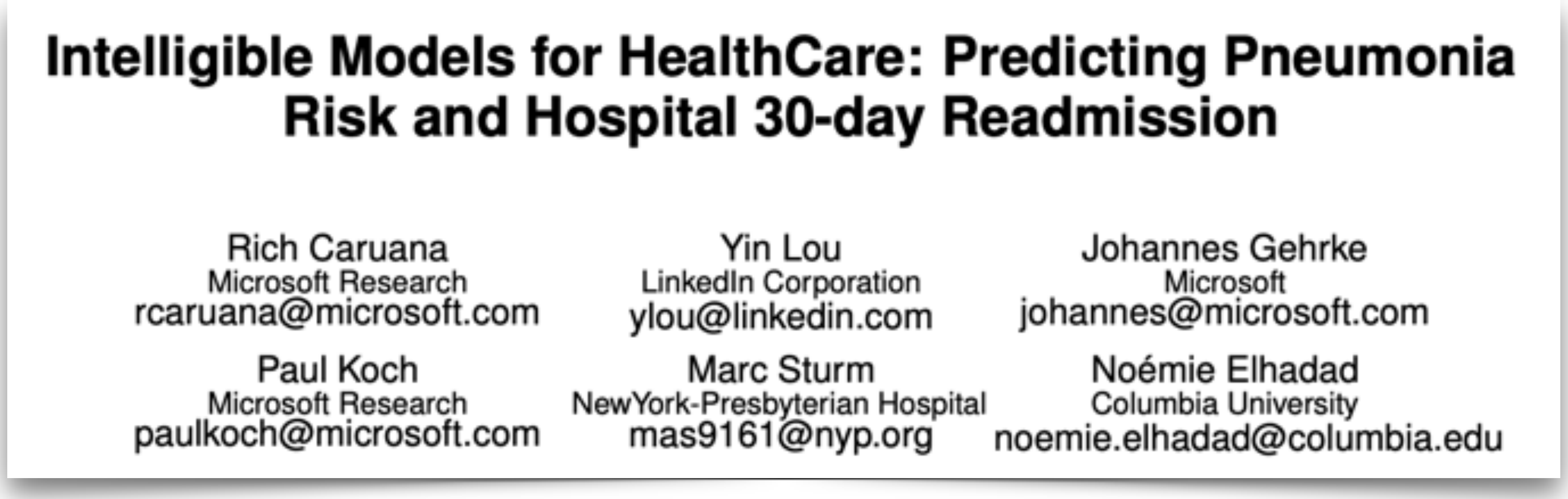
- There’s more to machine learning than predictive performance

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

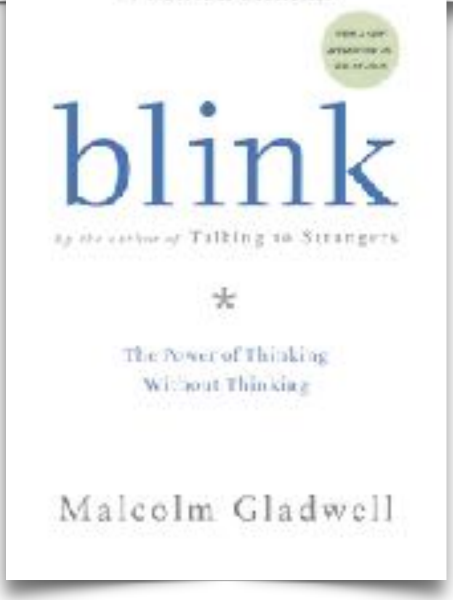
Yin Lou
LinkedIn Corporation
ylou@linkedin.com

Johannes Gehrke
Microsoft
johannes@microsoft.com

Paul Koch
Microsoft Research
paulkoch@microsoft.com

Marc Sturm
NewYork-Presbyterian Hospital
mas9161@nyp.org

Noémie Elhadad
Columbia University
noemie.elhadad@columbia.edu



#1 National Bestseller

blink

by the author of Talking to Strangers

The Power of Thinking Without Thinking

Malcolm Gladwell

[https://www.pbs.org/newshour/show/how-some-election-officials-are-trying-to-verify-the-vote-more-easily; Stark, Annals of Statistics, 2008; Caruana et al, KDD, 2015; https://www.gladwellbooks.com/titles/malcolm-gladwell/blink/9780316005043/]

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

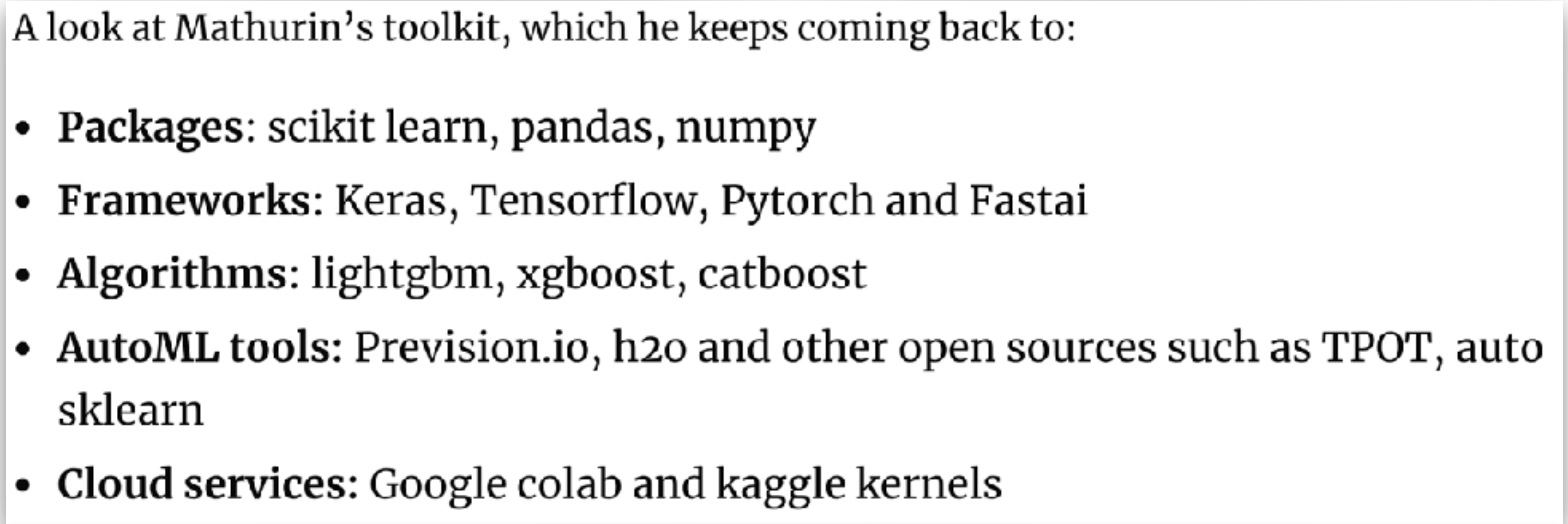
- Even if all you care about is vanilla predictive performance, you should care about trees

kaggle

Predictive performance and beyond

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

kaggle



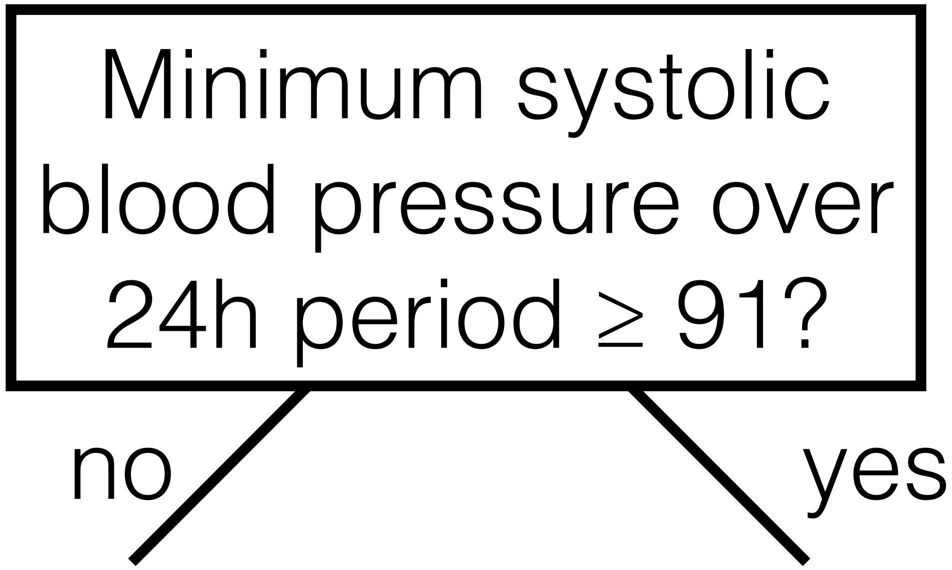
A look at Mathurin's toolkit, which he keeps coming back to:

- **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

[https://analyticsindiamag.com/kaggle-master-mathurin-ache-competition-data-science/]

Decision tree

Decision tree



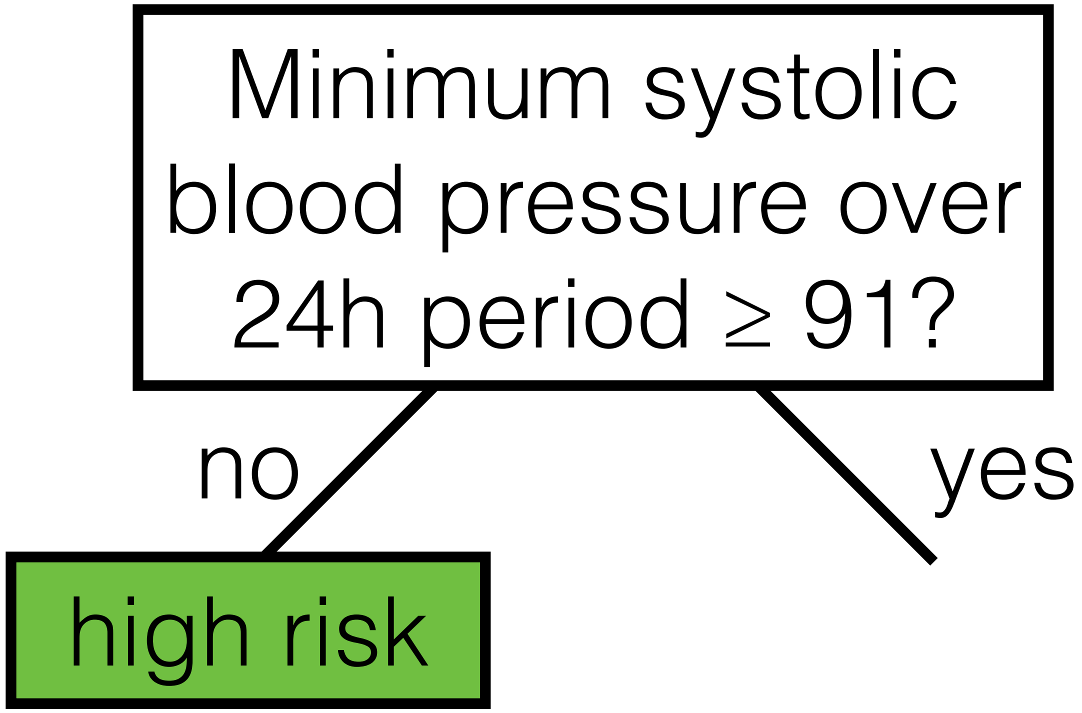
Minimum systolic blood pressure over 24h period ≥ 91?

no

yes

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree



Minimum systolic blood pressure over 24h period ≥ 91?

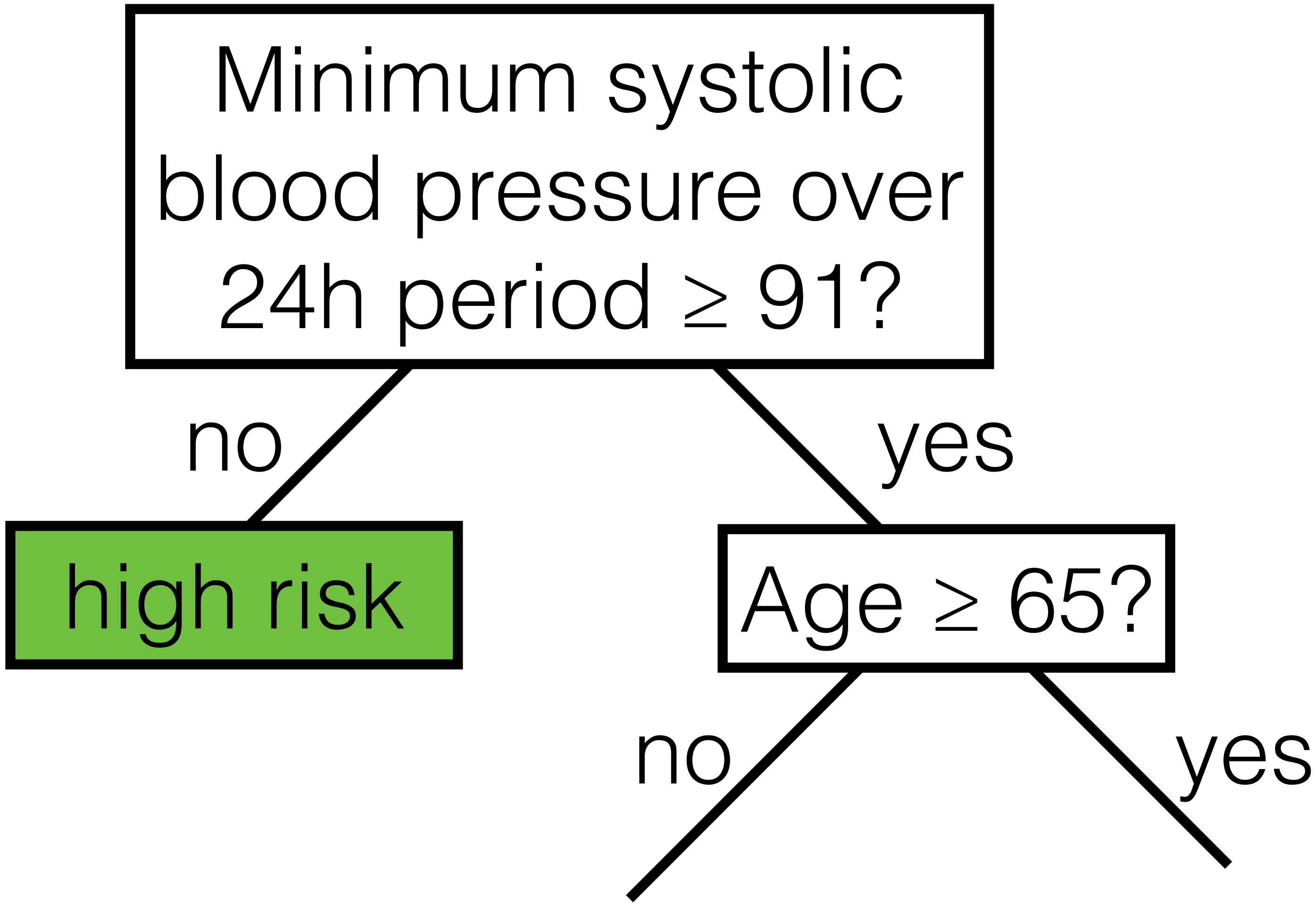
no

yes

high risk

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree



Minimum systolic blood pressure over 24h period ≥ 91?

no

high risk

yes

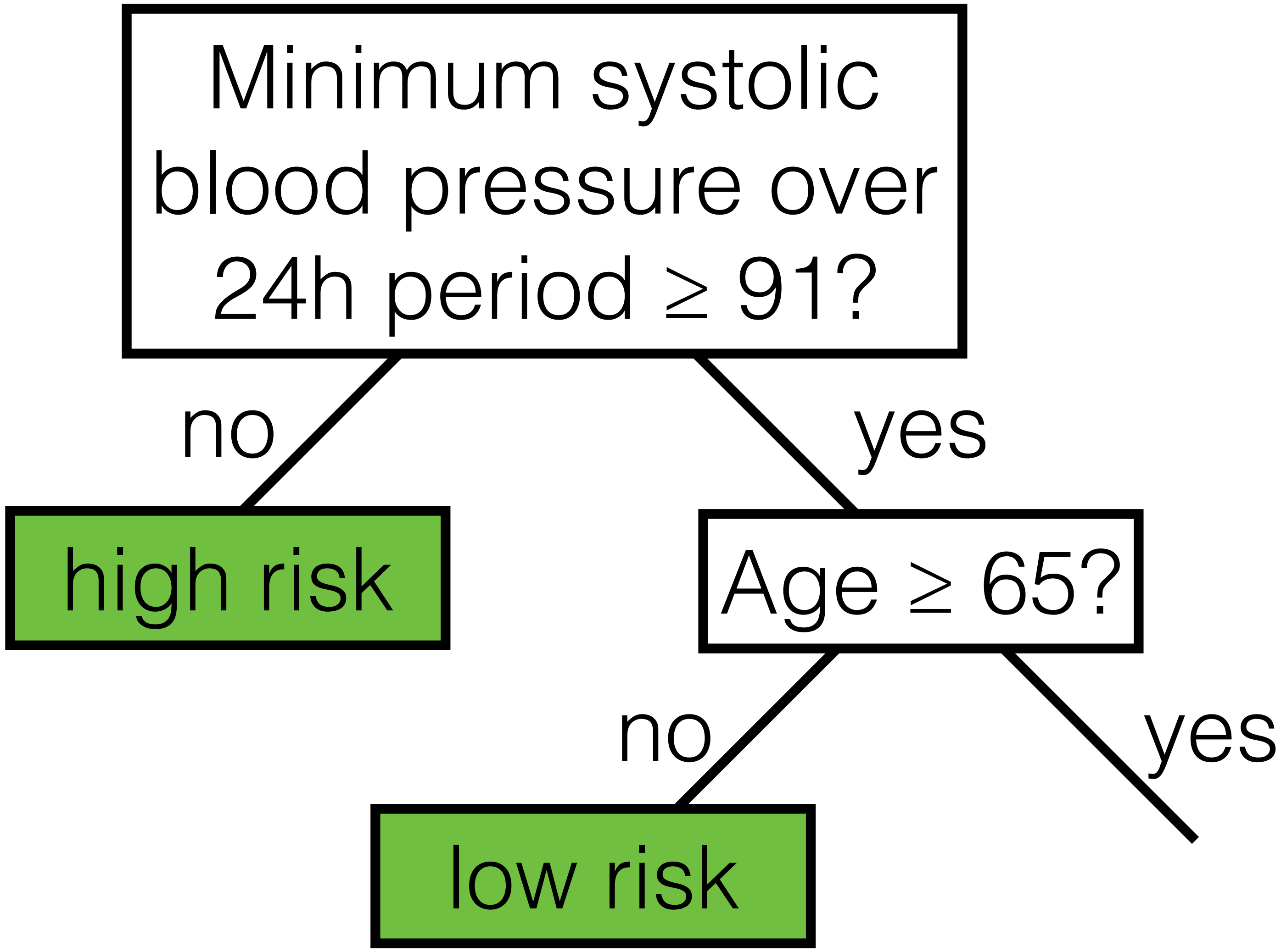
Age ≥ 65?

no

yes

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree



Minimum systolic blood pressure over 24h period ≥ 91?

no

high risk

yes

Age ≥ 65?

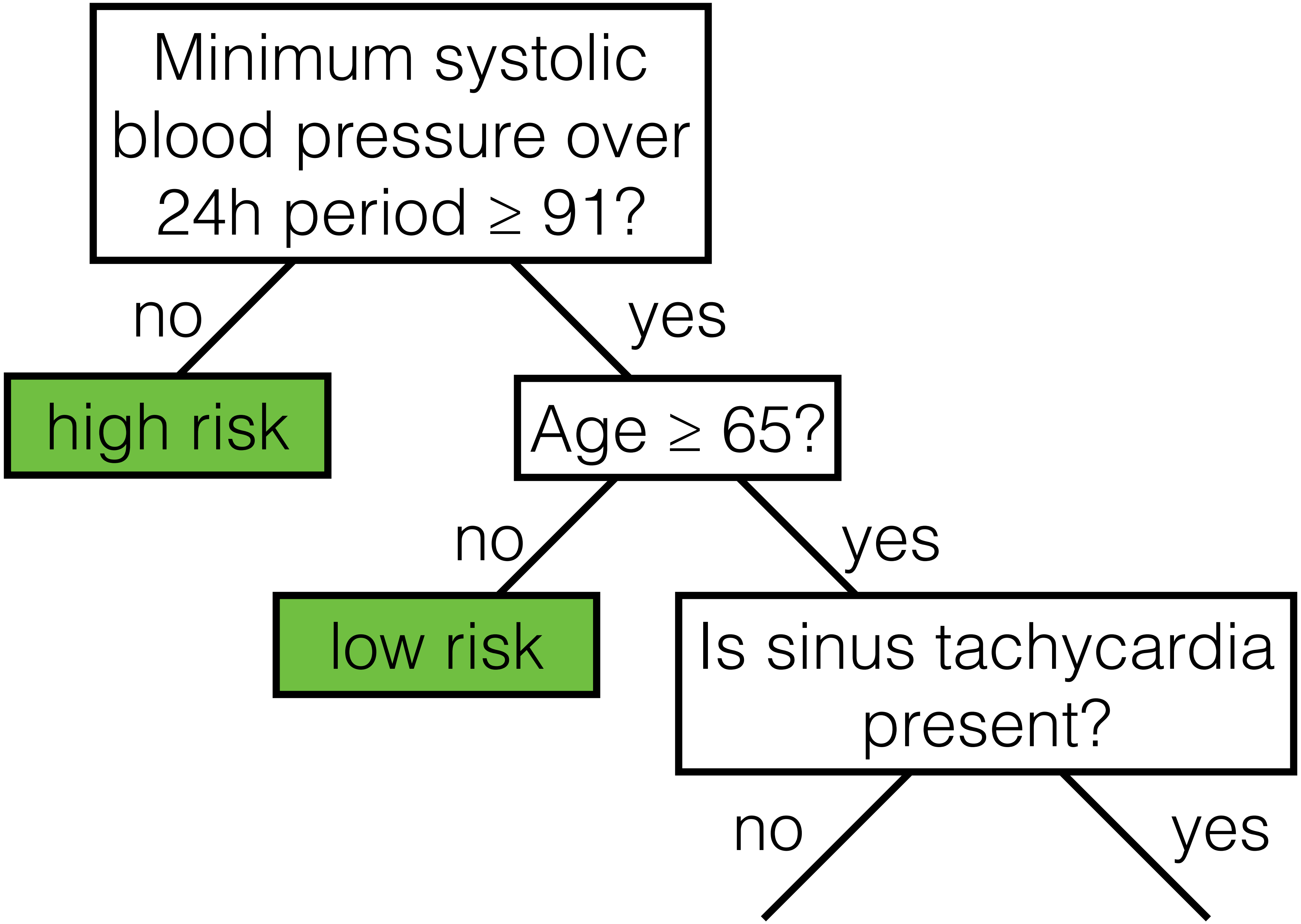
no

low risk

yes

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree



Minimum systolic blood pressure over 24h period ≥ 91?

no

high risk

yes

Age ≥ 65?

no

low risk

yes

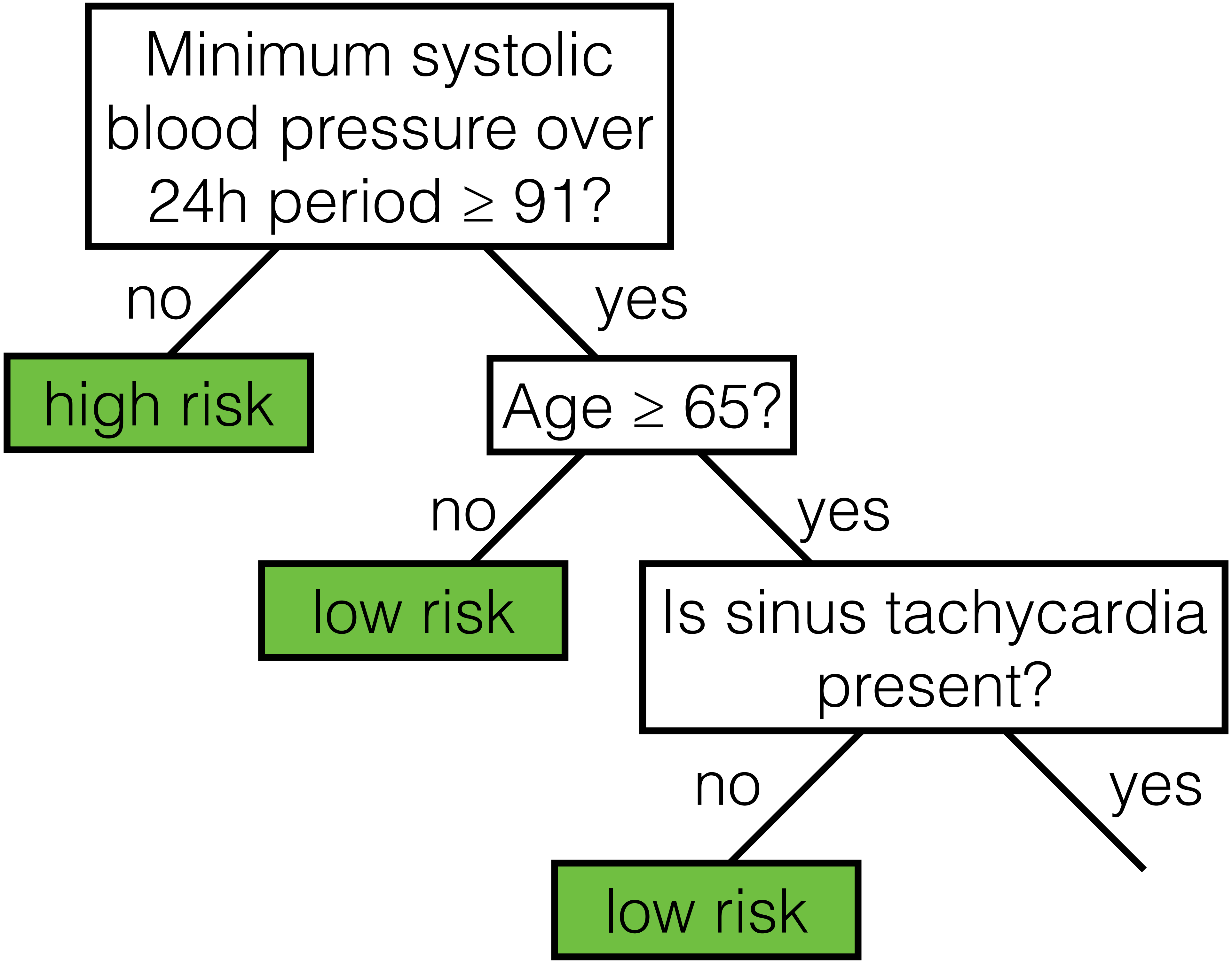
Is sinus tachycardia present?

no

yes

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree



Minimum systolic blood pressure over 24h period ≥ 91?

no

high risk

yes

Age ≥ 65?

no

low risk

yes

Is sinus tachycardia present?

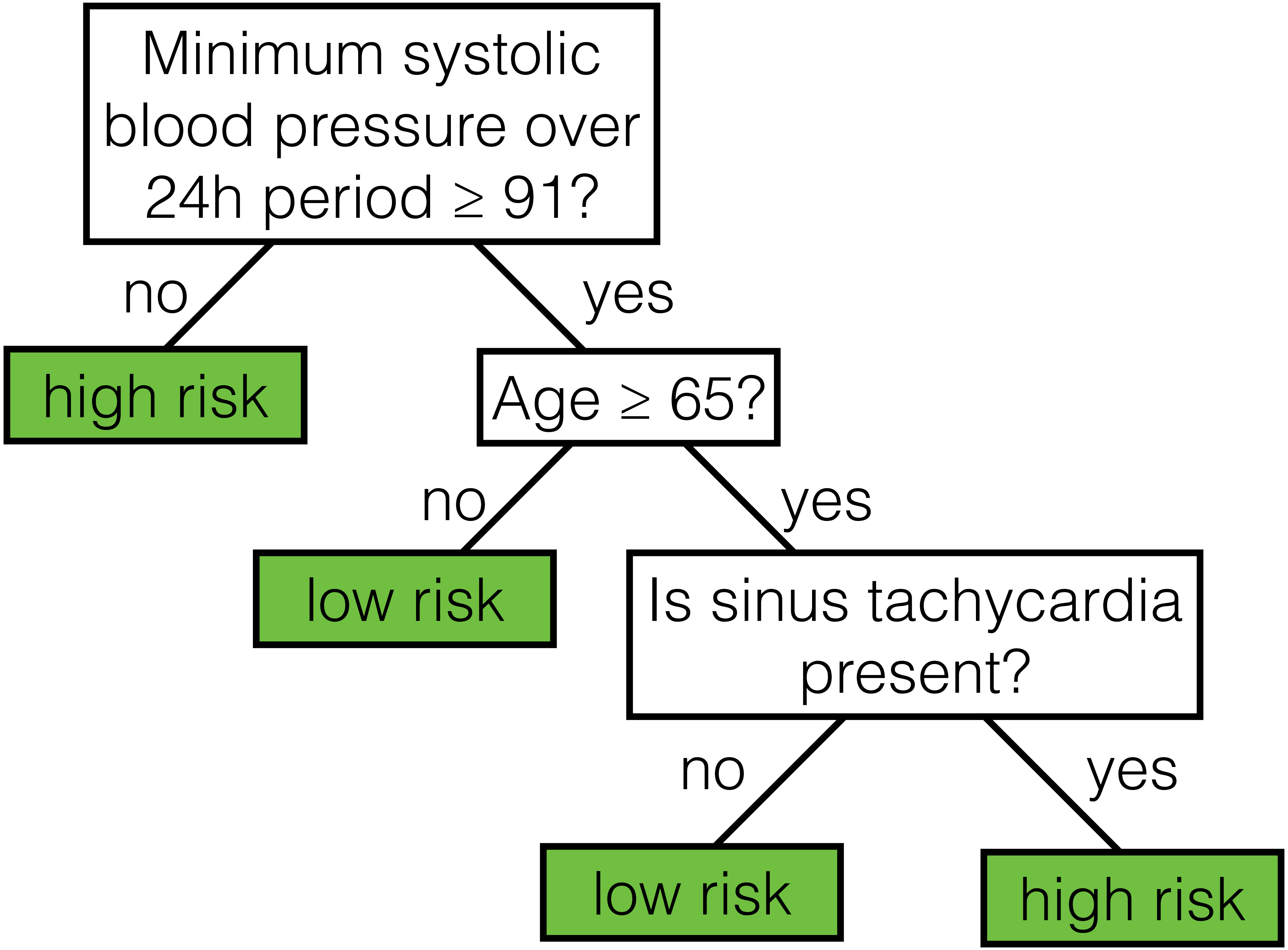
no

low risk

yes

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree


Minimum systolic blood pressure over 24h period ≥ 91?

no → high risk

yes → Age ≥ 65?

no → low risk

yes → Is sinus tachycardia present?

no → low risk

yes → high risk

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree

features:

x_1: date

x_2: age

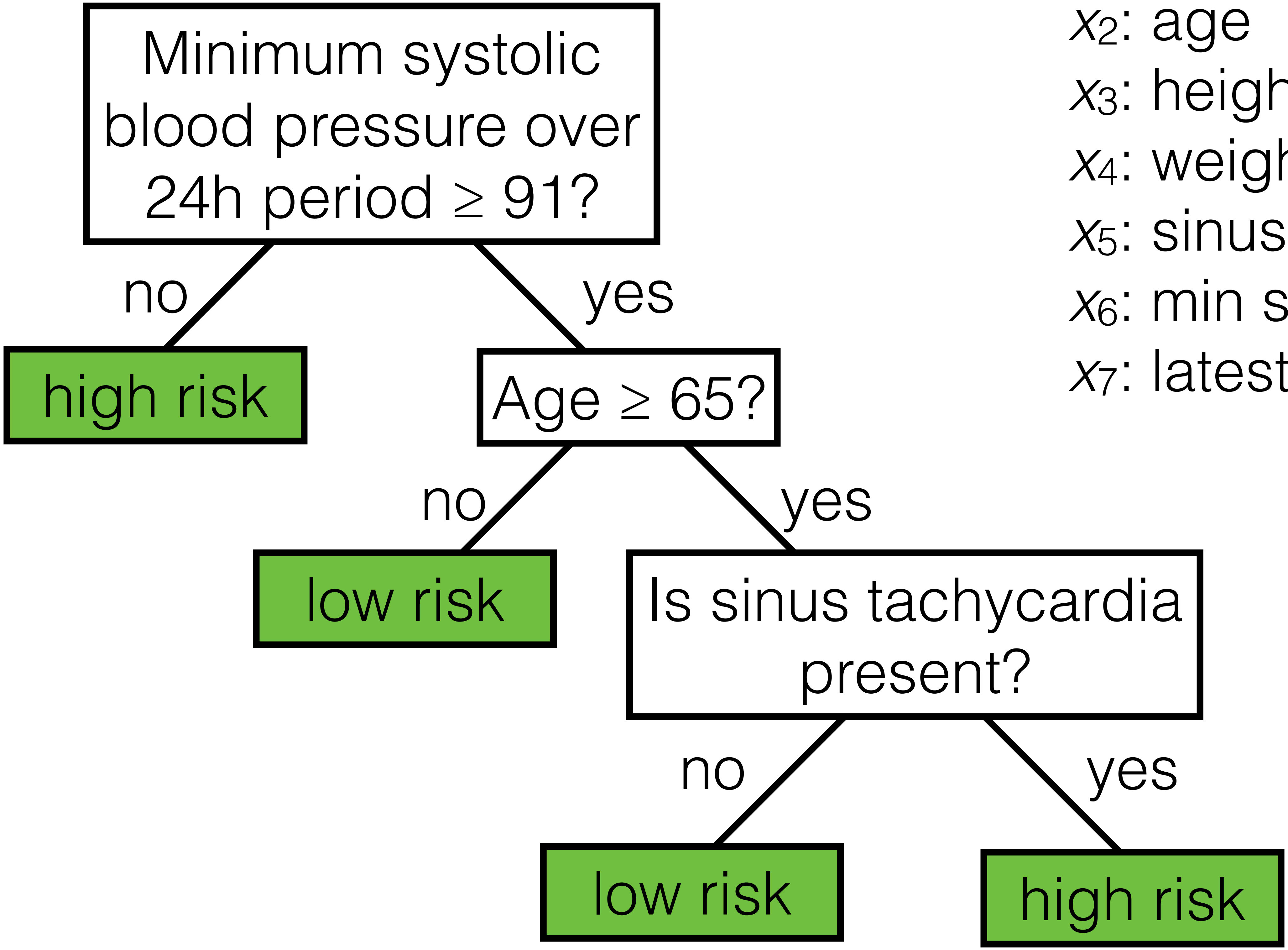
x_3: height

x_4: weight

x_5: sinus tachycardia?

x_6: min systolic bp, 24h

x_7: latest diastolic bp



Minimum systolic blood pressure over 24h period ≥ 91?

no

high risk

yes

Age ≥ 65?

no

low risk

yes

Is sinus tachycardia present?

no

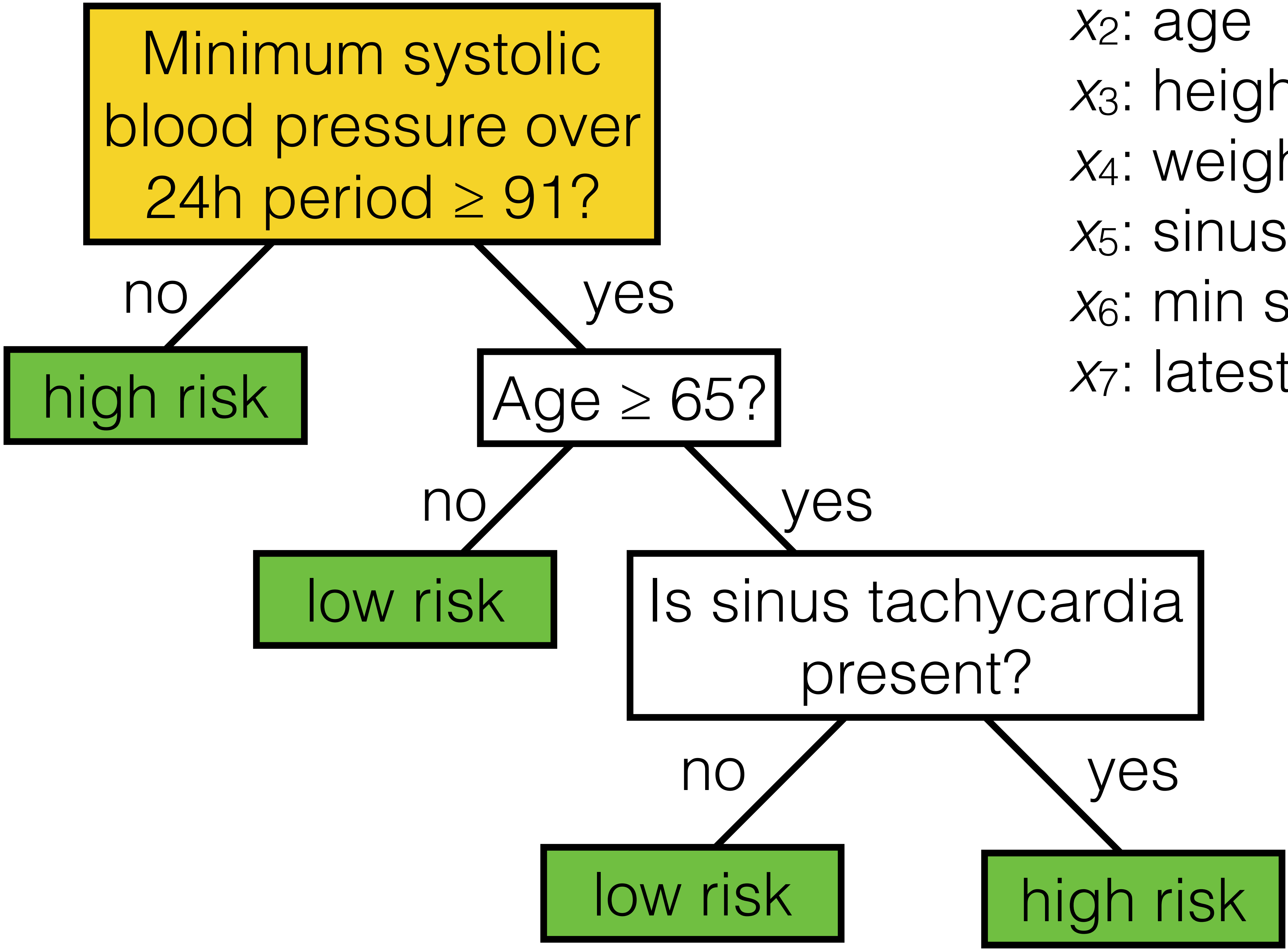
low risk

yes

high risk

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree



Minimum systolic blood pressure over 24h period ≥ 91?

no

high risk

yes

Age ≥ 65?

no

low risk

yes

Is sinus tachycardia present?

no

low risk

yes

high risk

features:

x_1: date

x_2: age

x_3: height

x_4: weight

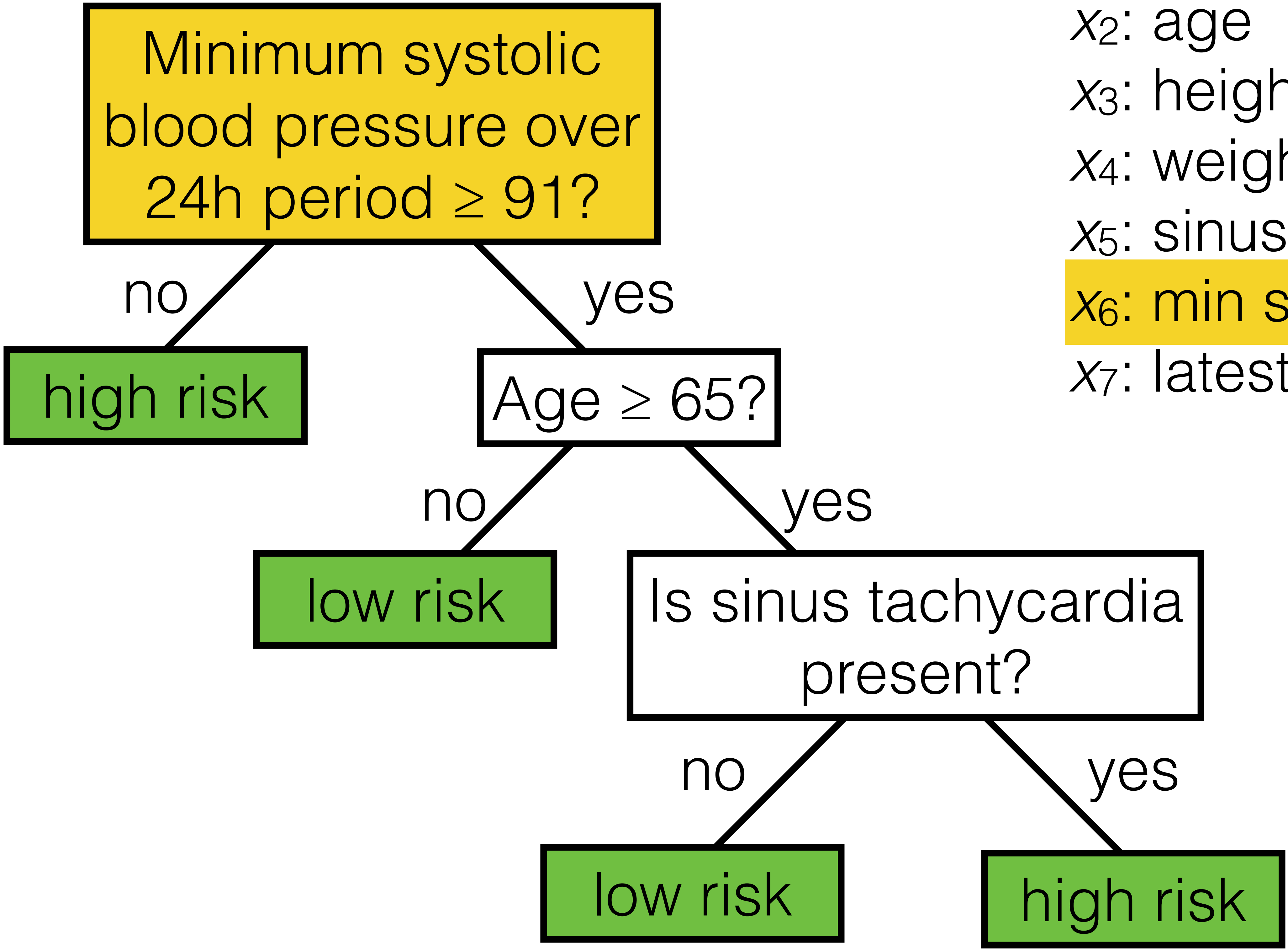
x_5: sinus tachycardia?

x_6: min systolic bp, 24h

x_7: latest diastolic bp

[Breiman, Friedman, Olshen, Stone, 1984]

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

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree

features:

x_1: date

x_2: age

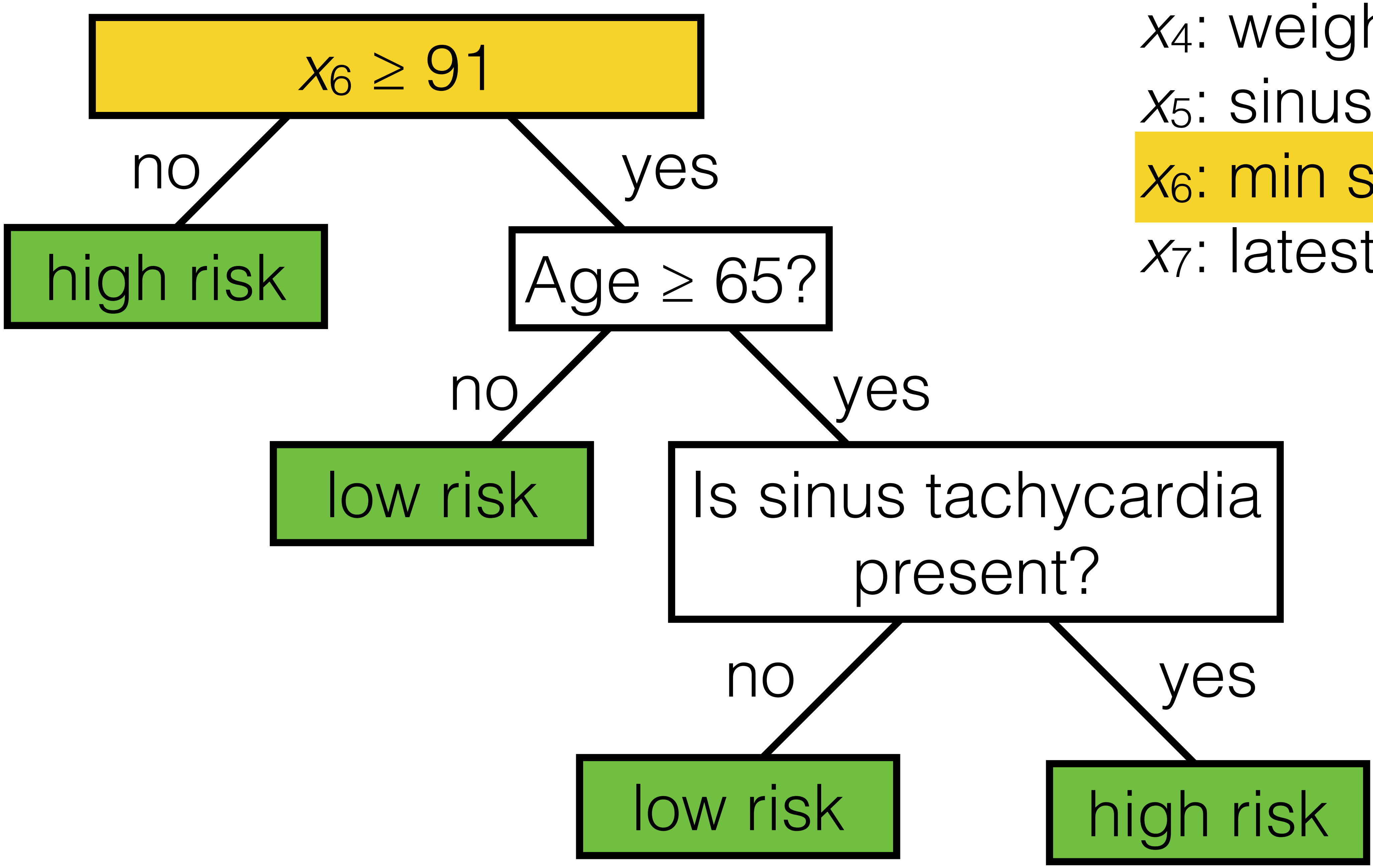
x_3: height

x_4: weight

x_5: sinus tachycardia?

x_6: min systolic bp, 24h

x_7: latest diastolic bp



$x_6 \ge 91$

no

high risk

yes

Age $\ge$ 65?

no

low risk

yes

Is sinus tachycardia present?

no

low risk

yes

high risk

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree

features:

x_1: date

x_2: age

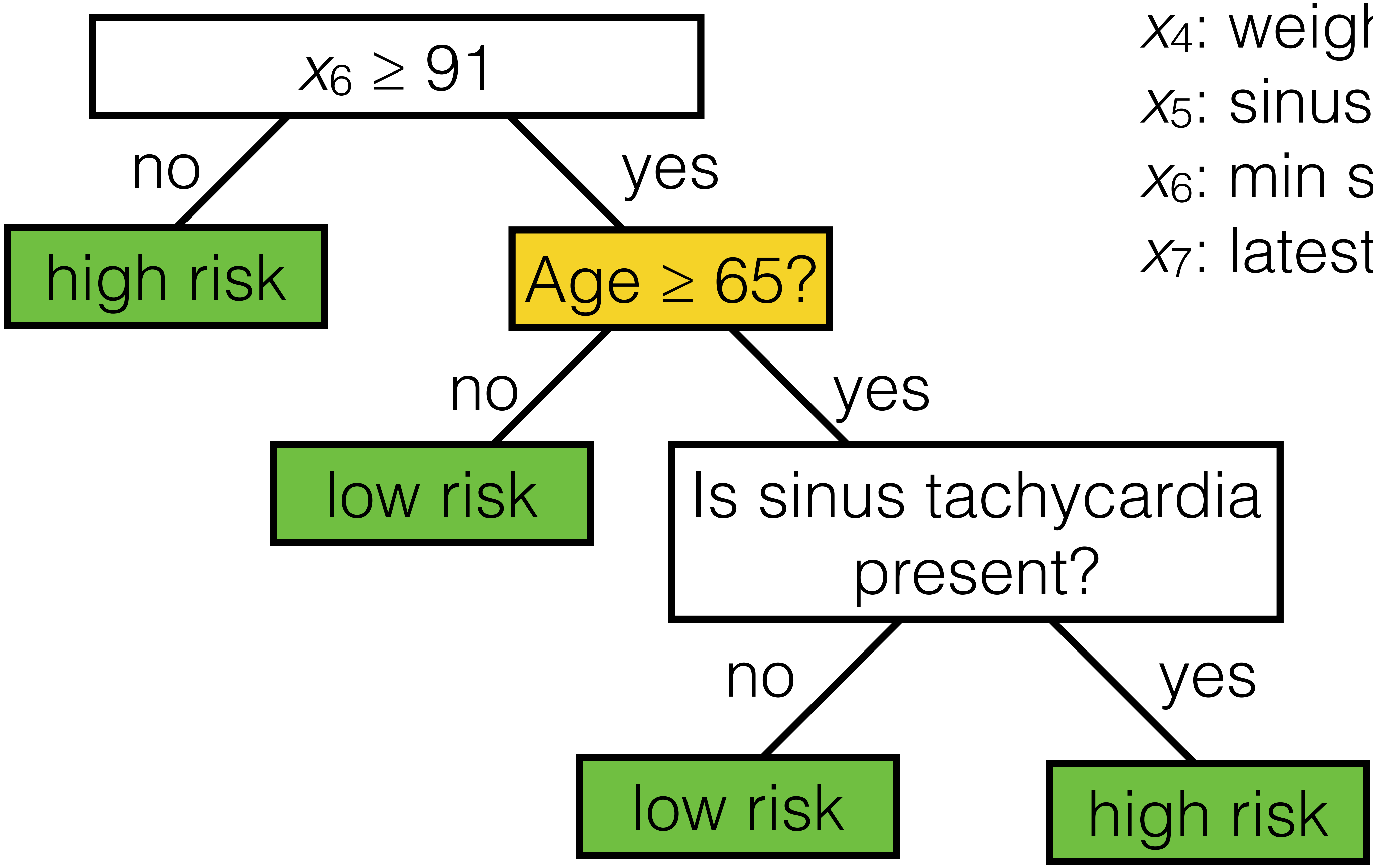
x_3: height

x_4: weight

x_5: sinus tachycardia?

x_6: min systolic bp, 24h

x_7: latest diastolic bp



[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree

features:

x_1: date

x_2: age

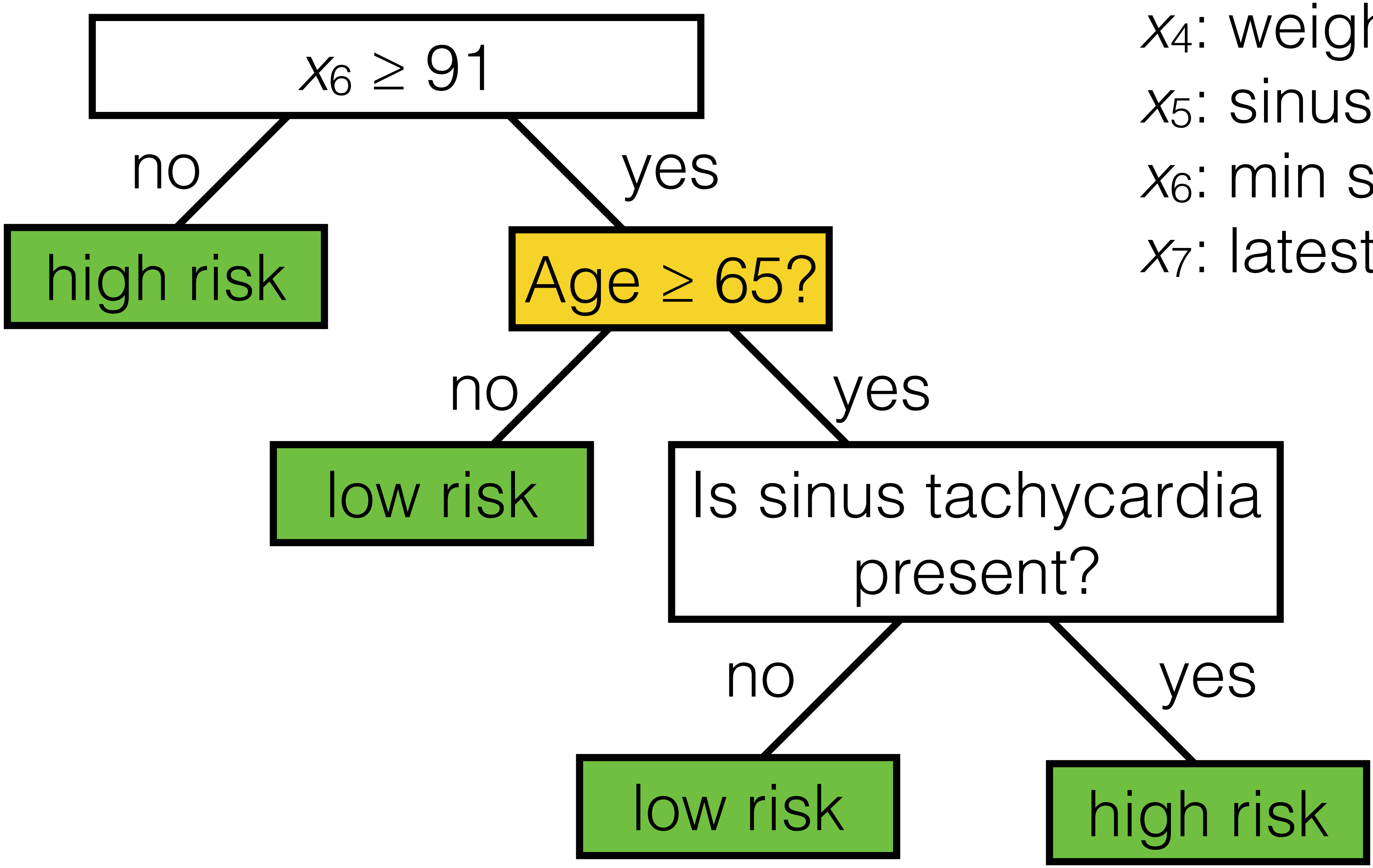
x_3: height

x_4: weight

x_5: sinus tachycardia?

x_6: min systolic bp, 24h

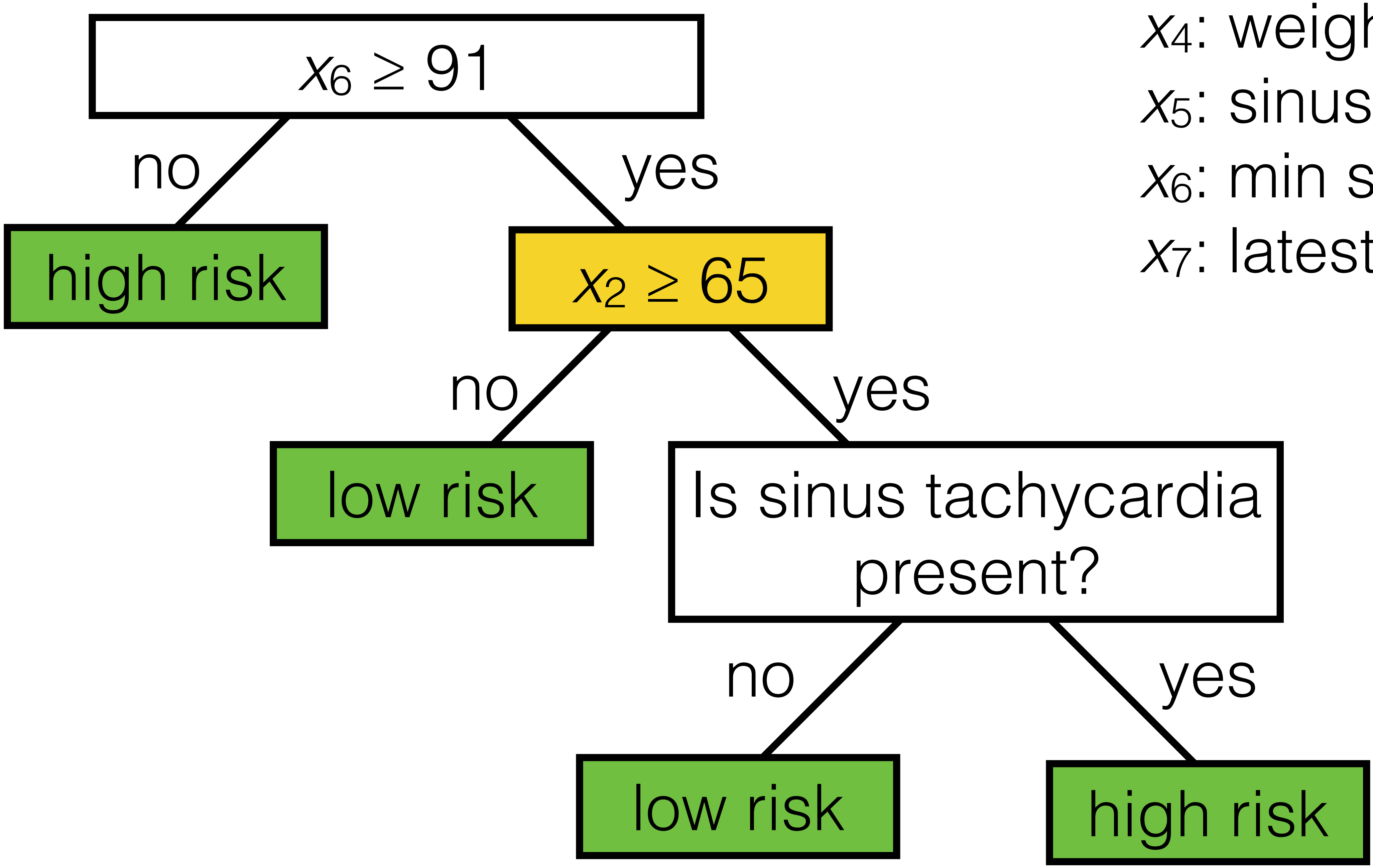
x_7: latest diastolic bp



[Breiman, Friedman, Olshen, Stone, 1984]

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



$x_6 \geq 91$

no

high risk

yes

$x_2 \geq 65$

no

low risk

yes

Is sinus tachycardia present?

no

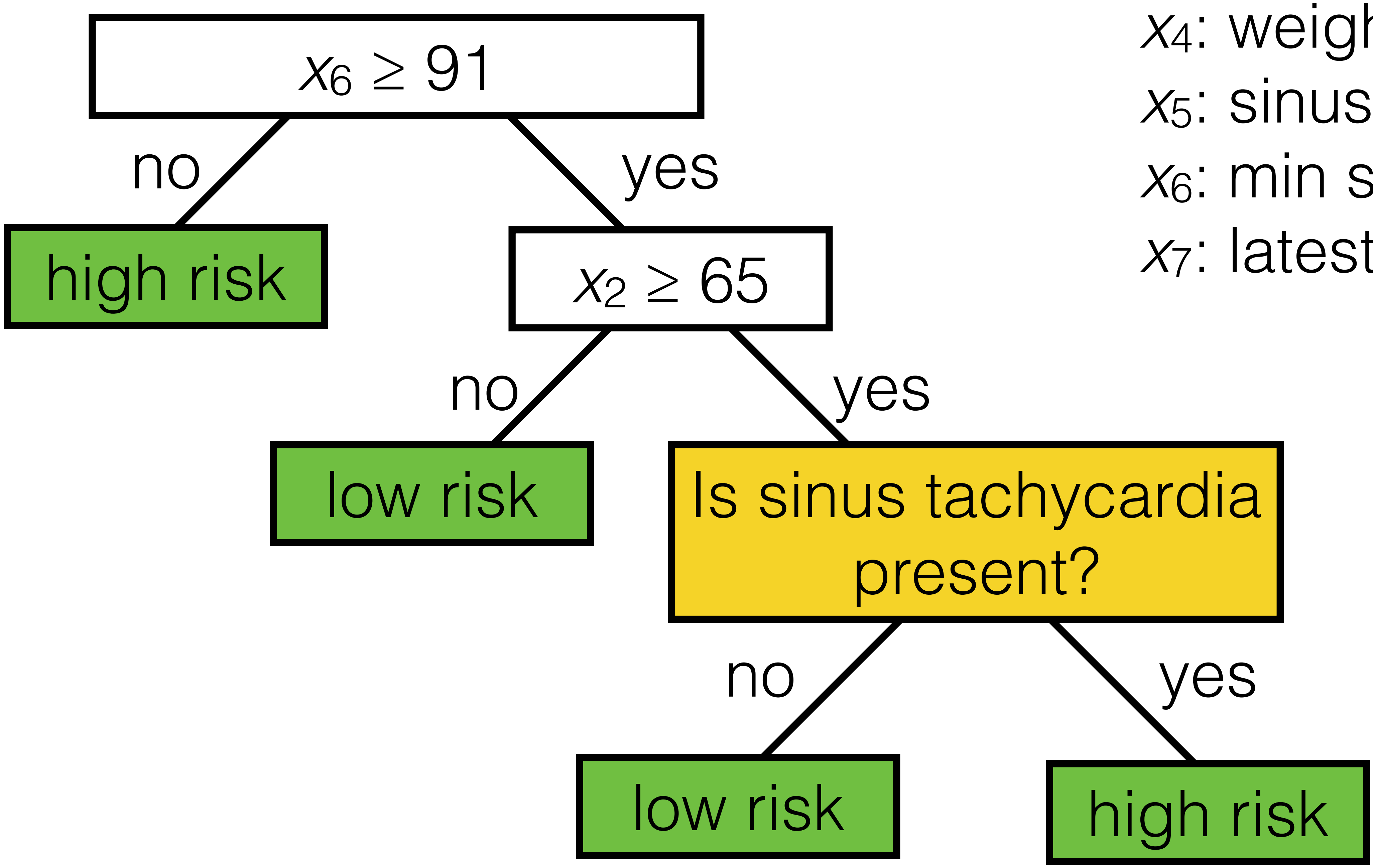
low risk

yes

high risk

[Breiman, Friedman, Olshen, Stone, 1984]

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

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree

features:

x_1: date

x_2: age

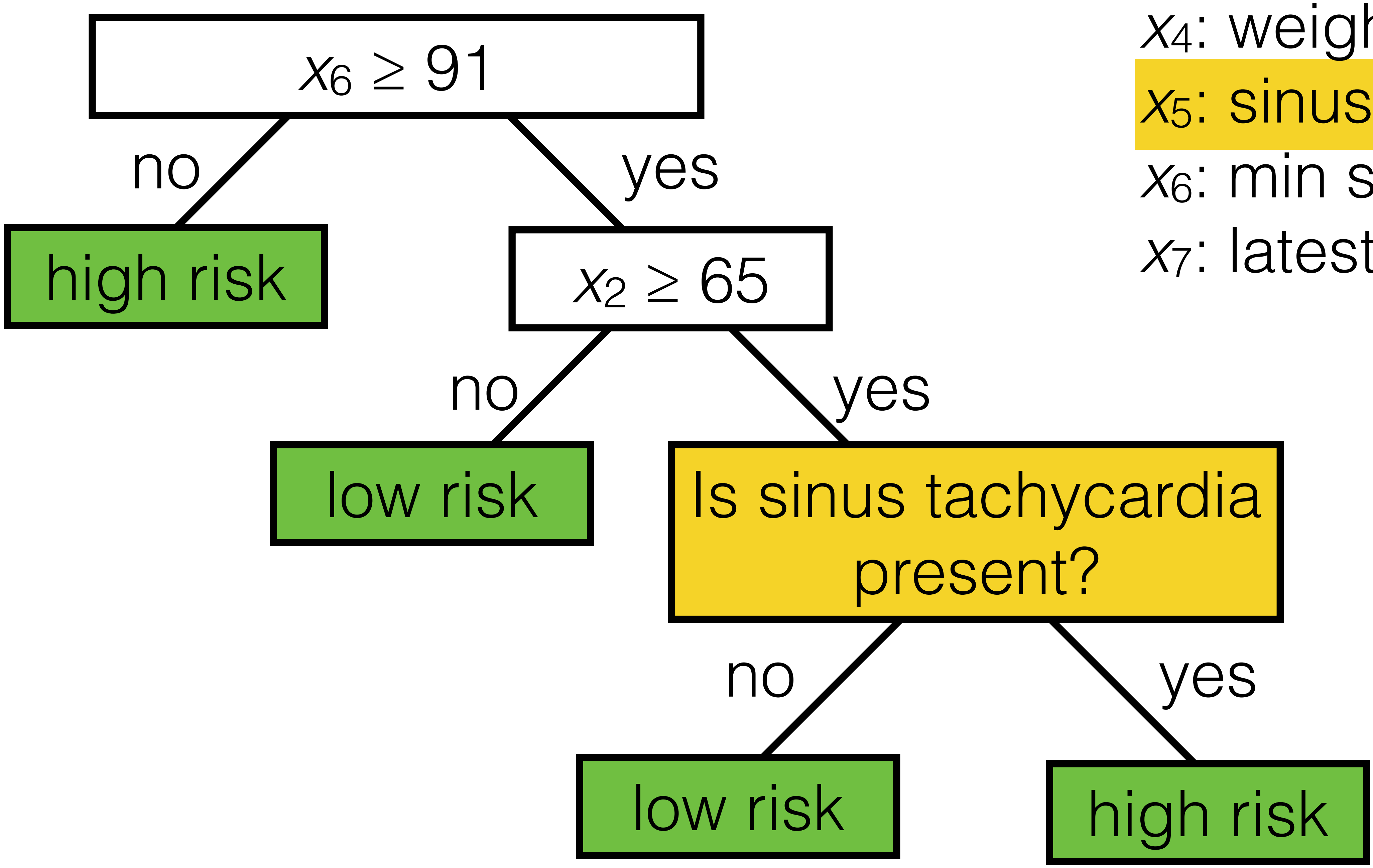
x_3: height

x_4: weight

x_5: sinus tachycardia?

x_6: min systolic bp, 24h

x_7: latest diastolic bp



$x_6 \geq 91$

no

high risk

yes

$x_2 \geq 65$

no

low risk

yes

Is sinus tachycardia present?

no

low risk

yes

high risk

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree

features:

x_1: date

x_2: age

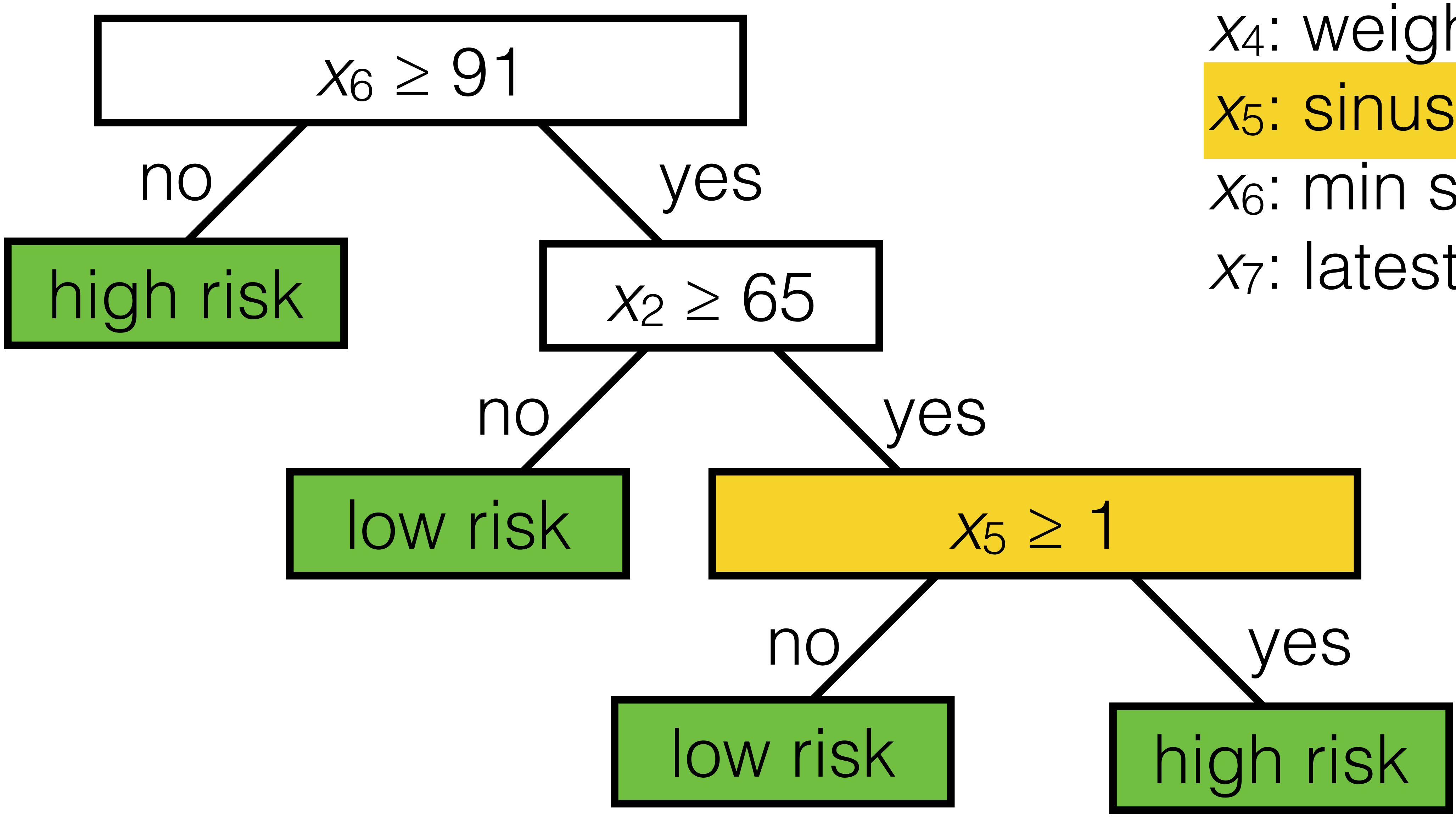
x_3: height

x_4: weight

x_5: sinus tachycardia?

x_6: min systolic bp, 24h

x_7: latest diastolic bp



[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree

features:

x_1: date

x_2: age

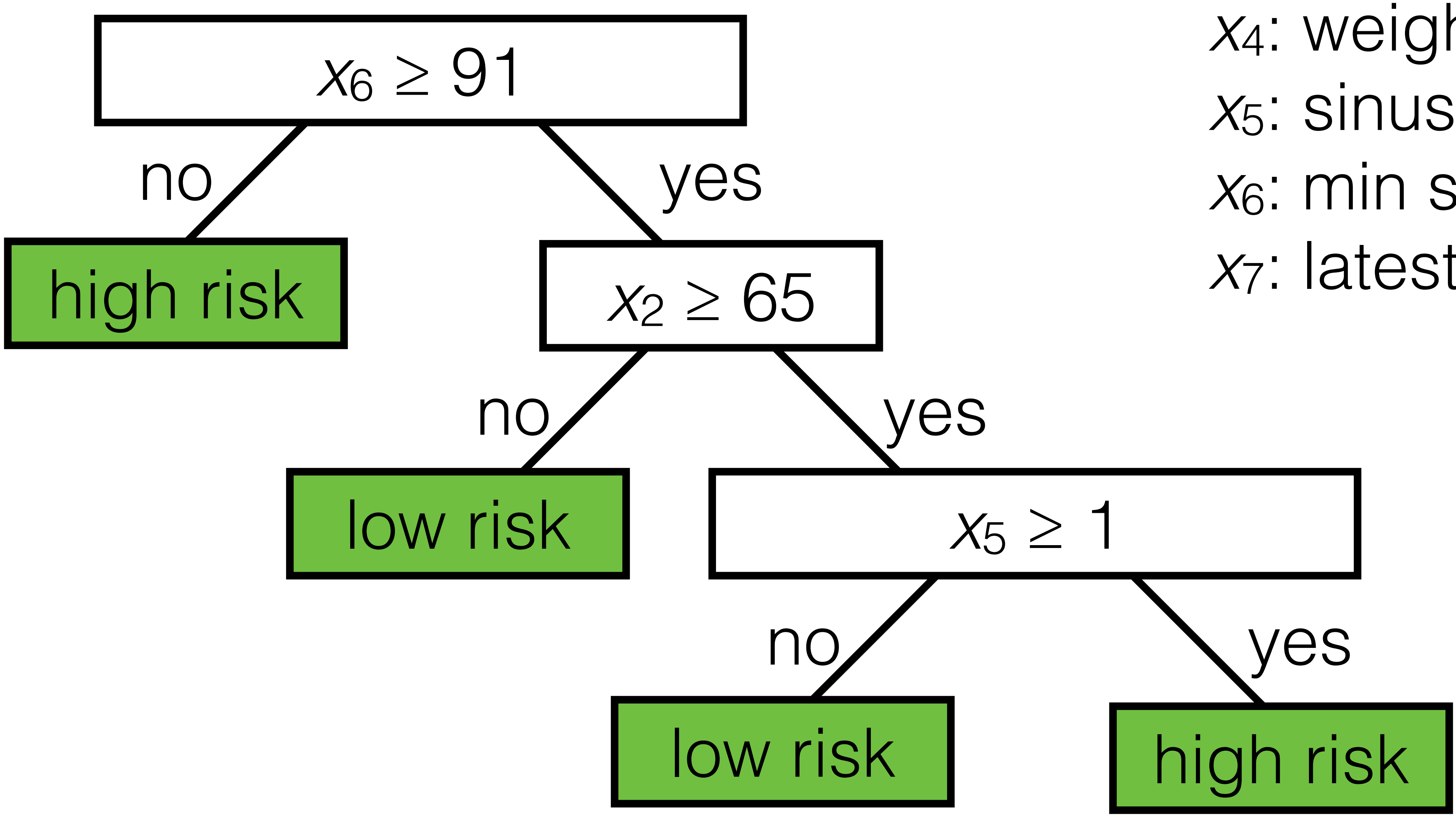
x_3: height

x_4: weight

x_5: sinus tachycardia?

x_6: min systolic bp, 24h

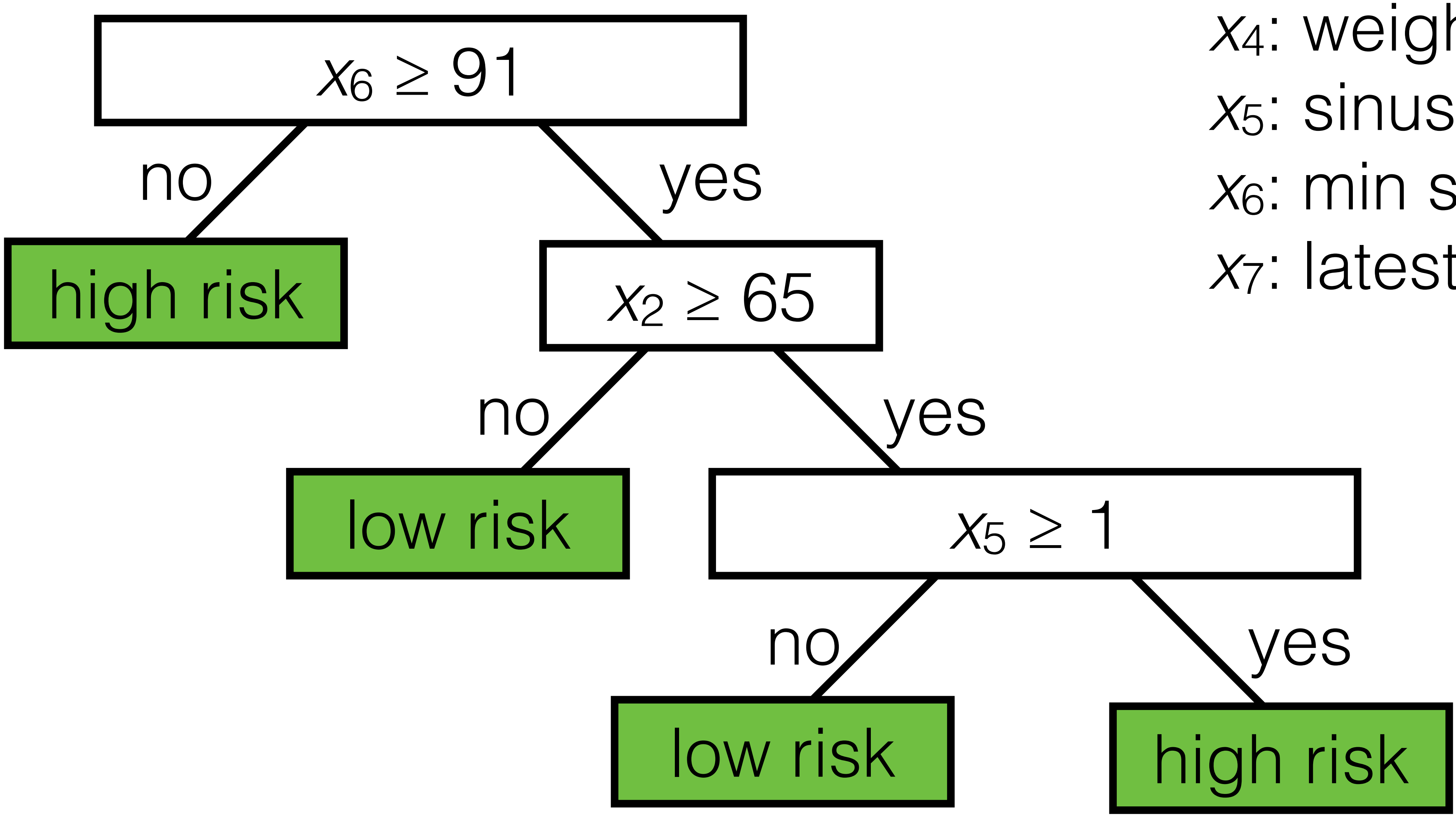
x_7: latest diastolic bp



[Breiman, Friedman, Olshen, Stone, 1984]

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



$x_6 \geq 91$

no

yes

high risk

$x_2 \geq 65$

no

yes

low risk

$x_5 \geq 1$

no

yes

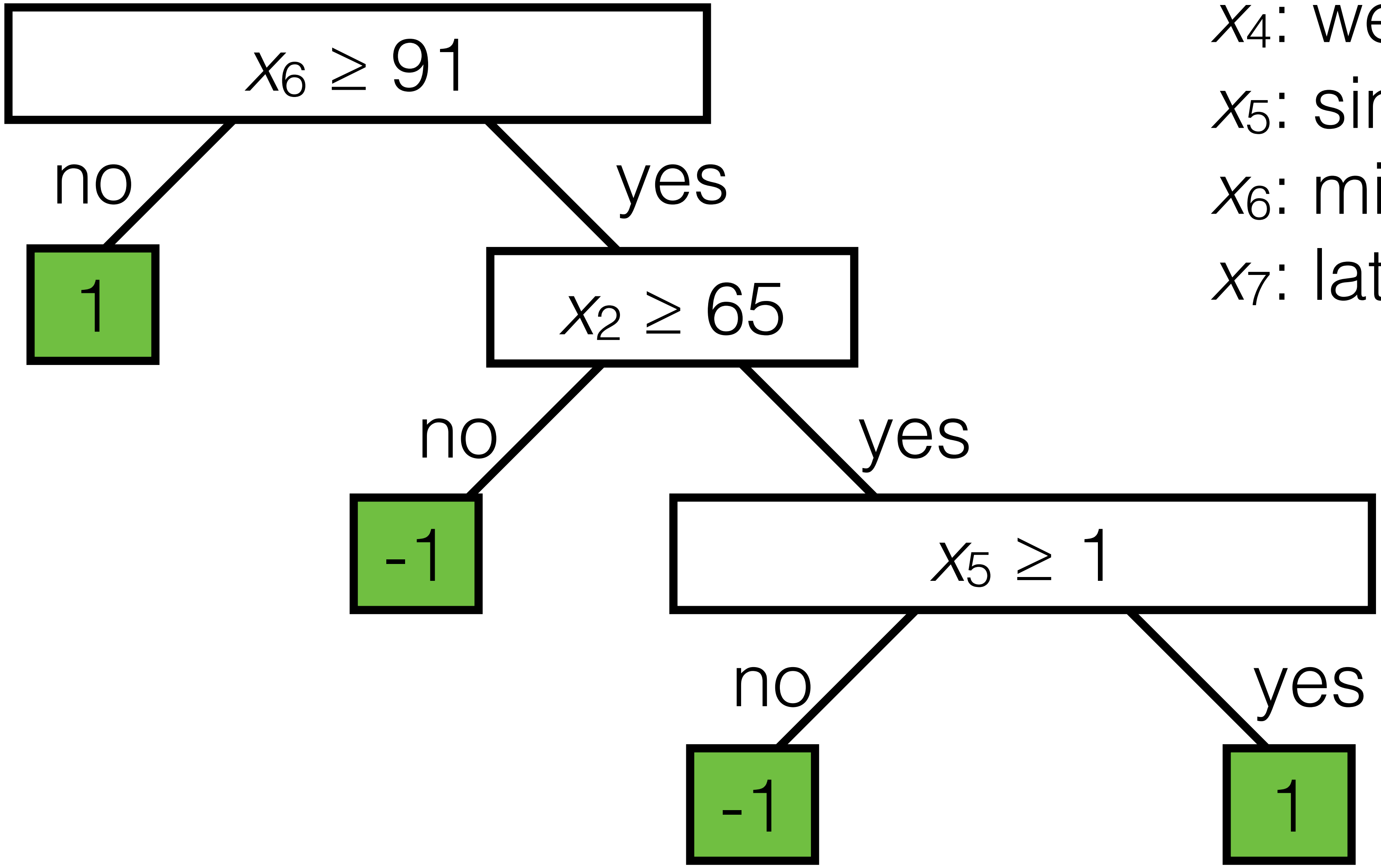
low risk

high risk

labels y:
1: high risk
-1: low risk

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree

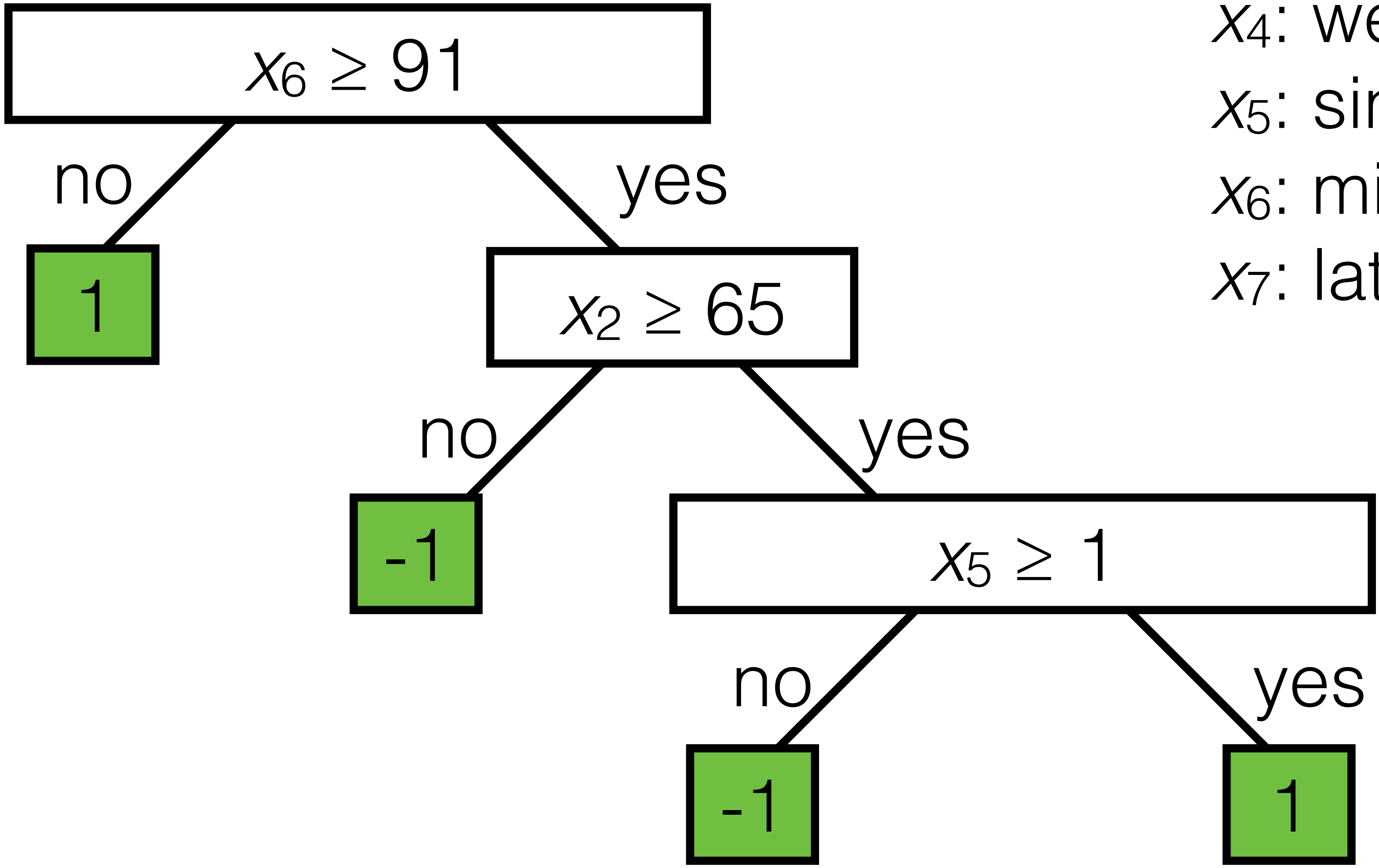


features:
x_1: date
x_2: age
x_3: height
x_4: weight
x_5: sinus tachycardia?
x_6: min systolic bp, 24h
x_7: latest diastolic bp

labels y:
1: high risk
-1: low risk

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree



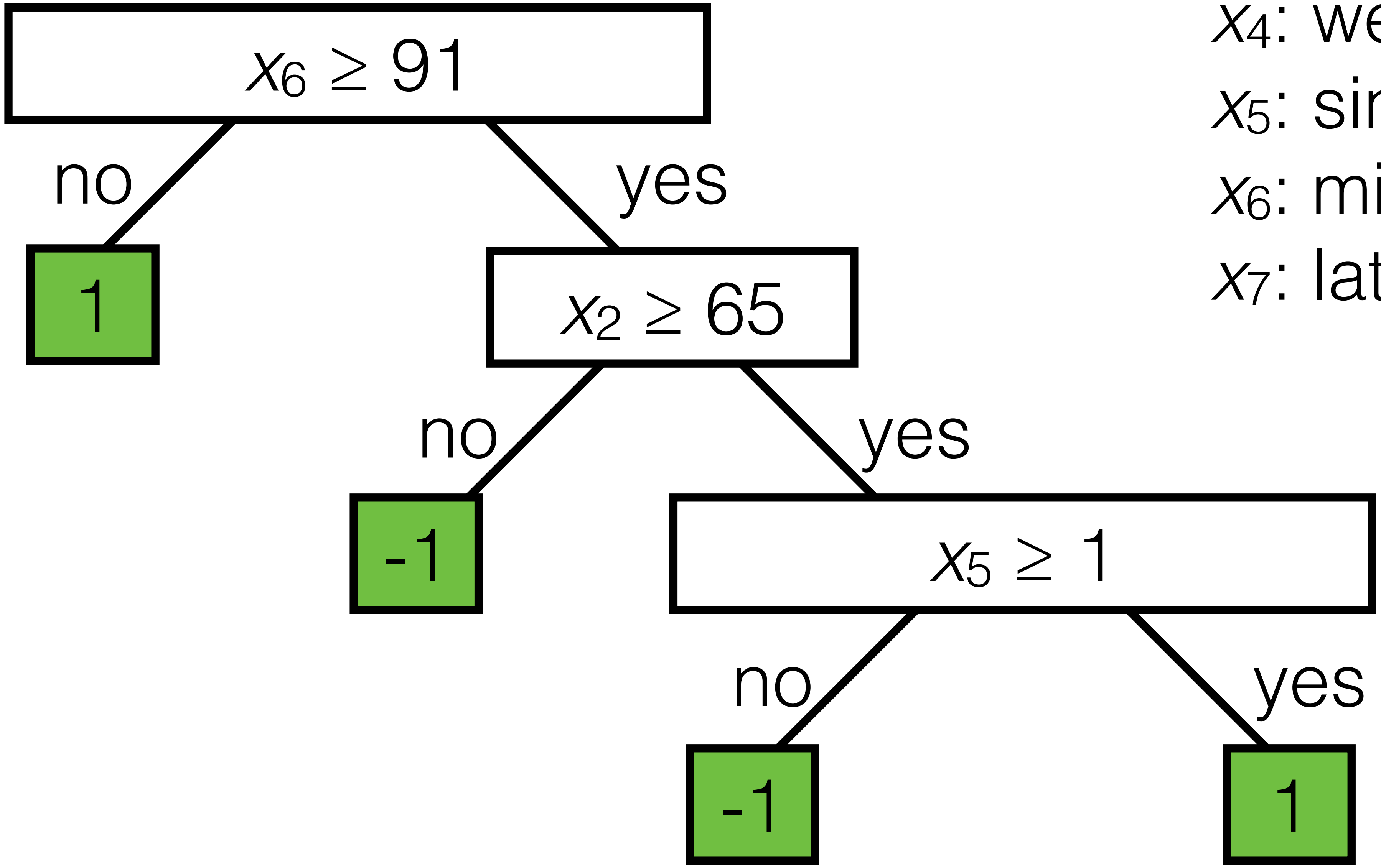
features:
x_1: date
x_2: age
x_3: height
x_4: weight
x_5: sinus tachycardia?
x_6: min systolic bp, 24h
x_7: latest diastolic bp

labels y:
1: high risk
-1: low risk

[Breiman, Friedman, Olshen, Stone, 1984]

Decision tree

internal node:



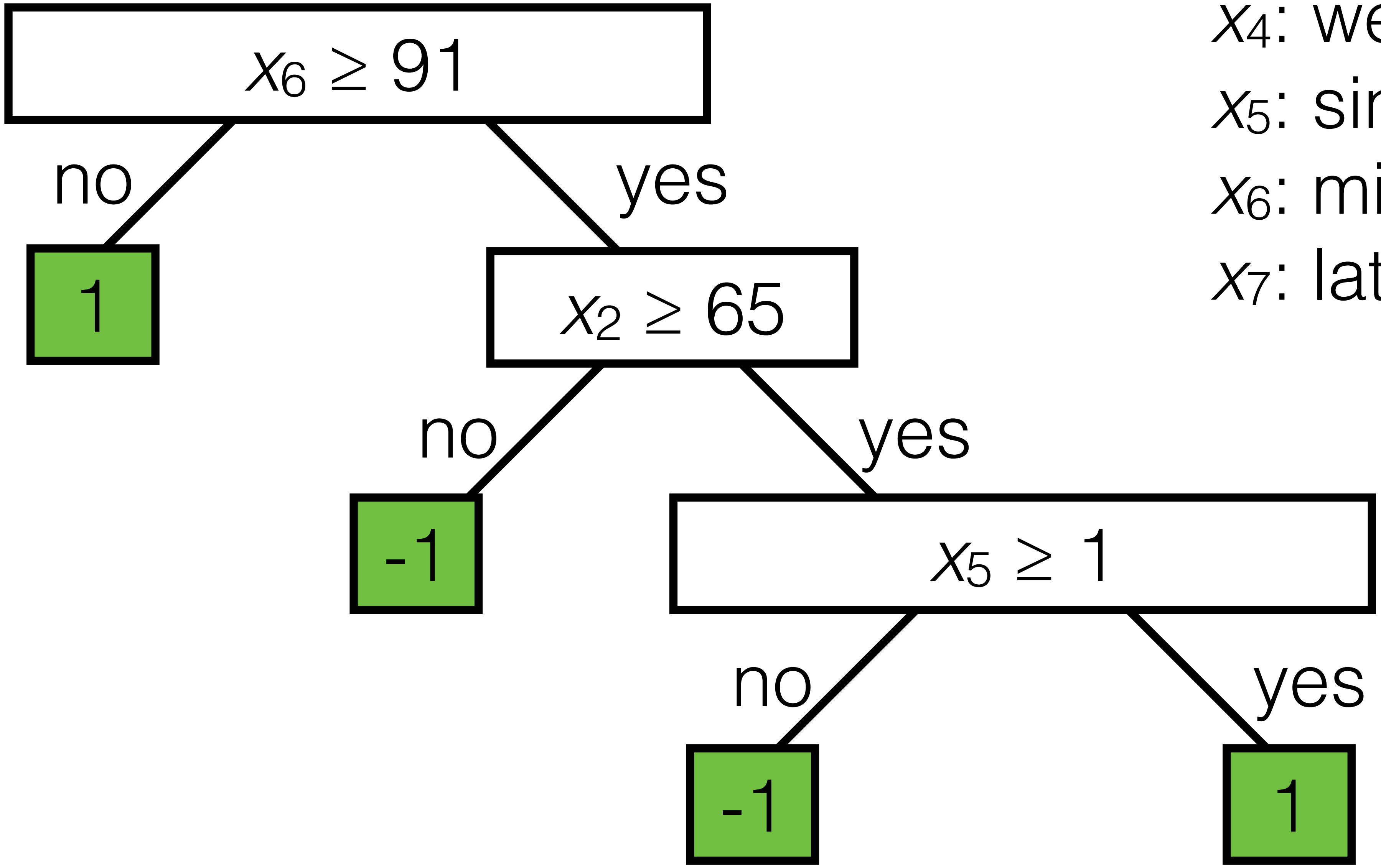
features:
x_1: date
x_2: age
x_3: height
x_4: weight
x_5: sinus tachycardia?
x_6: min systolic bp, 24h
x_7: latest diastolic bp

labels y:
1: high risk
-1: low risk

Decision tree

internal node:

- dimension index *j*; split value *s*



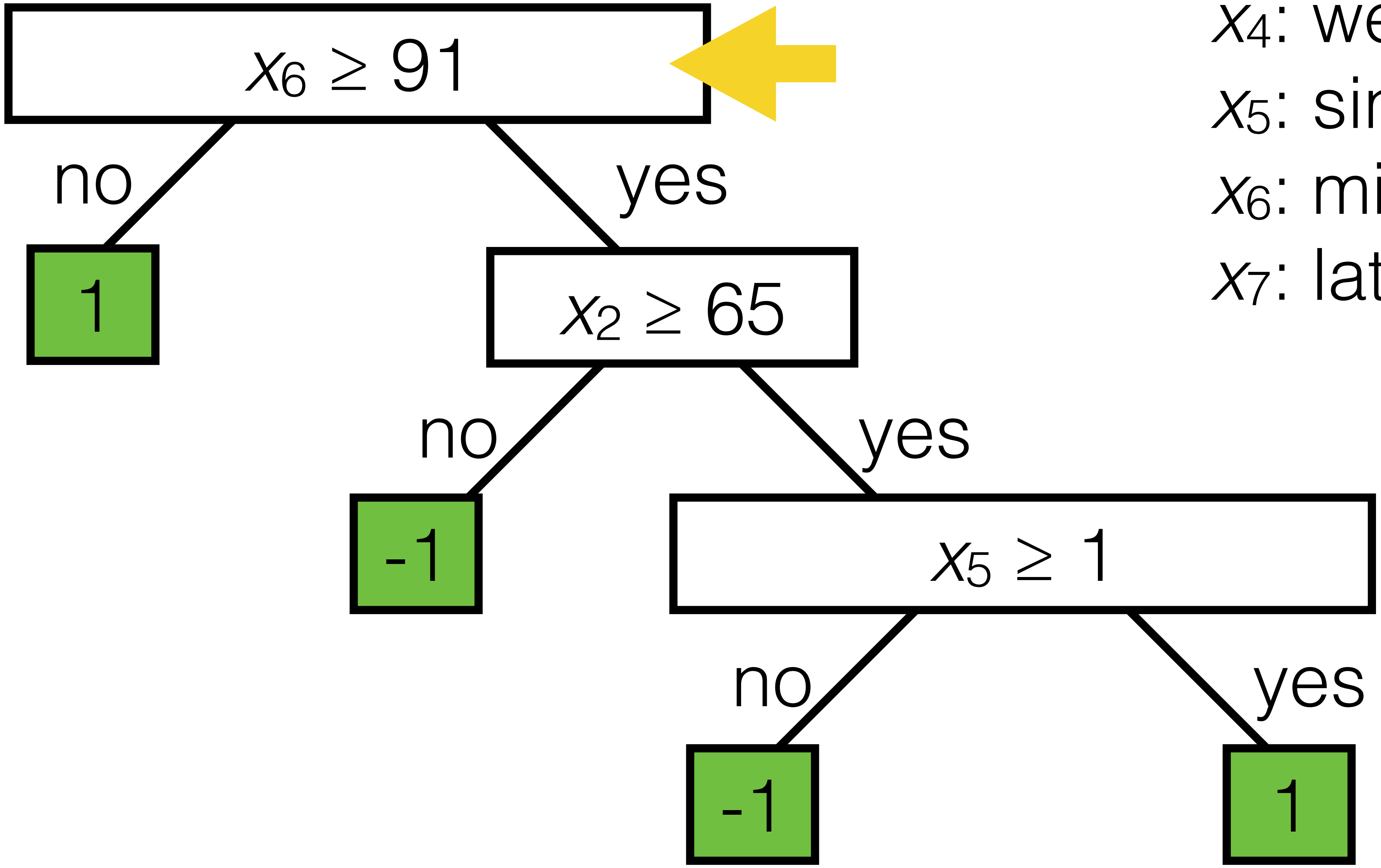
features:
x_1: date
x_2: age
x_3: height
x_4: weight
x_5: sinus tachycardia?
x_6: min systolic bp, 24h
x_7: latest diastolic bp

labels y:
1: high risk
-1: low risk

Decision tree

internal node:

- dimension index *j*; split value *s*



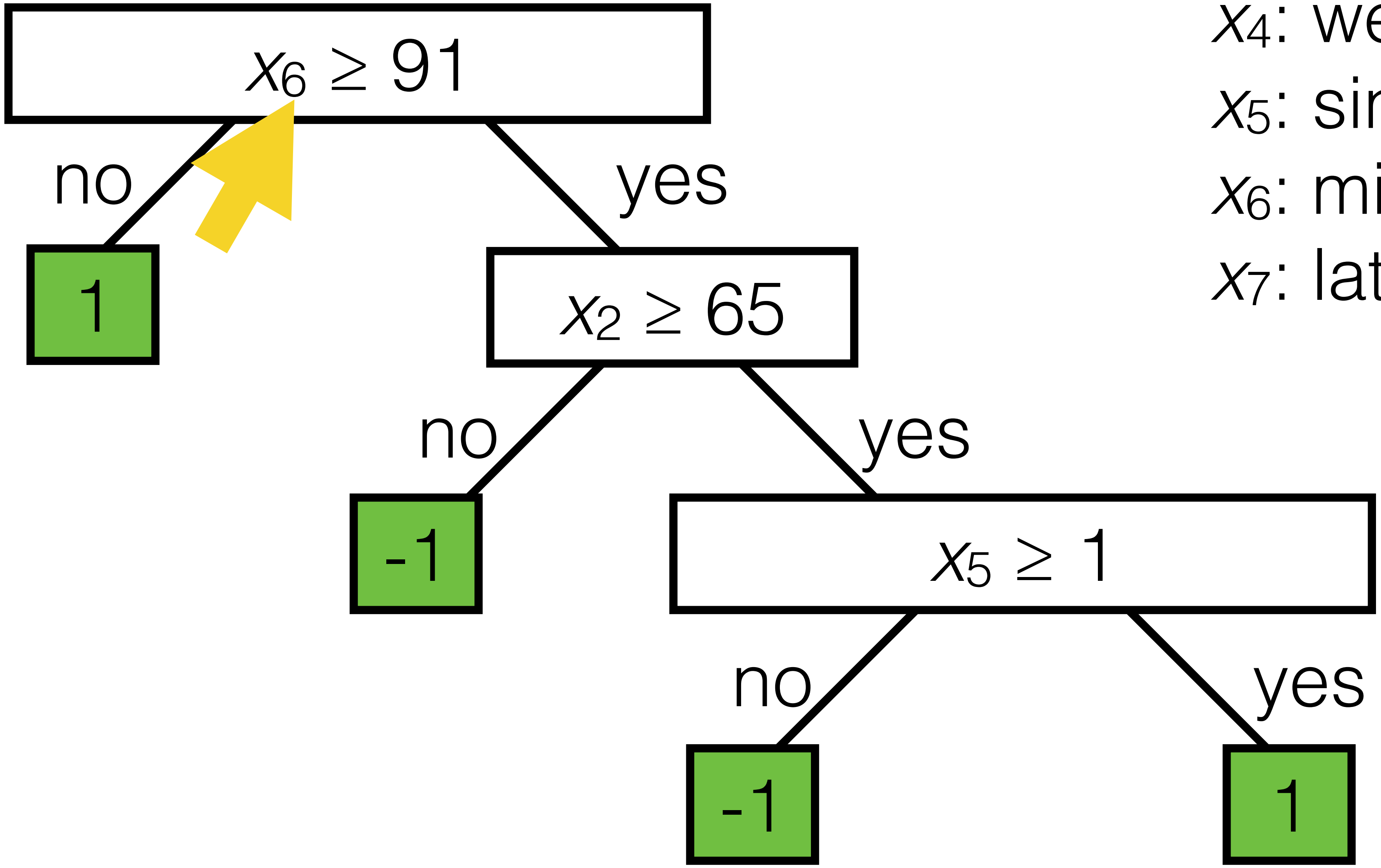
features:
x_1: date
x_2: age
x_3: height
x_4: weight
x_5: sinus tachycardia?
x_6: min systolic bp, 24h
x_7: latest diastolic bp

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*:

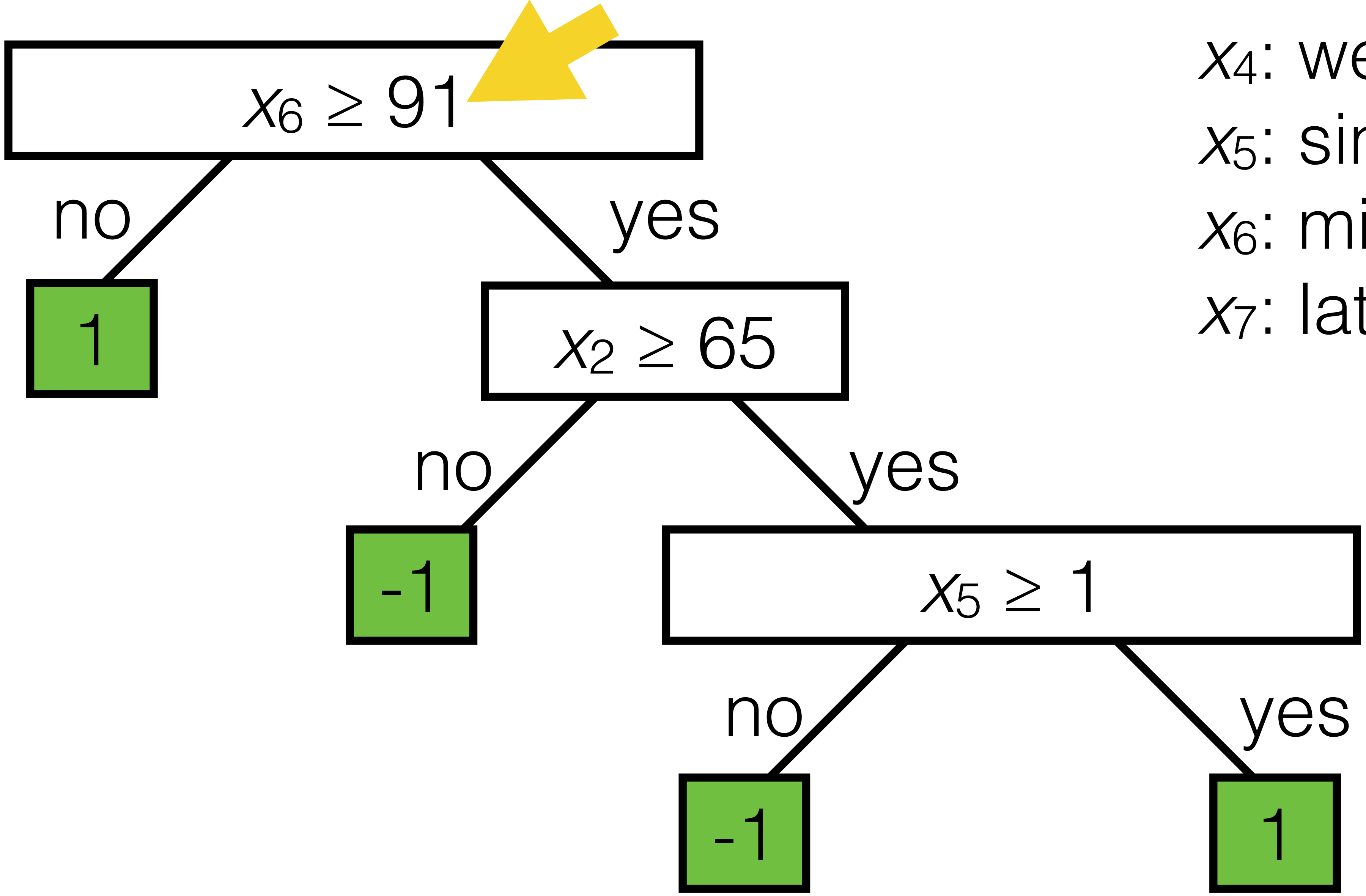
1: high risk

-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:

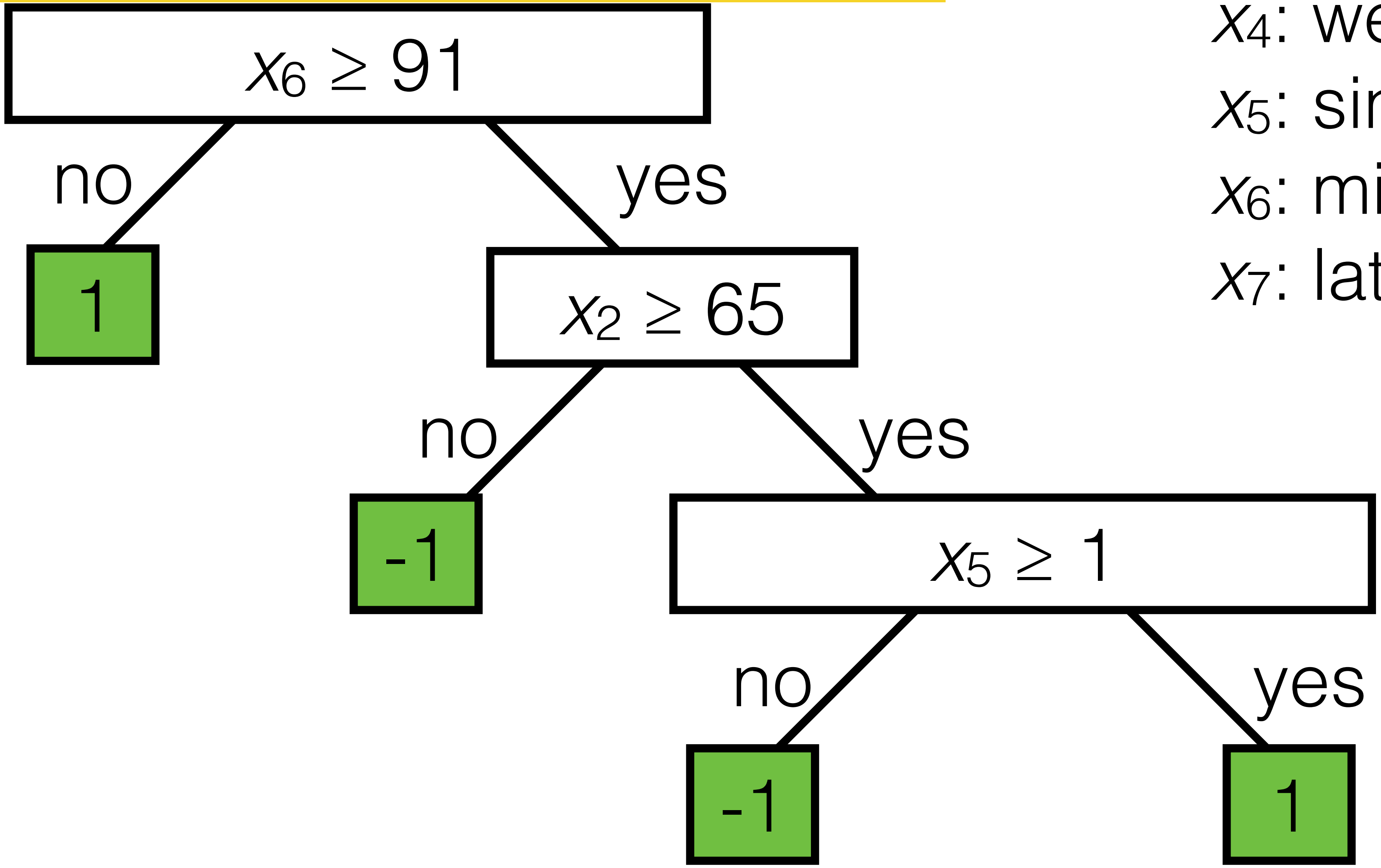
1: high risk

-1: low risk

Decision tree

internal node:

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



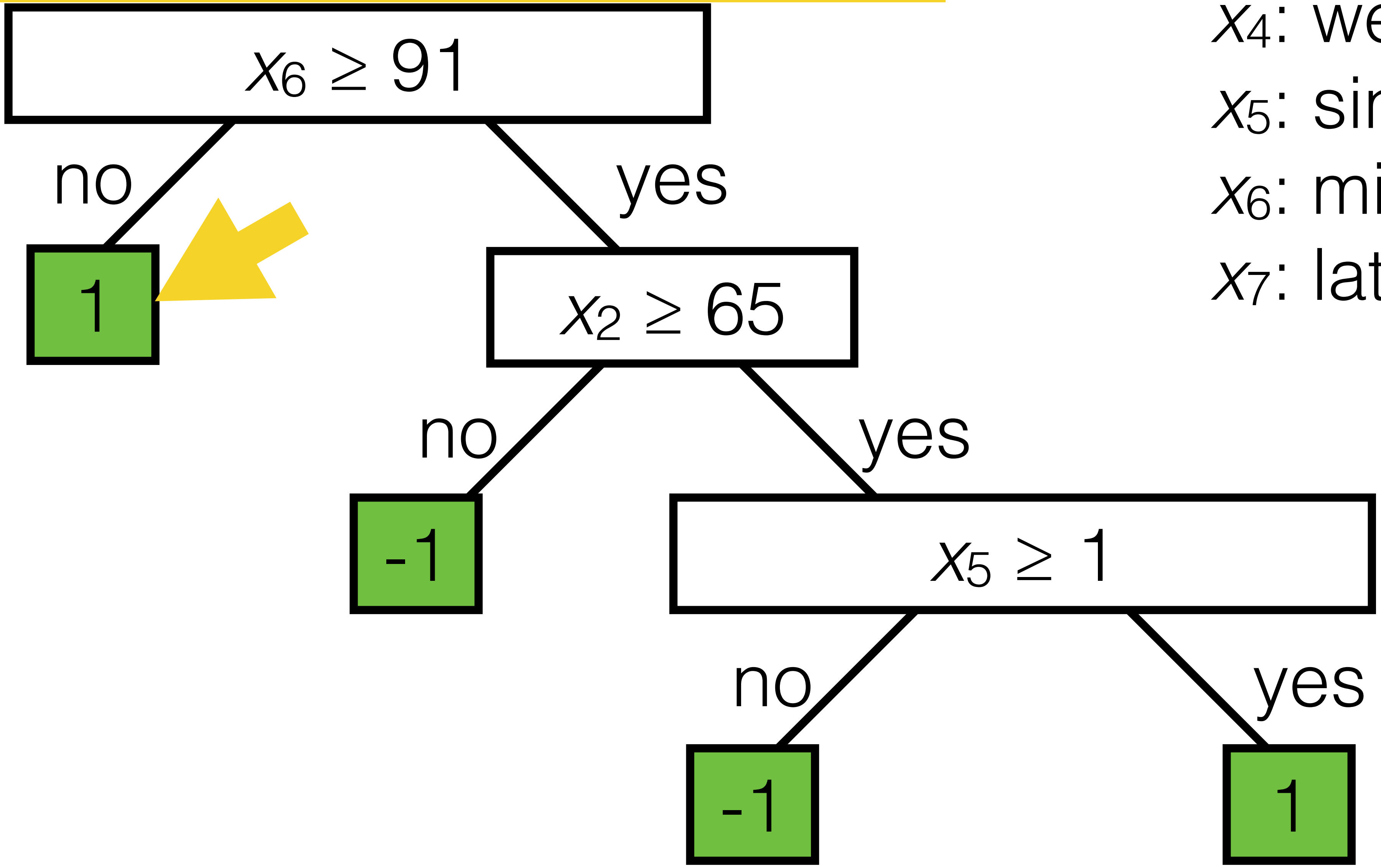
features:
x_1: date
x_2: age
x_3: height
x_4: weight
x_5: sinus tachycardia?
x_6: min systolic bp, 24h
x_7: latest diastolic bp

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*:

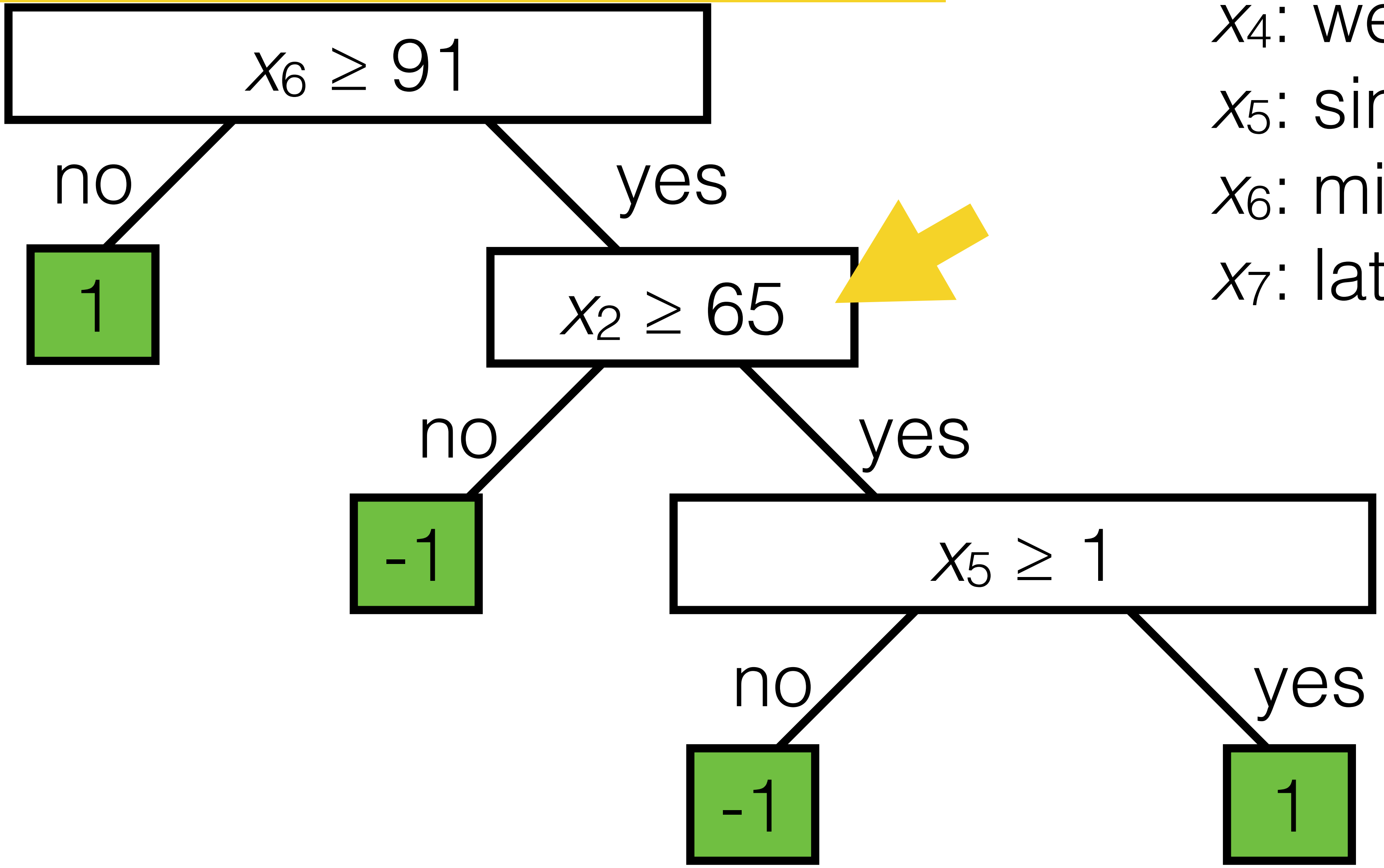
1: high risk

-1: low risk

Decision tree

internal node:

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

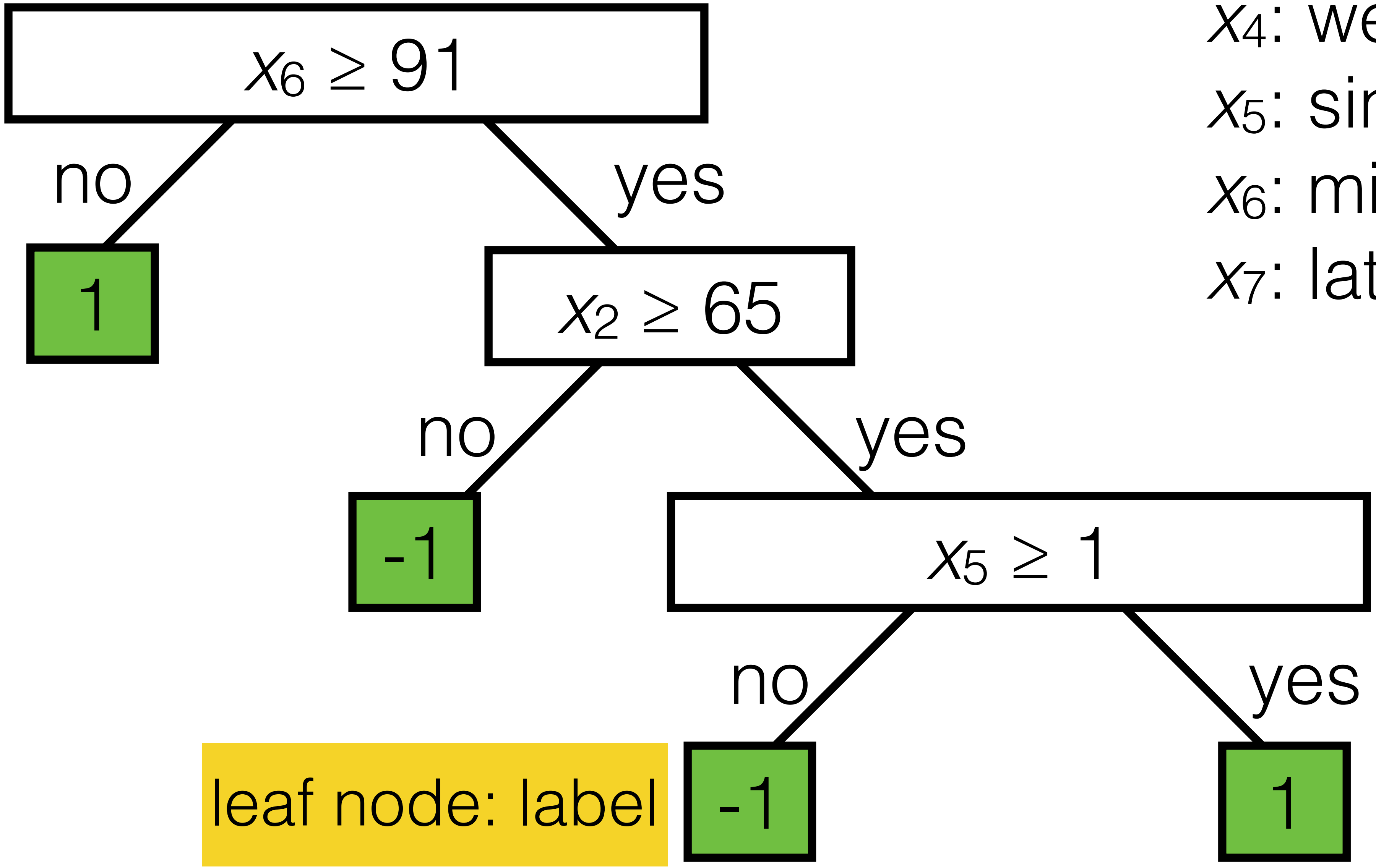


features:
x_1: date
x_2: age
x_3: height
x_4: weight
x_5: sinus tachycardia?
x_6: min systolic bp, 24h
x_7: latest diastolic bp

labels y:
1: high risk
-1: low risk

Decision tree

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



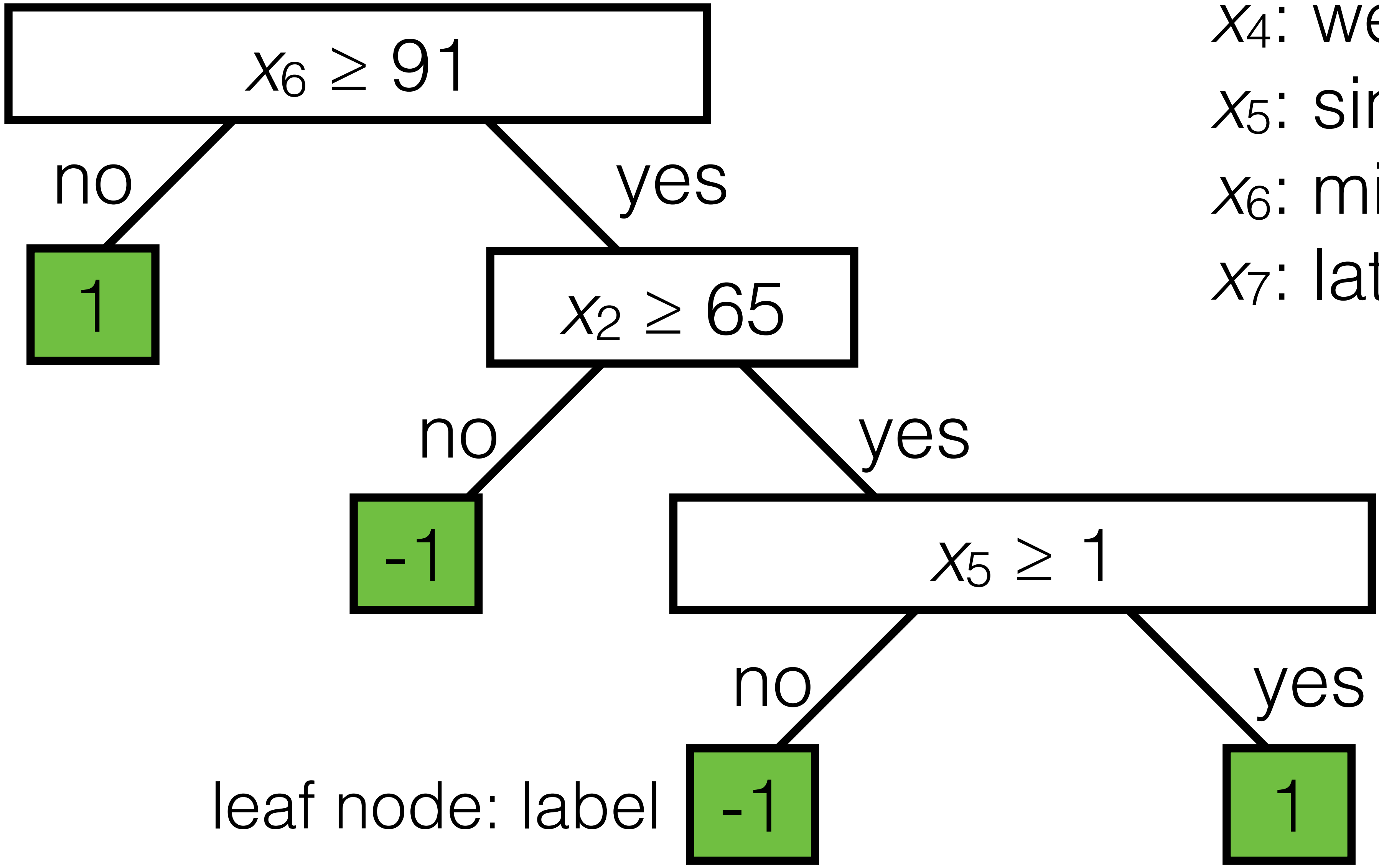
features:
x_1: date
x_2: age
x_3: height
x_4: weight
x_5: sinus tachycardia?
x_6: min systolic bp, 24h
x_7: latest diastolic bp

labels *y*:
1: high risk
-1: low risk

Classification tree

internal node:

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



$x_6 \geq 91$

no

yes

1

$x_2 \geq 65$

no

yes

-1

$x_5 \geq 1$

no

yes

leaf node: label

-1

1

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*:

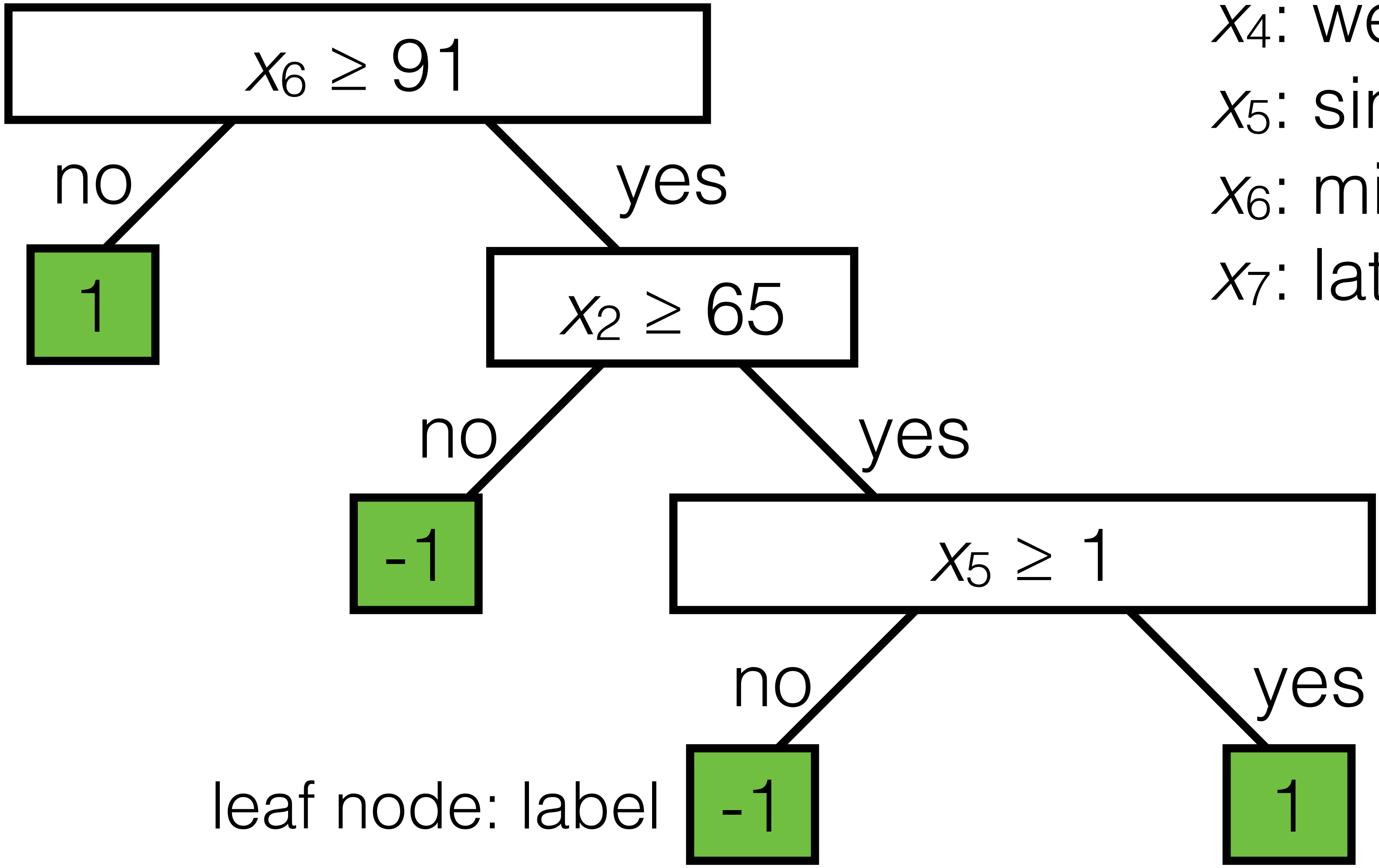
1: high risk

-1: low risk

Classification tree

internal node:

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



$x_6 \geq 91$

no

1

yes

$x_2 \geq 65$

no

-1

yes

$x_5 \geq 1$

no

leaf node: label

-1

yes

1

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

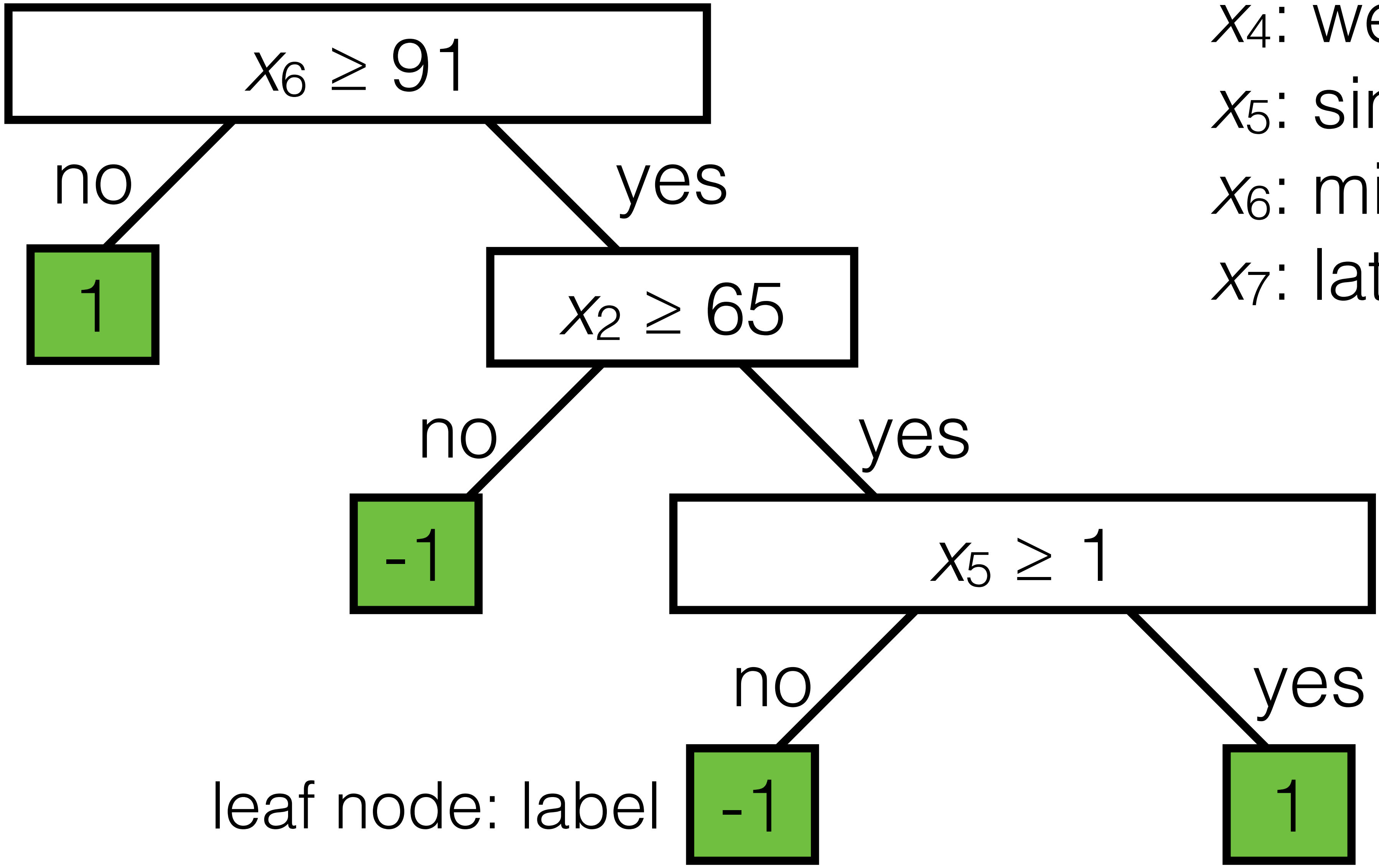
-1: low risk

$x^{(1)}$

Classification tree

internal node:

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



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

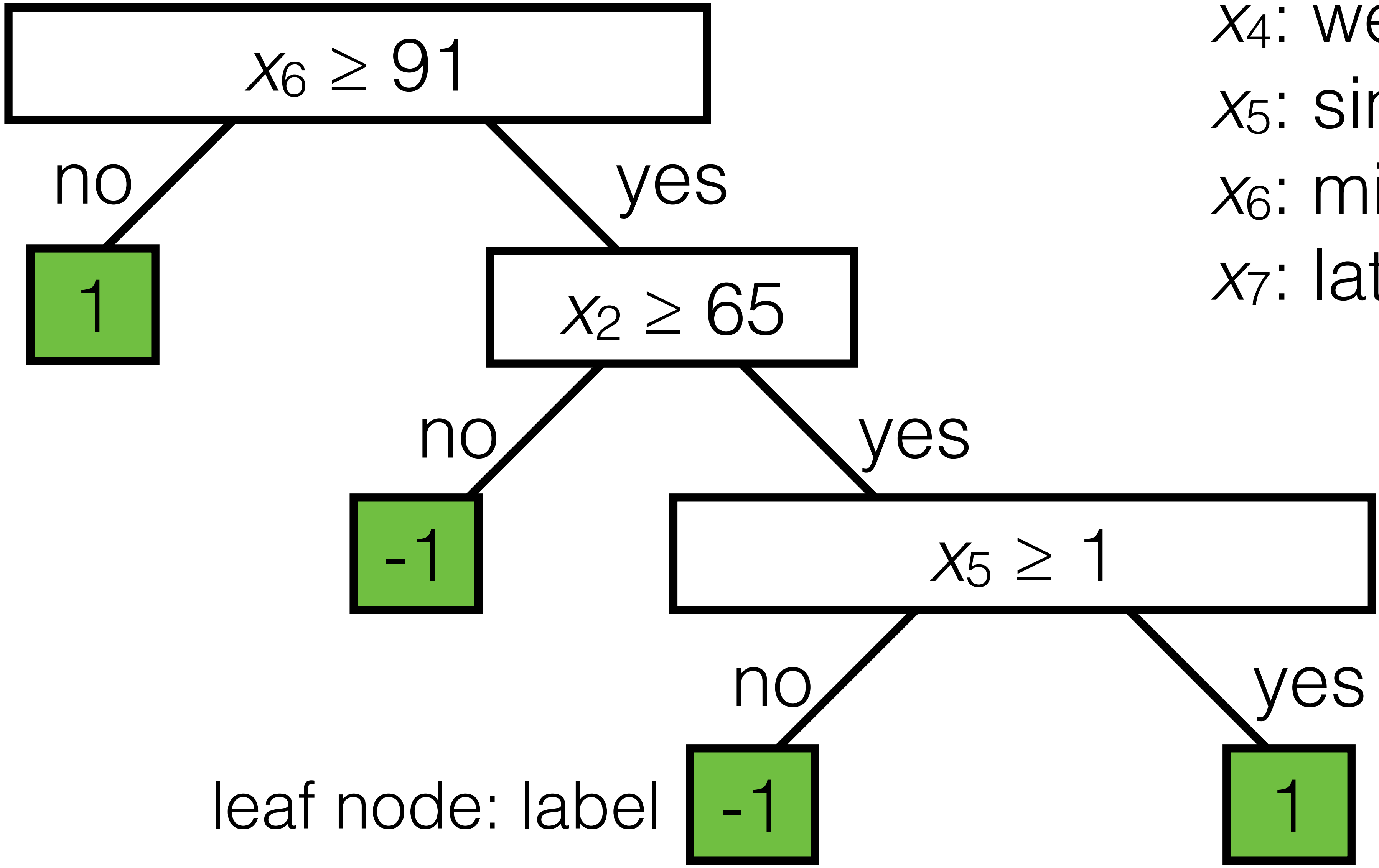
-1: low risk

$x^{(1)}$: 2020/11/17, 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

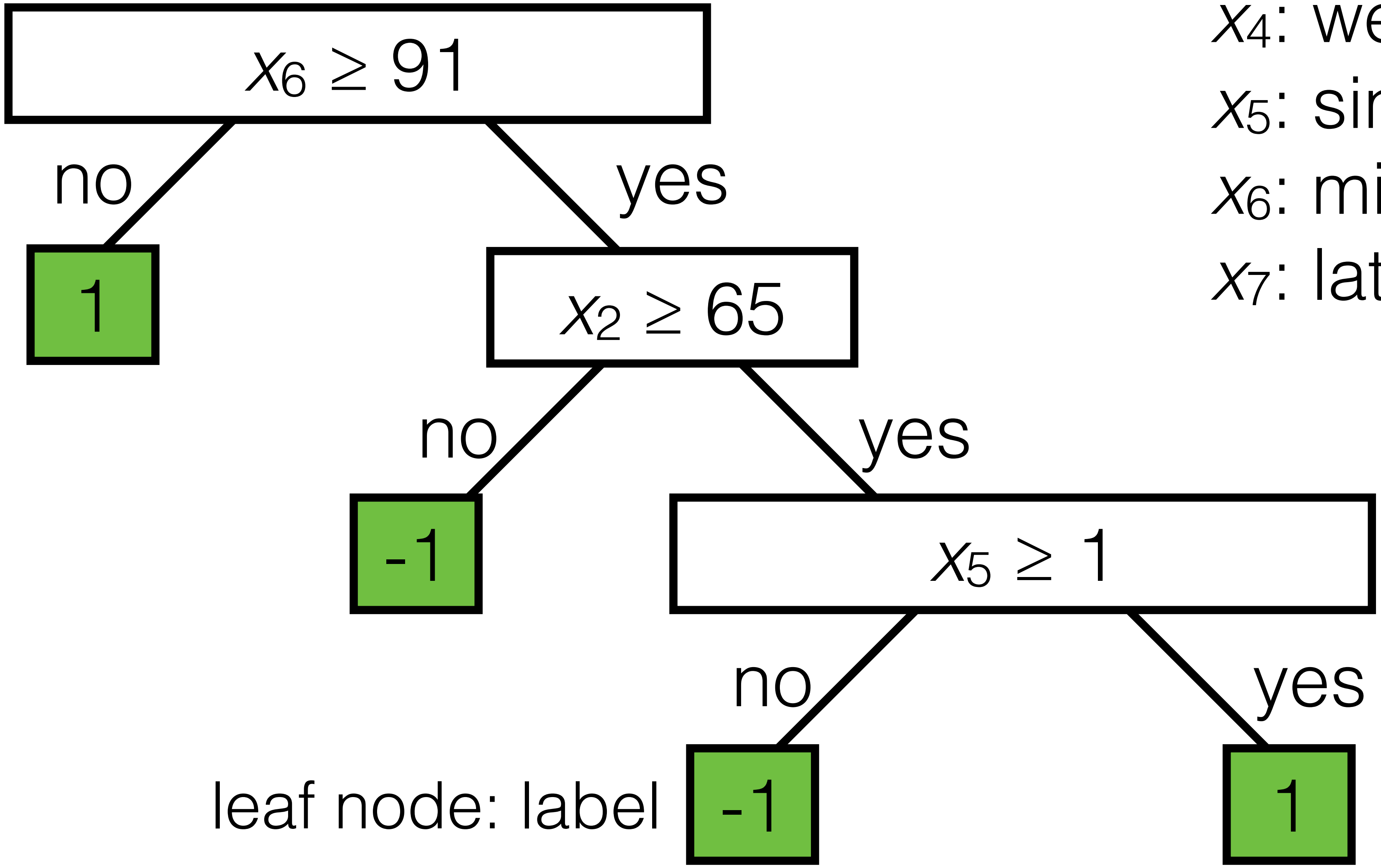
$x^{(1)}$: 2020/11/17, 49, 172 cm, 70.5 kg, 0, 115, 79

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

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

$x^{(1)}$: 2020/11/17, 49, 172 cm, 70.5 kg, 0, 115, 79

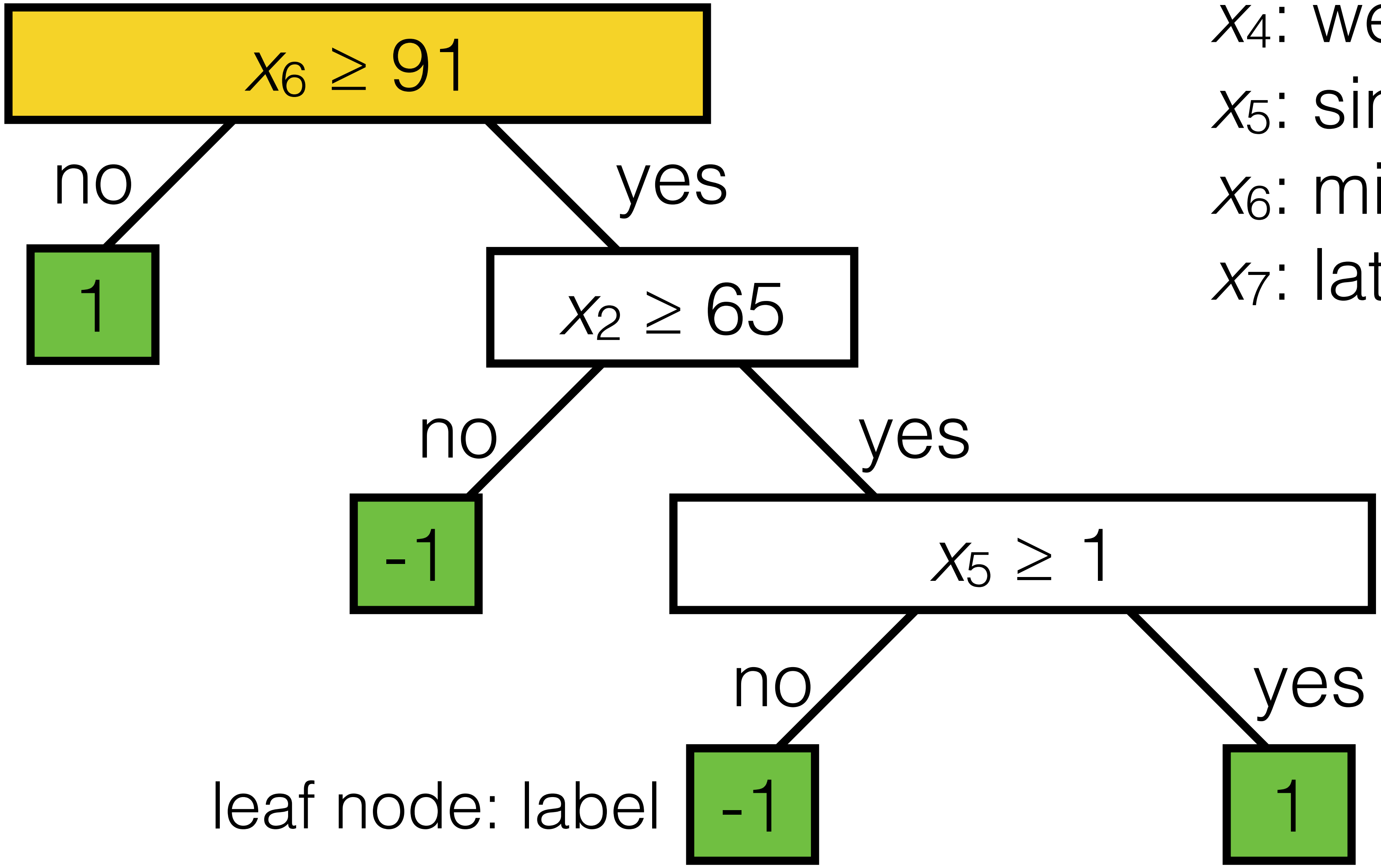
$T(x^{(1)}) =$

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



$x_6 \geq 91$
no
1
yes
$x_2 \geq 65$
no
-1
yes
$x_5 \geq 1$
no
-1
yes
1

leaf node: label

labels *y*:
1: high risk
-1: low risk

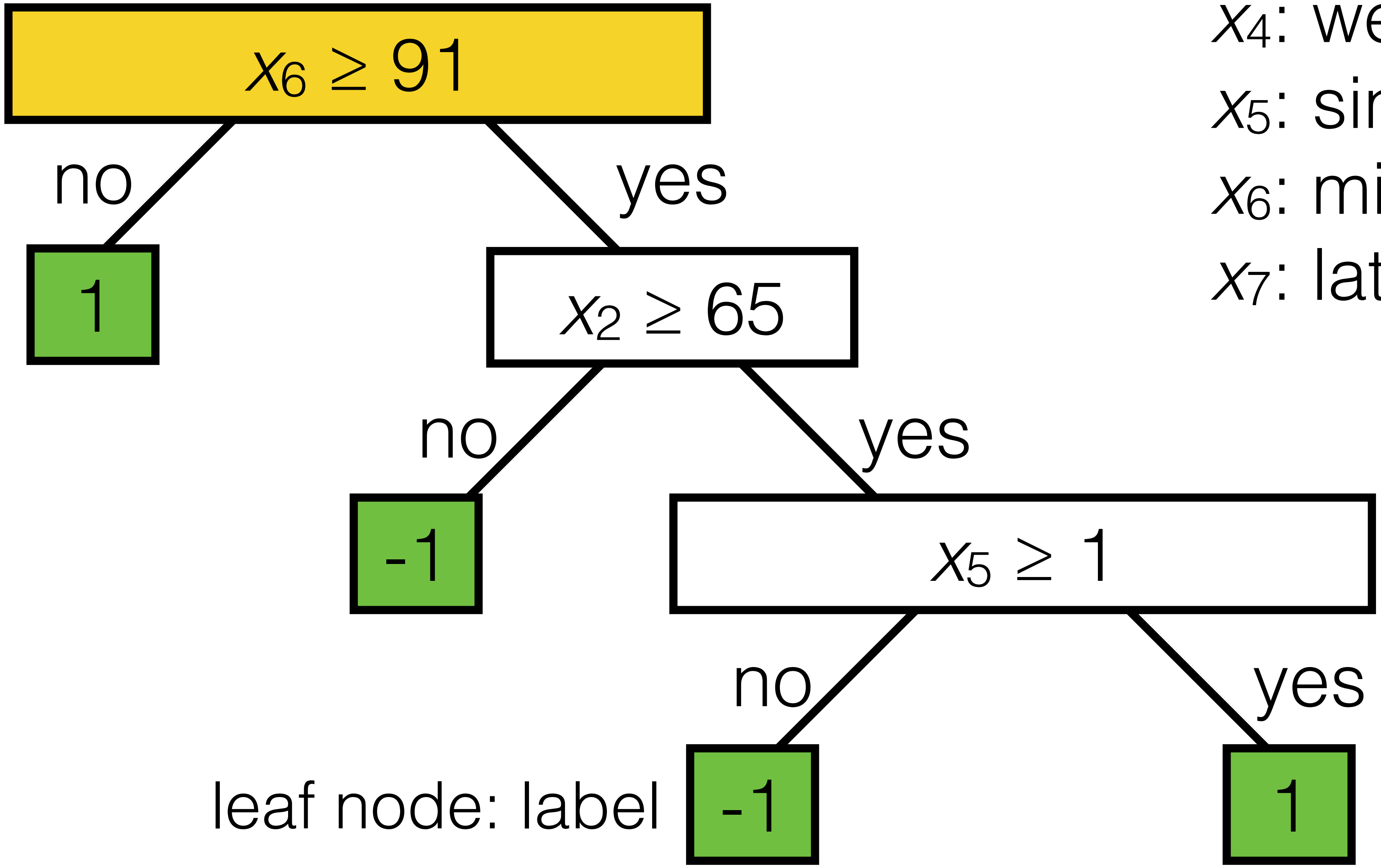
$x^{(1)}$: 2020/11/17, 49, 172 cm, 70.5 kg, 0, 115, 79

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

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

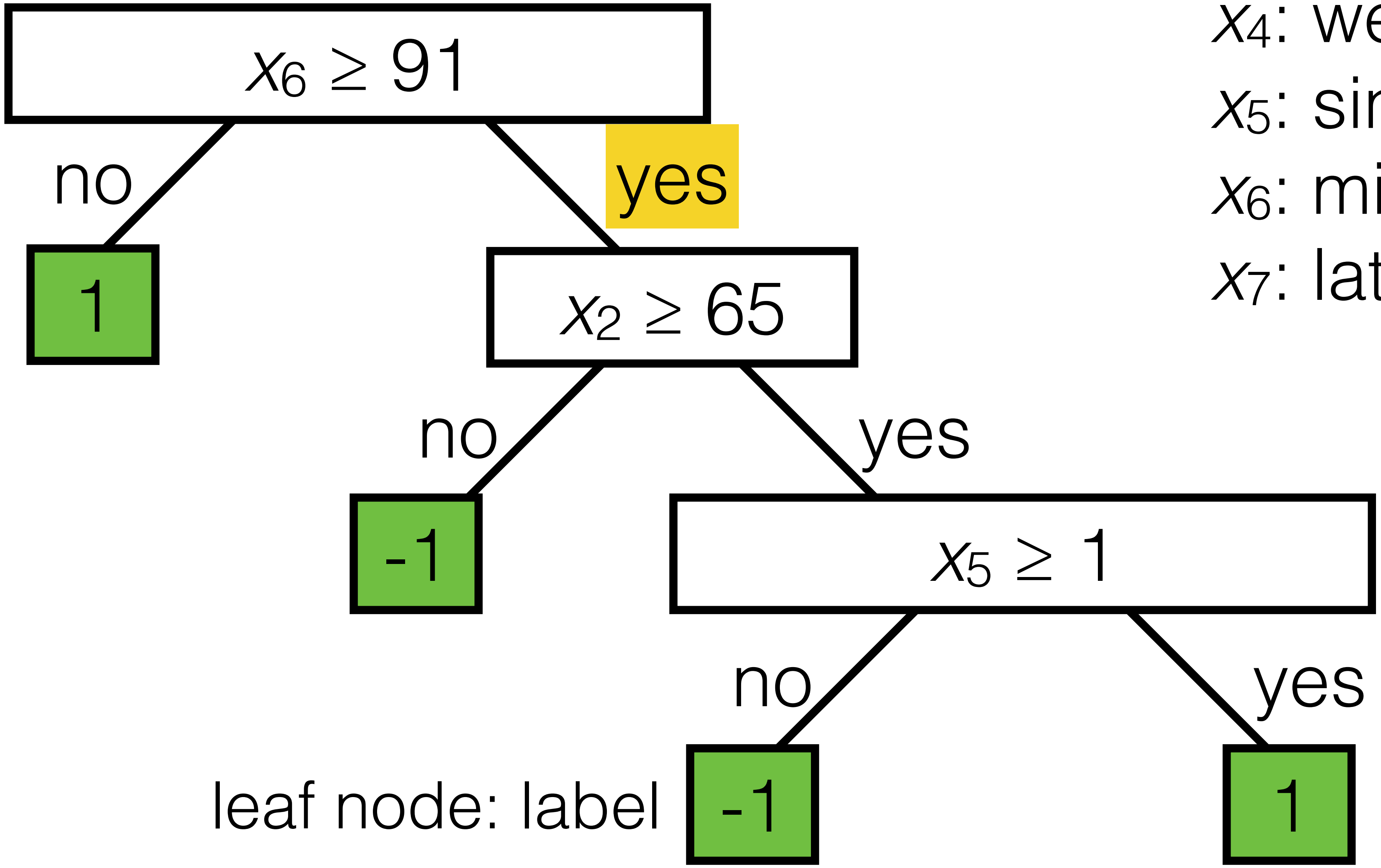
$x^{(1)}$: 2020/11/17, 49, 172 cm, 70.5 kg, 0, 115, 79

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

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

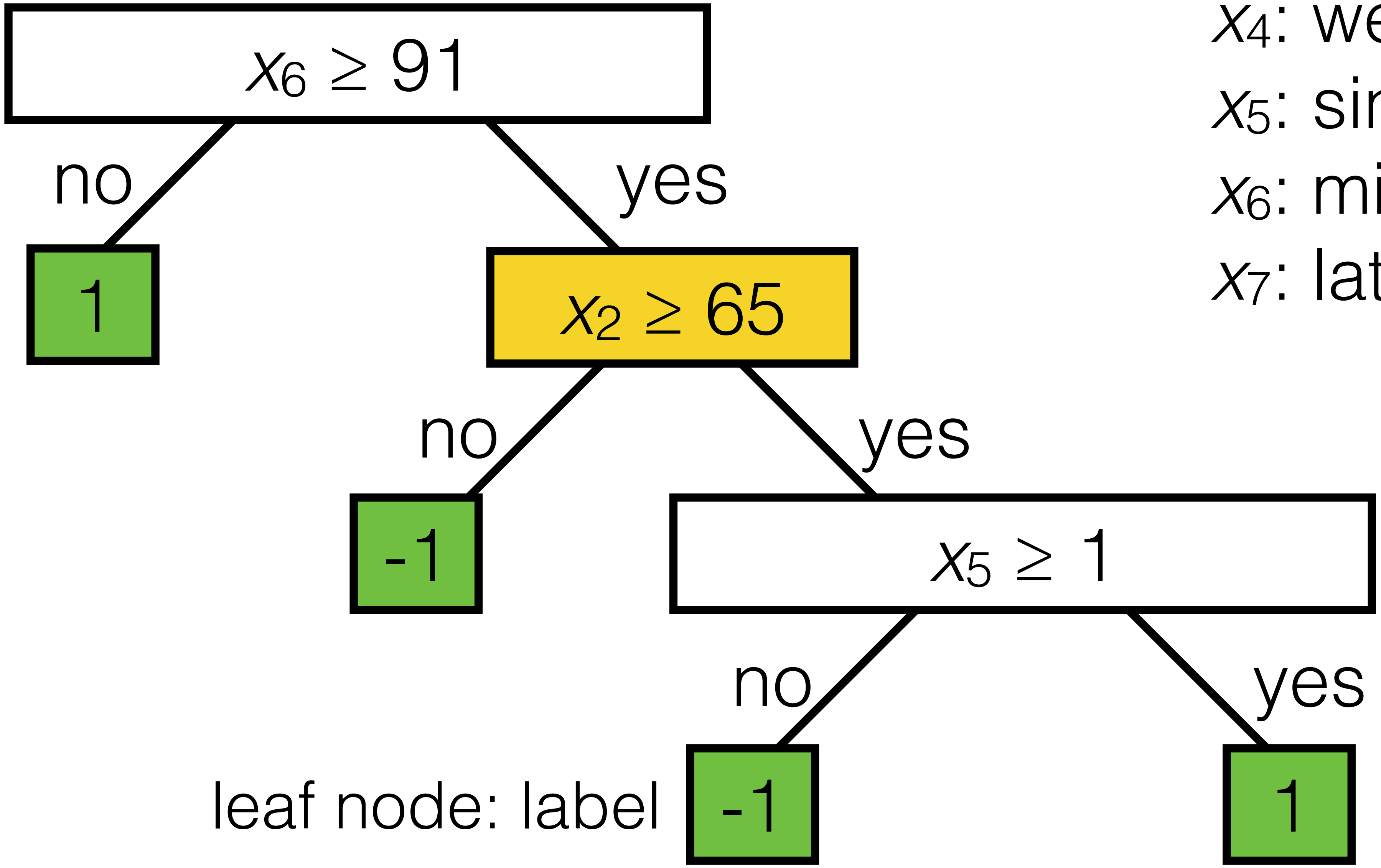
$x^{(1)}$: 2020/11/17, 49, 172 cm, 70.5 kg, 0, 115, 79

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

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

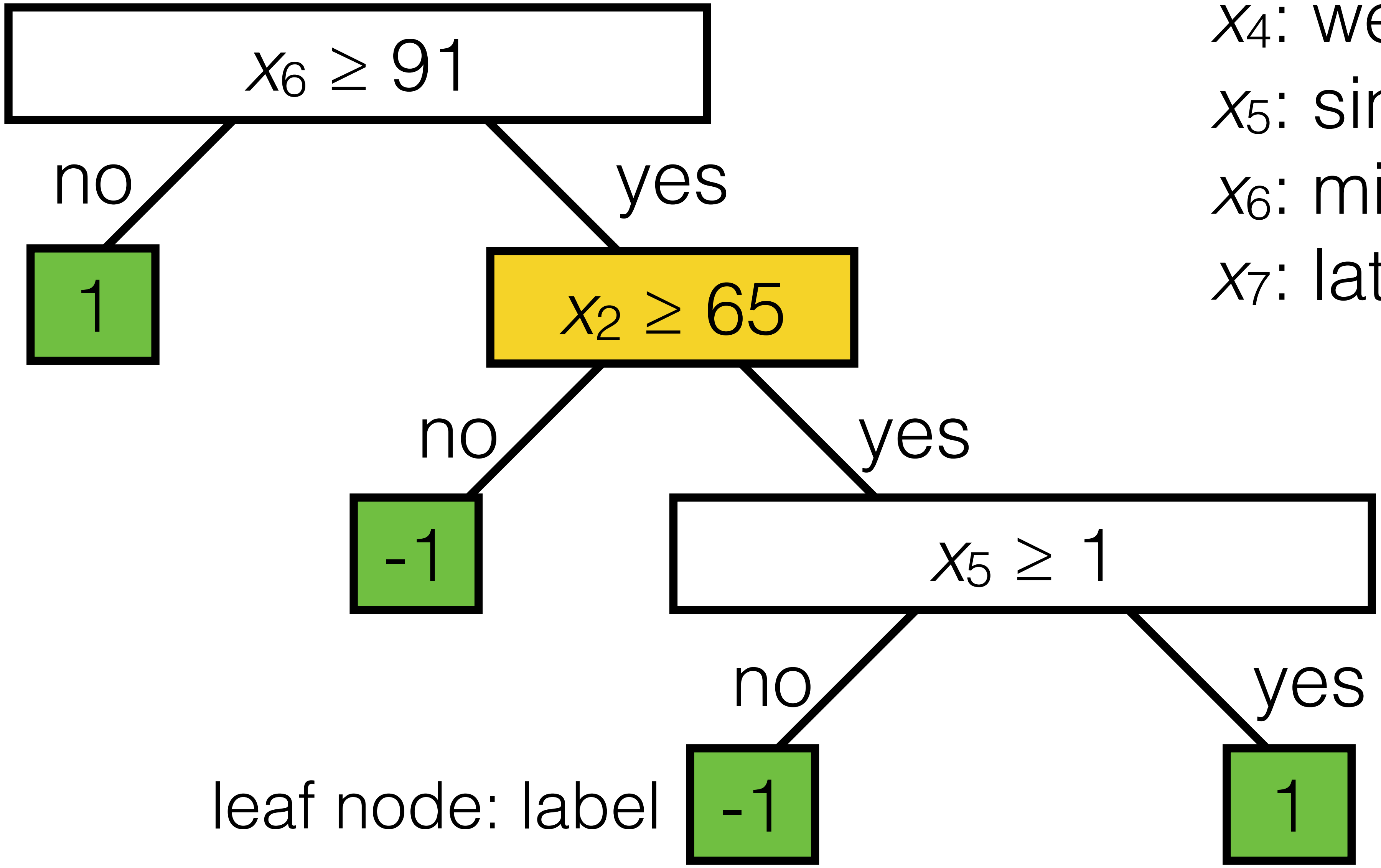
$x^{(1)}$: 2020/11/17, 49, 172 cm, 70.5 kg, 0, 115, 79

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

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

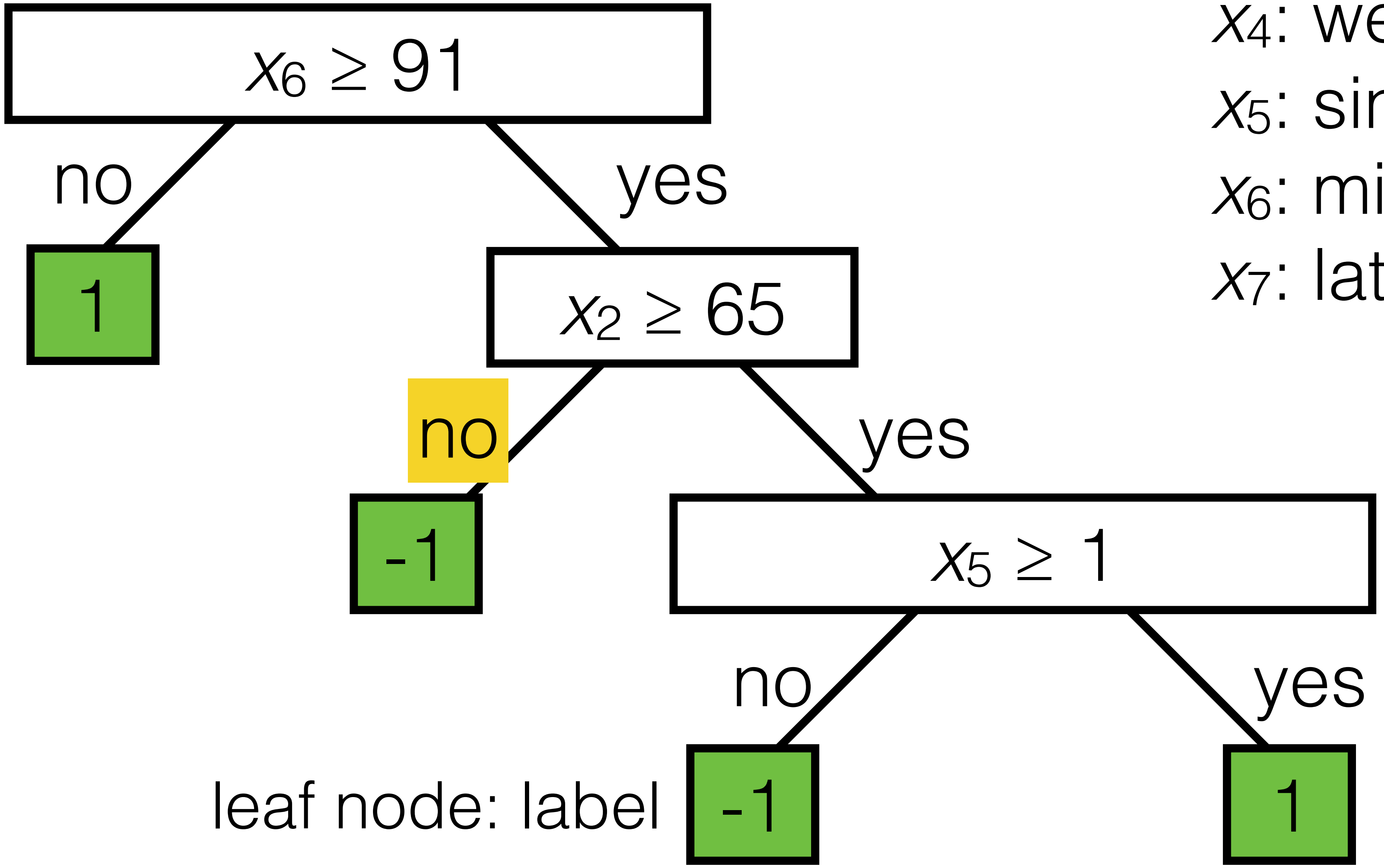
$x^{(1)}$: 2020/11/17, 49, 172 cm, 70.5 kg, 0, 115, 79

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

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

$x^{(1)}$: 2020/11/17, 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

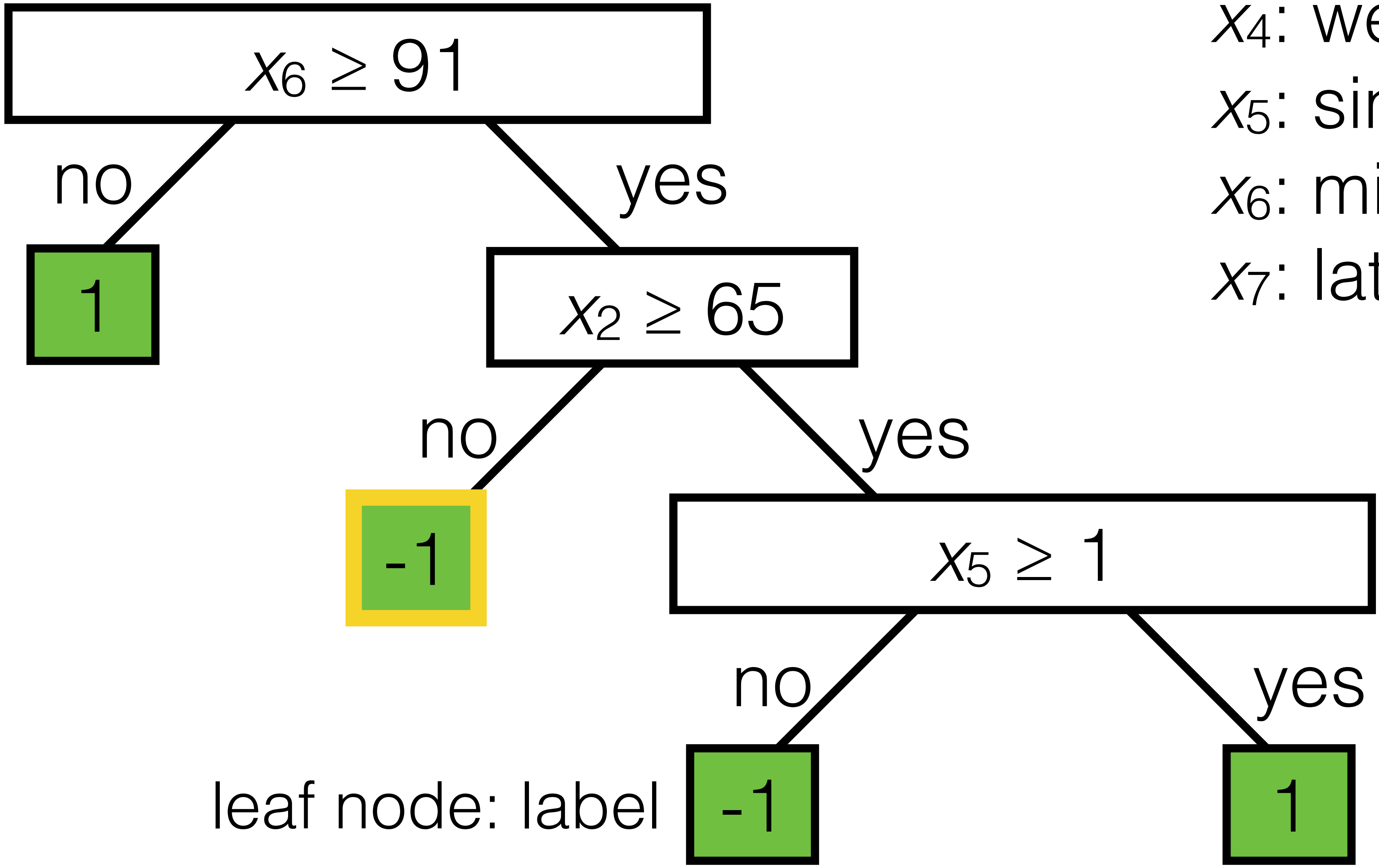
x_3: height

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

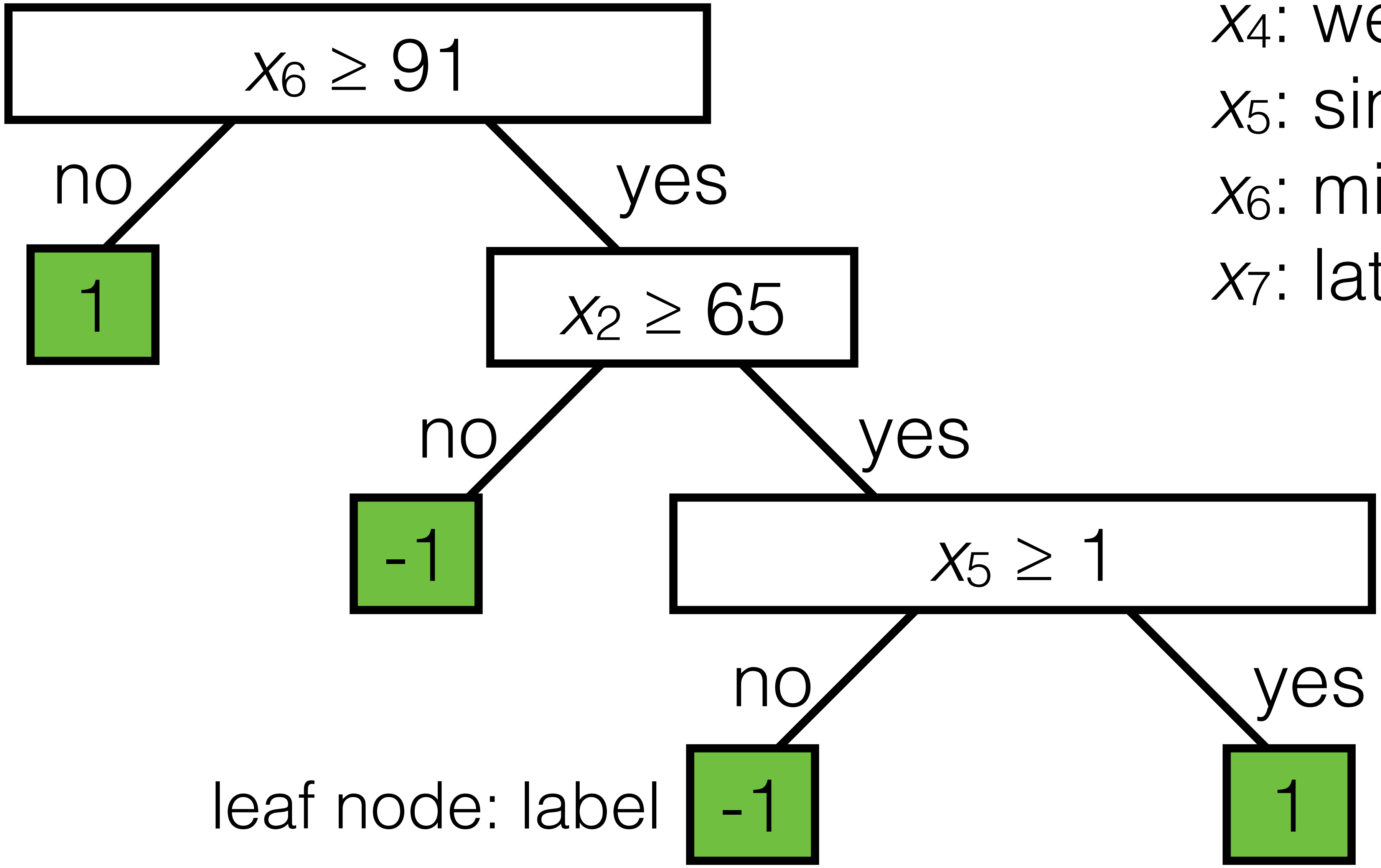
$x^{(1)}$: 2020/11/17, 49, 172 cm, 70.5 kg, 0, 115, 79

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

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

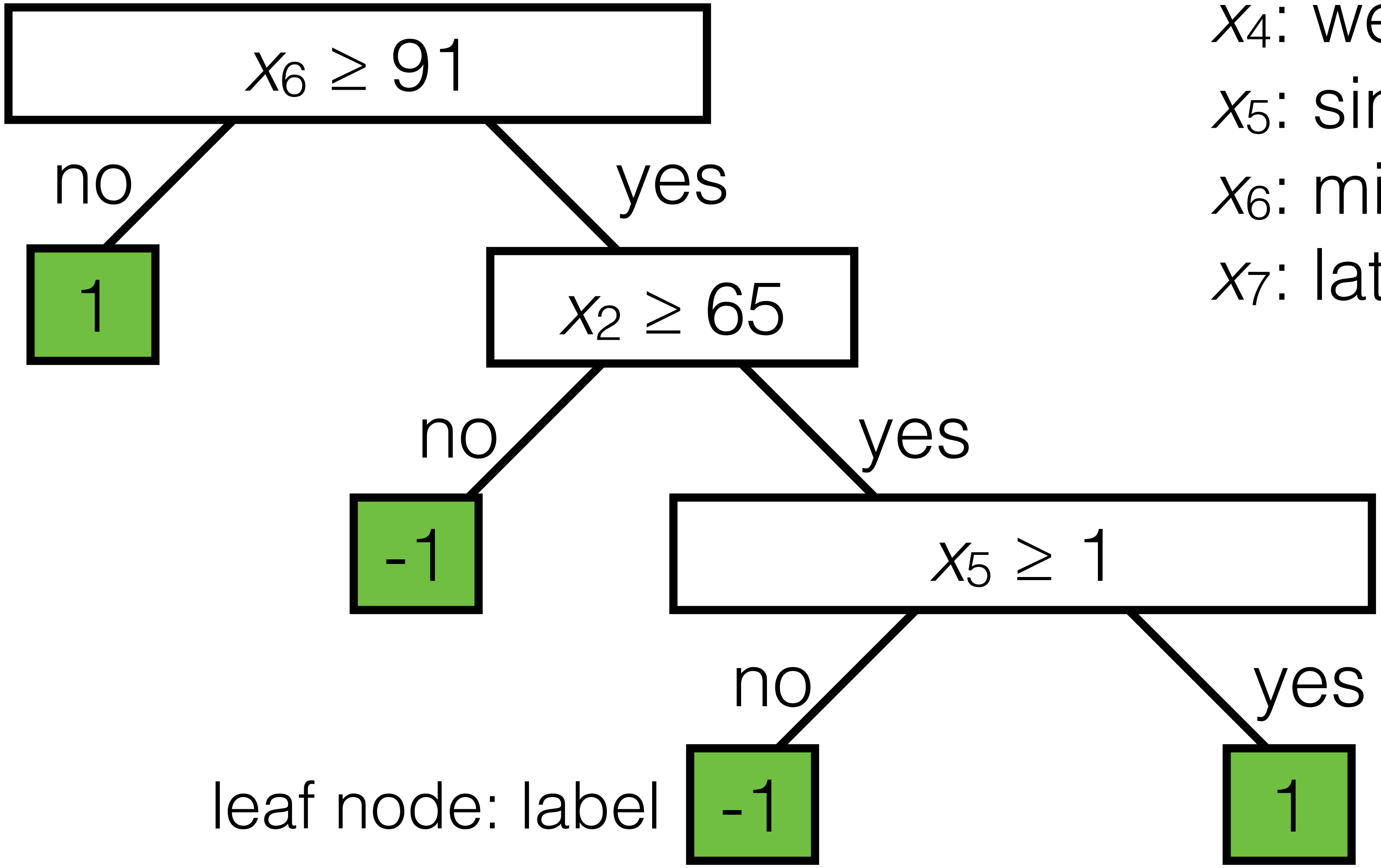
$x^{(1)}$: 2020/11/17, 49, 172 cm, 70.5 kg, 0, 115, 79

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

Classification tree

internal node:

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



$x_6 \geq 91$

no

yes

1

$x_2 \geq 65$

no

yes

-1

$x_5 \geq 1$

no

yes

leaf node: label

-1

1

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

$x^{(1)}$: 2020/11/17, 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:

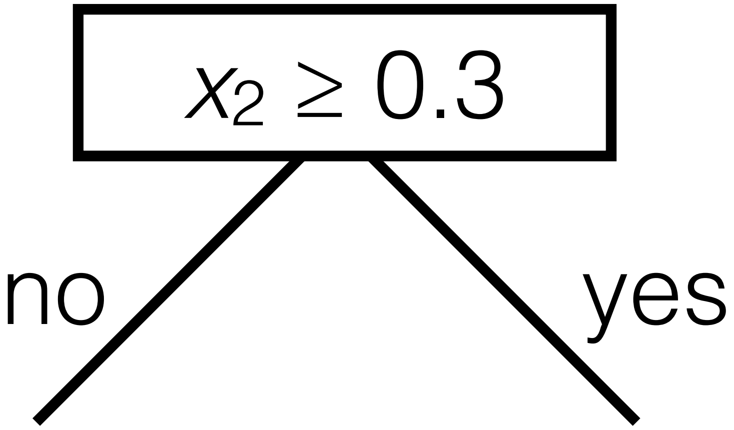
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

labels:

y: km run

Regression tree



$x_2 \geq 0.3$

no

yes

features:

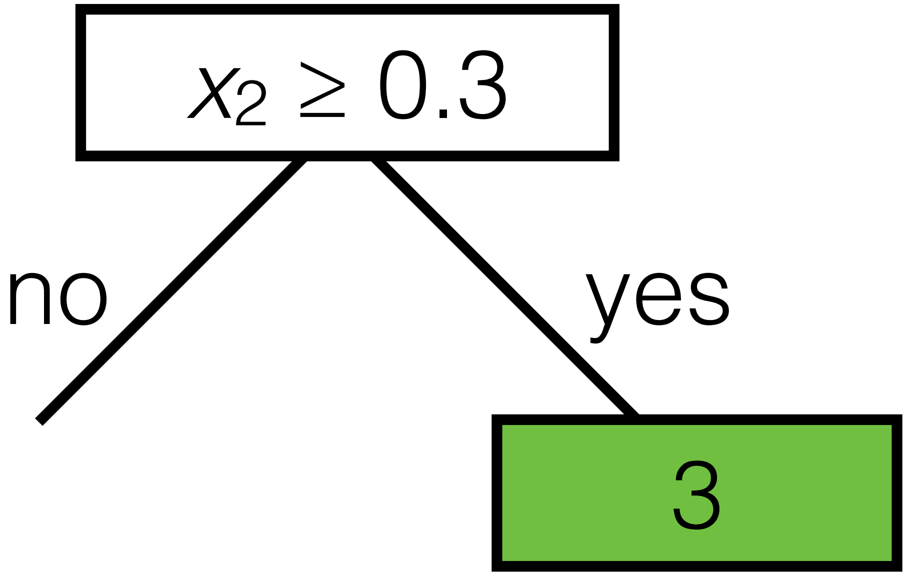
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

labels:

y: km run

Regression tree



features:

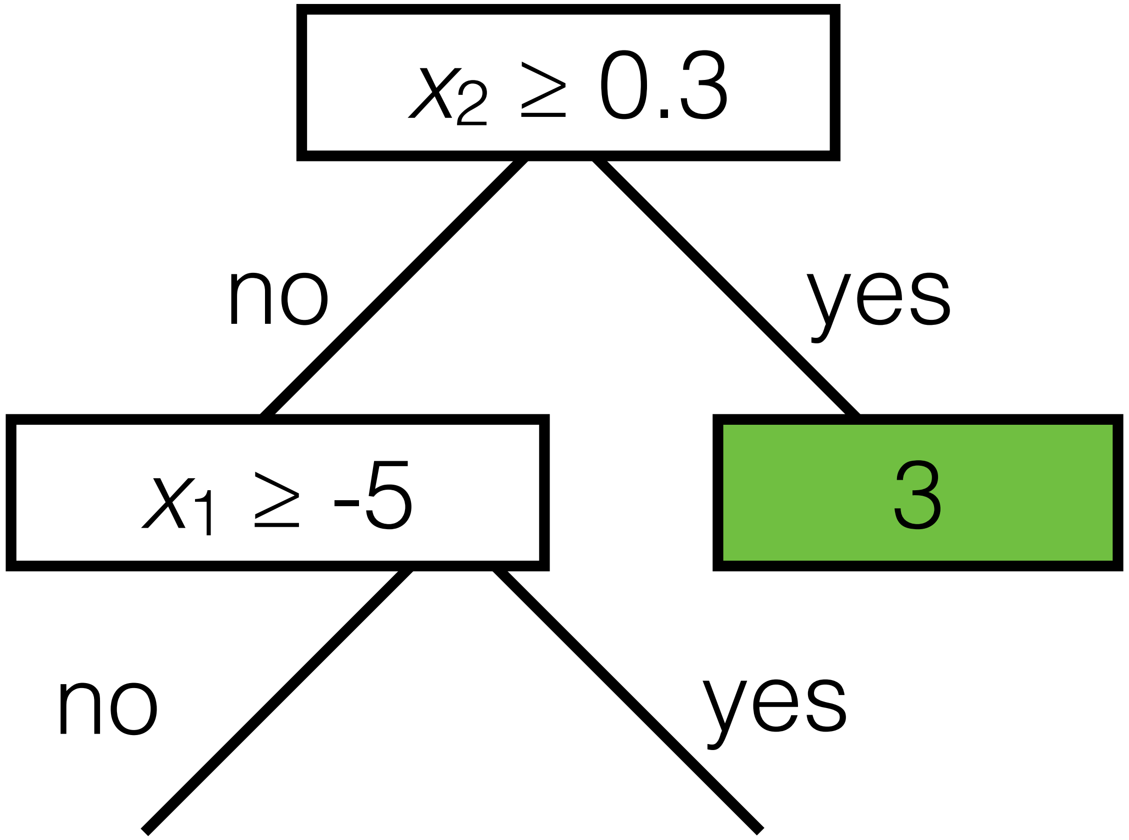
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

labels:

y: km run

Regression tree



features:

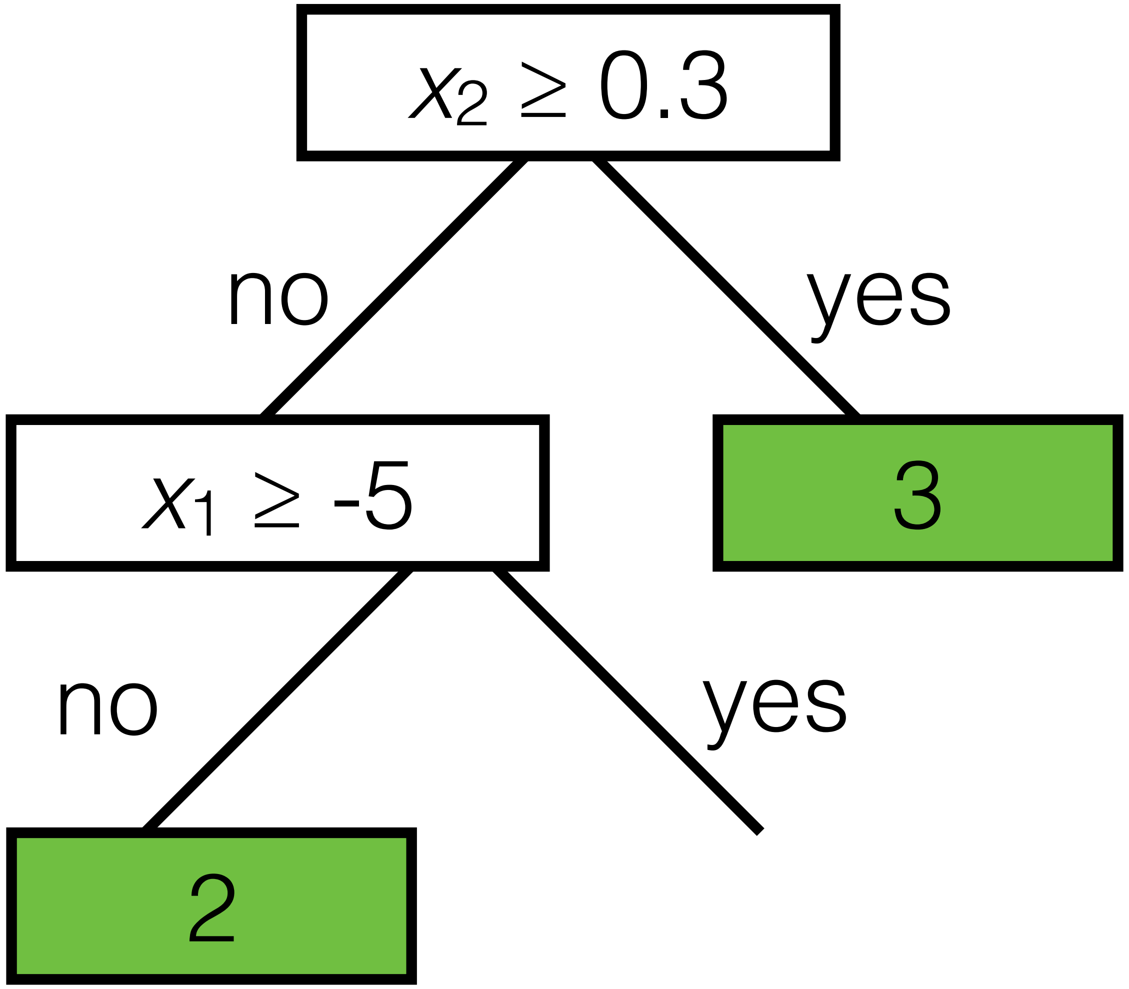
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

labels:

y: km run

Regression tree



features:

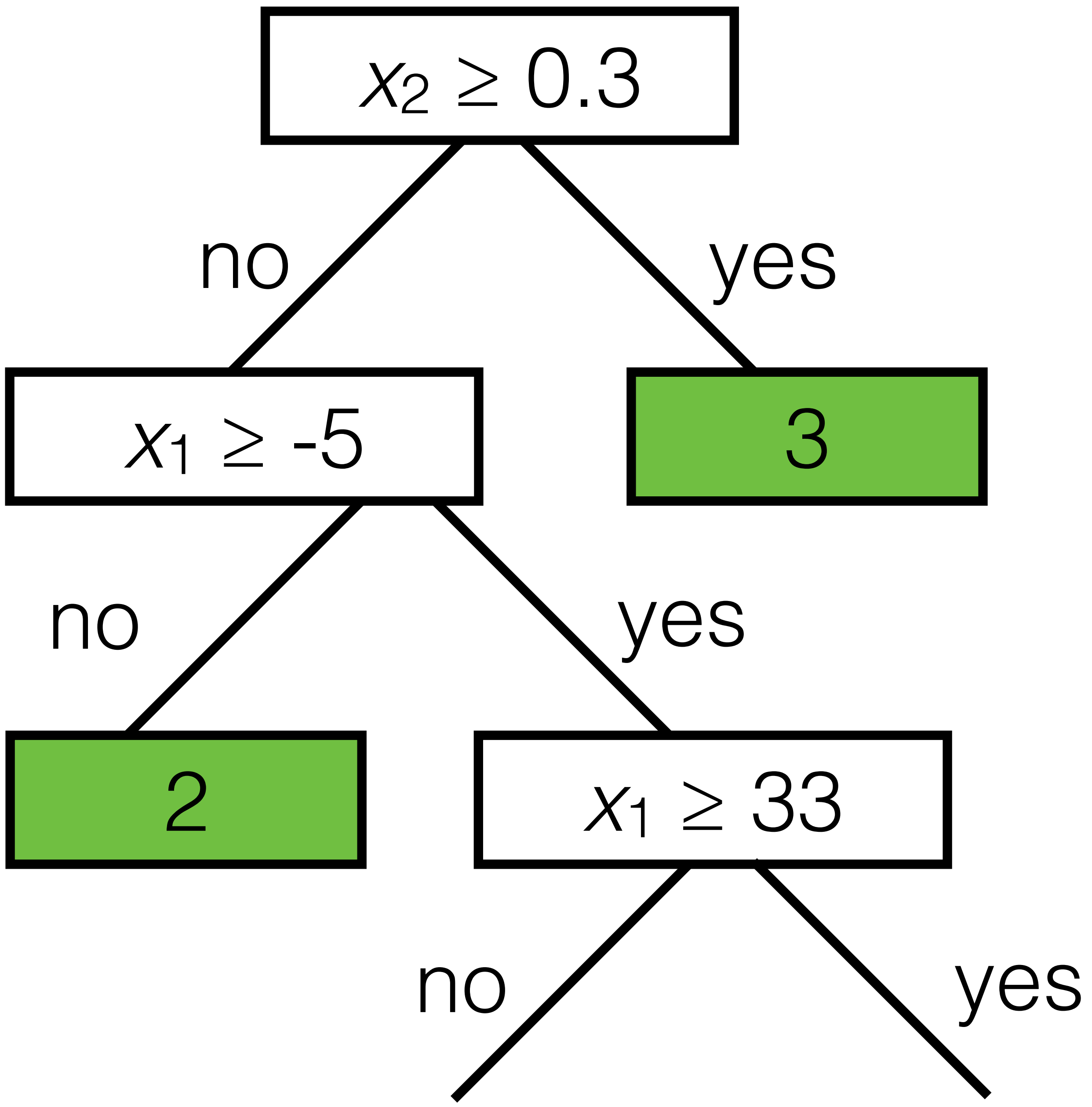
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

labels:

y: km run

Regression tree



features:

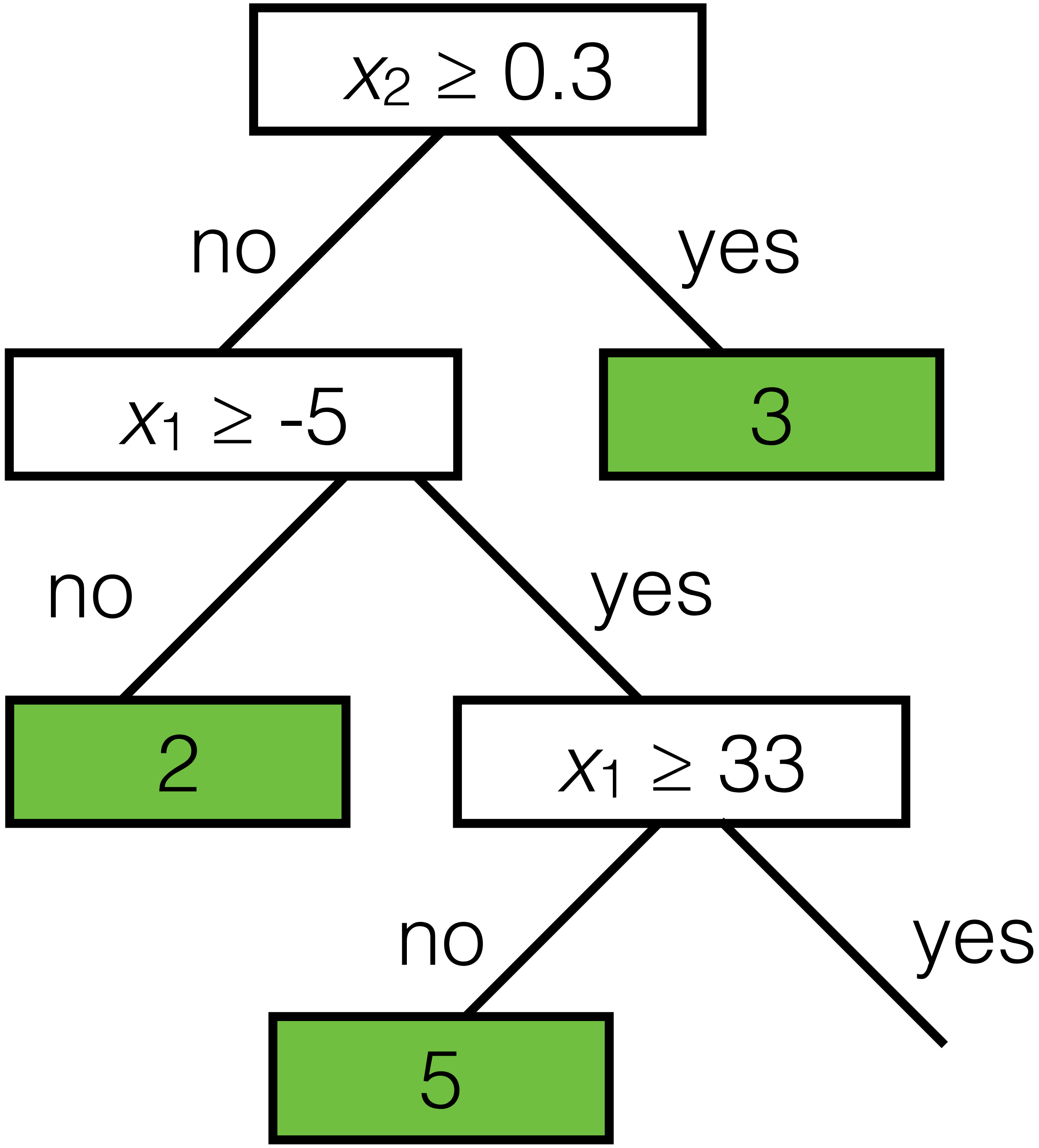
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

labels:

y: km run

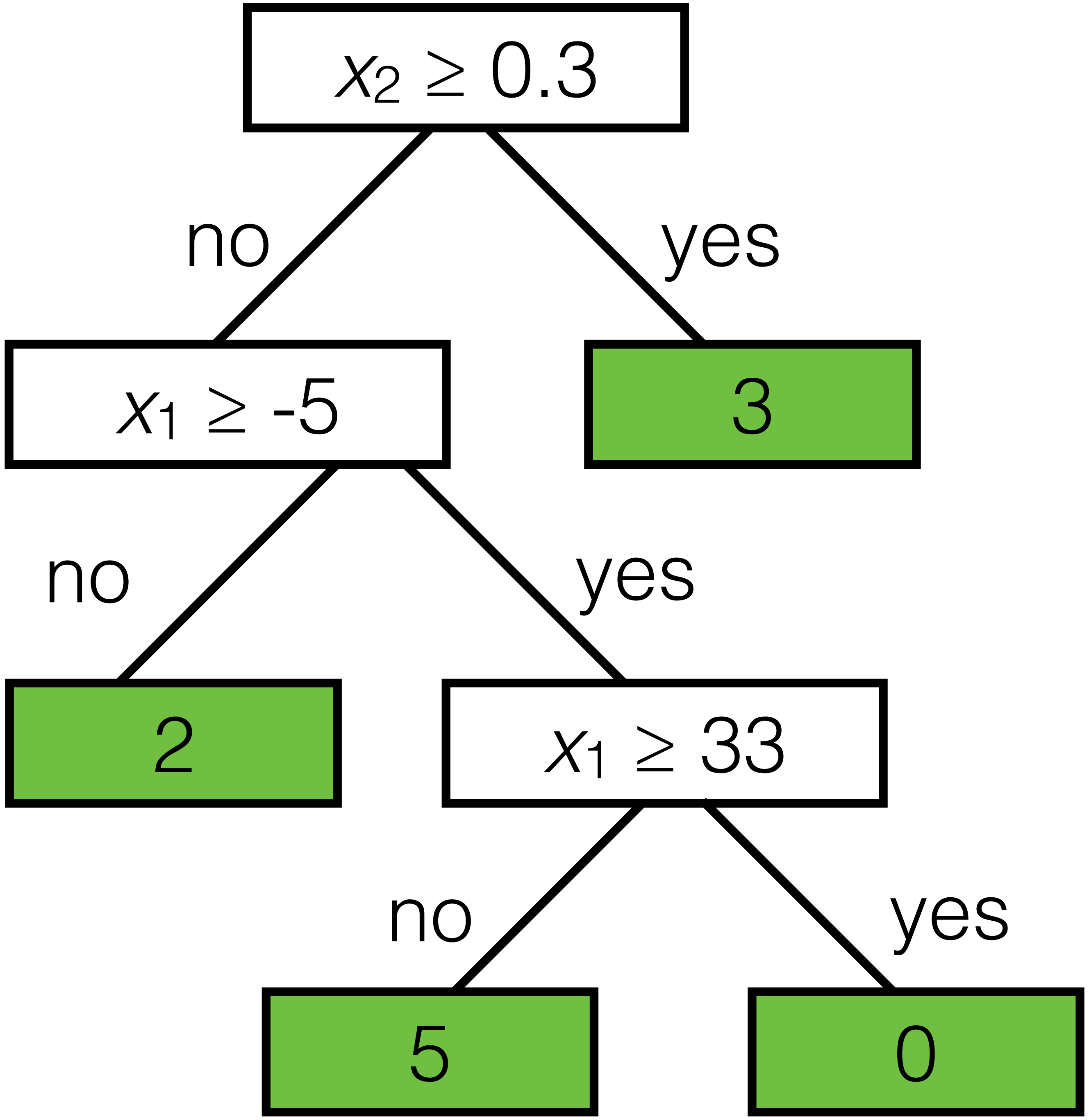
Regression tree



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

labels:
y: km run

Regression tree



features:

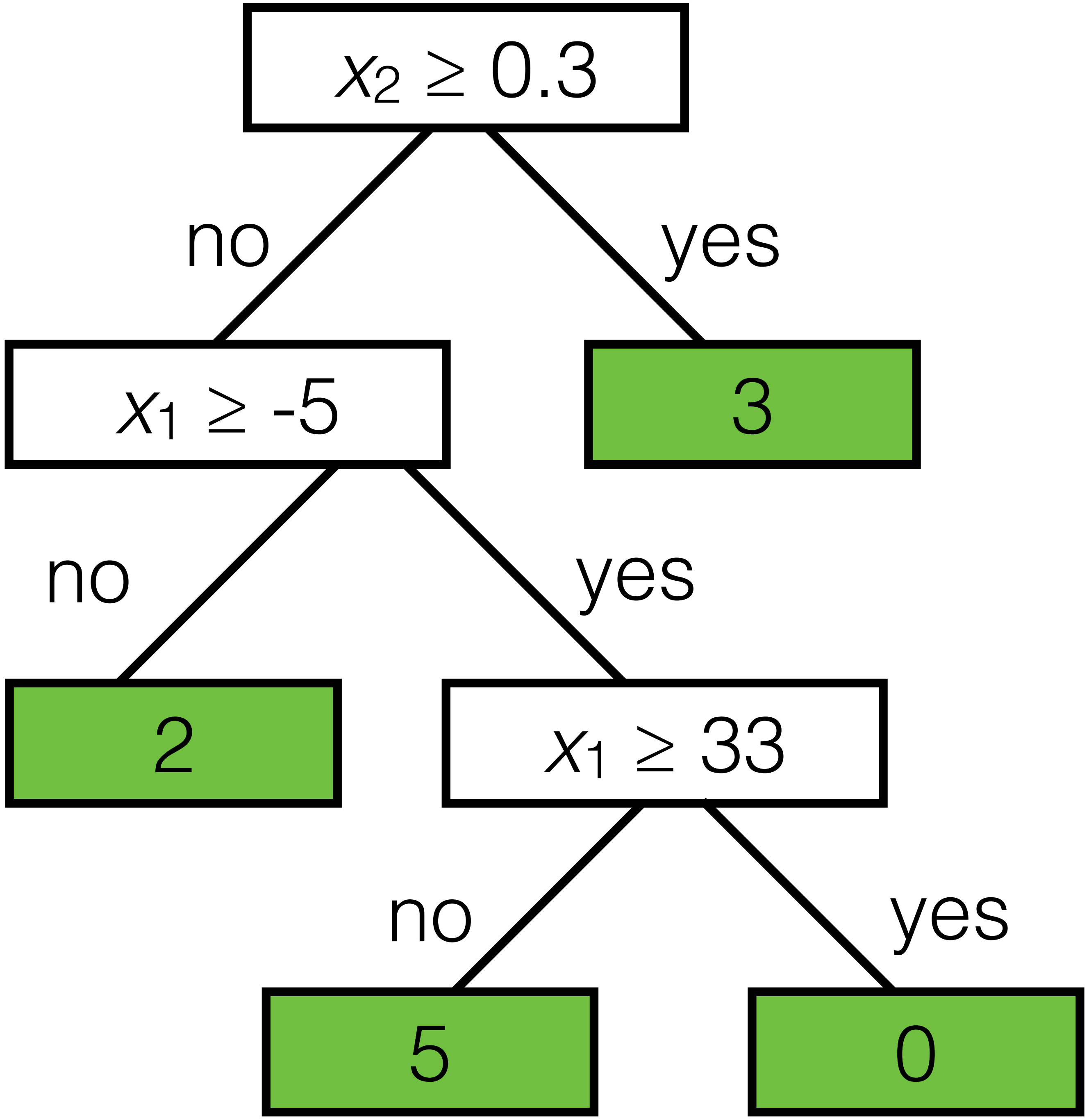
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

labels:

y: km run

Regression tree

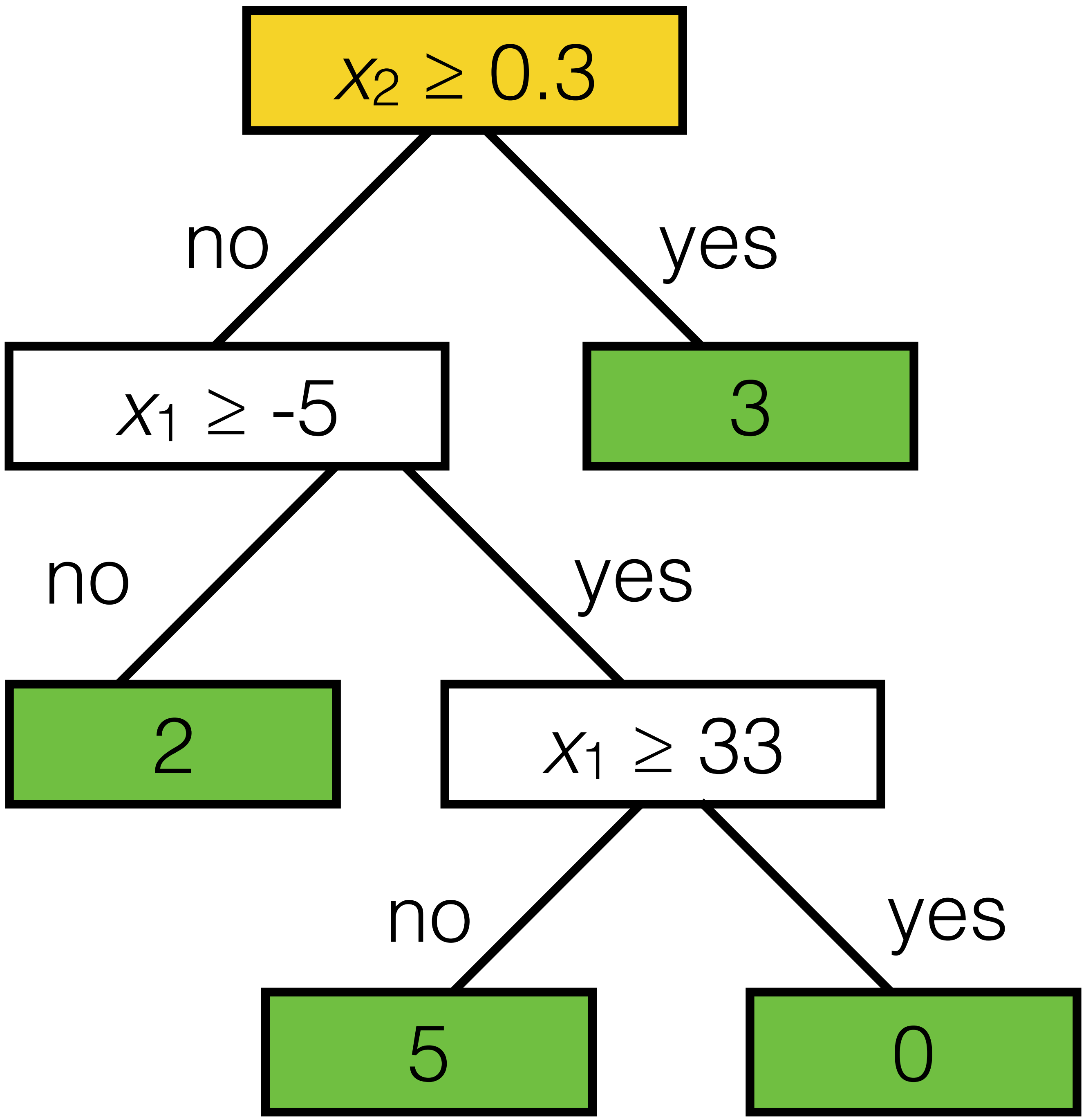


features:
x_1: temperature (deg C)
x_2: precipitation (cm/hr)

labels:
y: km run

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

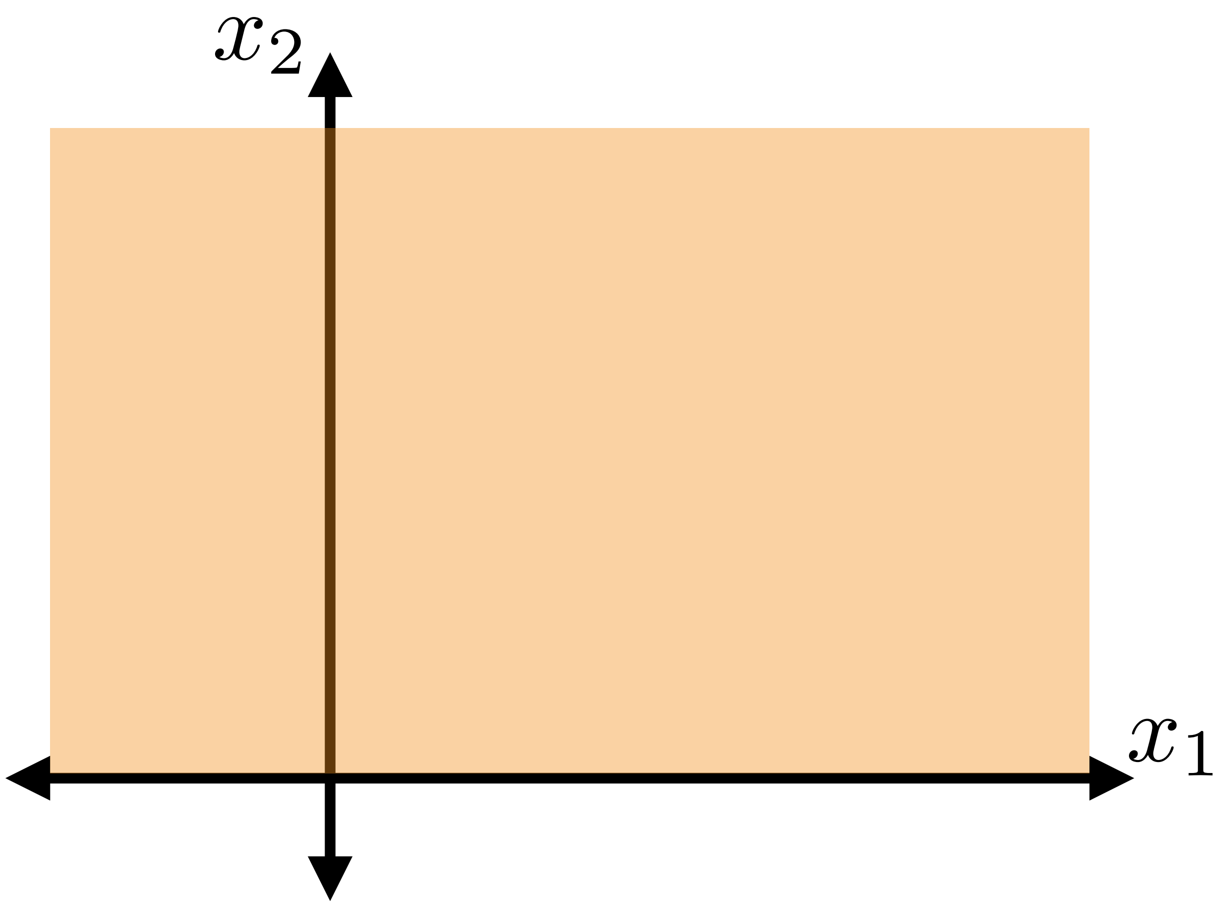
Regression tree



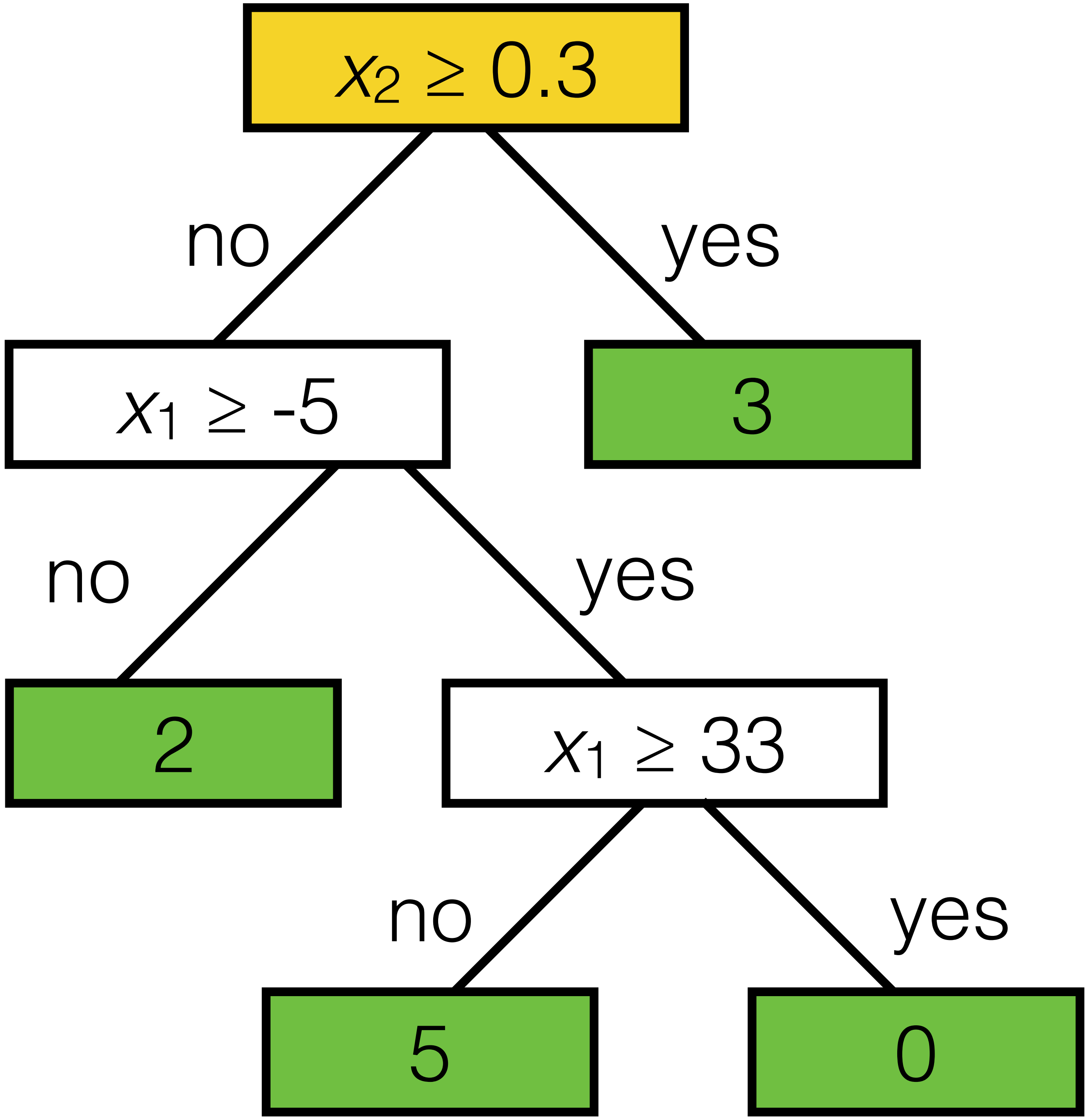
features:
x_1: temperature (deg C)
x_2: precipitation (cm/hr)

labels:
y: km run

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



Regression tree



$x_2 \geq 0.3$

no

yes

$x_1 \geq -5$

3

no

yes

2

$x_1 \geq 33$

no

yes

5

0

features:

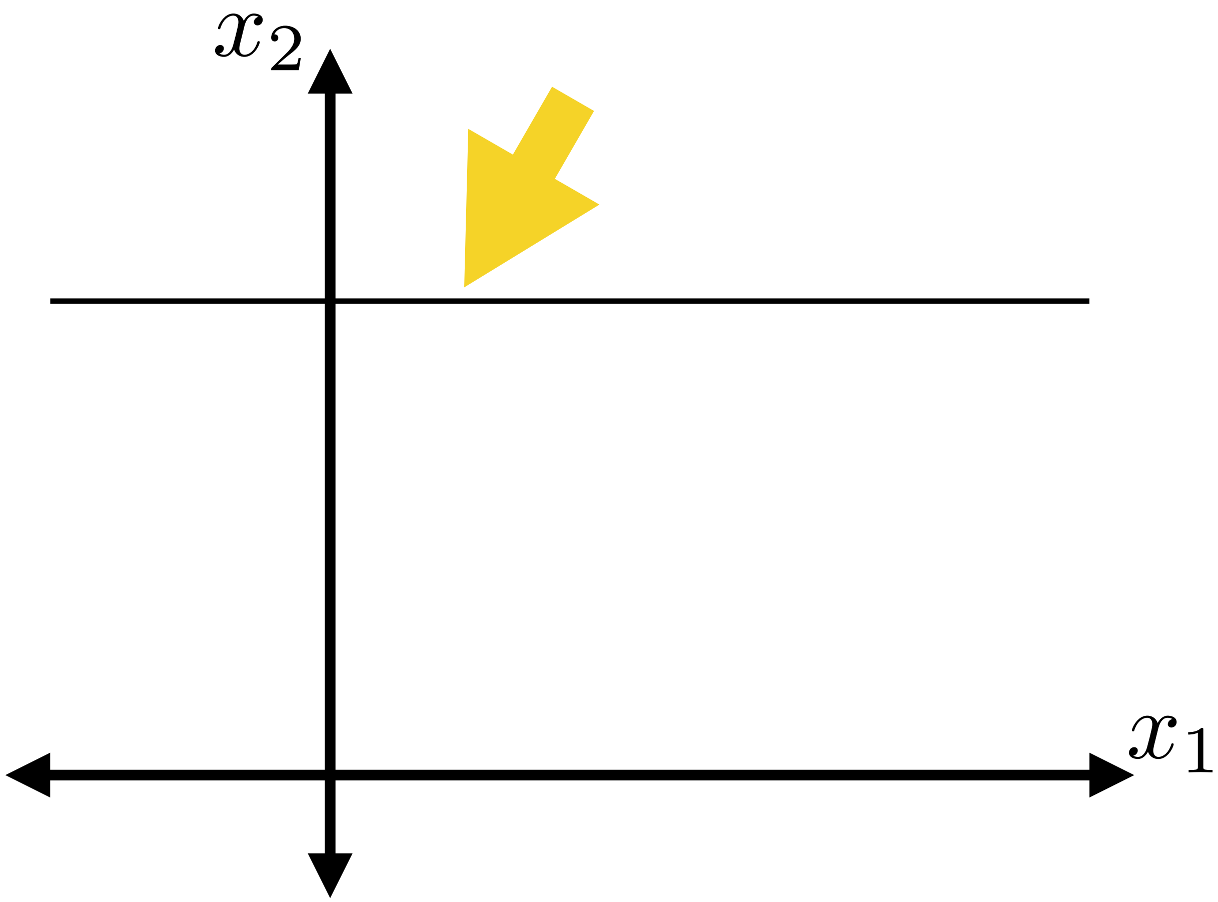
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

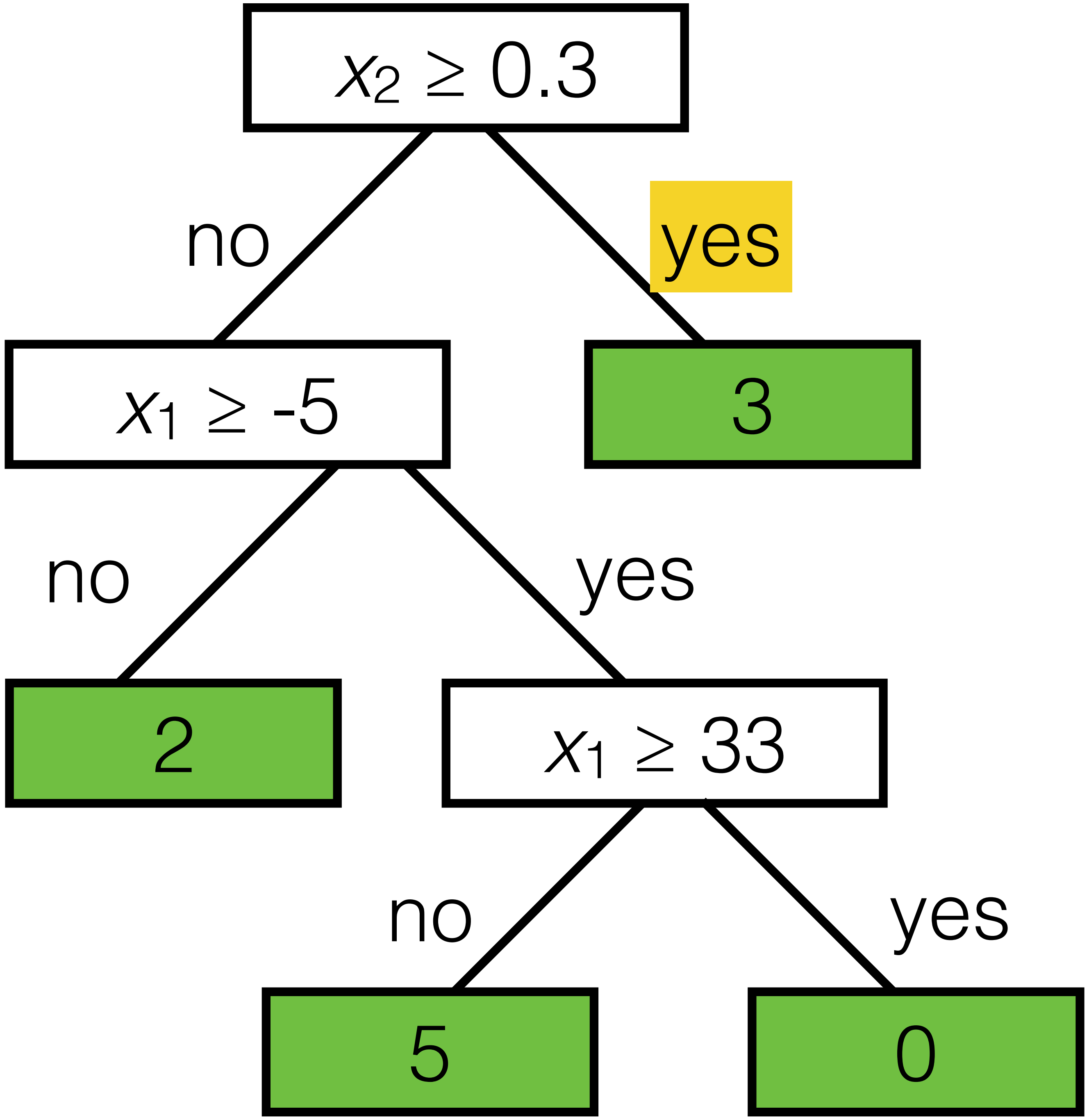
labels:

y: km run

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



Regression tree



$x_2 \geq 0.3$

no

yes

$x_1 \geq -5$

3

no

yes

2

$x_1 \geq 33$

no

yes

5

0

features:

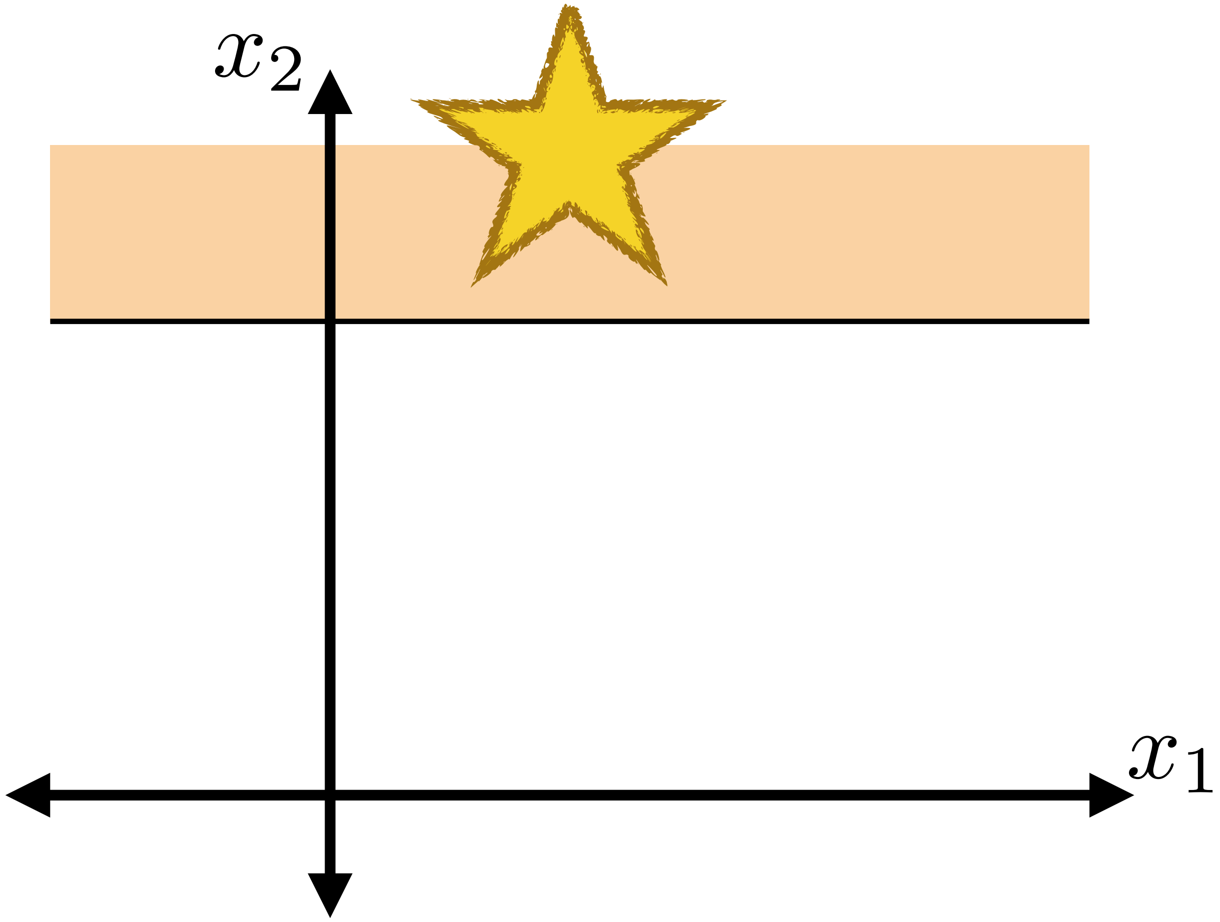
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

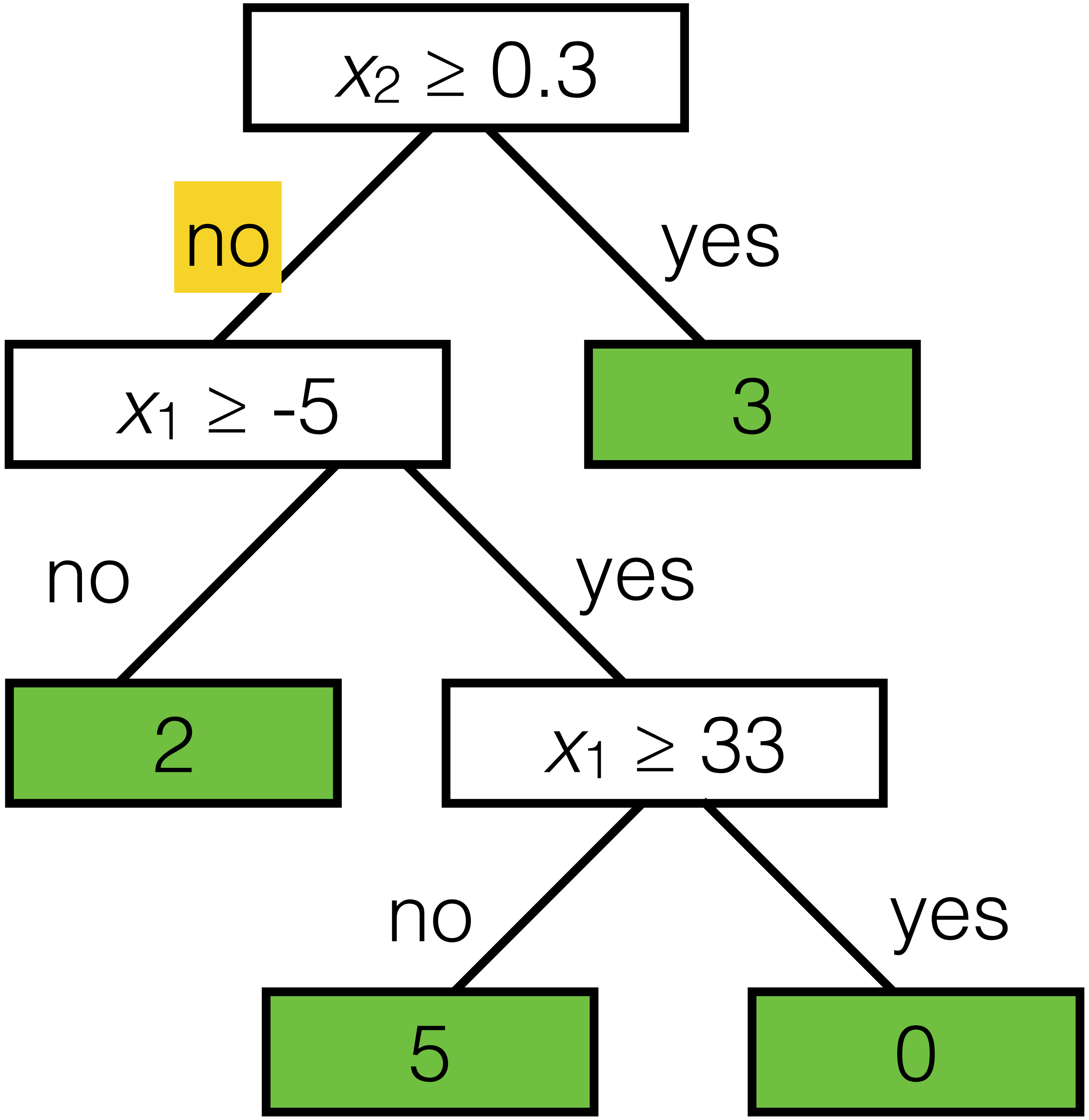
labels:

y: km run

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



Regression tree



$x_2 \geq 0.3$

no

yes

$x_1 \geq -5$

3

no

yes

2

$x_1 \geq 33$

no

yes

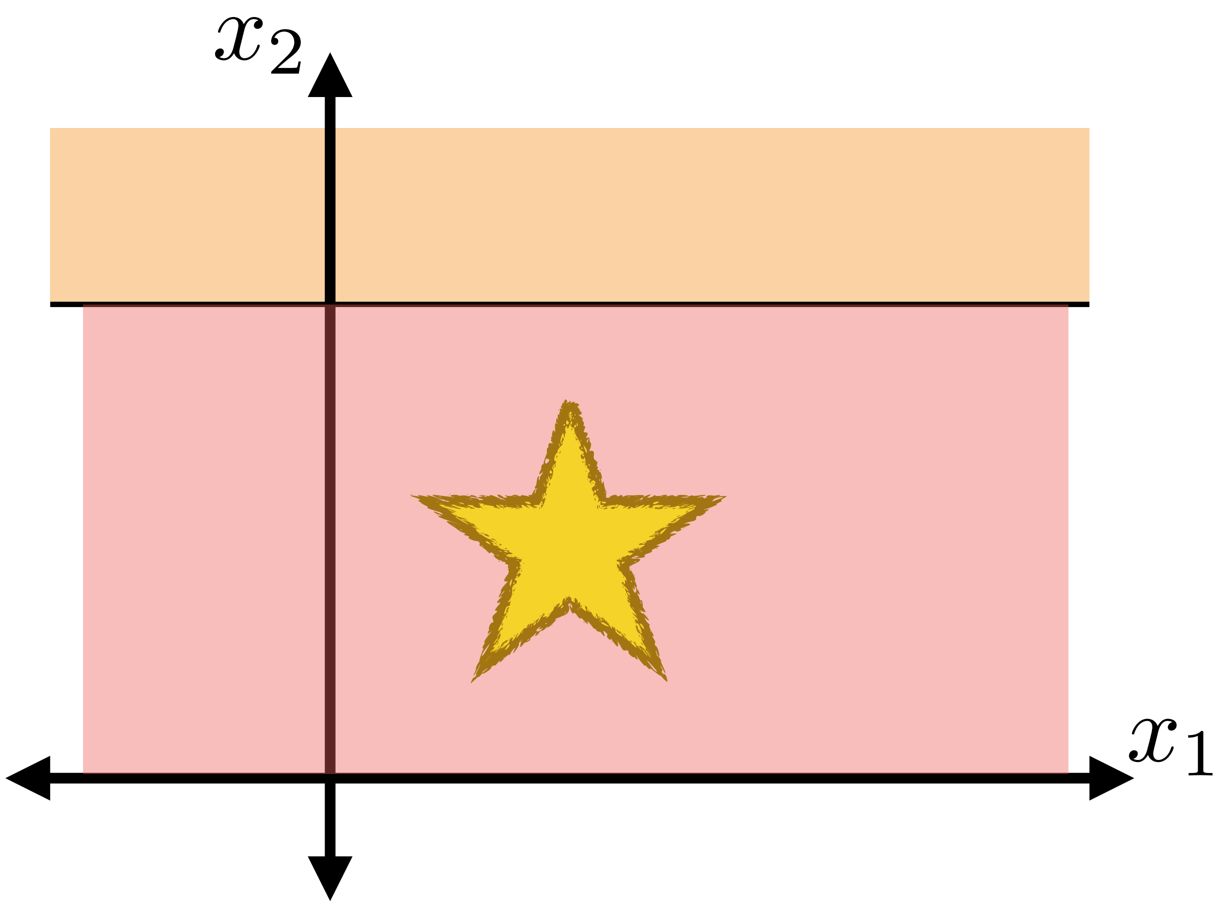
5

0

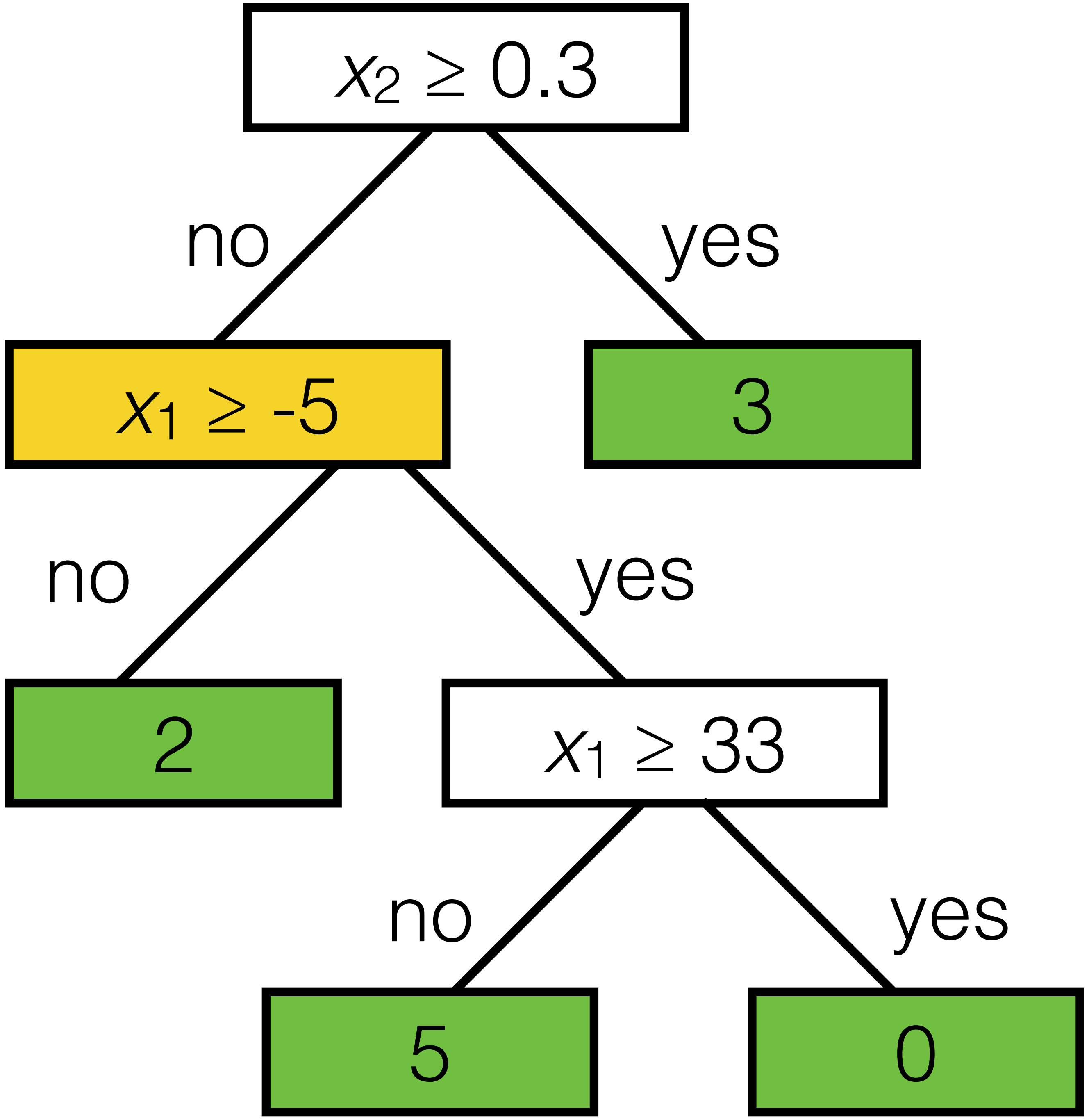
features:
x_1: temperature (deg C)
x_2: precipitation (cm/hr)

labels:
y: km run

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



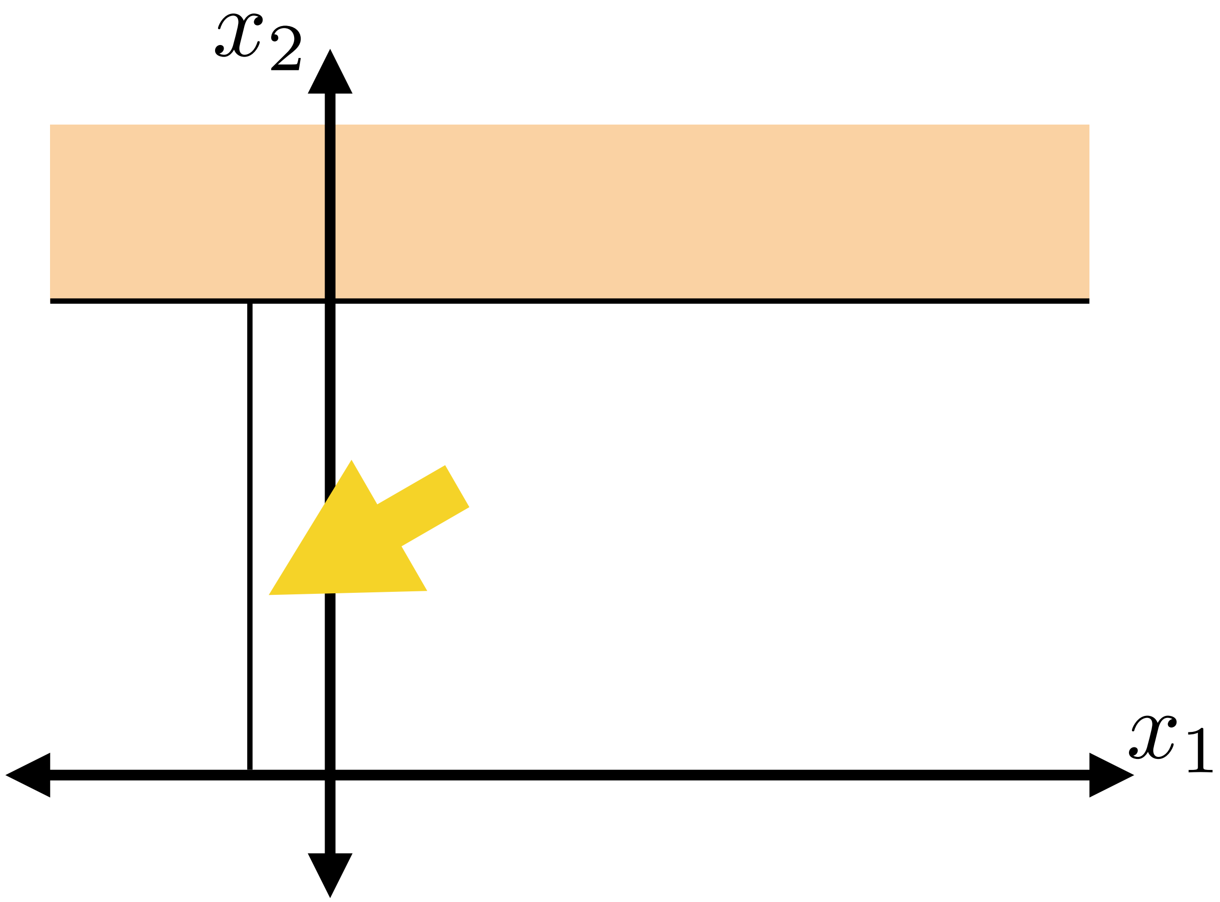
Regression tree



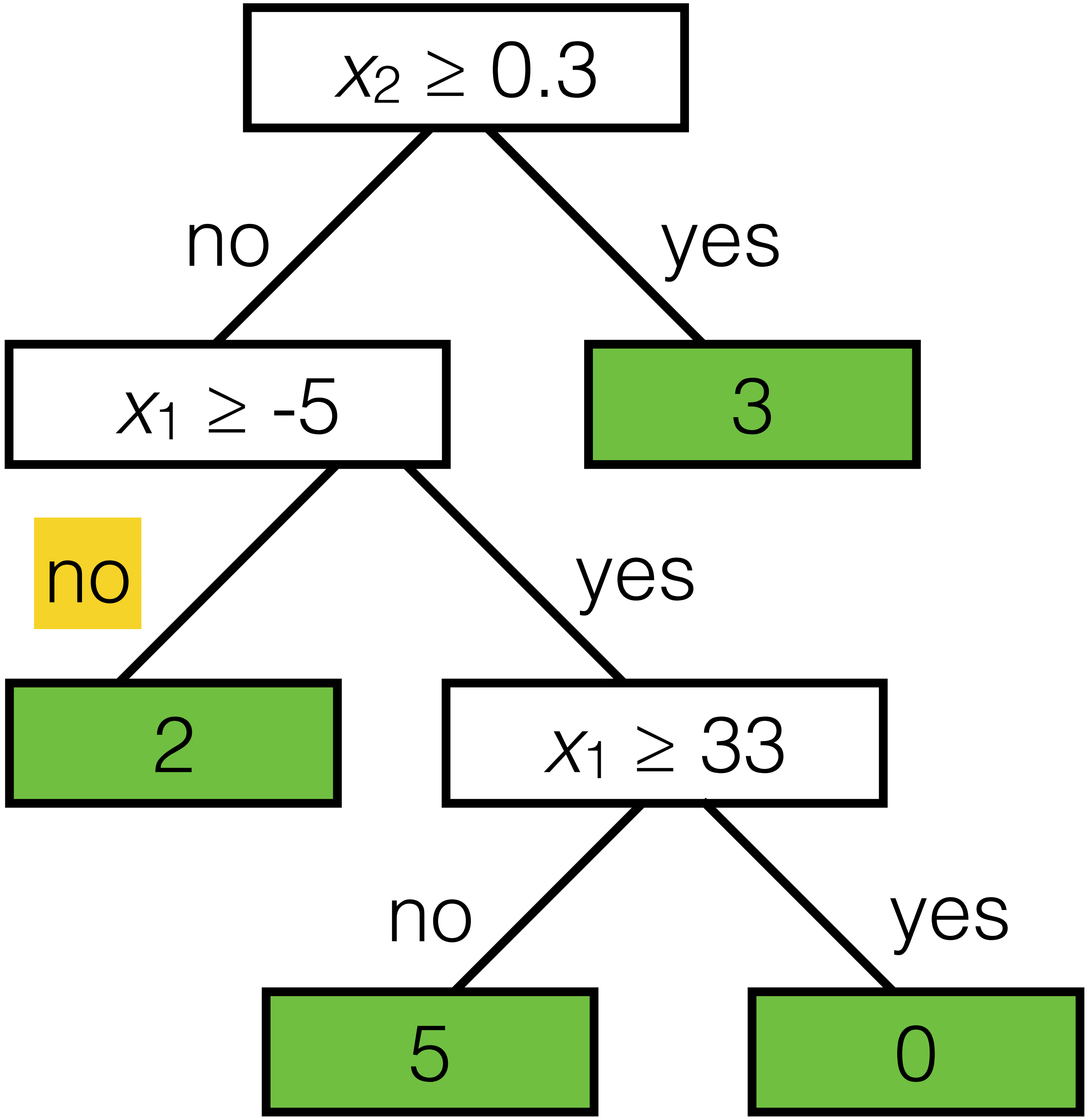
features:
x_1: temperature (deg C)
x_2: precipitation (cm/hr)

labels:
y: km run

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



Regression tree



$x_2 \geq 0.3$

no

yes

$x_1 \geq -5$

3

no

yes

2

$x_1 \geq 33$

no

yes

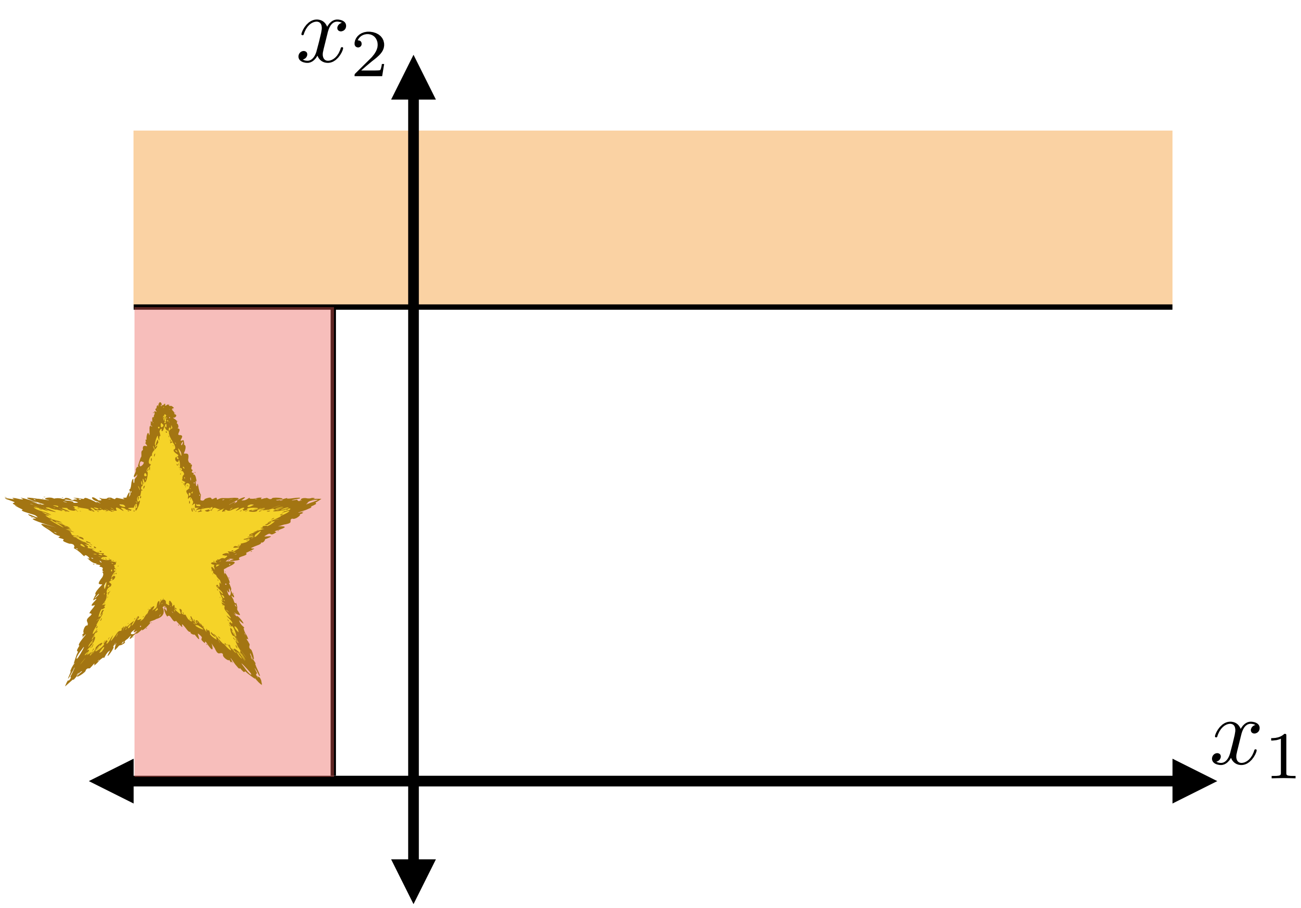
5

0

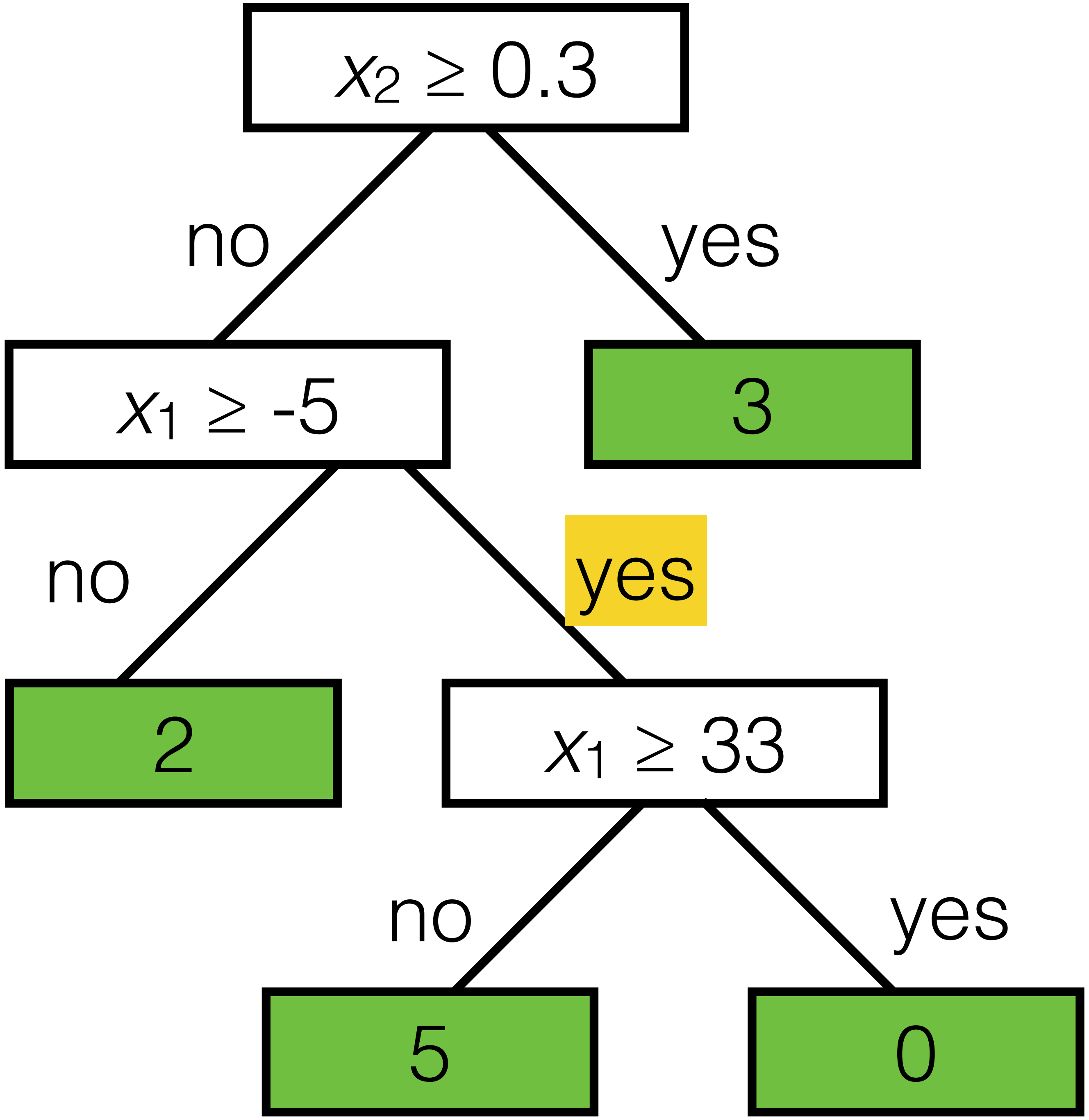
features:
x_1: temperature (deg C)
x_2: precipitation (cm/hr)

labels:
y: km run

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



Regression tree



$x_2 \geq 0.3$

no

yes

$x_1 \geq -5$

3

no

yes

2

$x_1 \geq 33$

no

yes

5

0

features:

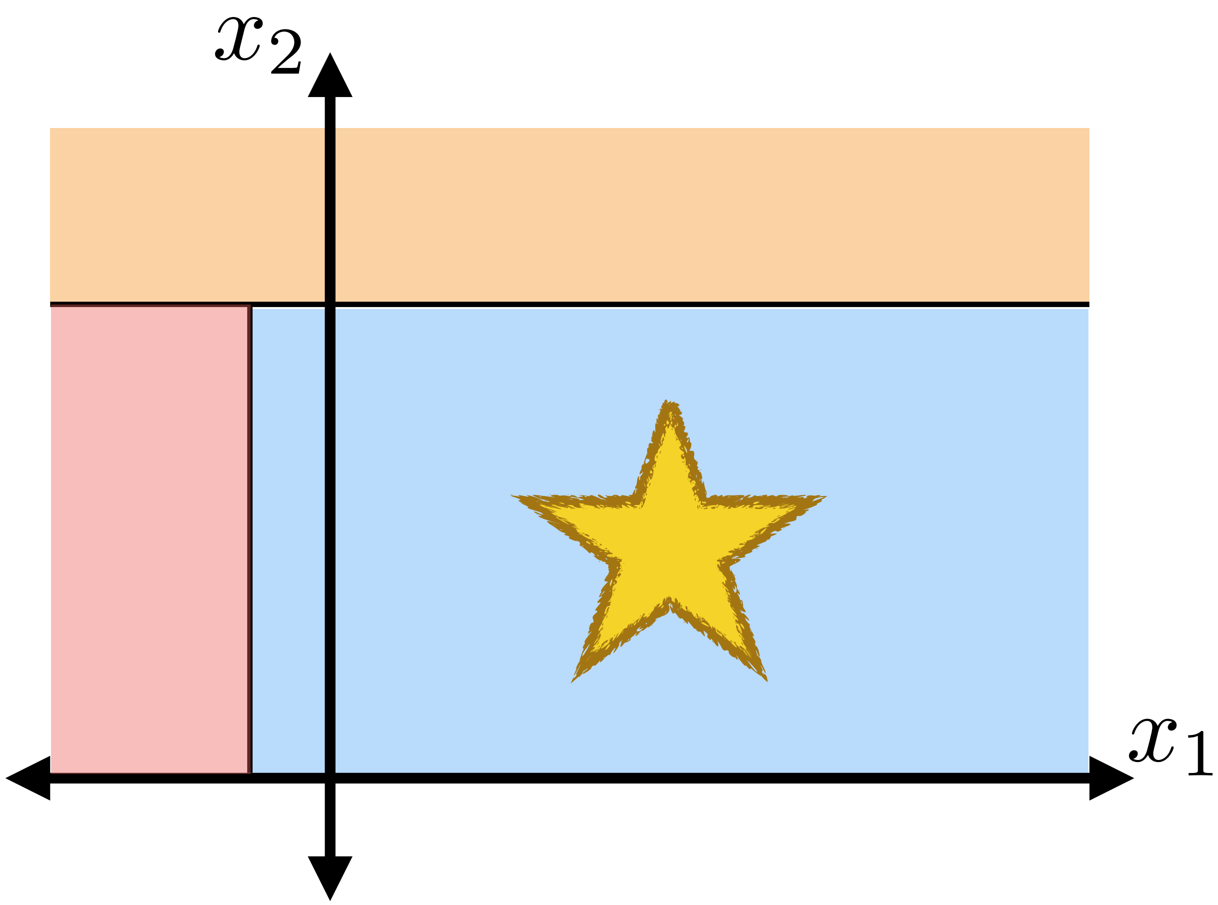
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

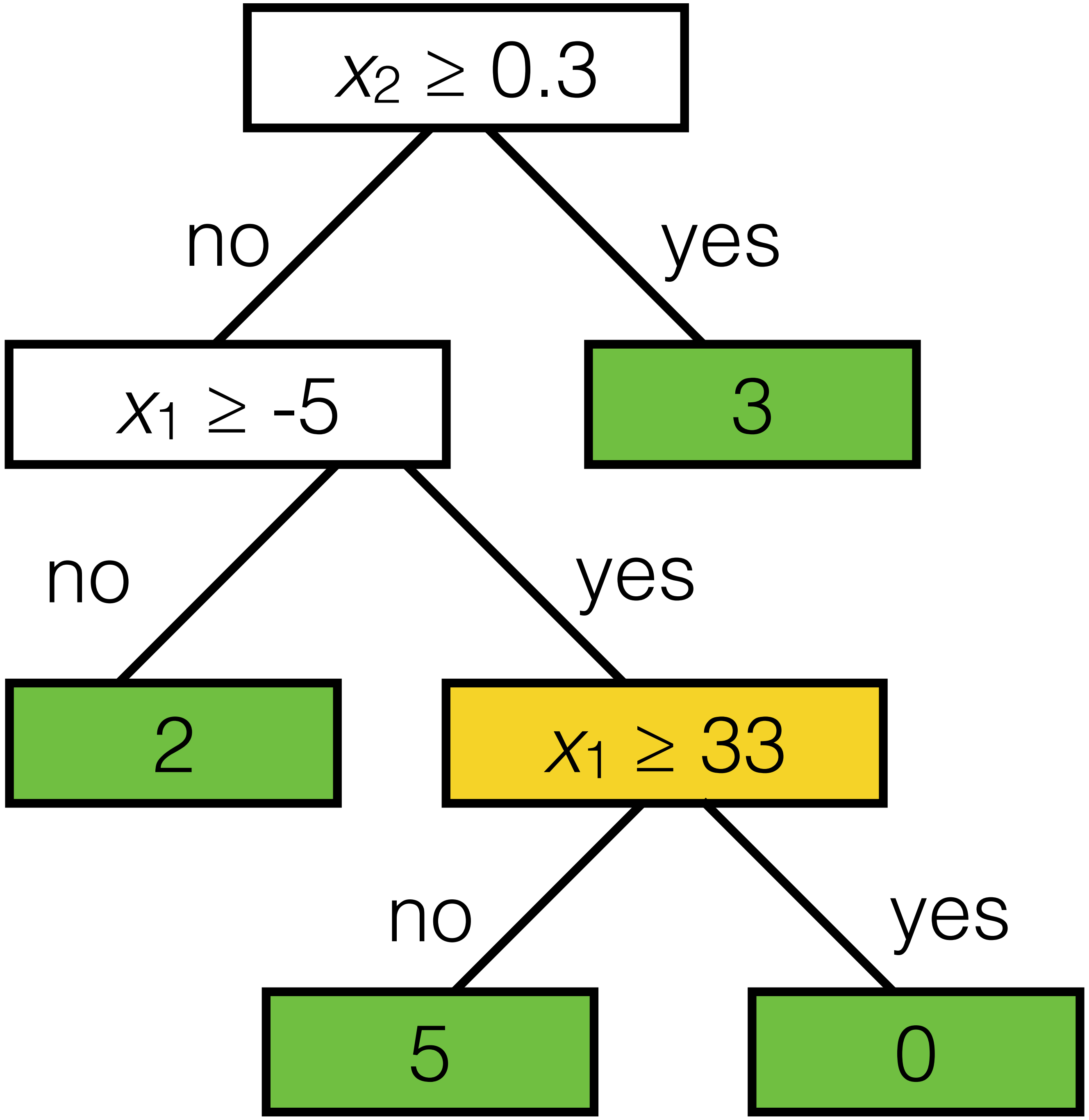
labels:

y: km run

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



Regression tree



$x_2 \geq 0.3$

no

yes

$x_1 \geq -5$

3

no

yes

2

$x_1 \geq 33$

no

yes

5

0

features:

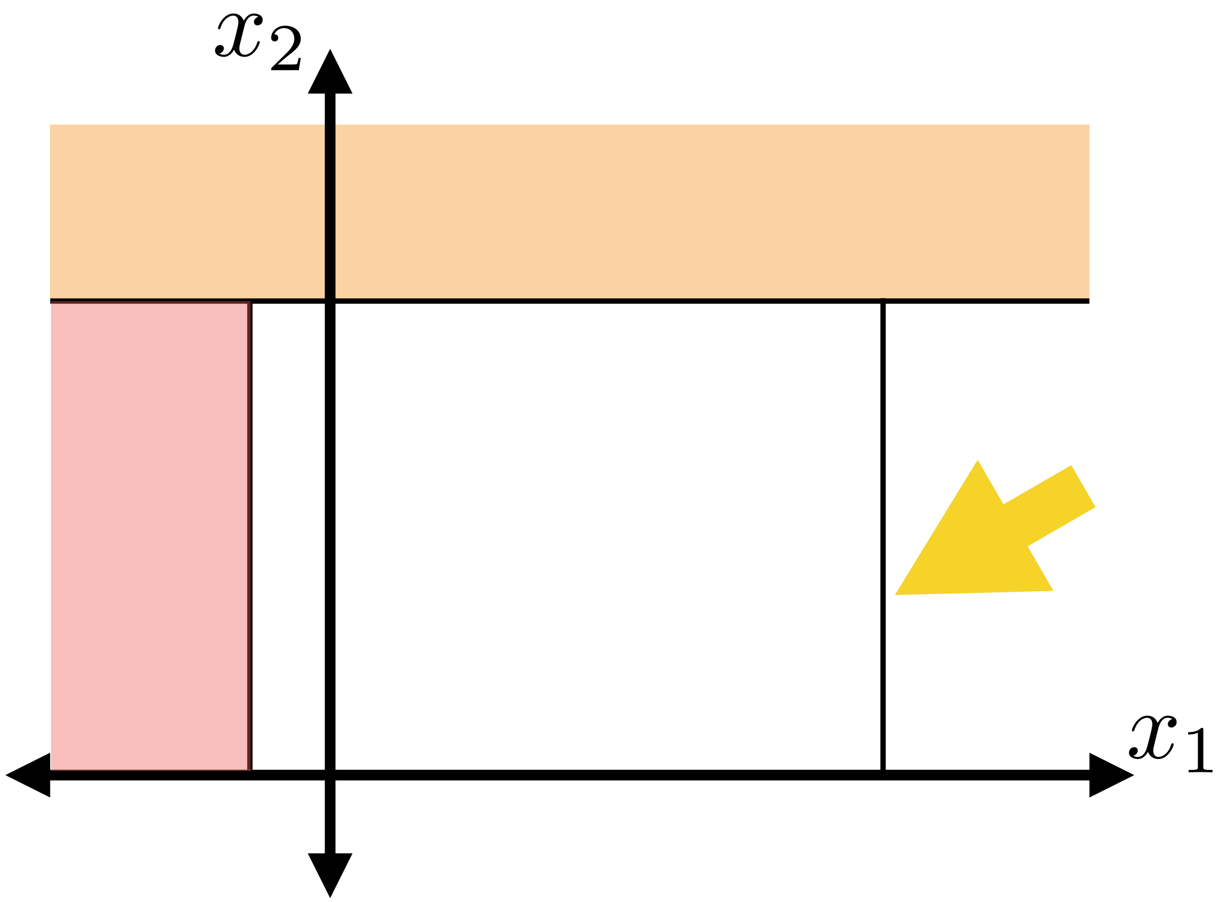
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

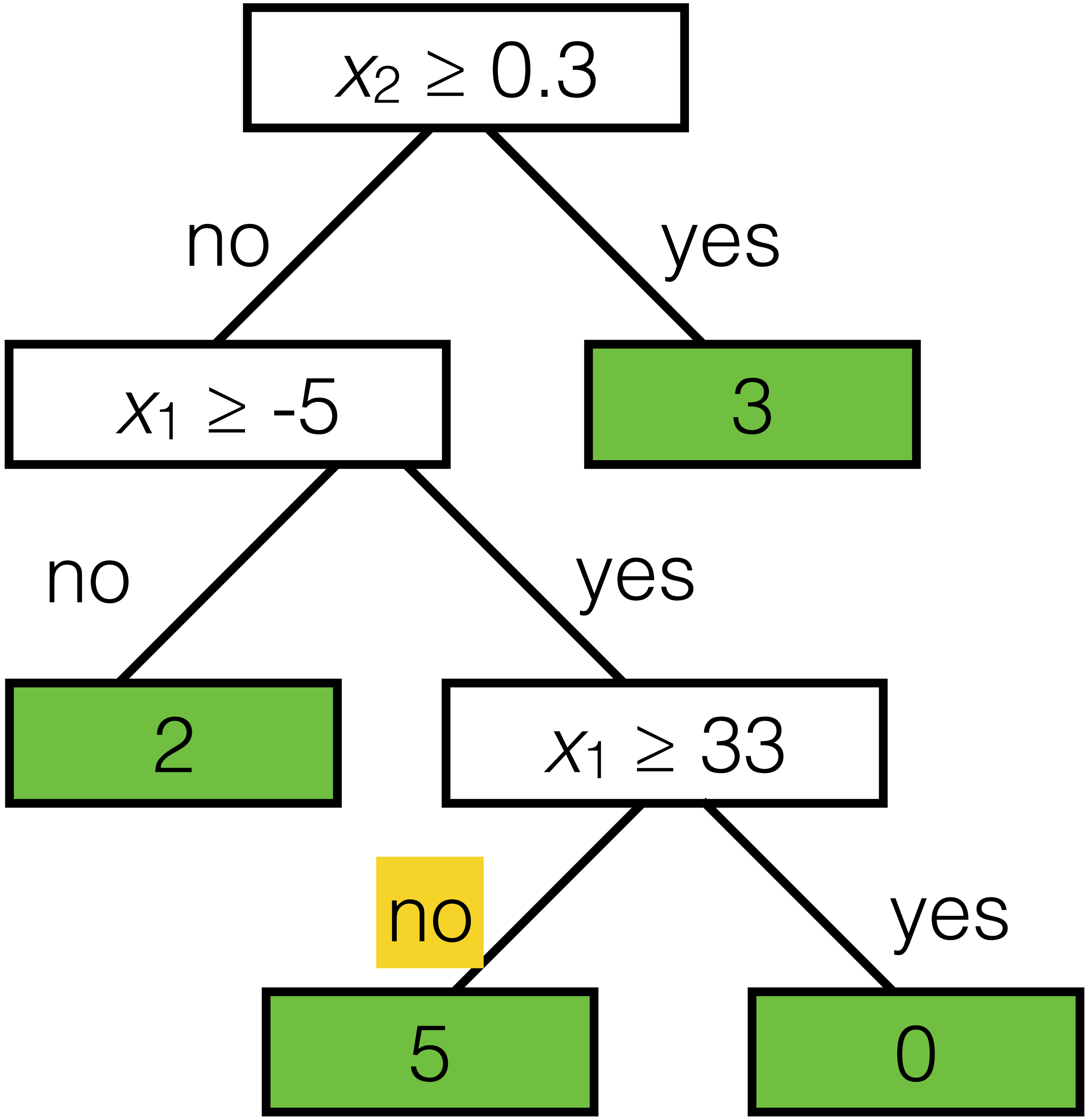
labels:

y: km run

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



Regression tree



$x_2 \geq 0.3$

no

yes

$x_1 \geq -5$

3

no

yes

2

$x_1 \geq 33$

no

yes

5

0

features:

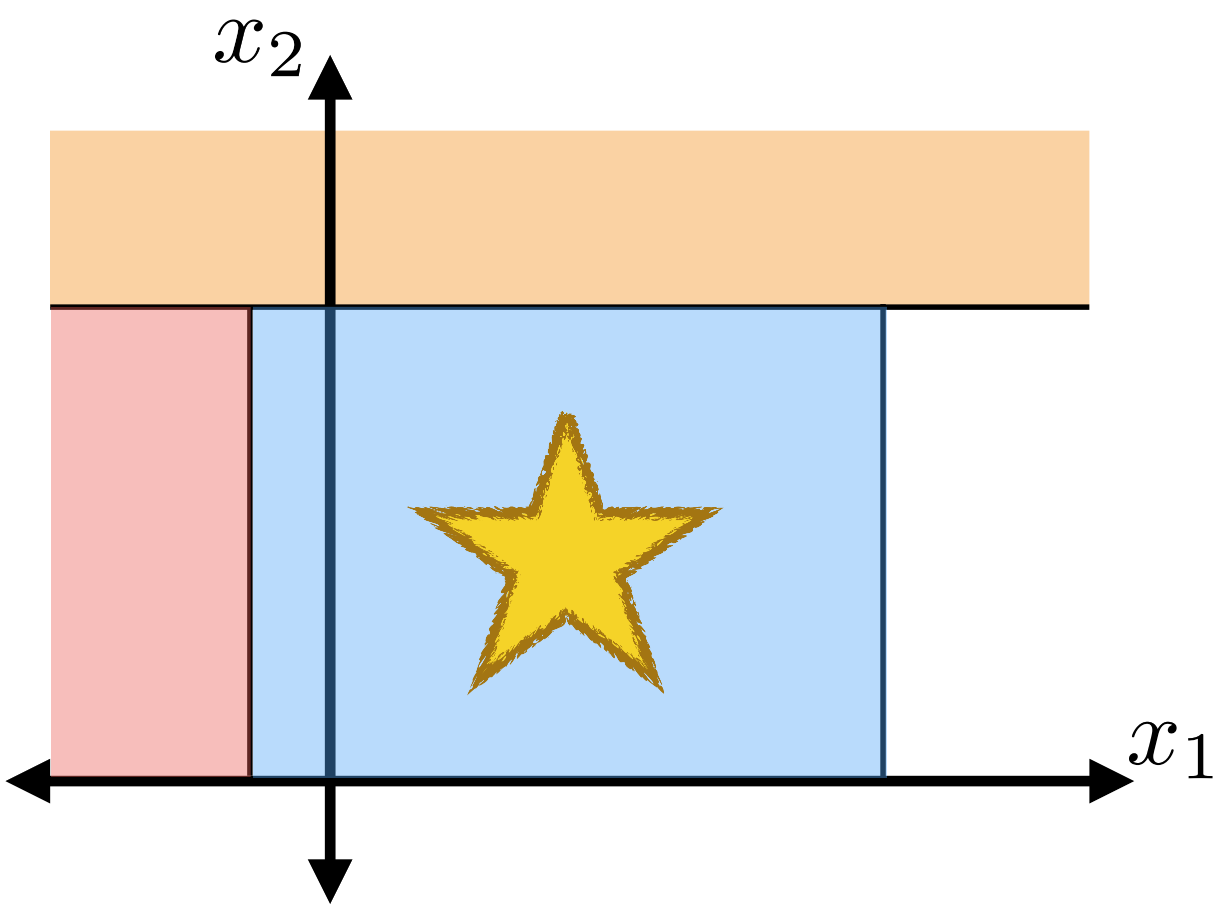
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

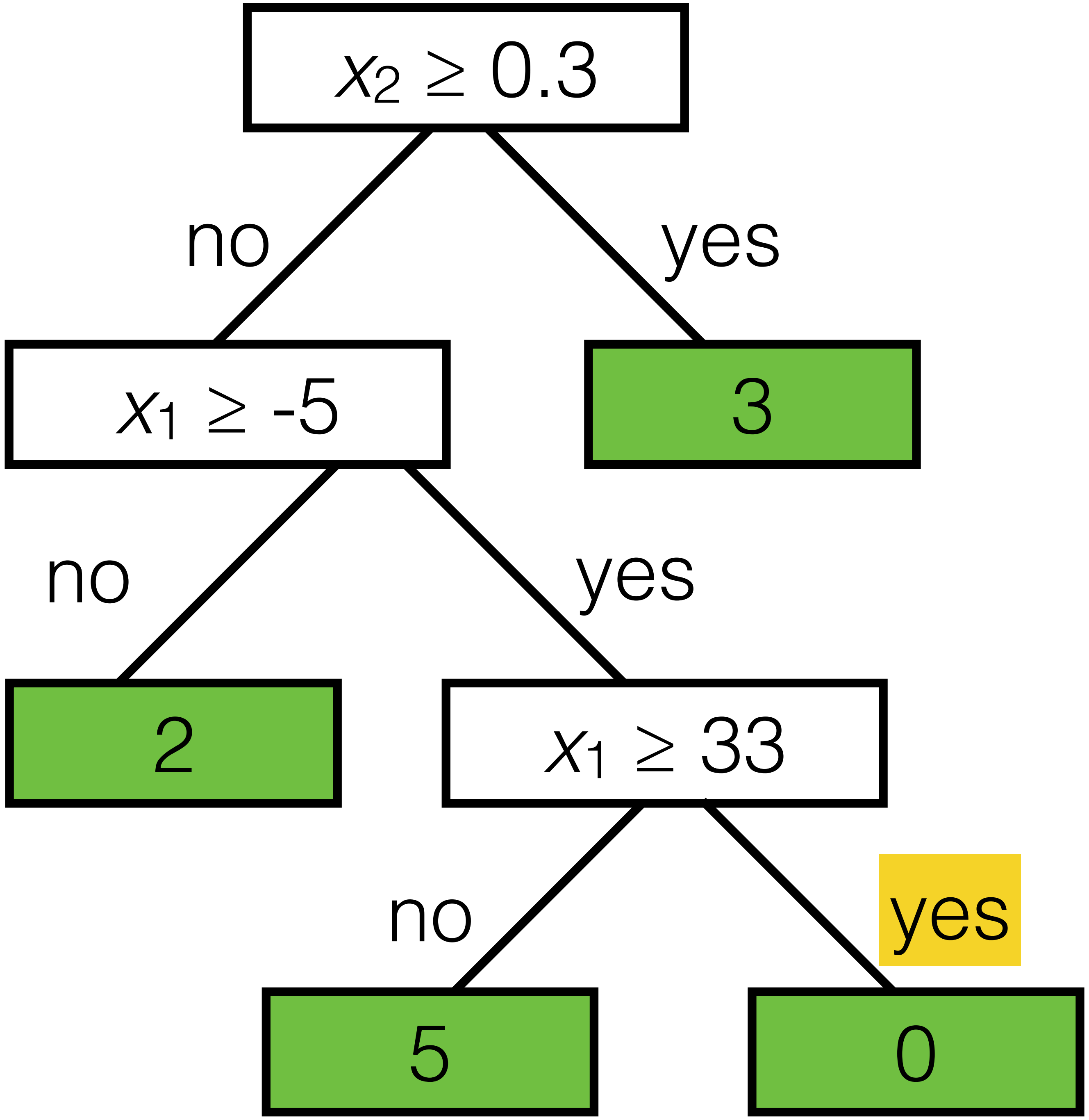
labels:

y: km run

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



Regression tree



$x_2 \geq 0.3$

no

yes

$x_1 \geq -5$

3

no

yes

2

$x_1 \geq 33$

no

yes

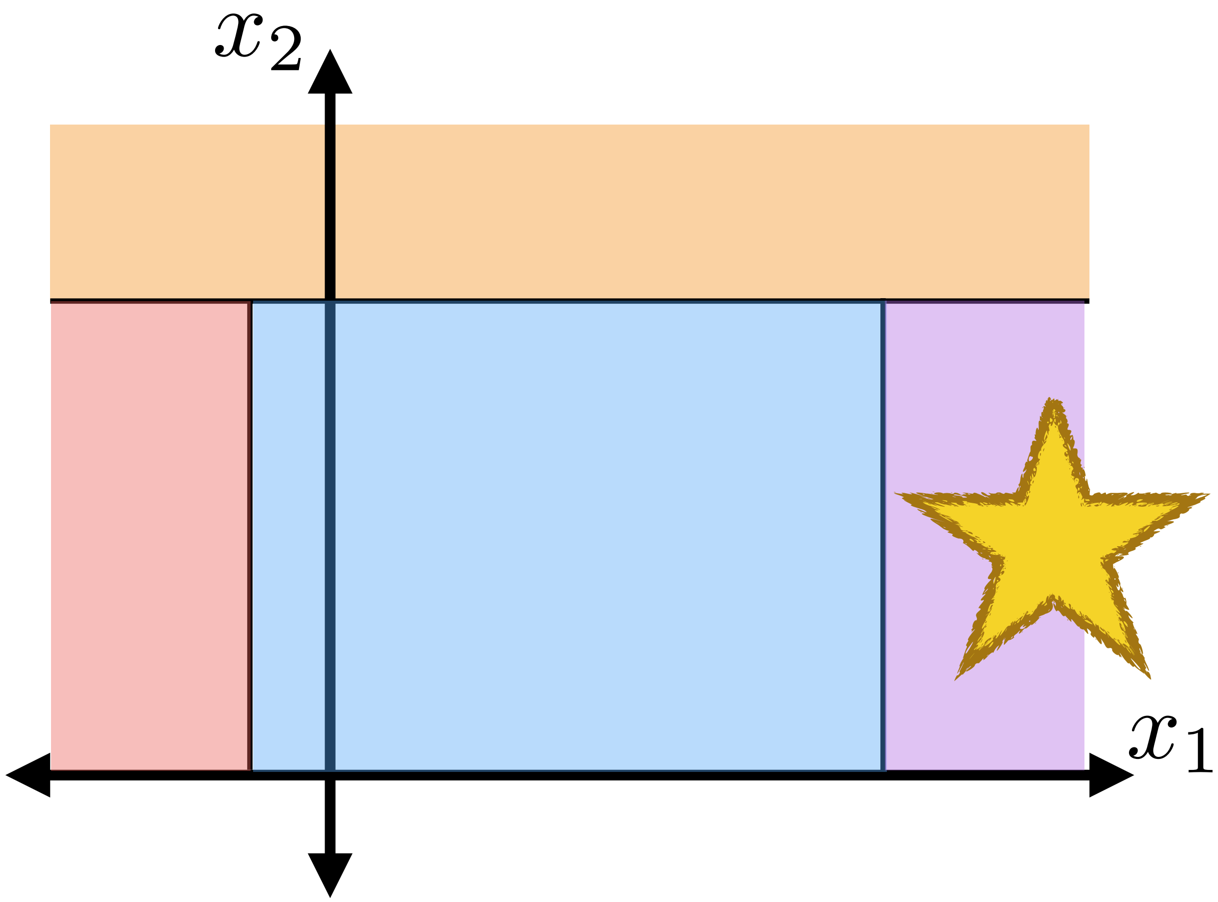
5

0

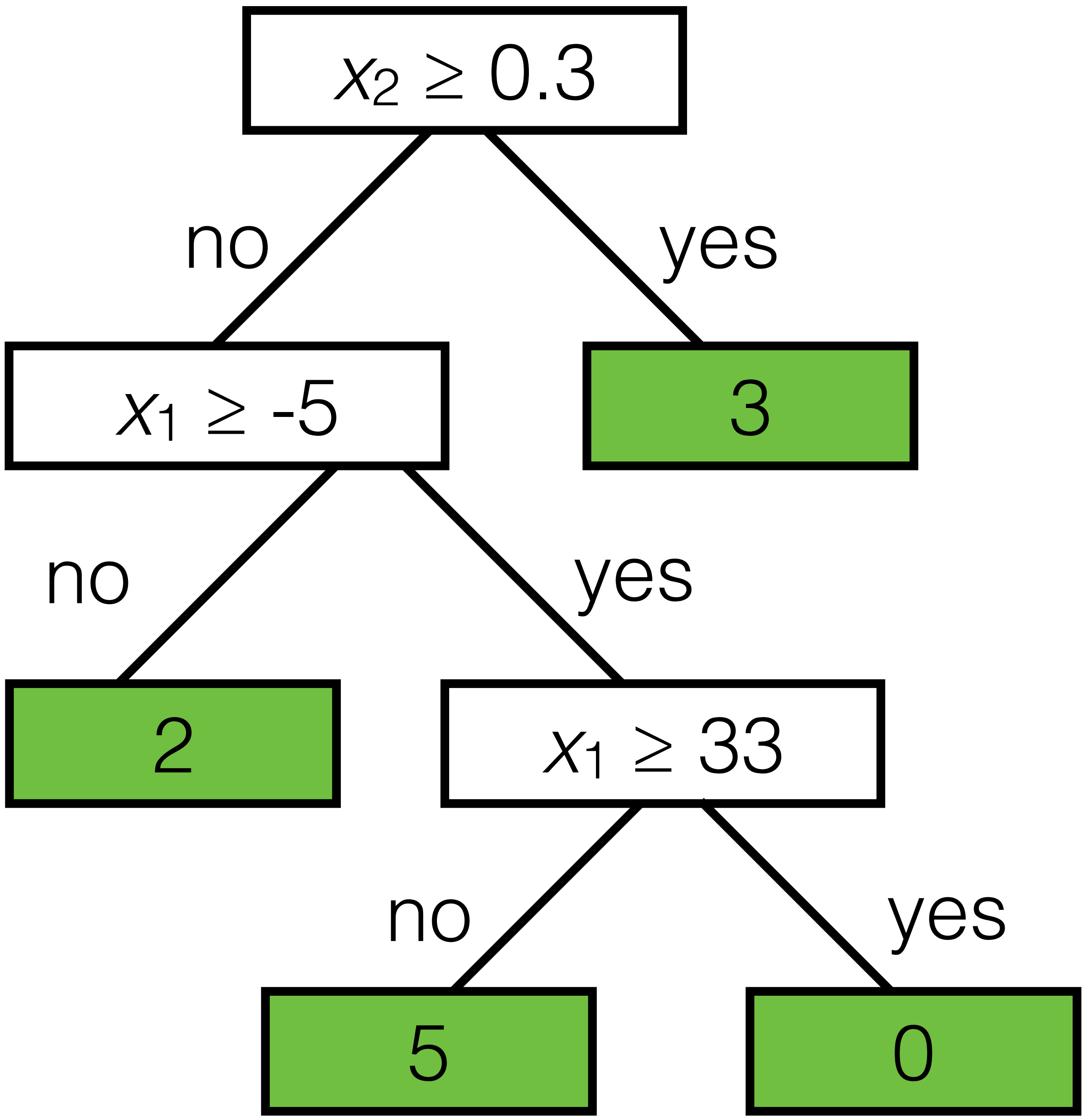
features:
x_1: temperature (deg C)
x_2: precipitation (cm/hr)

labels:
y: km run

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



Regression tree


$x_2 \geq 0.3$

no

yes

$x_1 \geq -5$

3

no

yes

2

$x_1 \geq 33$

no

yes

5

0

features:

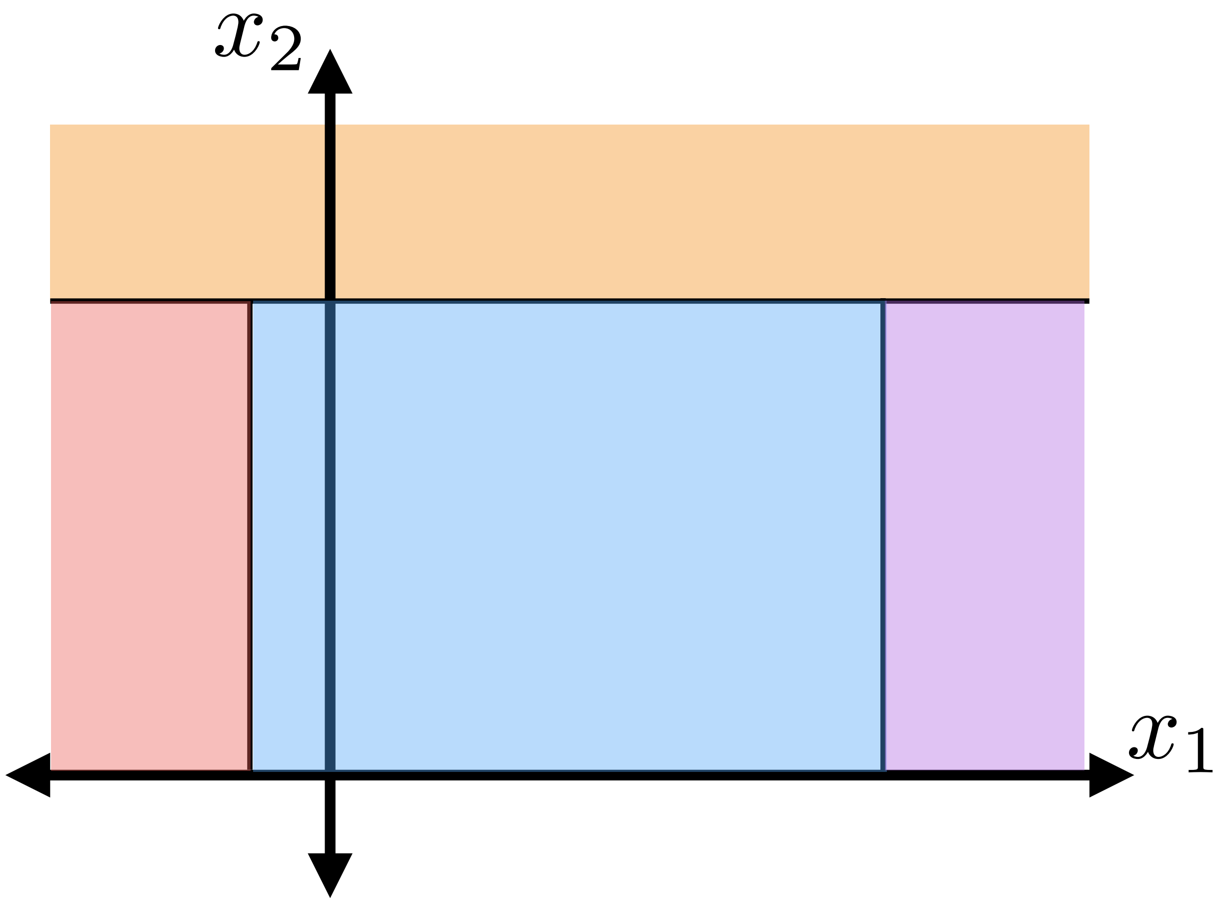
x_1: temperature (deg C)

x_2: precipitation (cm/hr)

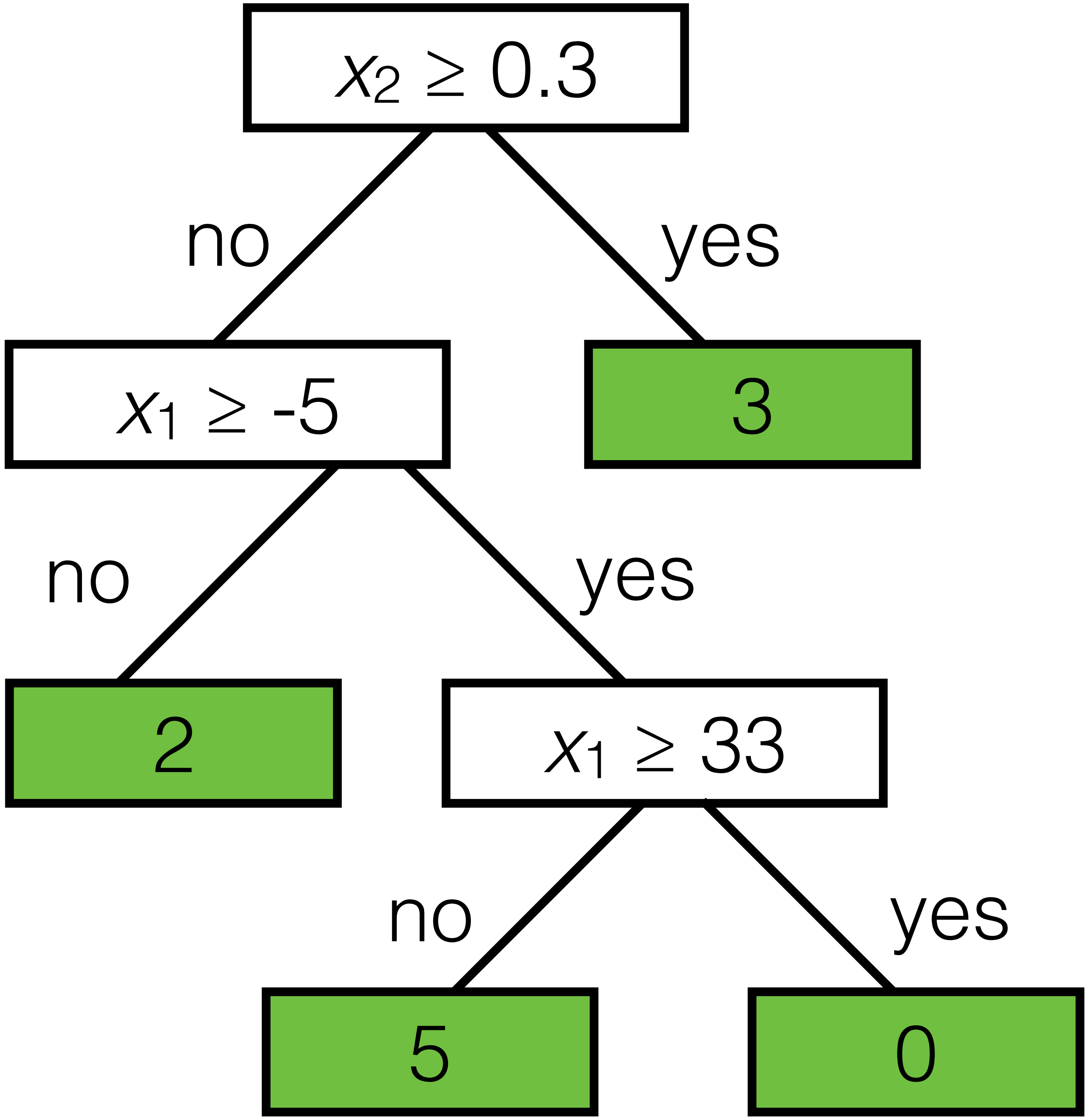
labels:

y: km run

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

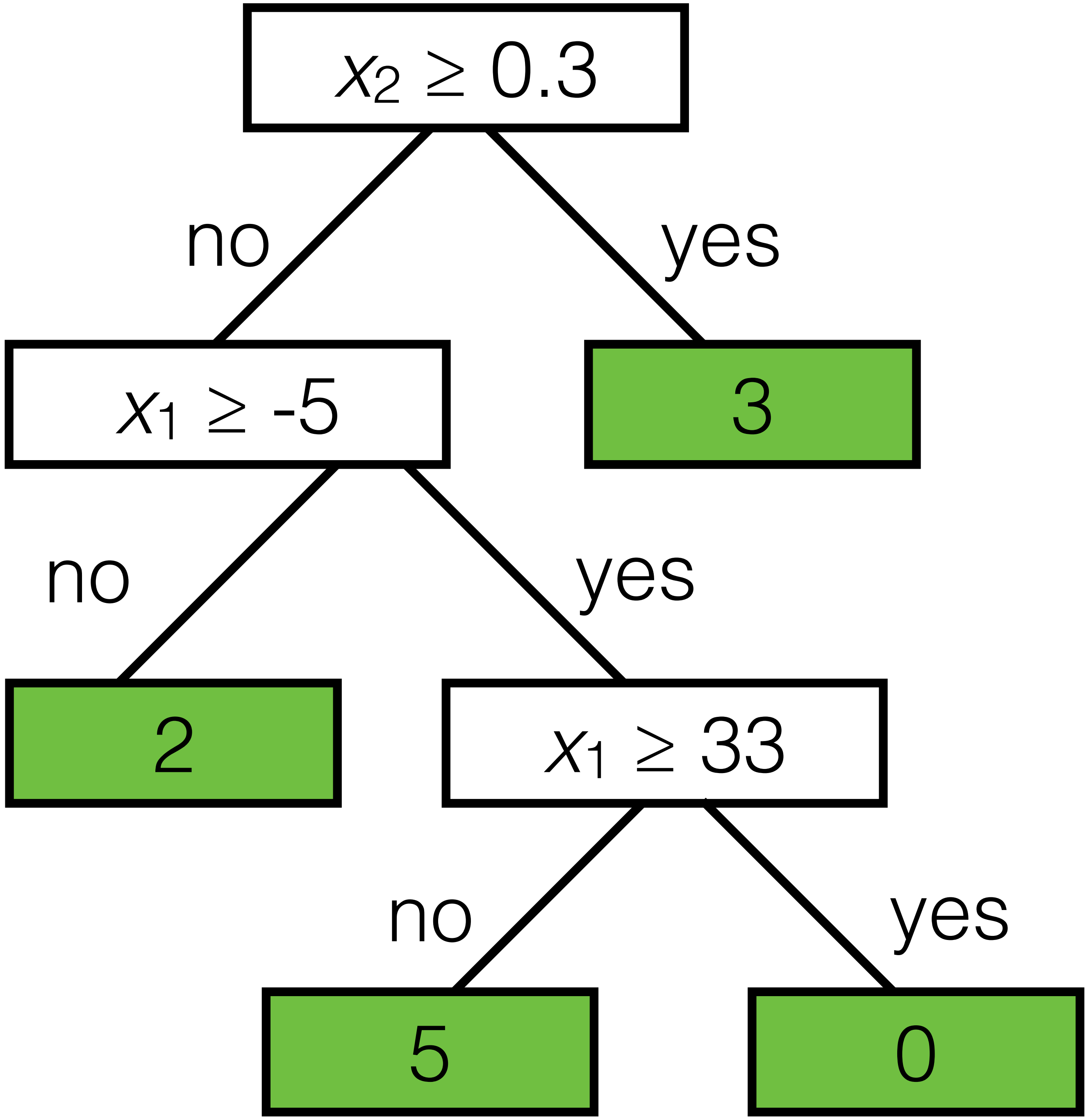


Decision tree

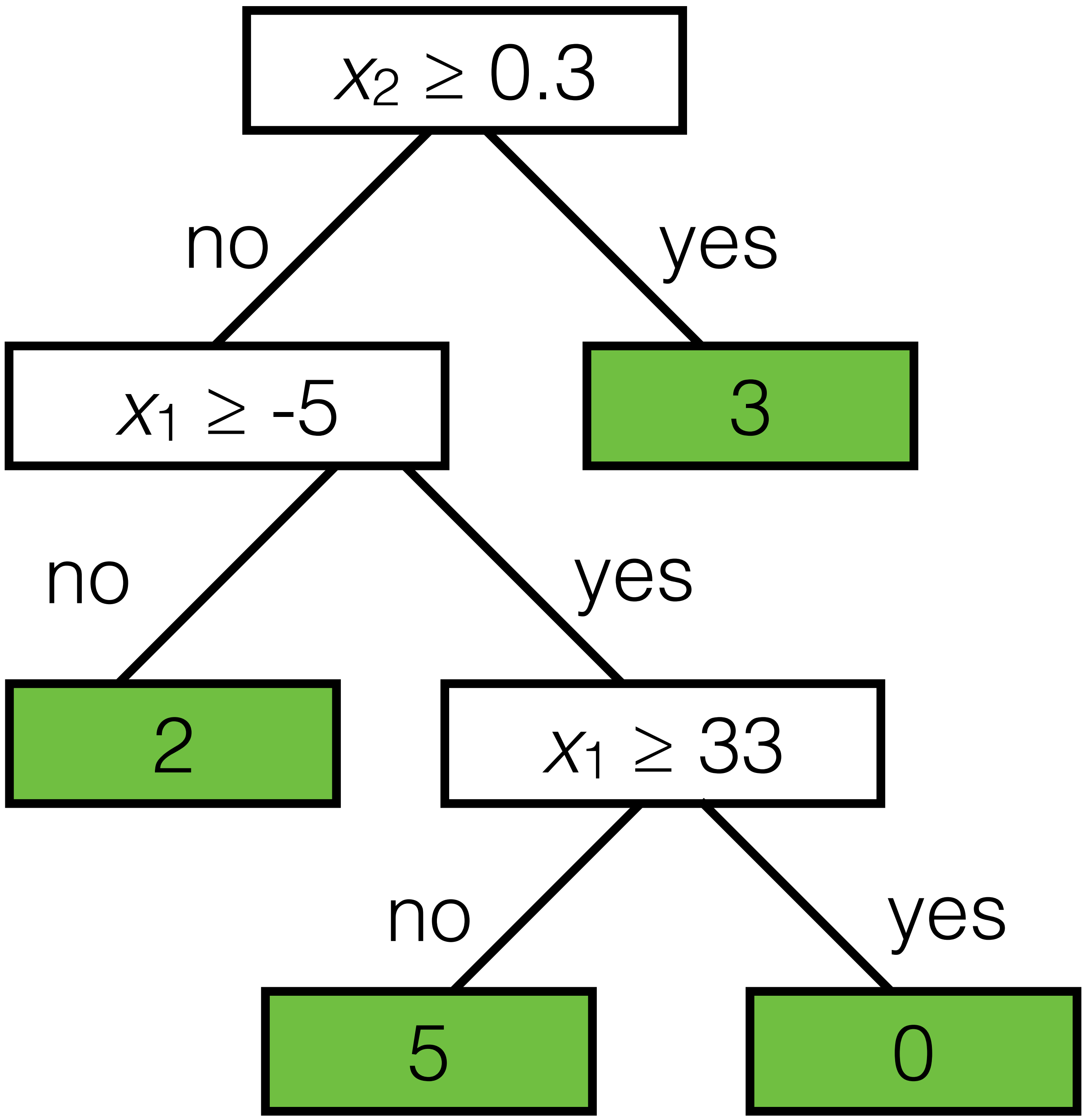


Decision tree

Recall: familiar pattern



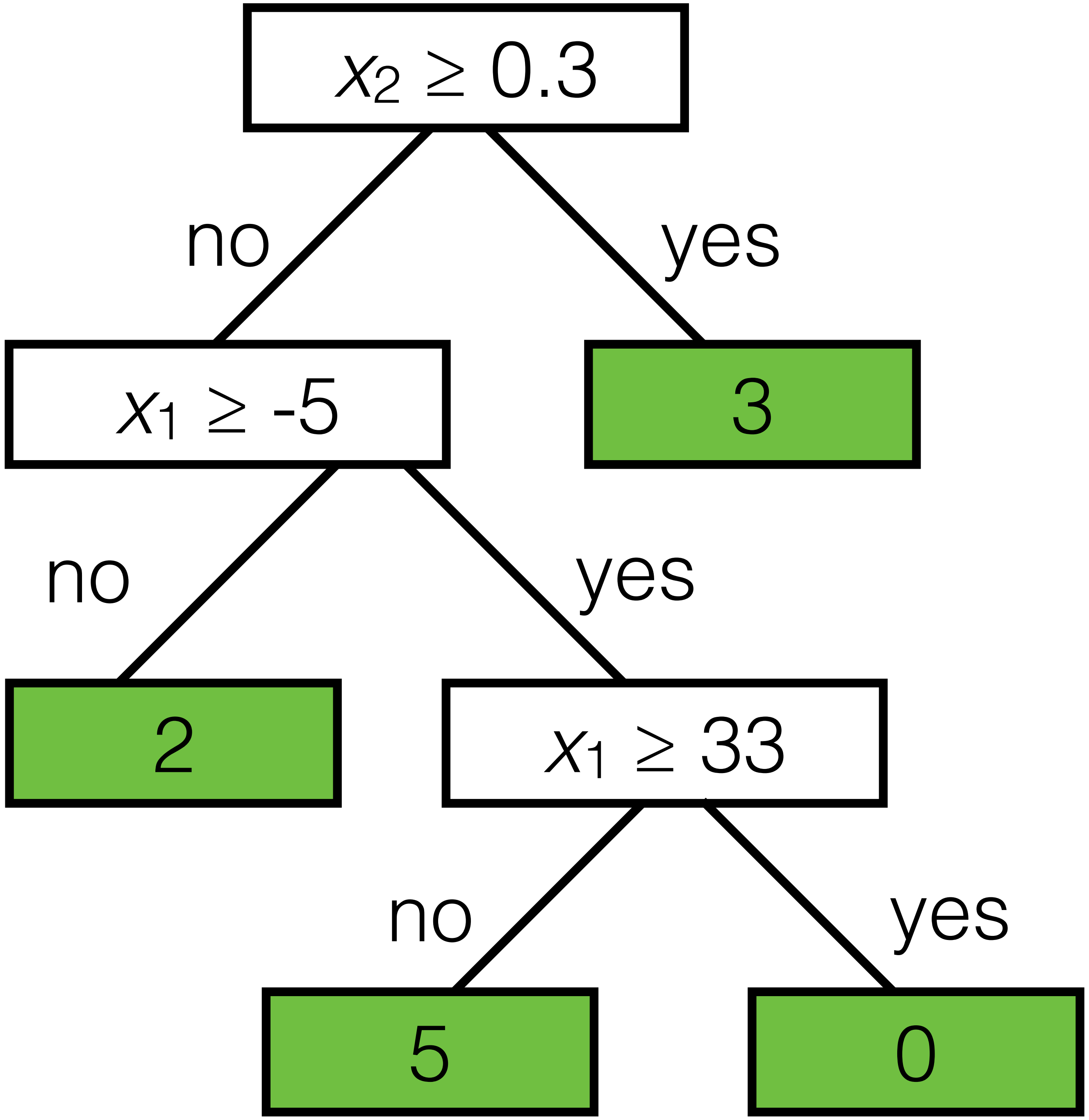
Decision tree



Recall: familiar pattern

1. Choose how to predict label (given features & parameters)

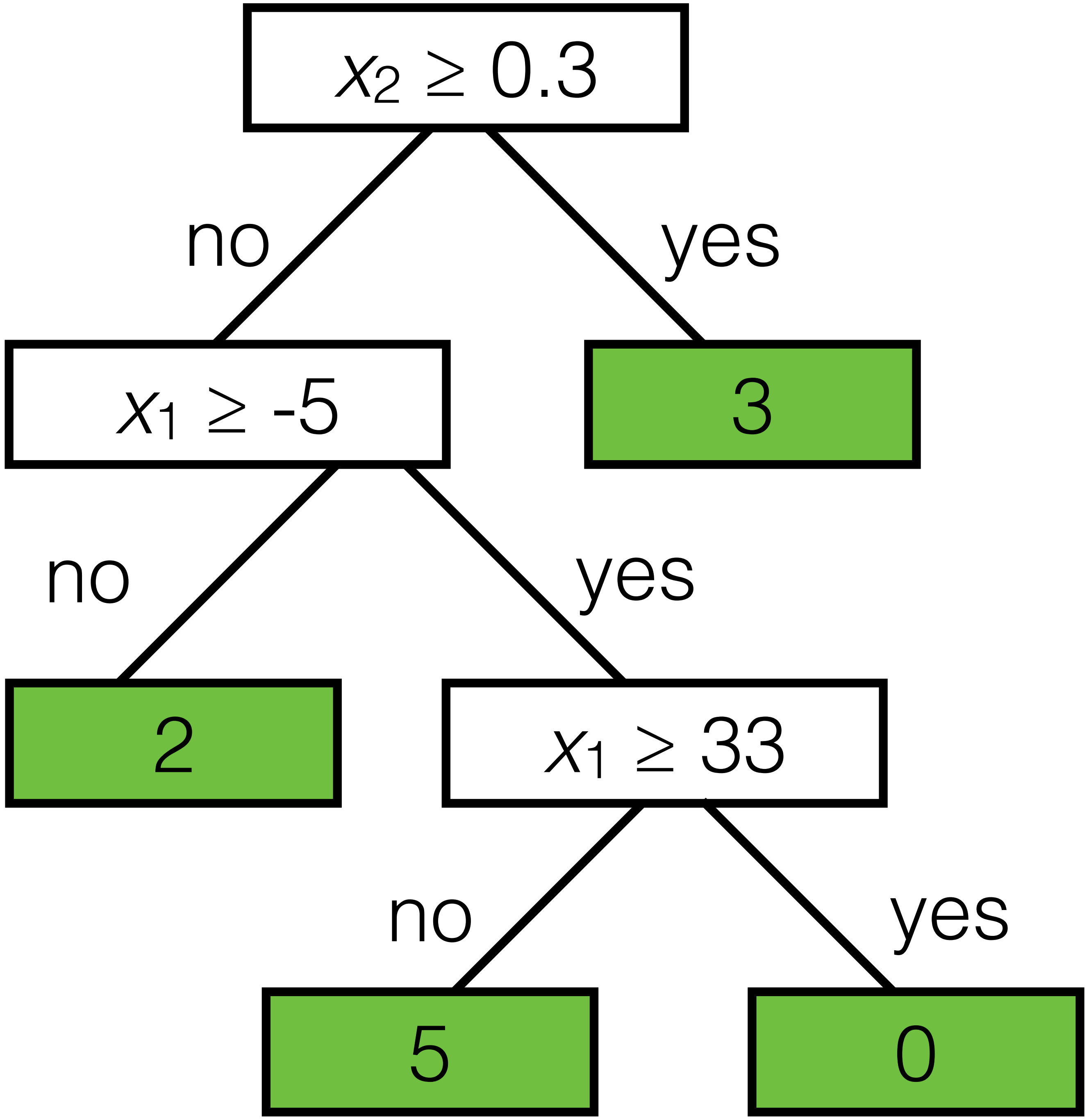
Decision tree



Recall: familiar pattern

1. Choose how to predict label (given features & parameters)
2. Choose a loss (between guess & actual label)

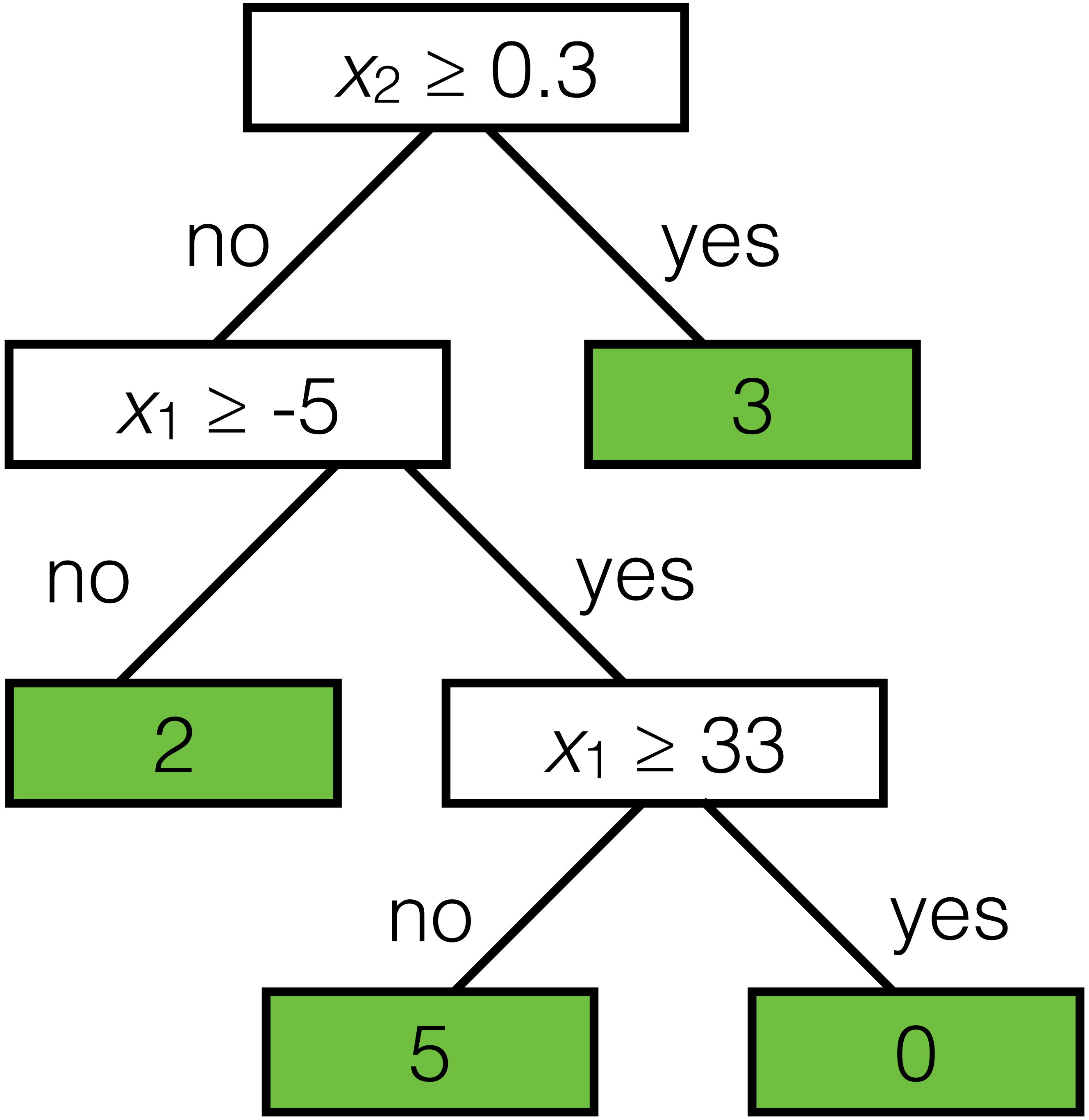
Decision tree



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

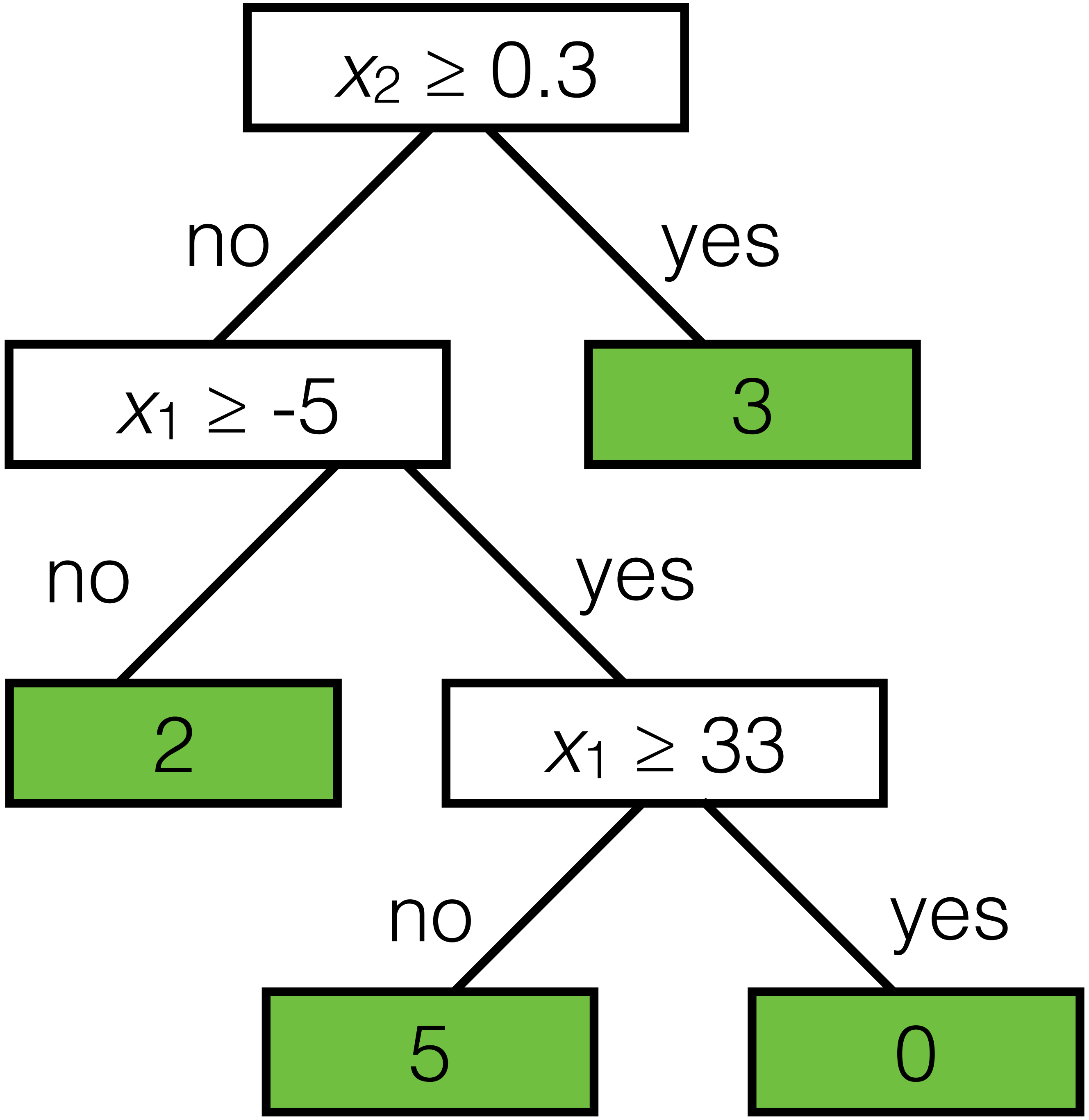


Recall: familiar pattern

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

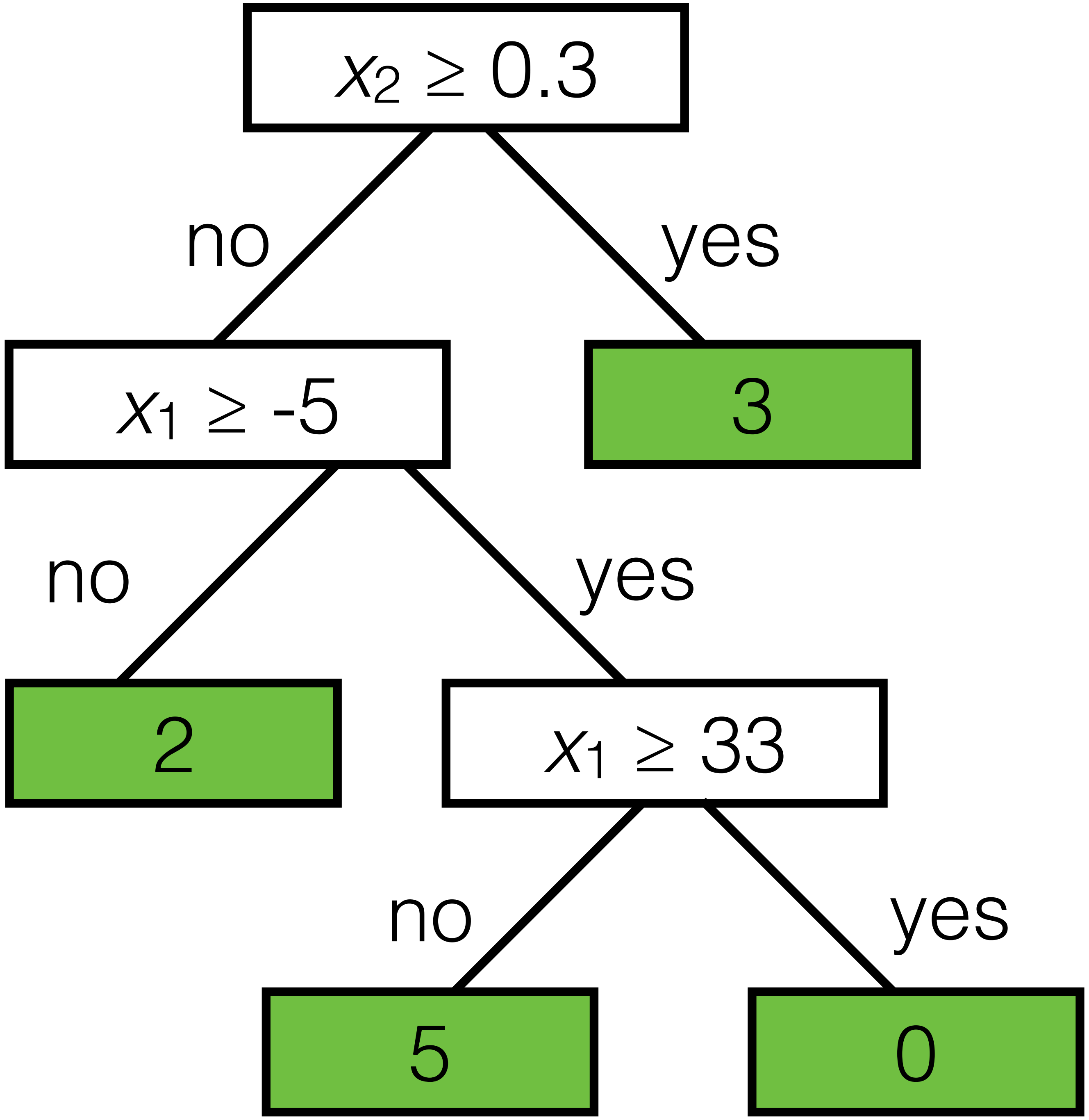


Recall: familiar pattern

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

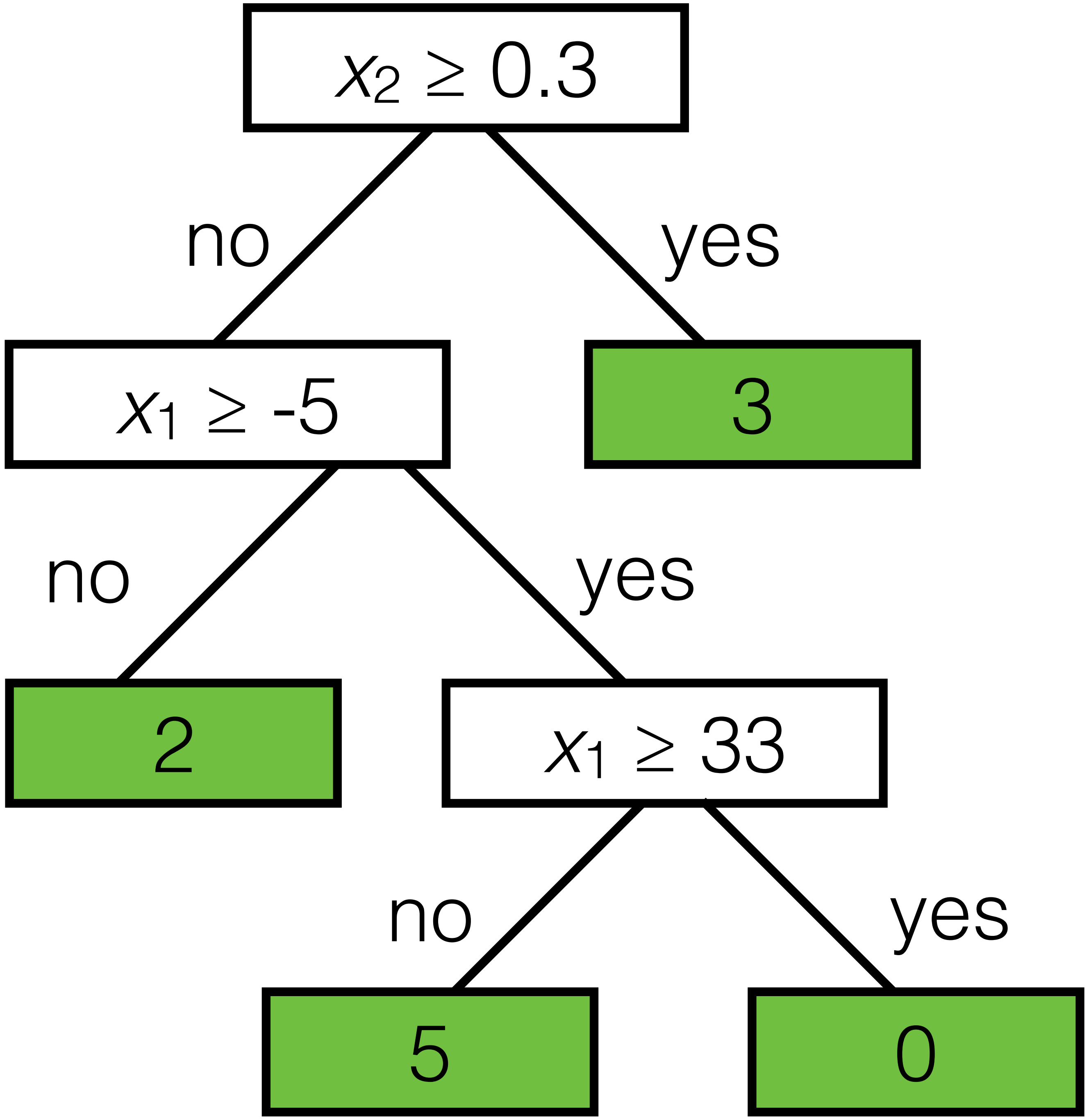


Recall: familiar pattern

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

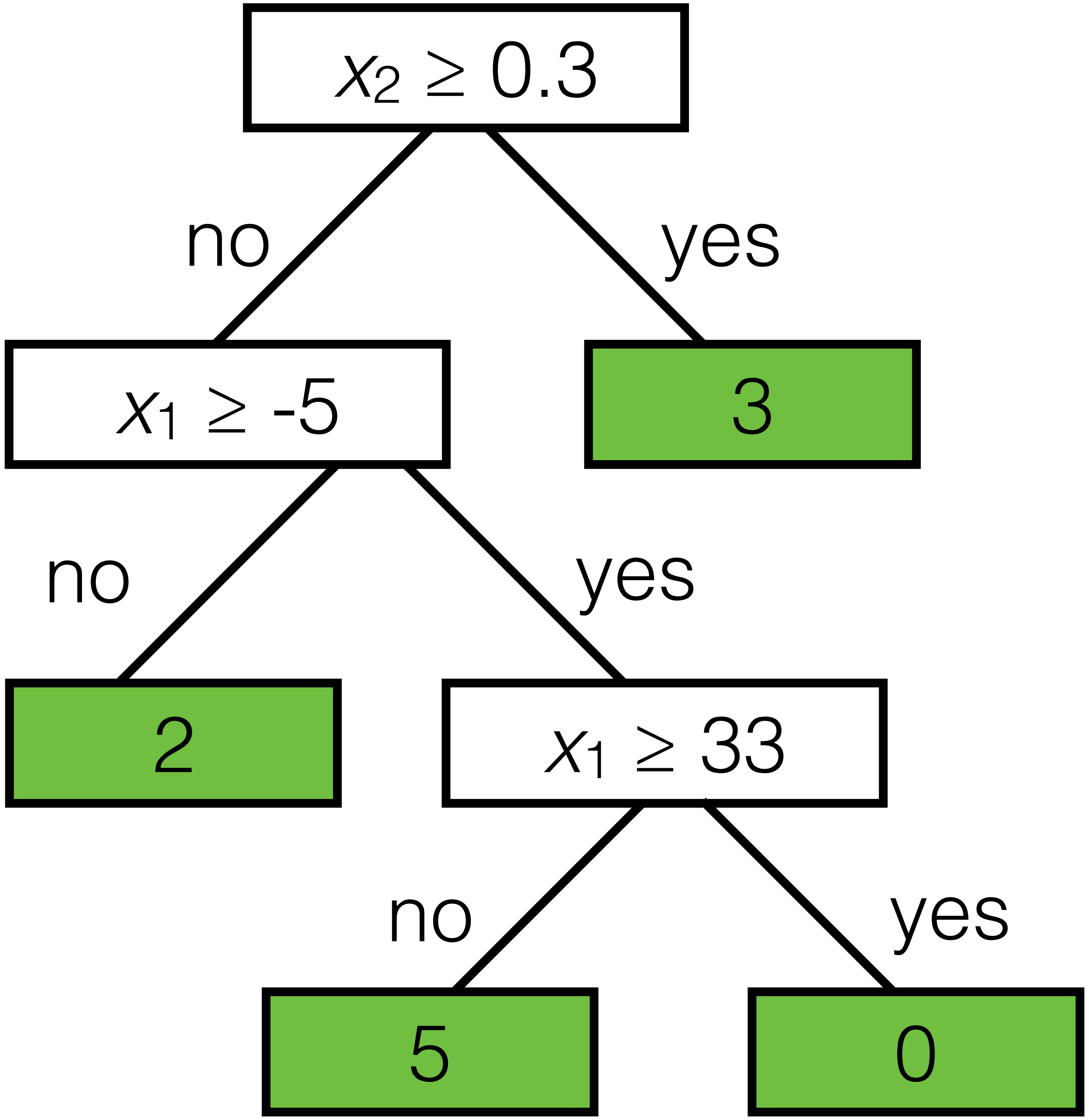


Recall: familiar pattern

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

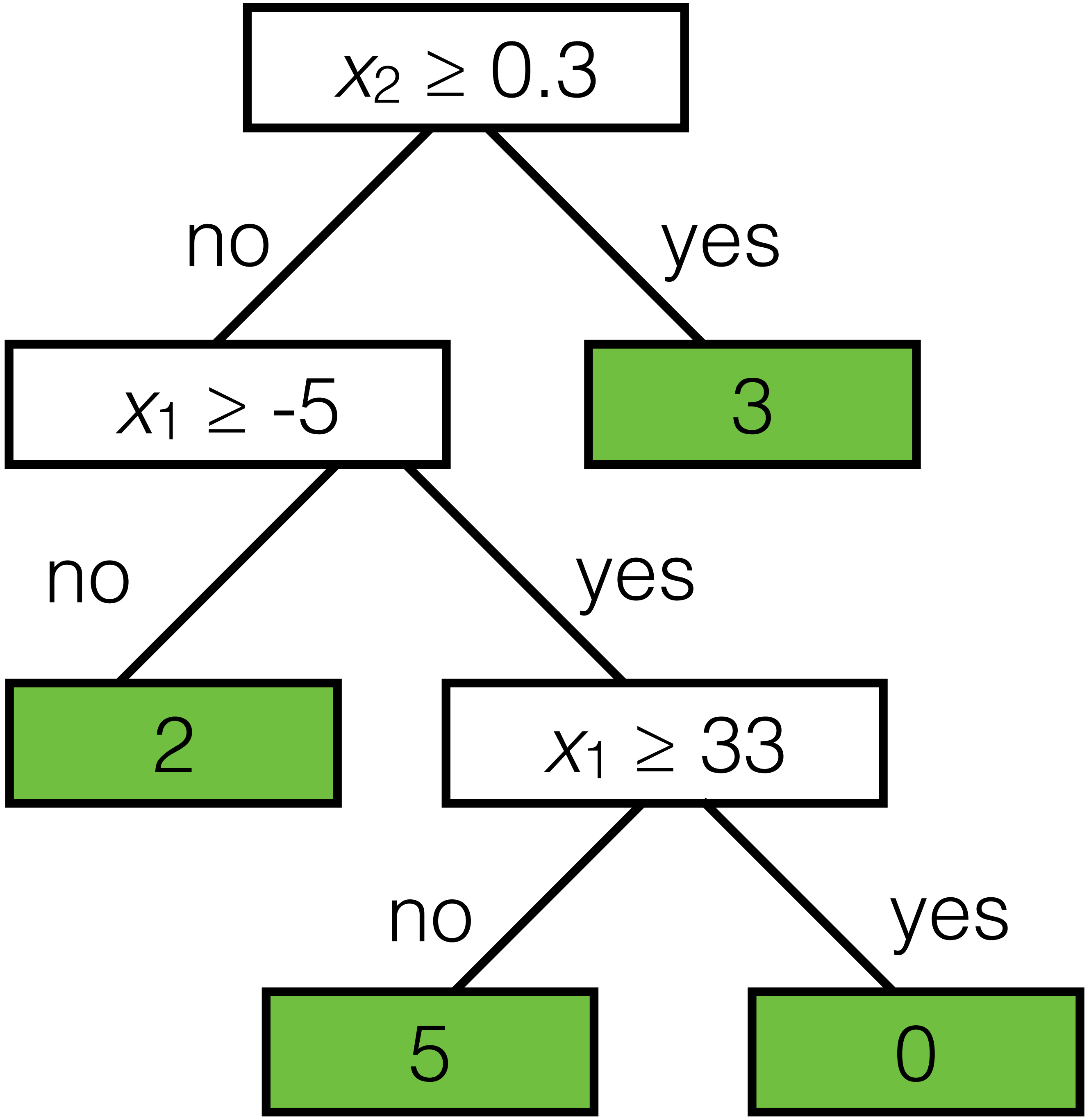


Recall: familiar pattern

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

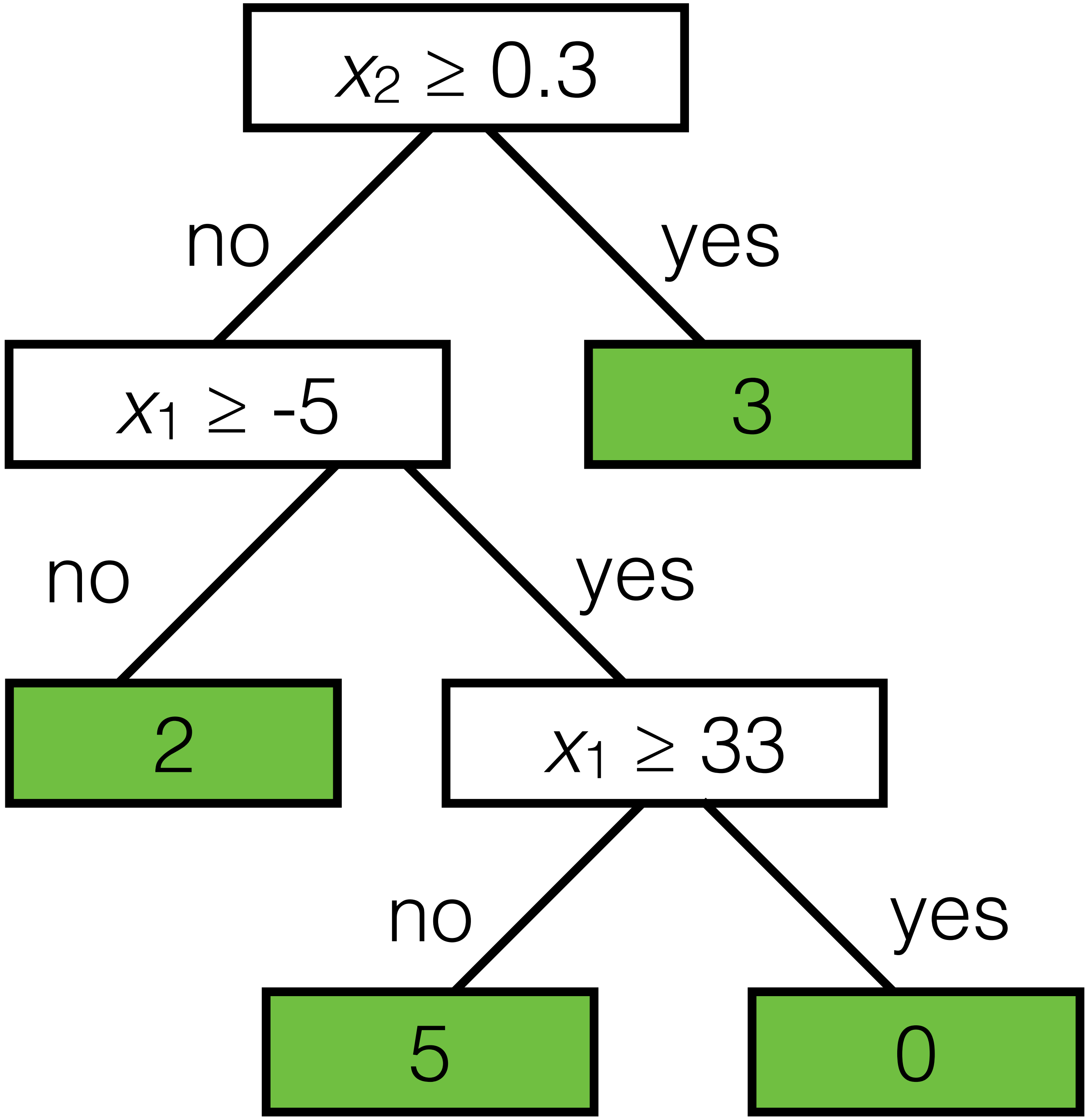


Recall: familiar pattern

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



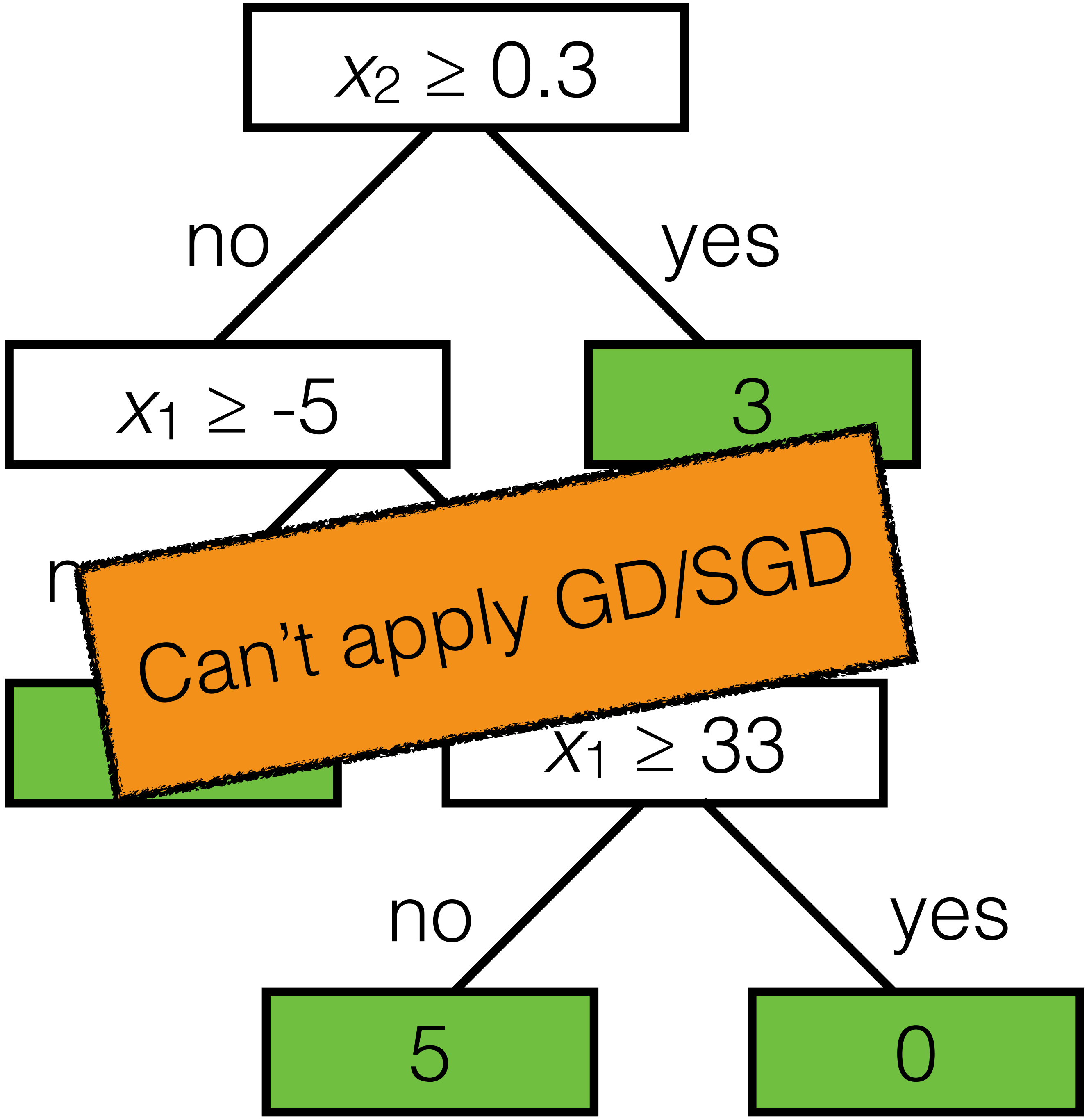
- Note: parameters here don't have a fixed dimension

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



Can't apply GD/SGD

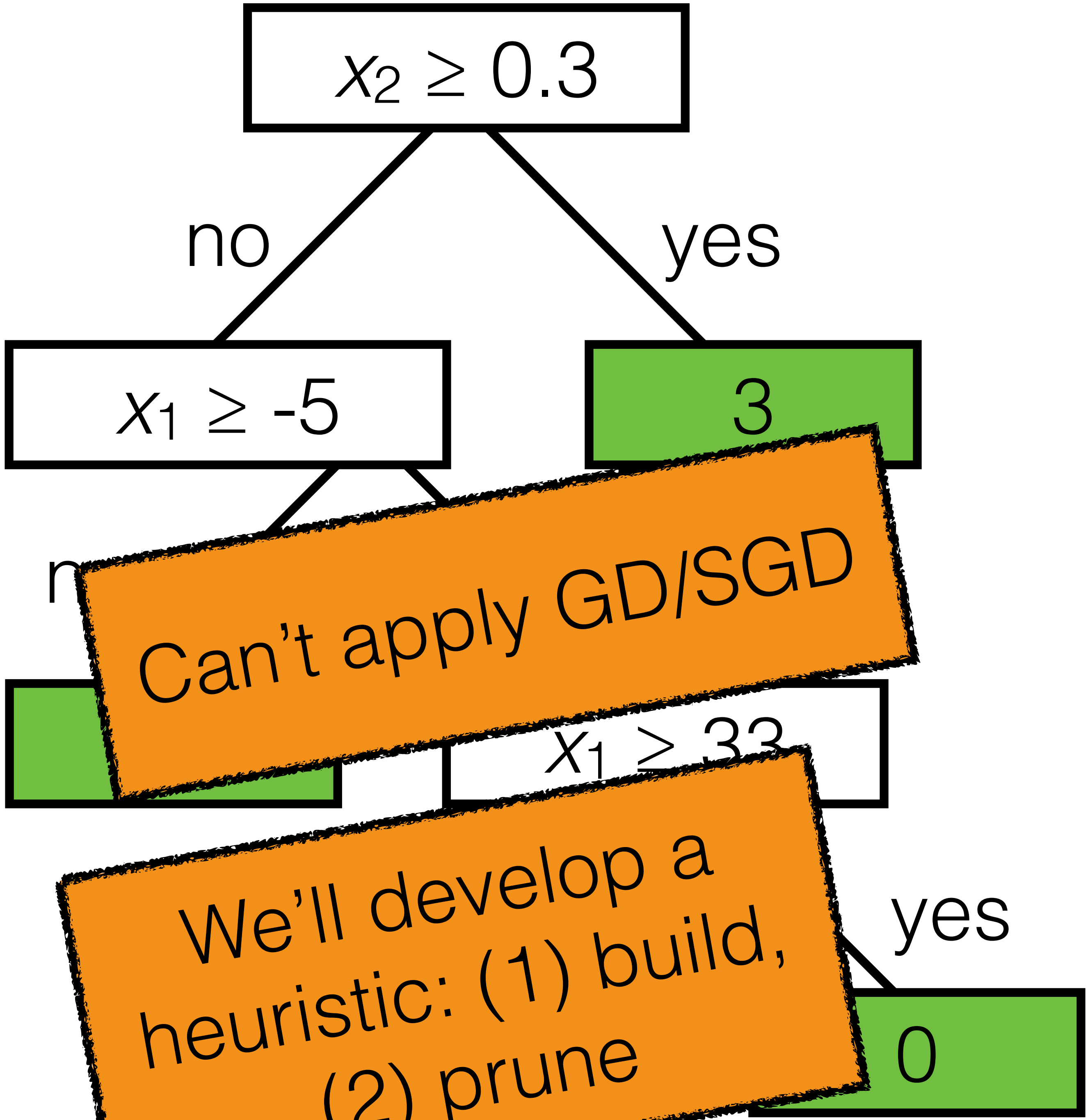
- Note: parameters here don't have a fixed dimension

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



Can't apply GD/SGD

We'll develop a heuristic: (1) build, (2) prune

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

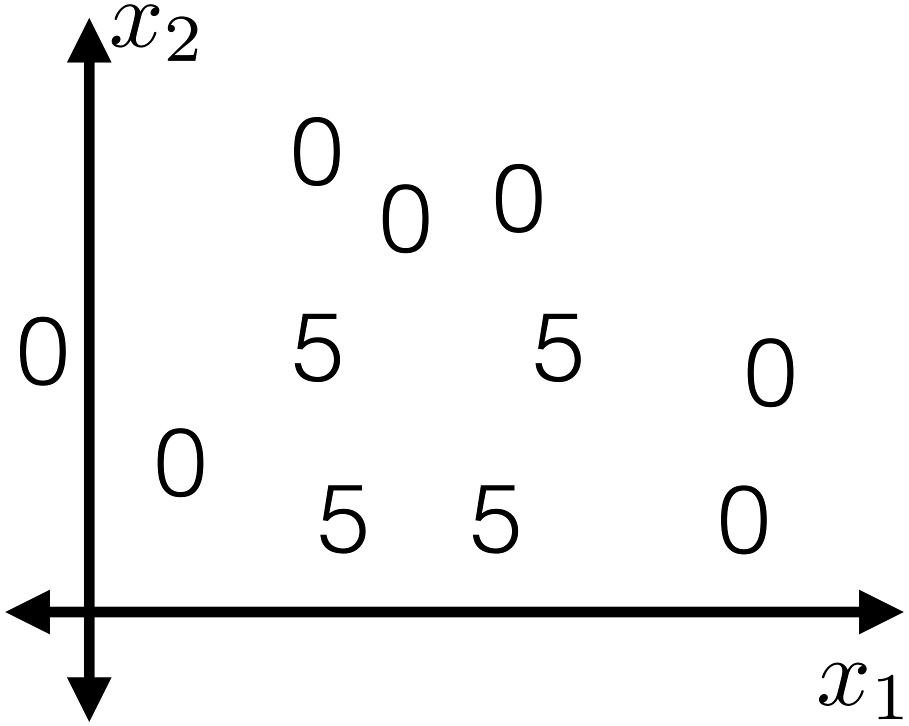
Recall: familiar pattern

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

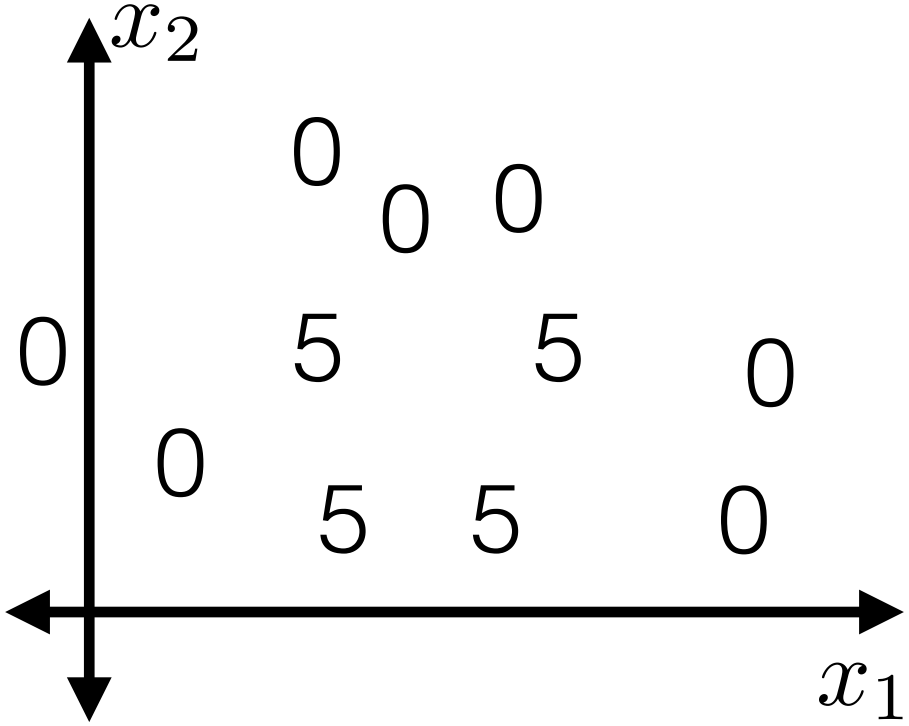
Building a decision tree

Building a decision tree



Building a decision tree

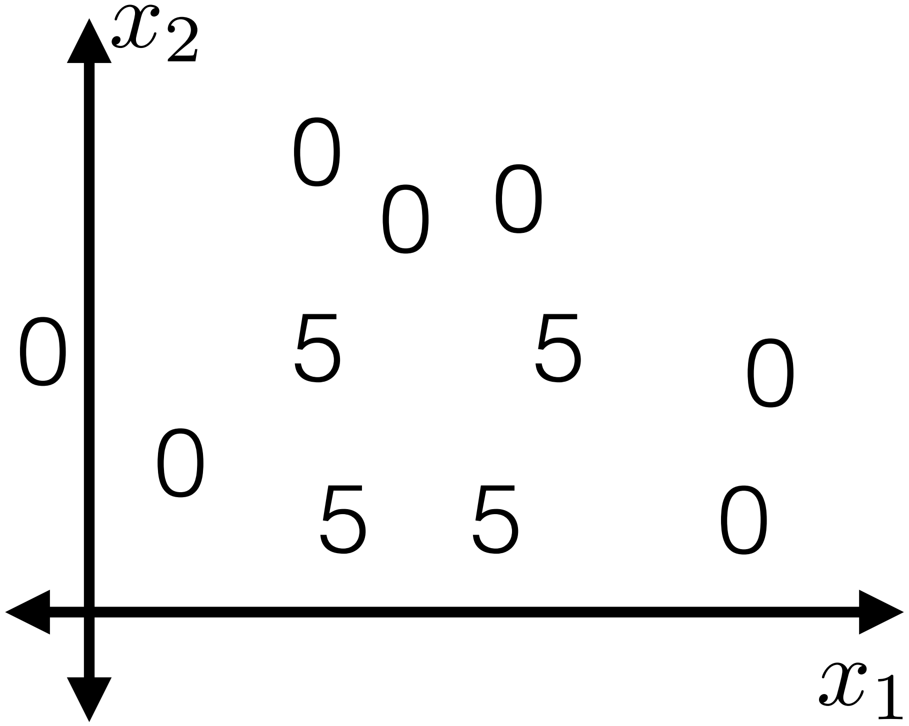
- Regression tree with squared error loss



Building a decision tree

- Regression tree with squared error loss

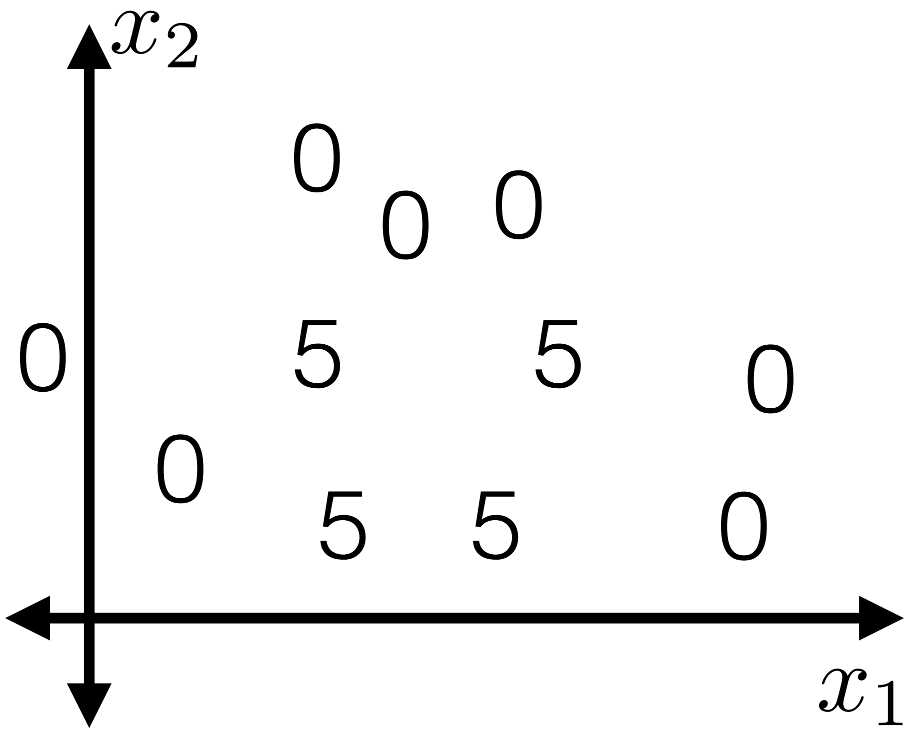
`BuildTree`



Building a decision tree

- Regression tree with squared error loss

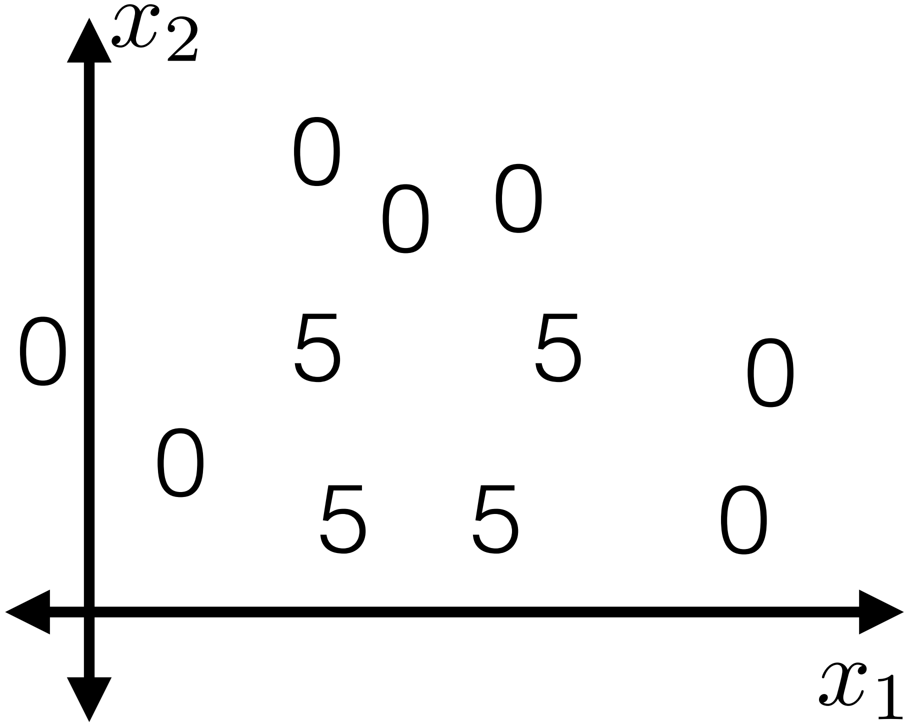
`BuildTree(`I`;`k`)`



Building a decision tree

- Regression tree with squared error loss

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

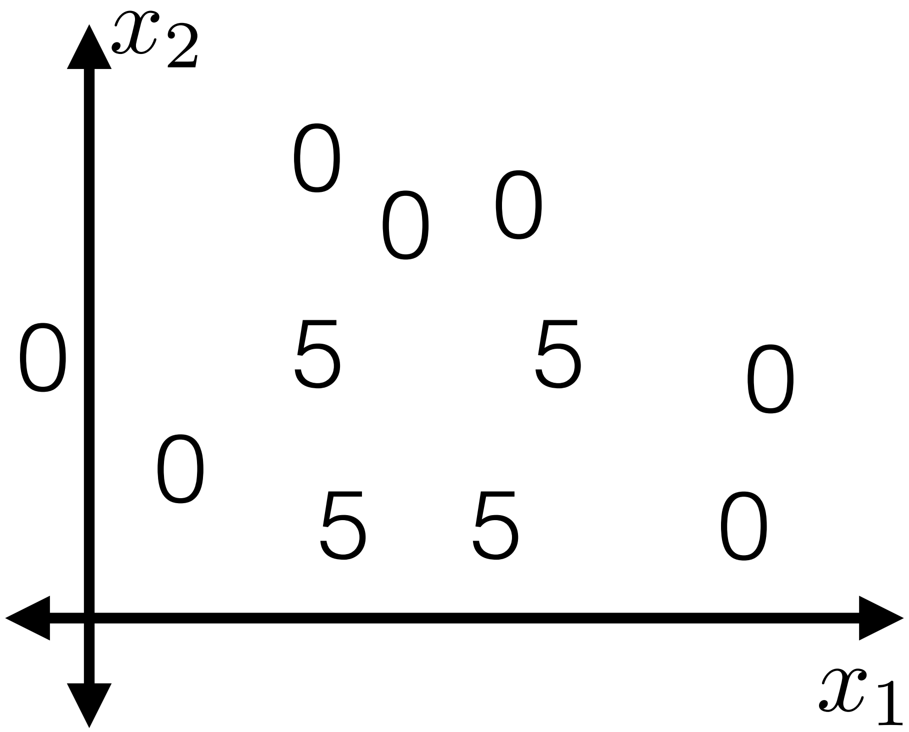


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

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

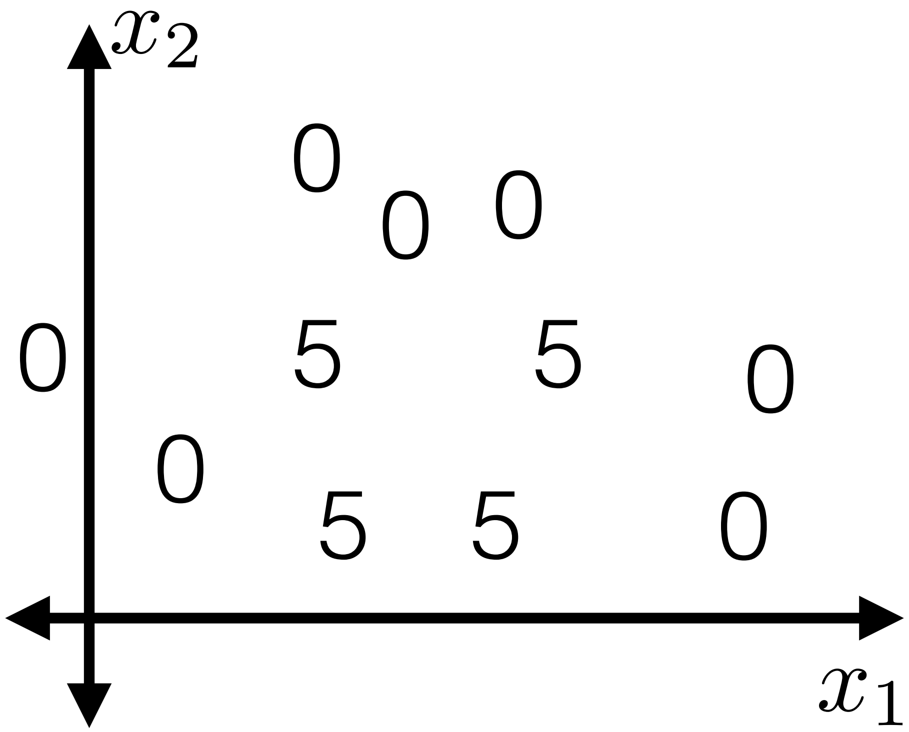


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

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

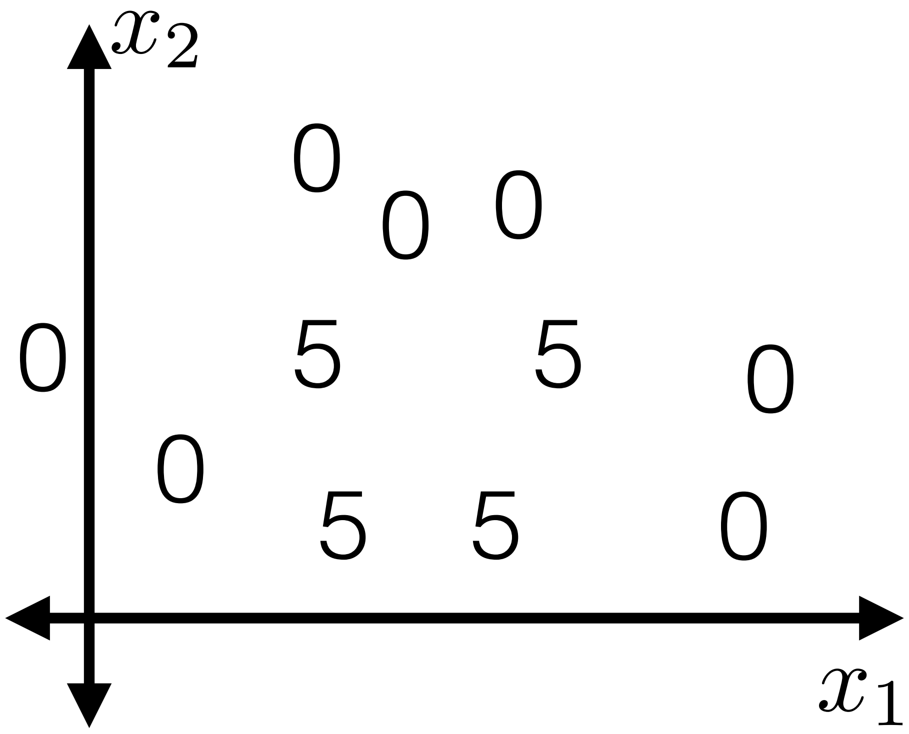


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

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

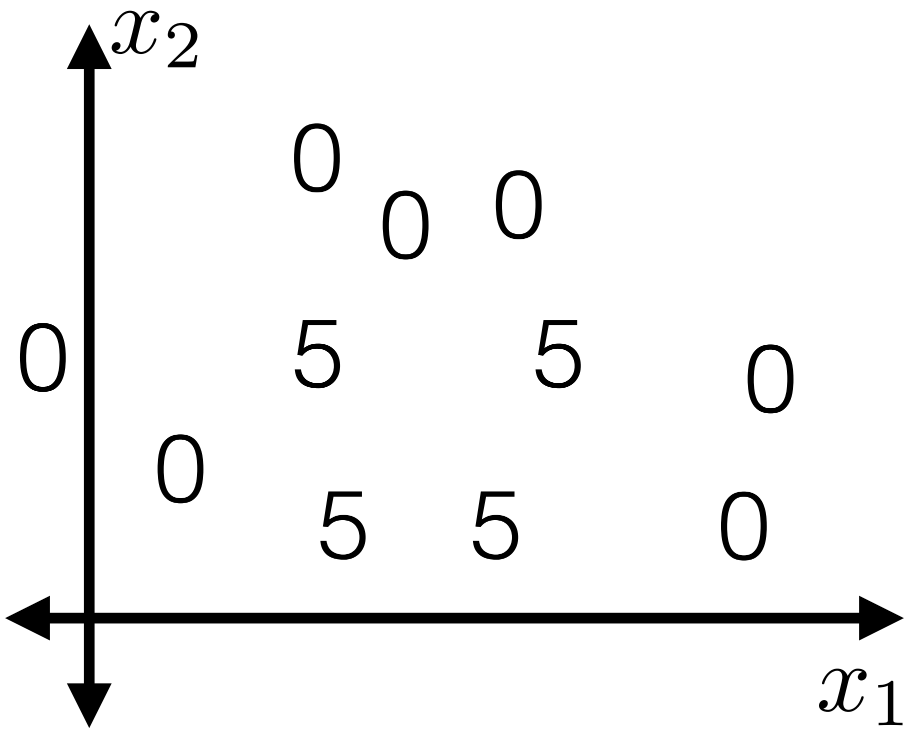


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| ≤ k
```

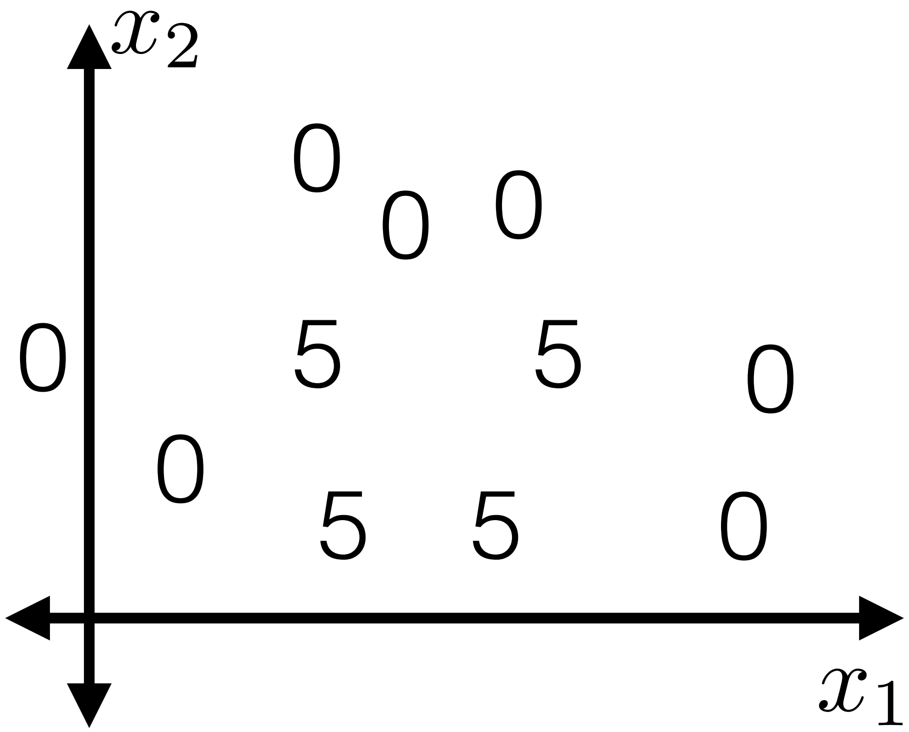


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| ≤ k
```

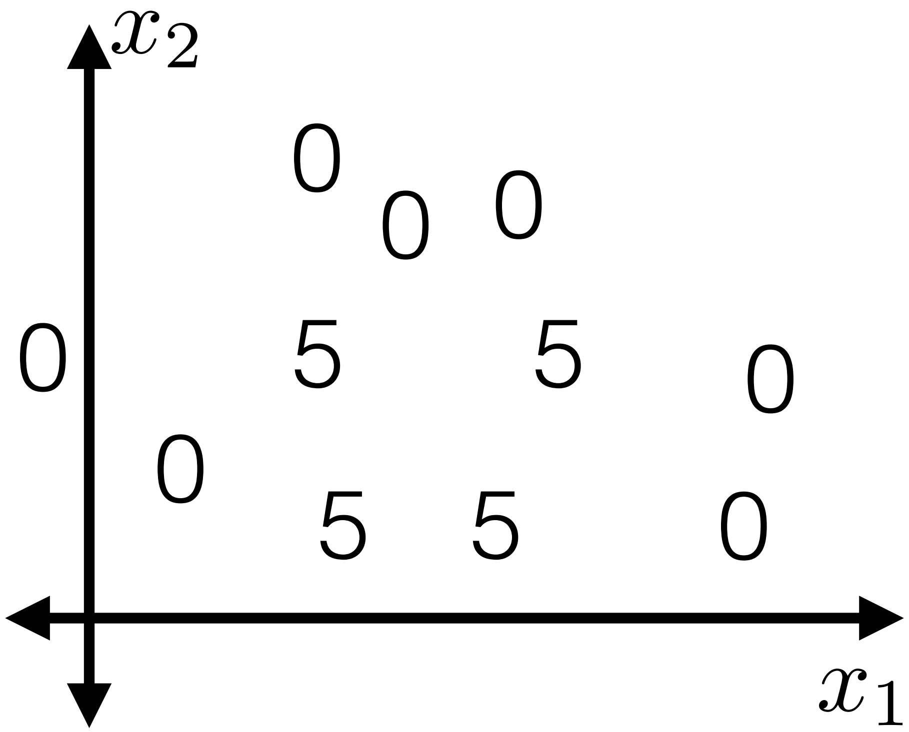


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| $\leq$ k
```

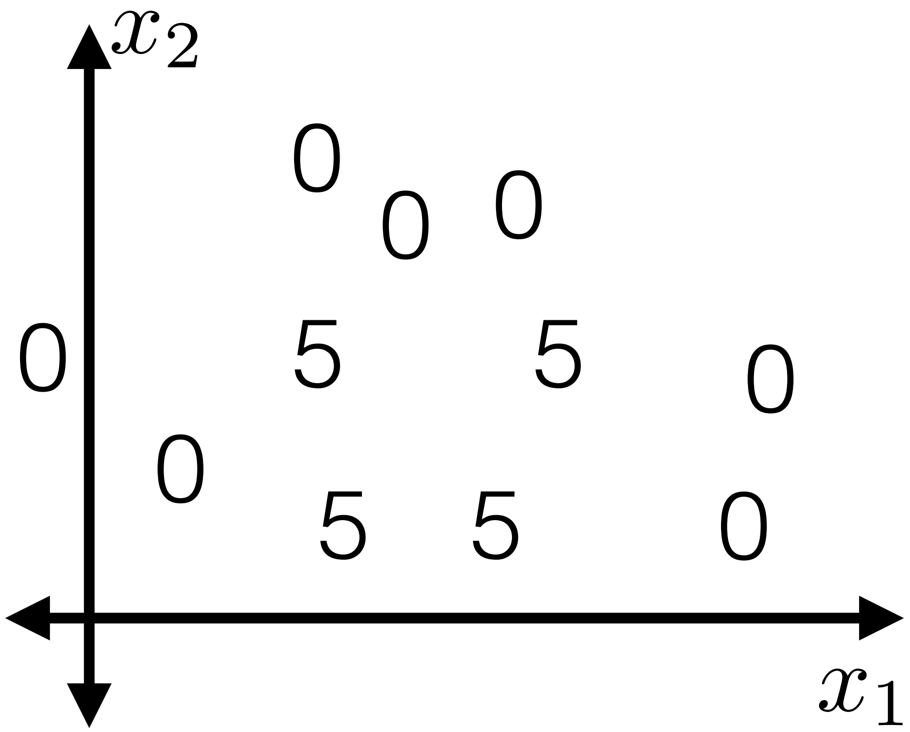


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| $\leq$ k
    Set $\hat{y}$ =
```

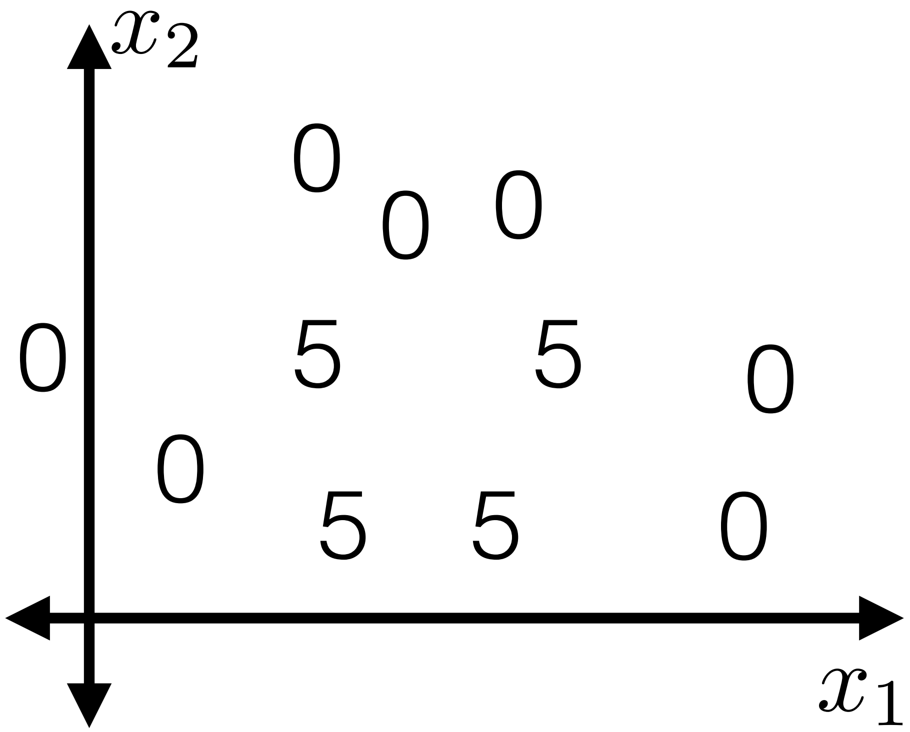


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if $|I| \leq$ k
    Set $\hat{y}$ =
```



BuildTree({1,…,n};2)

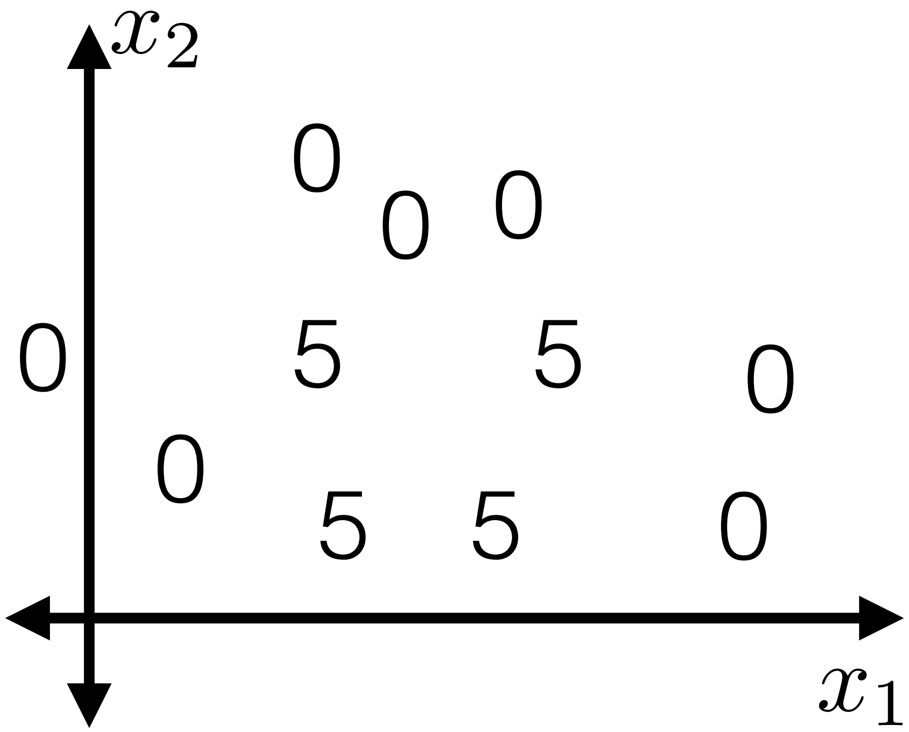
Want to minimize loss/error:

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

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| ≤ k
    Set $\hat{y}$ = $\text{average}_{i\in I} y^{(i)}$
```



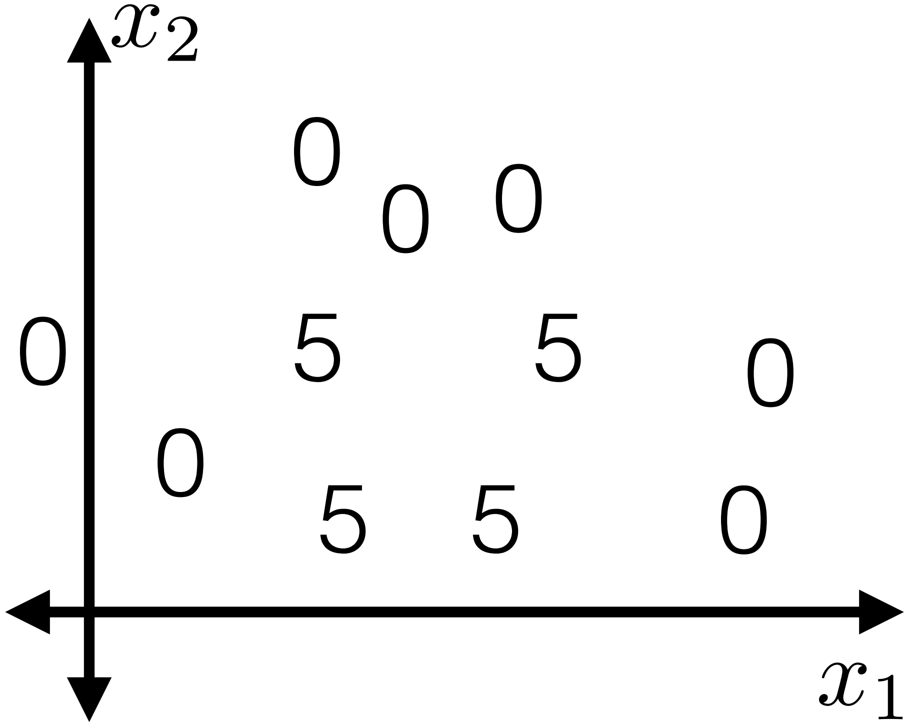
BuildTree({1,…,n};2)

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

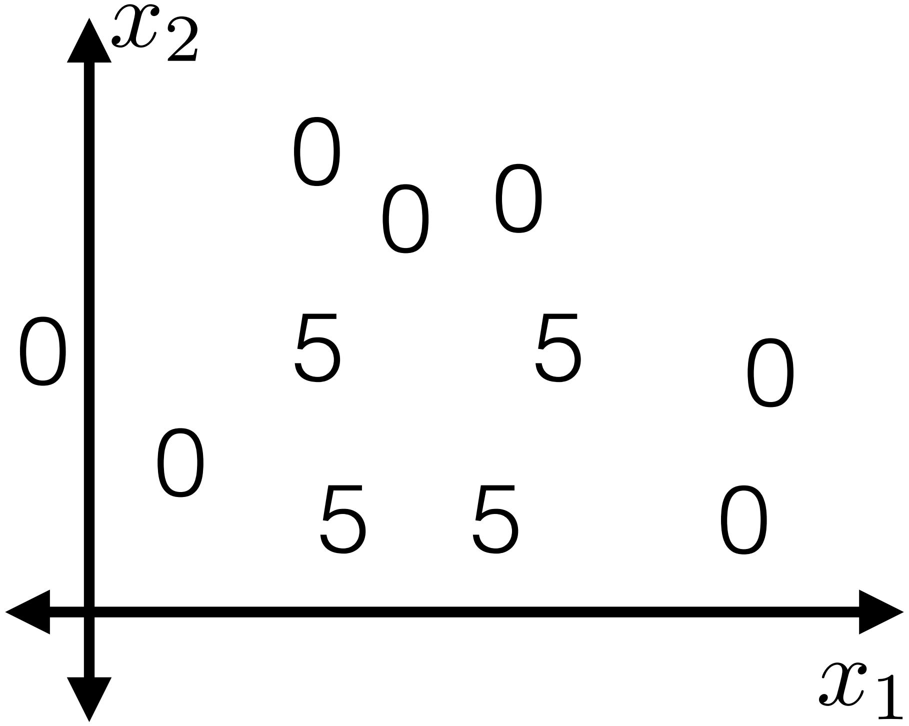


BuildTree({1,…,n};2)

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
```

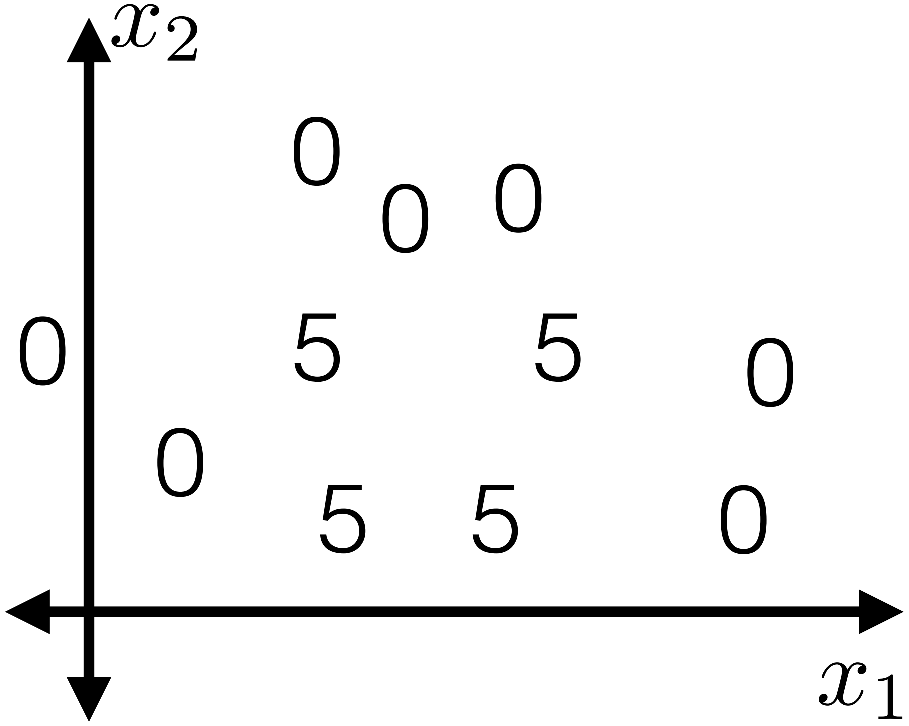


BuildTree({1,…,n};2)

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 $s$
```

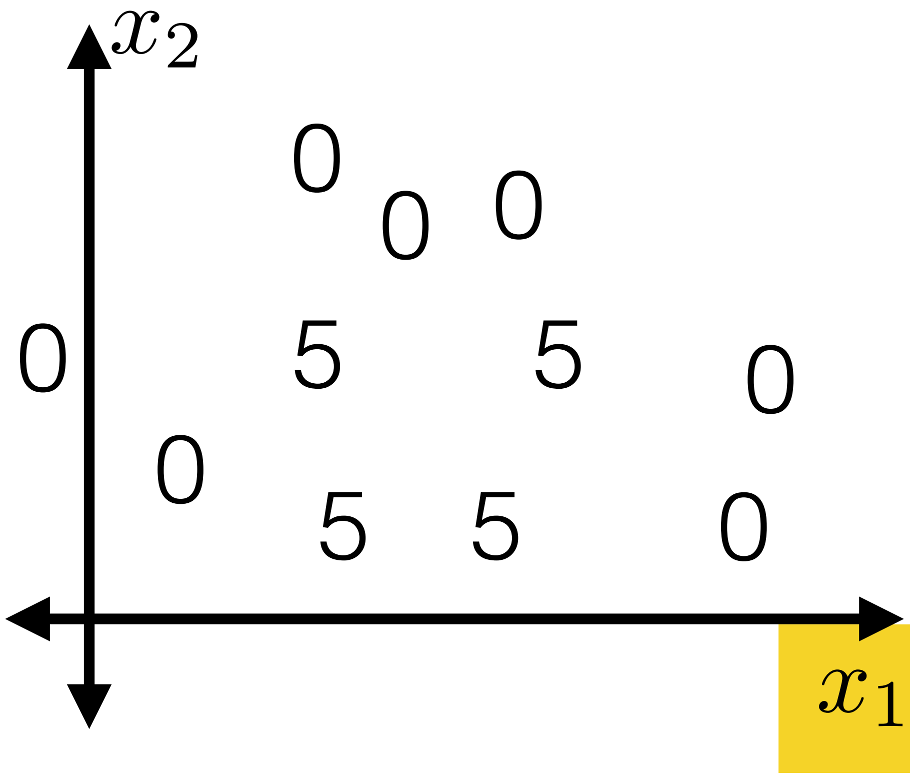


BuildTree({1,…,n};2)

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 $s$
```

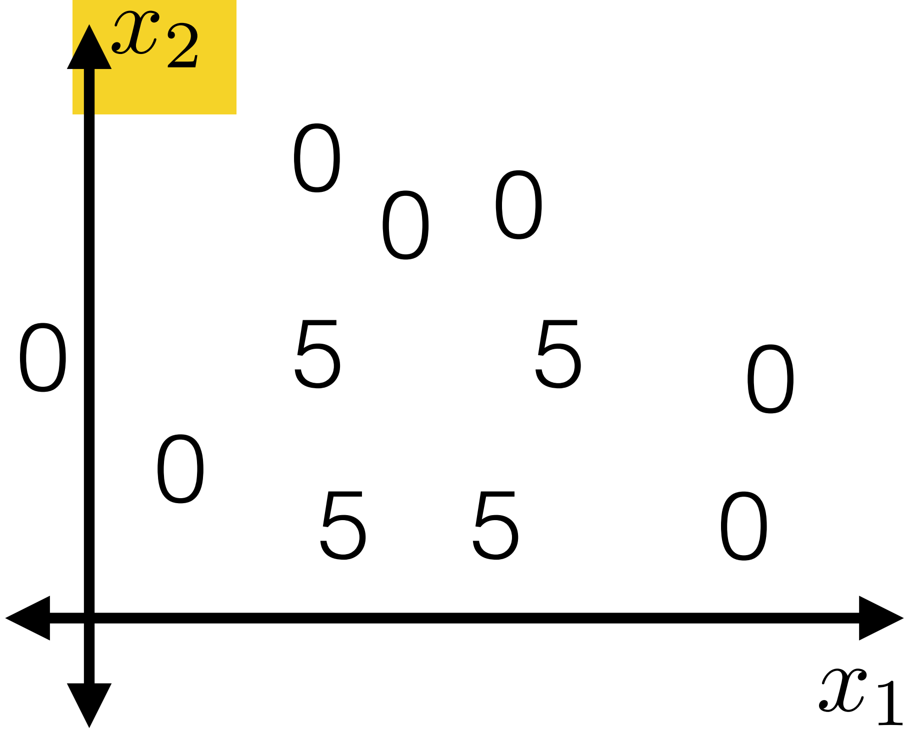


BuildTree({1,…,n};2)

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 $s$
```

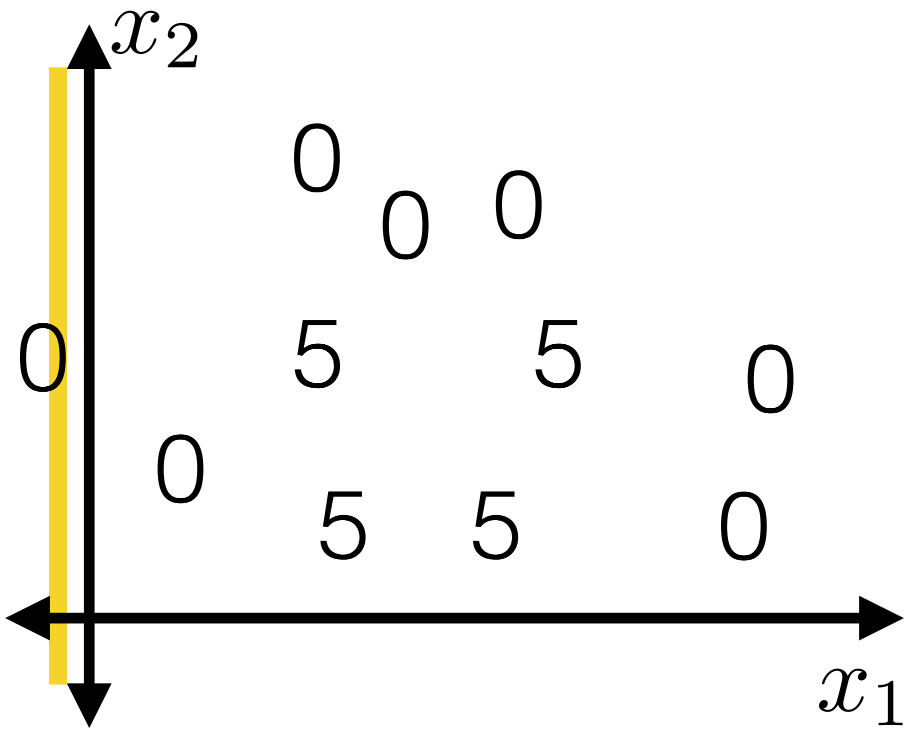


BuildTree({1,…,n};2)

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 $s$
```

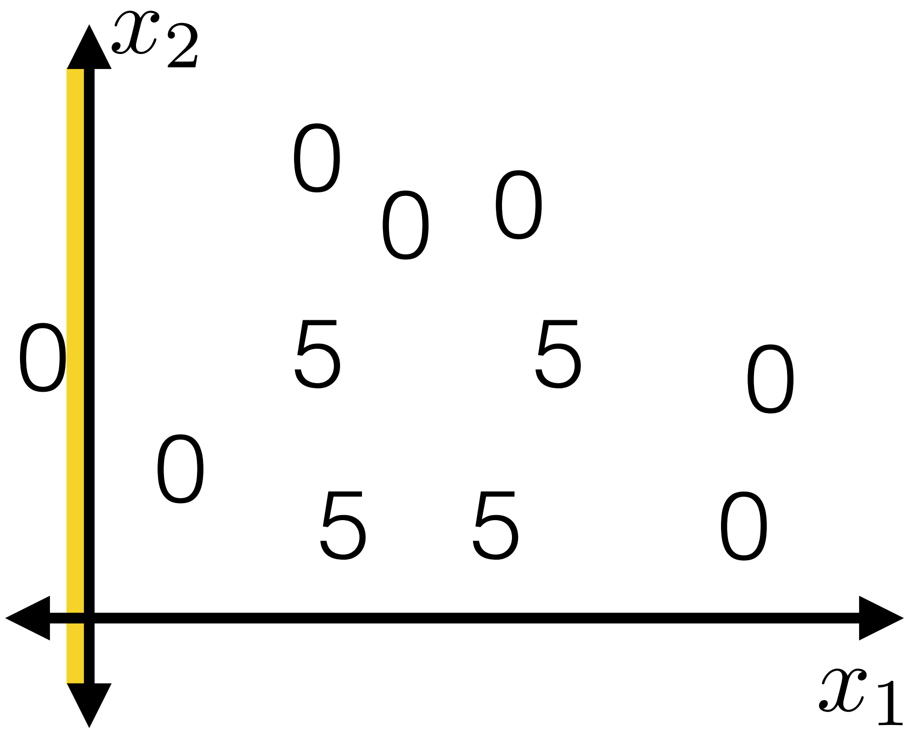


BuildTree({1,…,n};2)

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 $s$
```

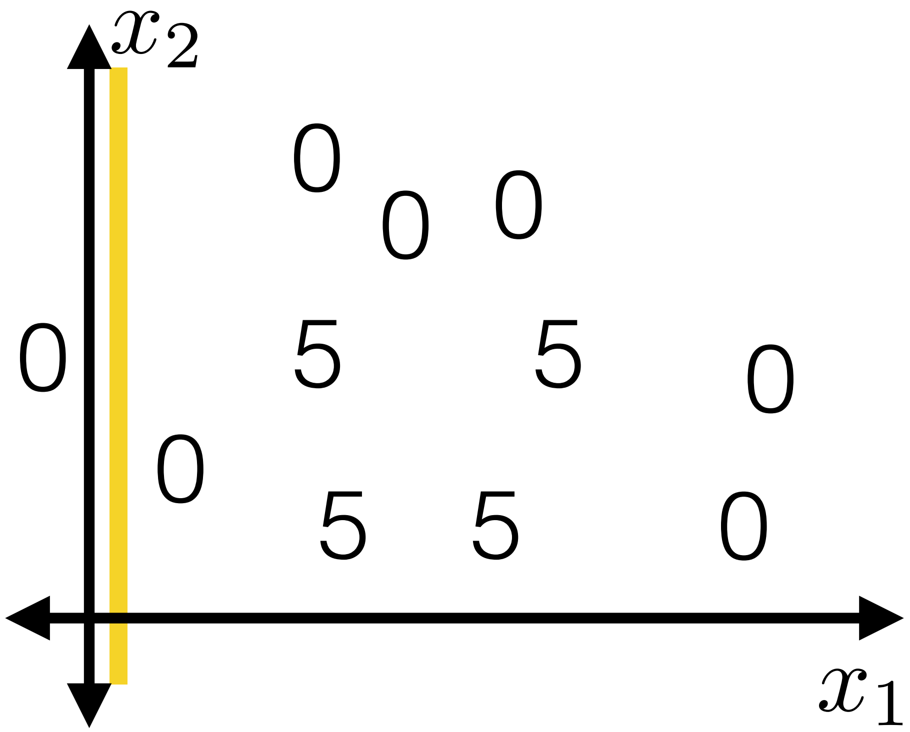


`BuildTree({1,…,n};2)`

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 $s$
```

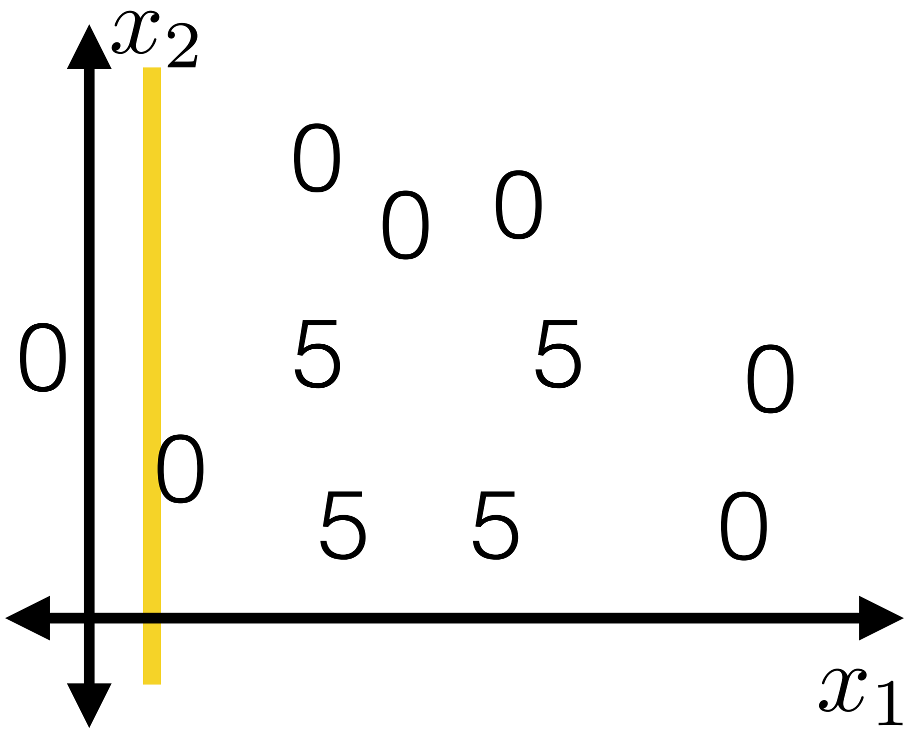


BuildTree({1,…,n};2)

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 $s$
```

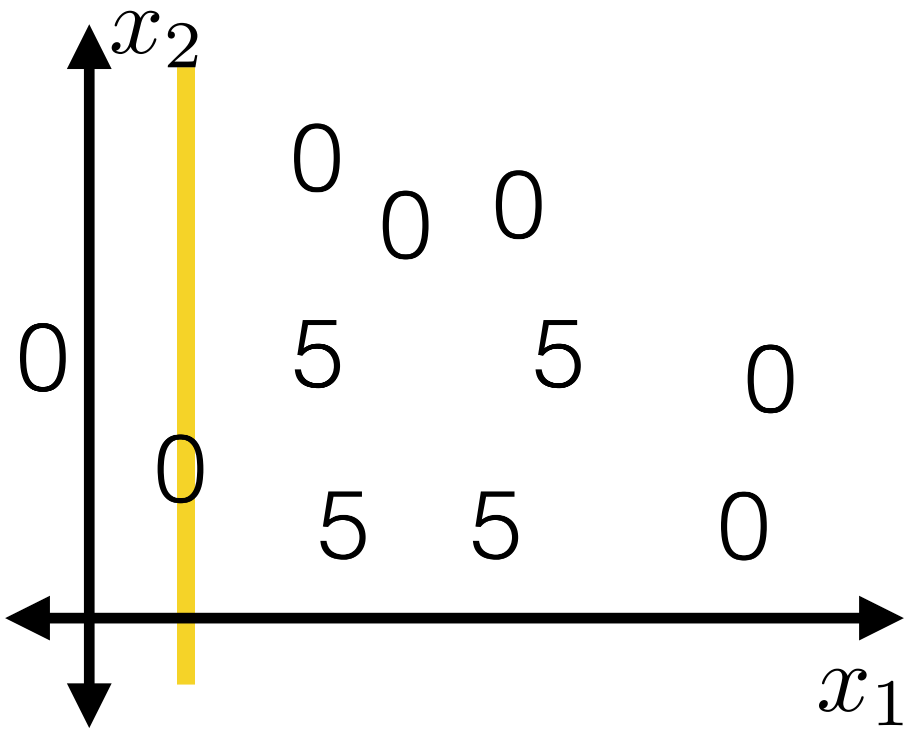


BuildTree({1,…,n};2)

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 $s$
```

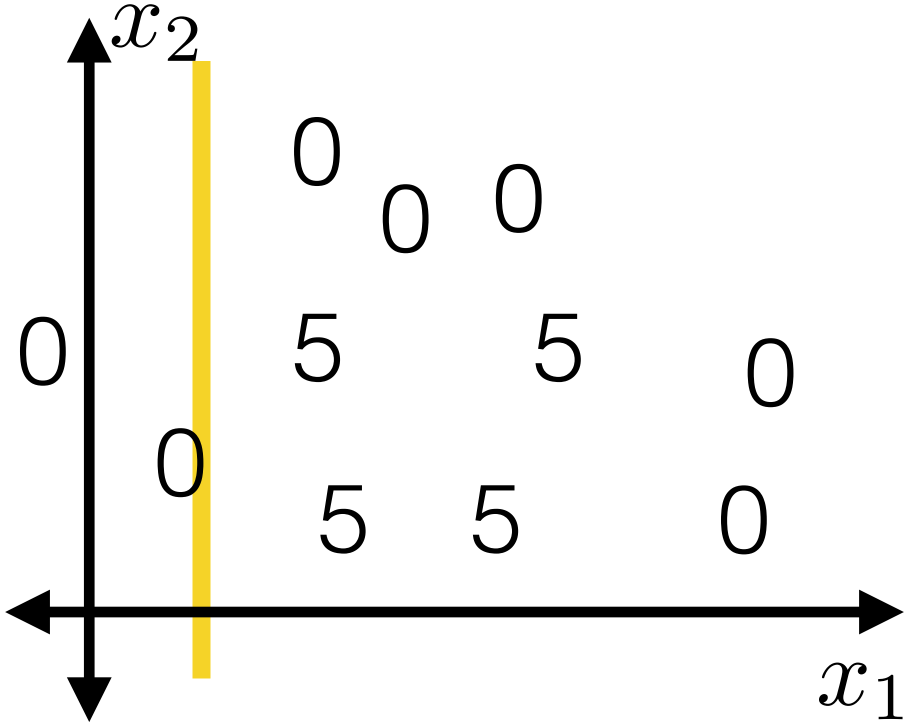


BuildTree({1,…,n};2)

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 $s$
```

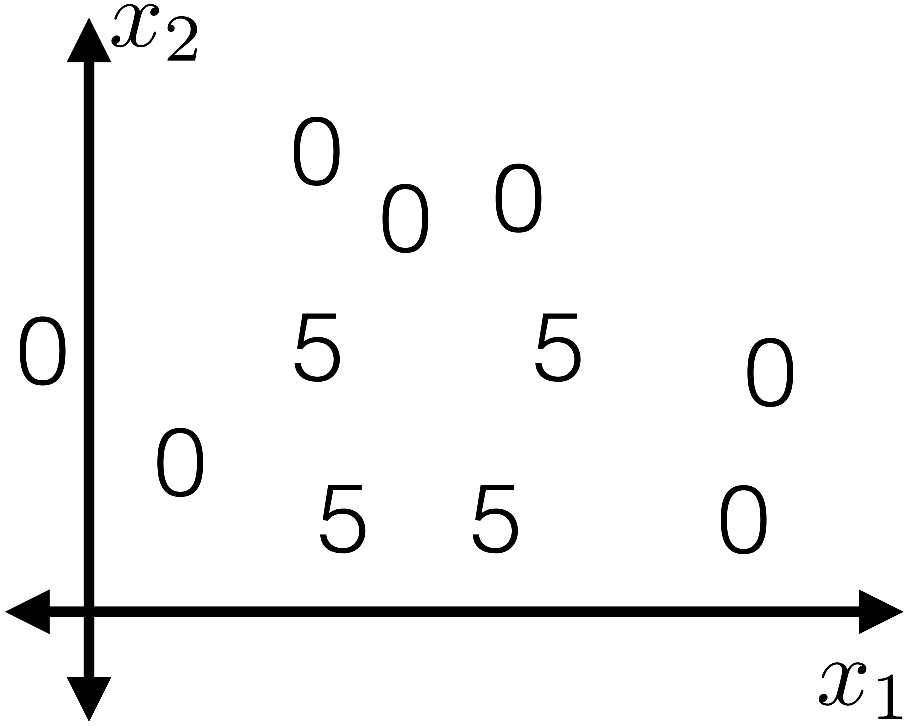


BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| ≤ k
    Set $\hat{y}$ = $\text{average}_{i \in I} y^{(i)}$
    return Leaf(label = $\hat{y}$)
  else
    for each split dim $j$ & value $s$
```

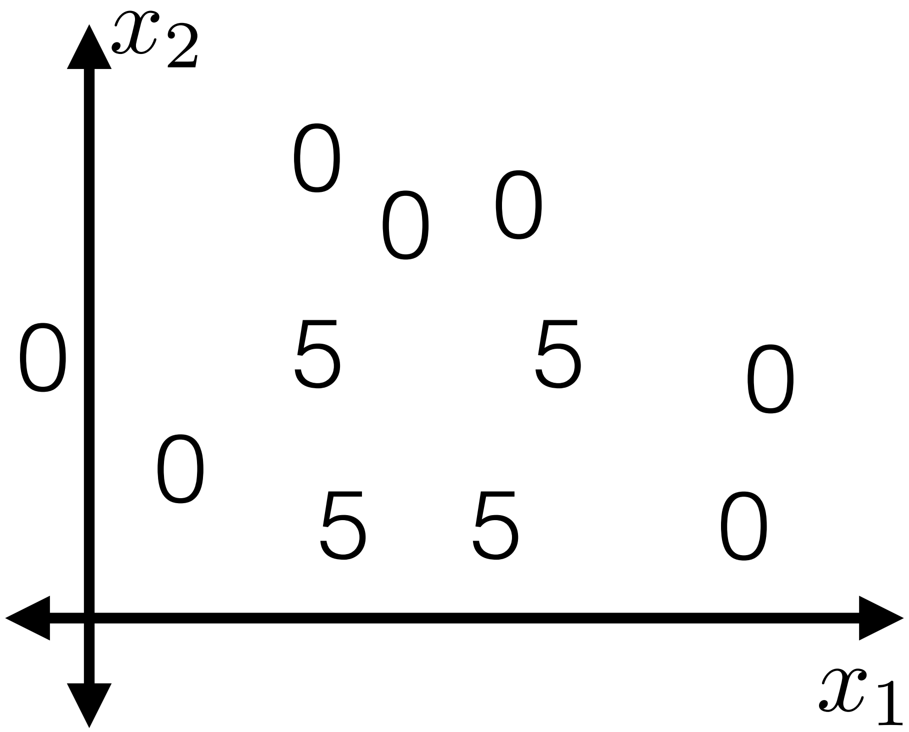


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I; k$)
  if |$I$| ≤ 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 $I^{-}_{j,s} = \{i \in I | x_j^{(i)} < s\}$
```

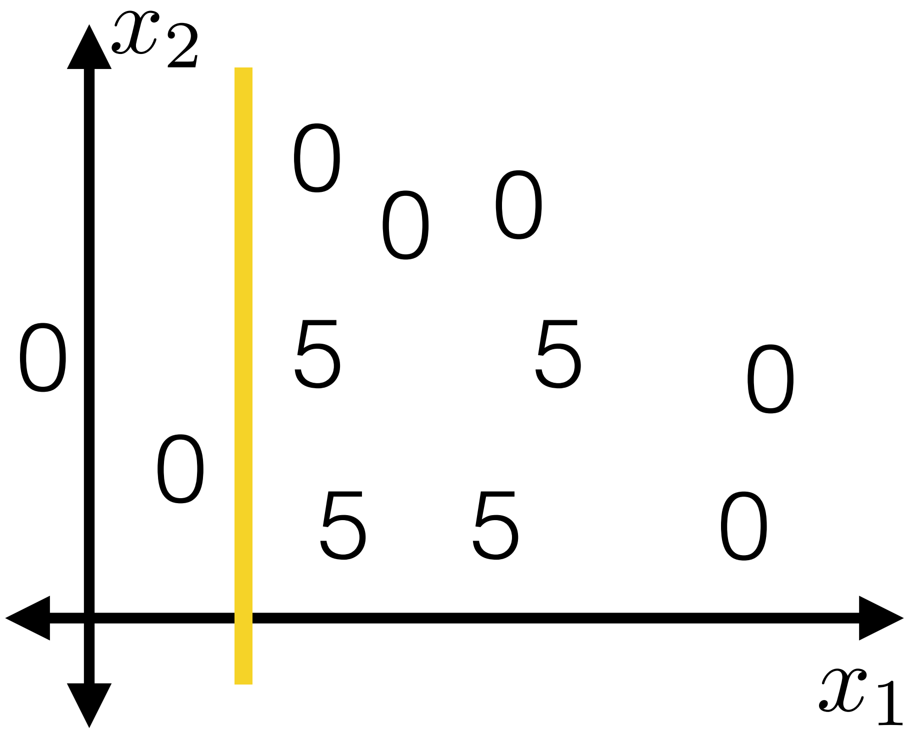


BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

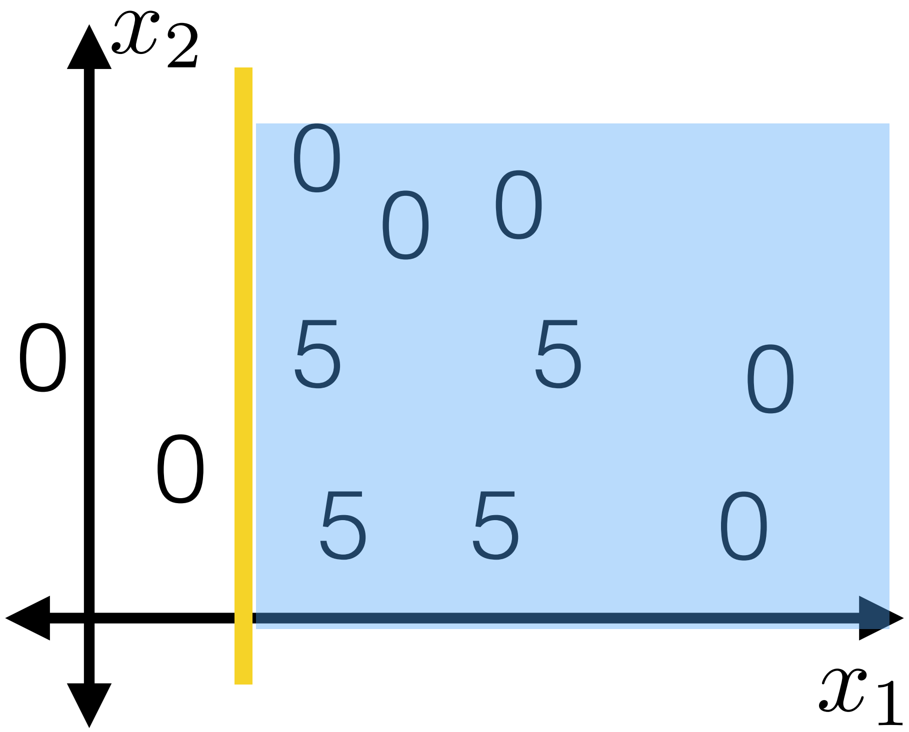
```
BuildTree($I;k$)
  if |$I$| $\leq$ k
    Set $\hat{y}$ = $\text{average}_{i\in I} y^{(i)}$
    return Leaf(label = $\hat{y}$)
  else
    for each split dim $j$ & value $s$
      Set $I^+_{j,s} = \{i \in I | x_j^{(i)} \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x_j^{(i)} < s\}$
```



BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

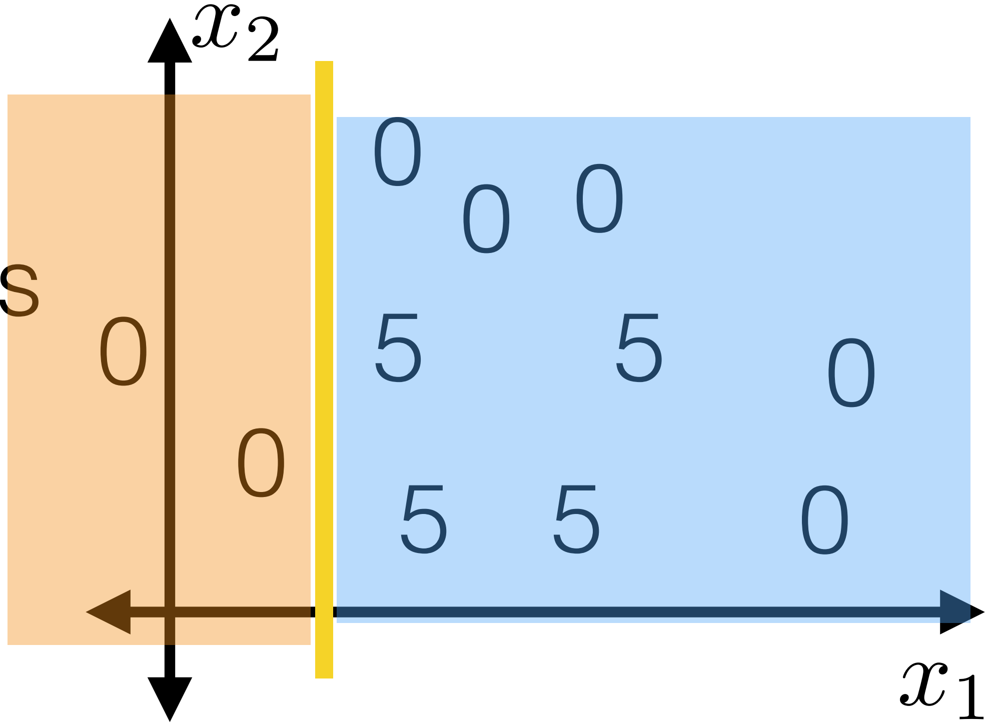


BuildTree({1,…,n};2)

```
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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
```

Building a decision tree

- Regression tree with squared error loss



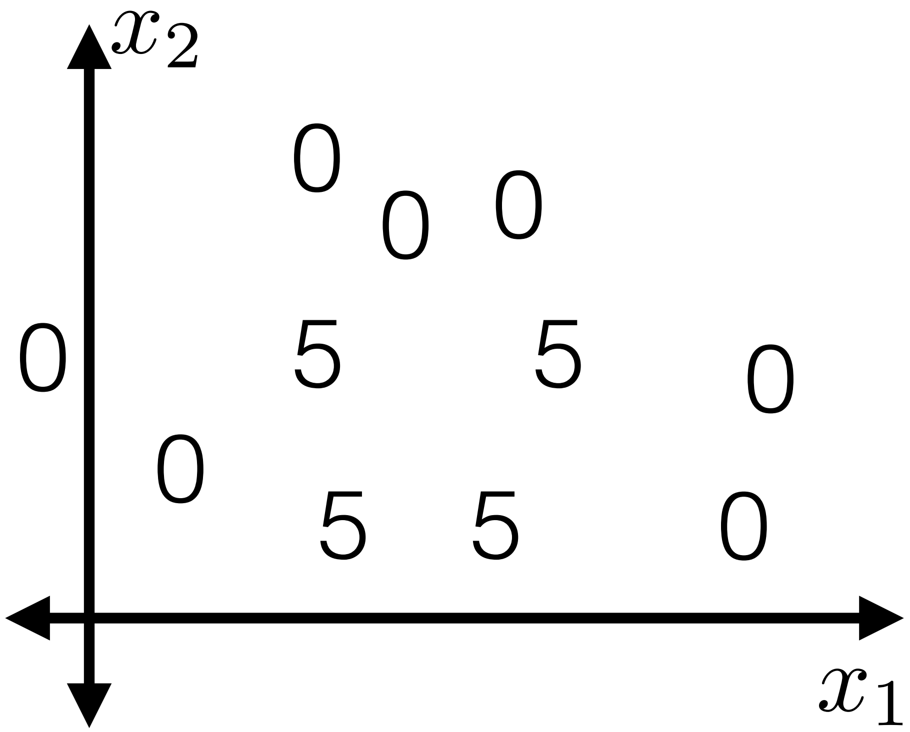
```
BuildTree($I$;$k$)
  if |$I$| ≤ k
    Set $\hat{y}$ = $\text{average}_{i \in I} y^{(i)}$
    return Leaf(label = $\hat{y}$)
  else
    for each split dim $j$ & value $s$
      Set $I^{+}_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_j < s\}$
```

`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

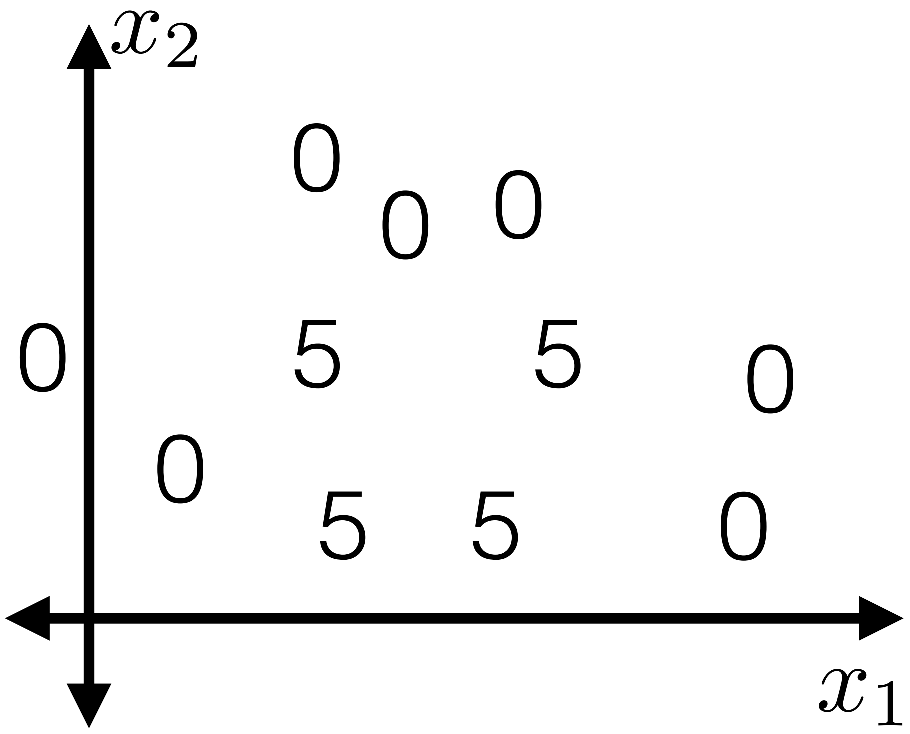
```
BuildTree($I$;$k$)
  if |$I$| ≤ k
    Set $\hat{y}$ = $\text{average}_{i \in I} y^{(i)}$
    return Leaf(label = $\hat{y}$ )
  else
    for each split dim $j$ & value $s$
      Set $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
          $\hat{y}^+_{j,s}$
          $\hat{y}^-_{j,s}$
```



BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss



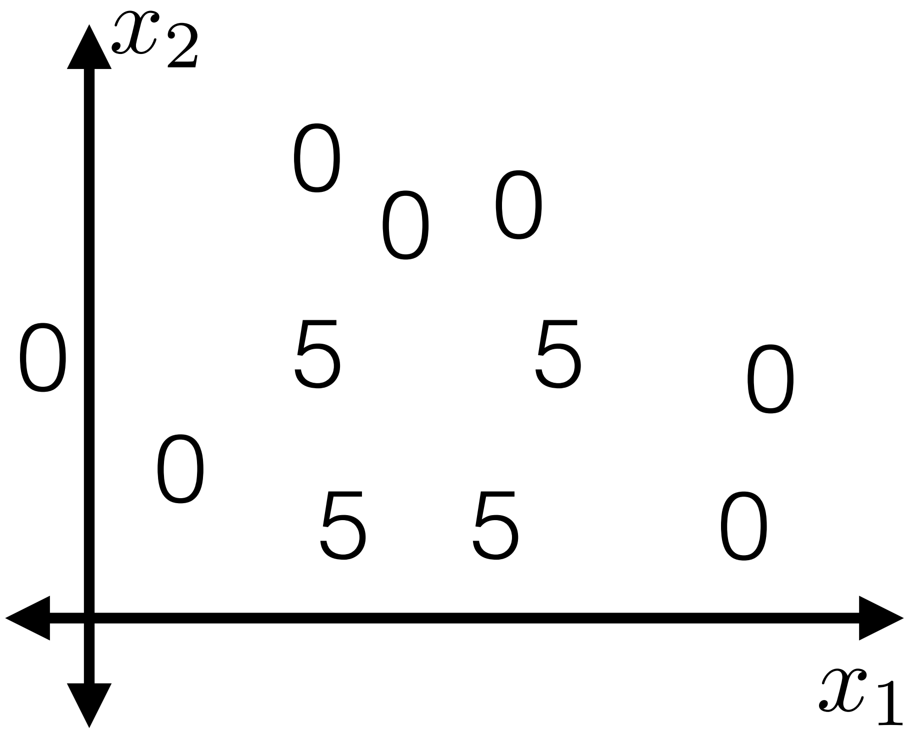
BuildTree({1,…,n};2)

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
          $\hat{y}^+_{j,s}$
          $\hat{y}^-_{j,s}$
      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 |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
          $\hat{y}^+_{j,s}$
          $\hat{y}^-_{j,s}$
      Set $E_{j,s} = \sum_{i \in I^+_{j,s}} (y^{(i)} - \hat{y}^+_{j,s})^2 + \sum_{i \in I^-_{j,s}} (y^{(i)} - \hat{y}^-_{j,s})^2$
```

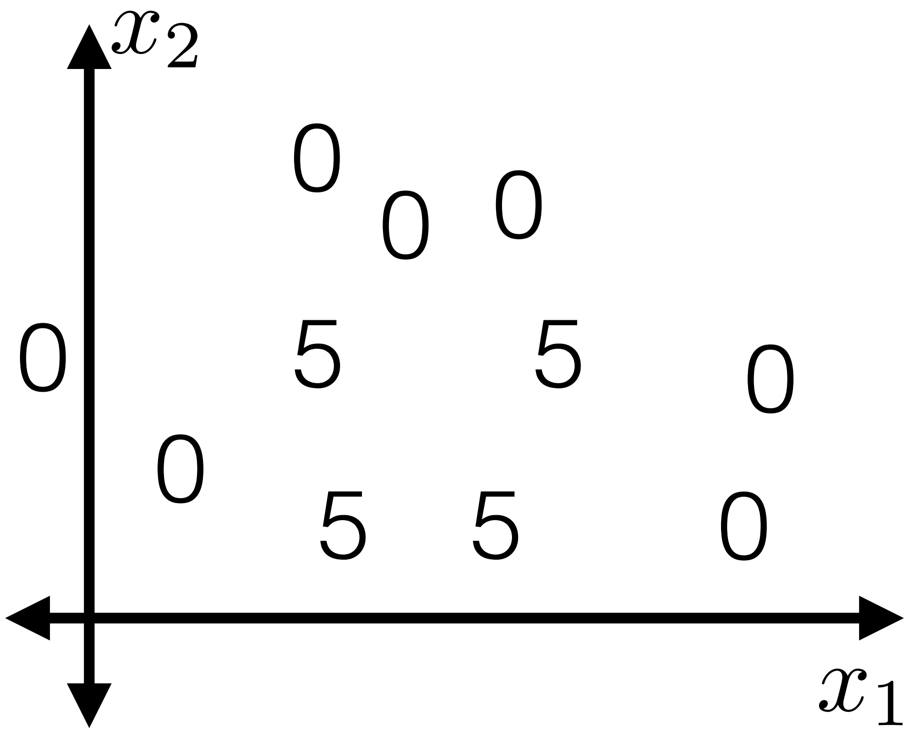


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $I^{+}_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $E_{j,s} = \sum_{i \in I^{+}_{j,s}} (y^{(i)} - \hat{y}^{+}_{j,s})^2 + \sum_{i \in I^{-}_{j,s}} (y^{(i)} - \hat{y}^{-}_{j,s})^2$
```

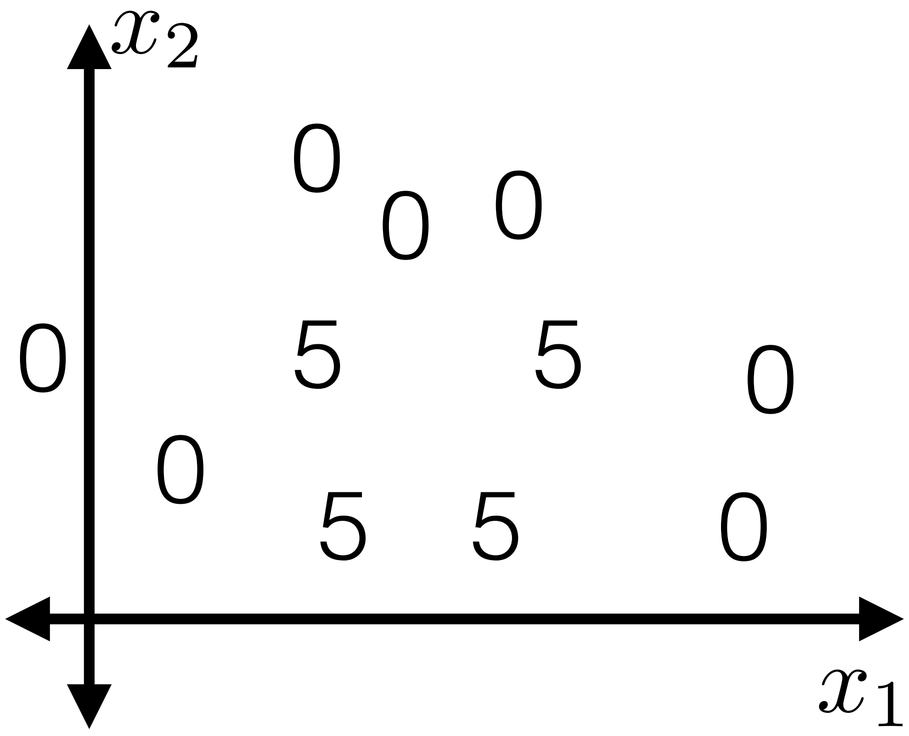


`BuildTree({1,…,n};2)`

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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $E_{j,s} = \sum_{i \in I^+_{j,s}} (y^{(i)} - \hat{y}^+_{j,s})^2 + \sum_{i \in I^-_{j,s}} (y^{(i)} - \hat{y}^-_{j,s})^2$
```

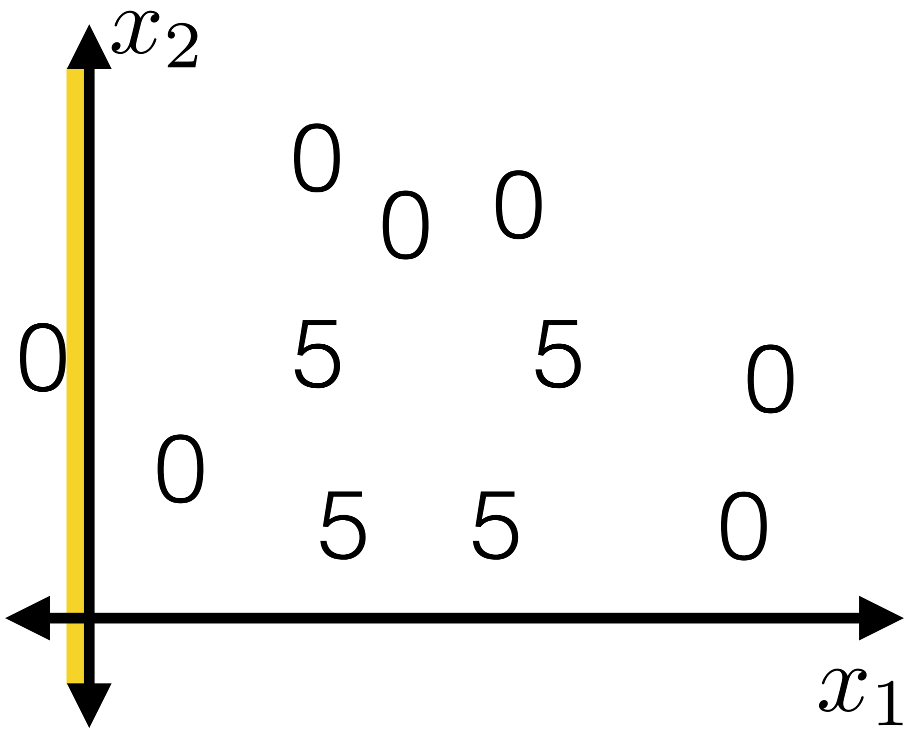


BuildTree({1,…,n};2)

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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $E_{j,s} = \sum_{i \in I^+_{j,s}} (y^{(i)} - \hat{y}^+_{j,s})^2 + \sum_{i \in I^-_{j,s}} (y^{(i)} - \hat{y}^-_{j,s})^2$
```

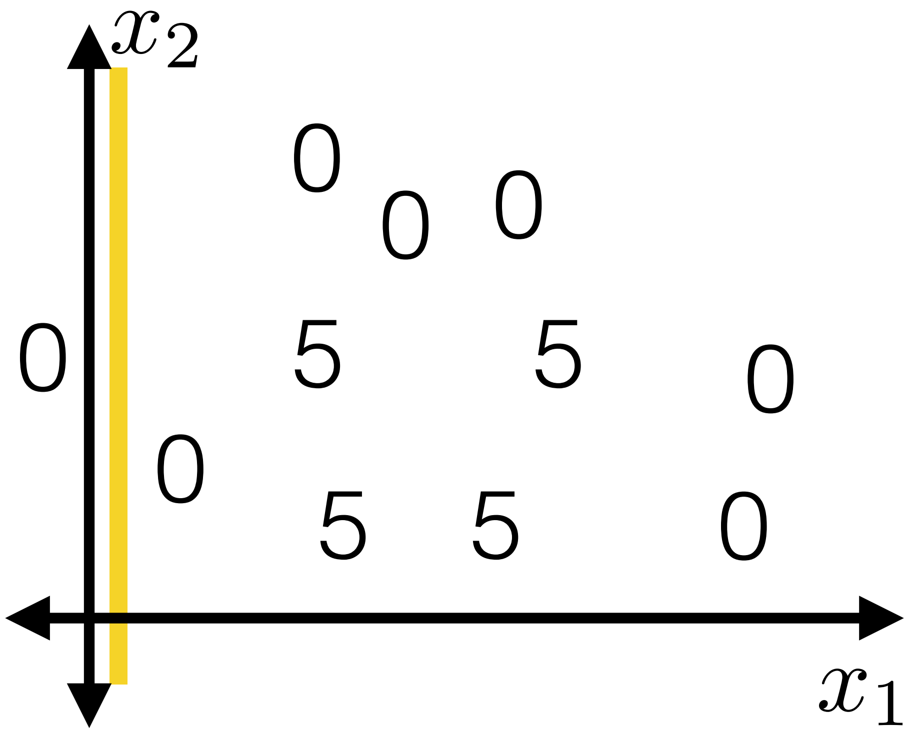


BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $E_{j,s} = \sum_{i \in I^+_{j,s}} (y^{(i)} - \hat{y}^+_{j,s})^2 + \sum_{i \in I^-_{j,s}} (y^{(i)} - \hat{y}^-_{j,s})^2$
```

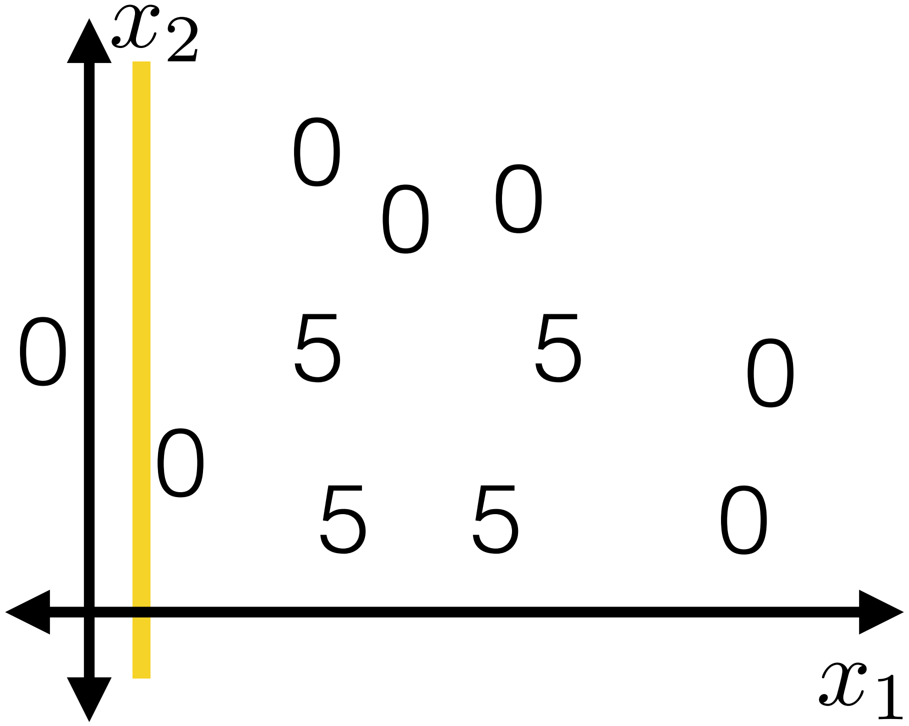


BuildTree({1,…,n};2)

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 $I^{+}_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $E_{j,s} = \sum_{i \in I^{+}_{j,s}} (y^{(i)} - \hat{y}^{+}_{j,s})^2 + \sum_{i \in I^{-}_{j,s}} (y^{(i)} - \hat{y}^{-}_{j,s})^2$
```

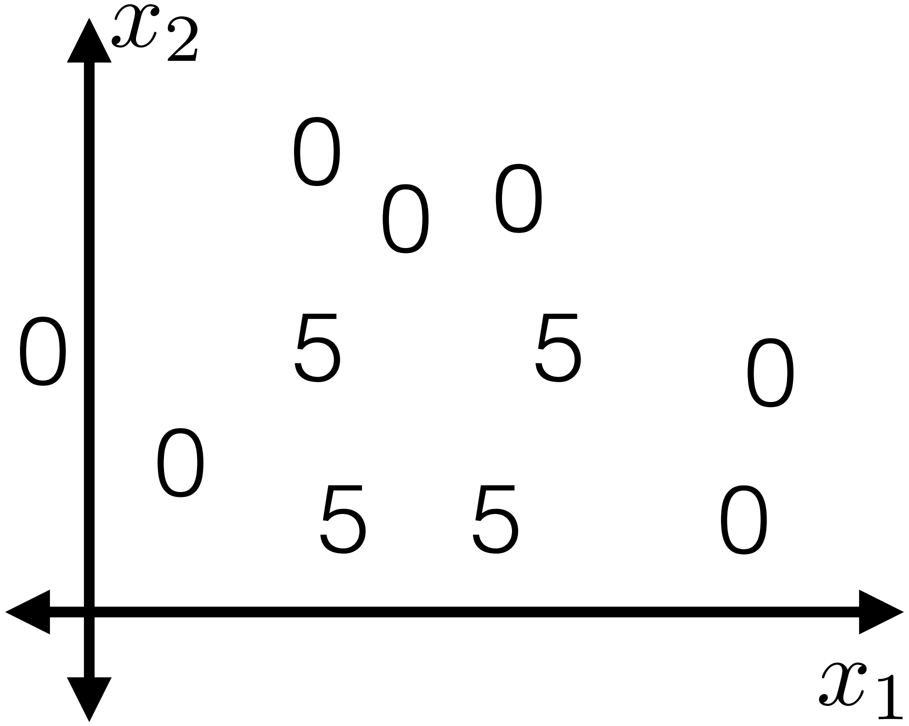


`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$; $k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $E_{j,s} = \sum_{i \in I^+_{j,s}} (y^{(i)} - \hat{y}^+_{j,s})^2 + \sum_{i \in I^-_{j,s}} (y^{(i)} - \hat{y}^-_{j,s})^2$
```

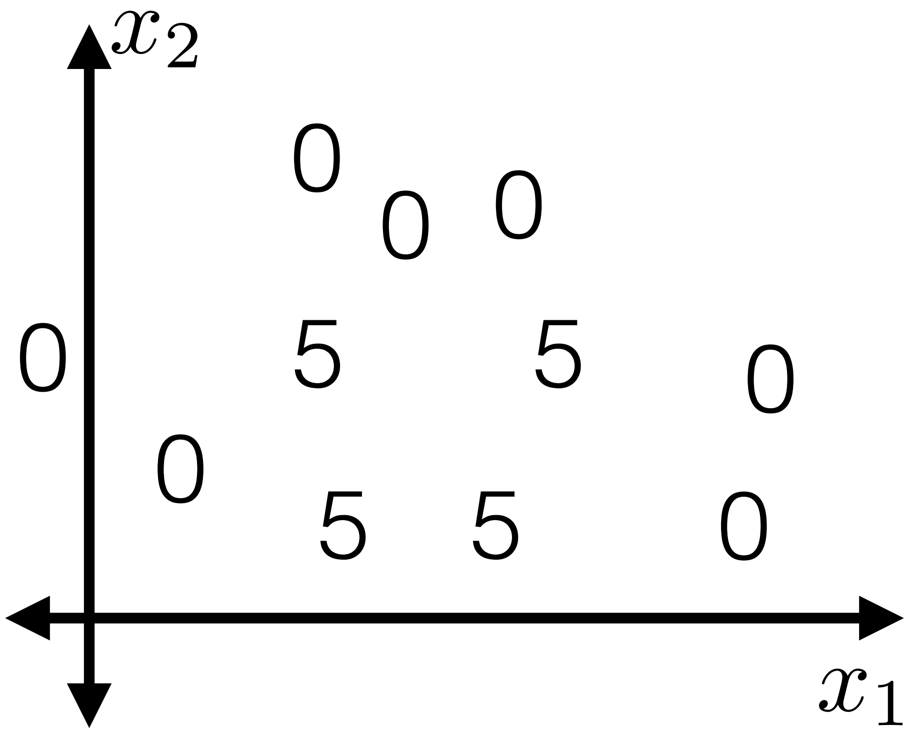


BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

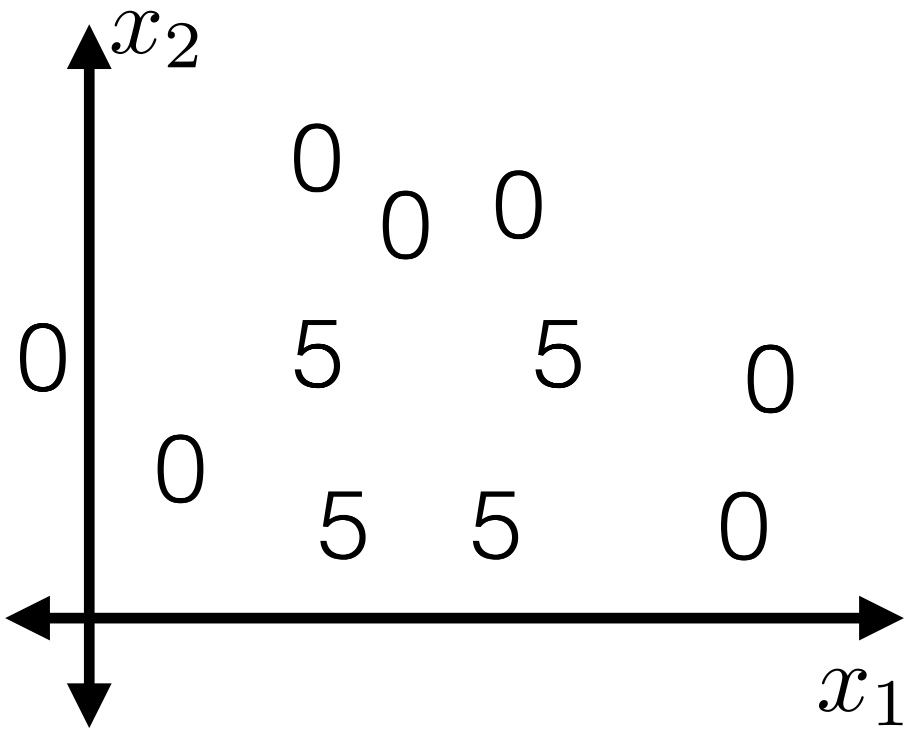
```
BuildTree($I; k$)
  if $|I| \leq$ k
    Set $\hat{y}$ = $\text{average}_{i \in I} y^{(i)}$
    return Leaf(label = $\hat{y}$)
  else
    for each split dim $j$ & value $s$
      Set $I_{j,s}^{+} = \{i \in I | x_j^{(i)} \geq s\}$
      Set $I_{j,s}^{-} = \{i \in I | x_j^{(i)} < s\}$
      Set $\hat{y}_{j,s}^{+} = \text{average}_{i \in I_{j,s}^{+}} y^{(i)}$
      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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
```



BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

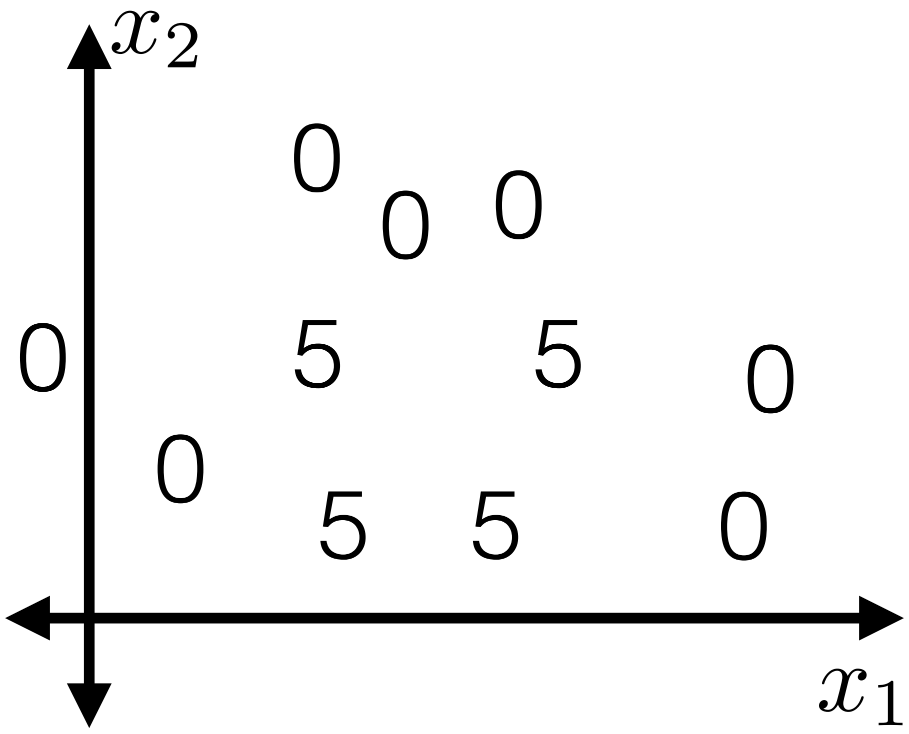


BuildTree({1,…,n};2)

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node
```

Building a decision tree

- Regression tree with squared error loss



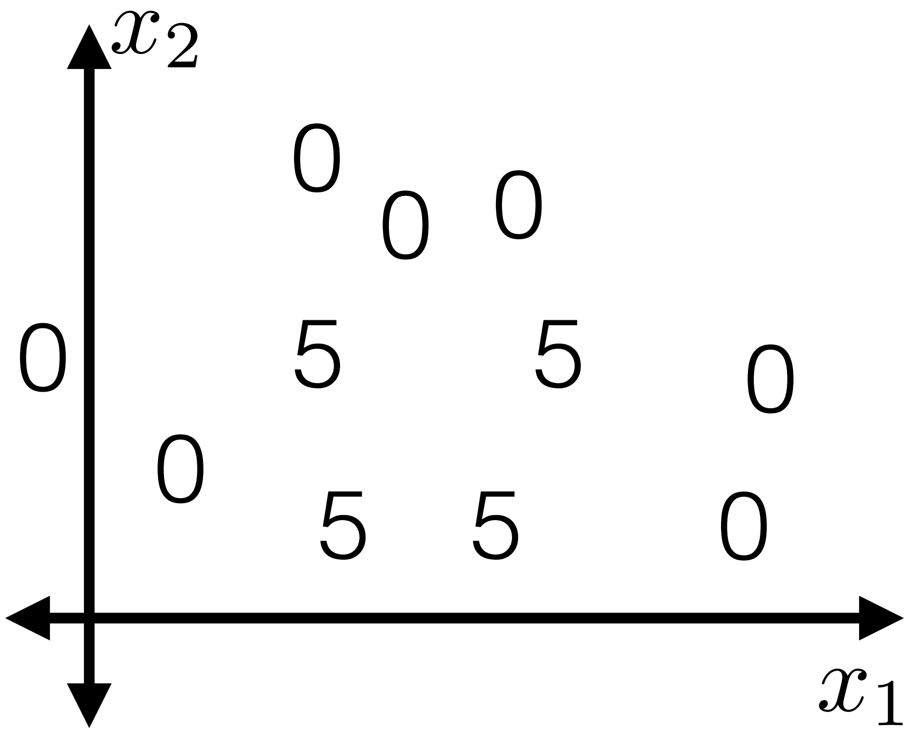
BuildTree({1,…,n};2)

```
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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node(dim,val,left-child,right-child)
```

Building a decision tree

- Regression tree with squared error loss

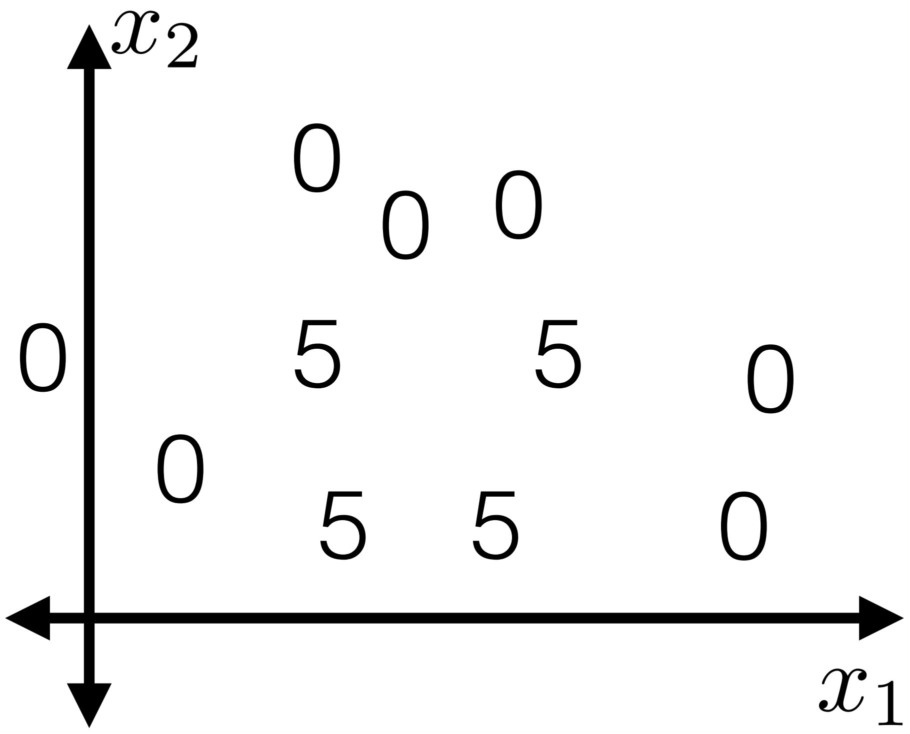
```
BuildTree($I; k$)
  if $|I| \leq$ k
    Set $\hat{y}$ = $\text{average}_{i \in I} y^{(i)}$
    return Leaf(label = $\hat{y}$ )
  else
    for each split dim $j$ & value $s$
      Set $I^+_{j,s} = \{i \in I | x_j^{(i)} \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x_j^{(i)} < s\}$
      Set $\hat{y}^+_{j,s} = \text{average}_{i \in I^+_{j,s}} y^{(i)}$
      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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node($j^*$, val, left-child, right-child)
```



`BuildTree({1,…,n};2)`

Building a decision tree

- Regression tree with squared error loss

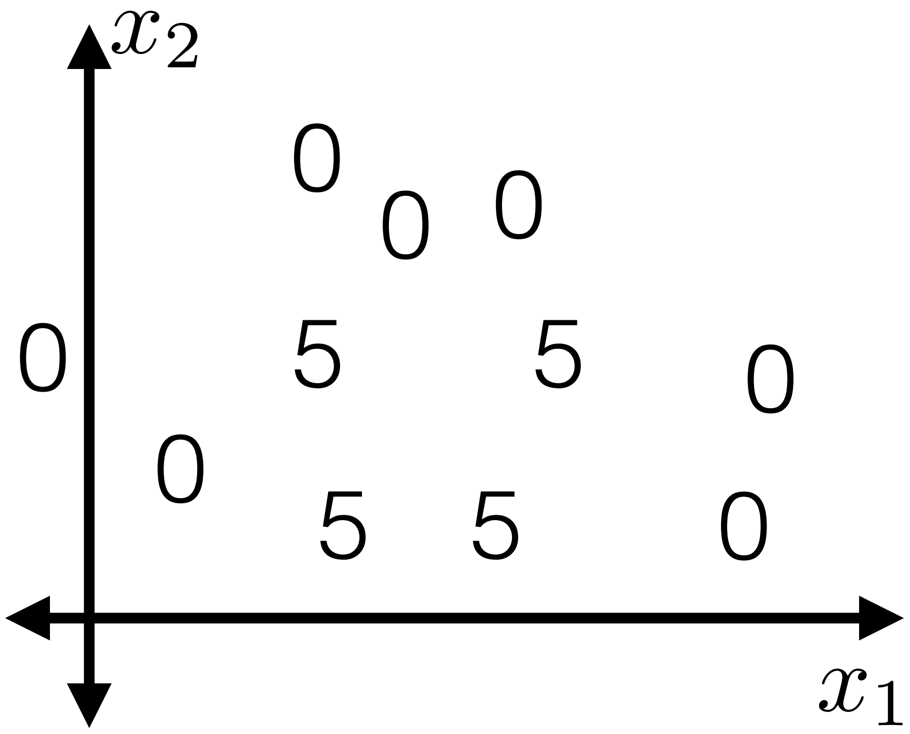


BuildTree({1,…,n};2)

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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

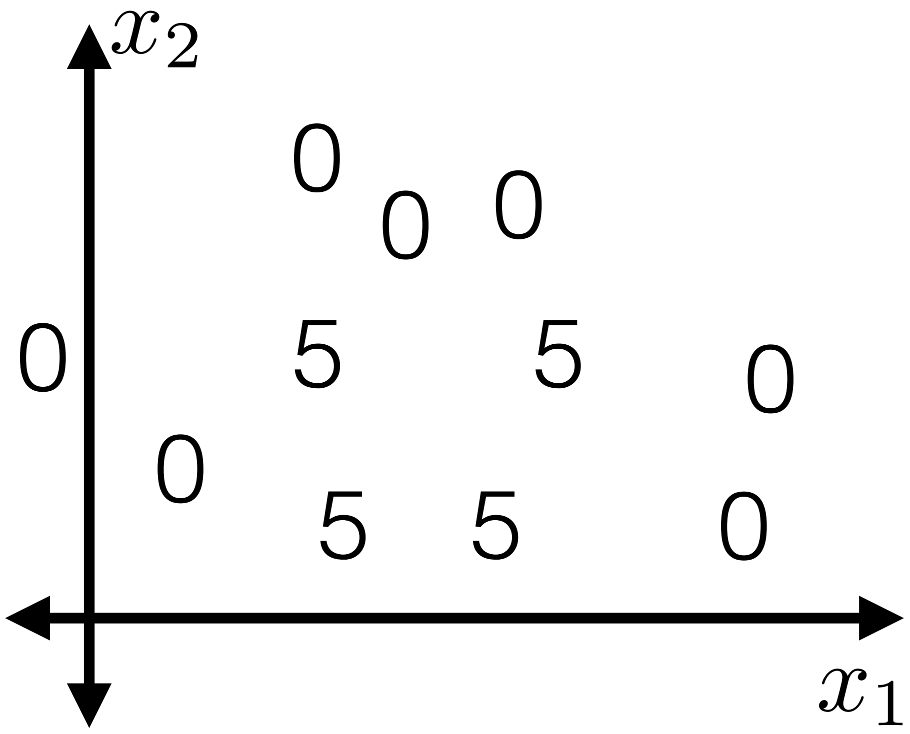


BuildTree({1,…,n};2)

```
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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*$ ,left-child,right-child)
```

Building a decision tree

- Regression tree with squared error loss



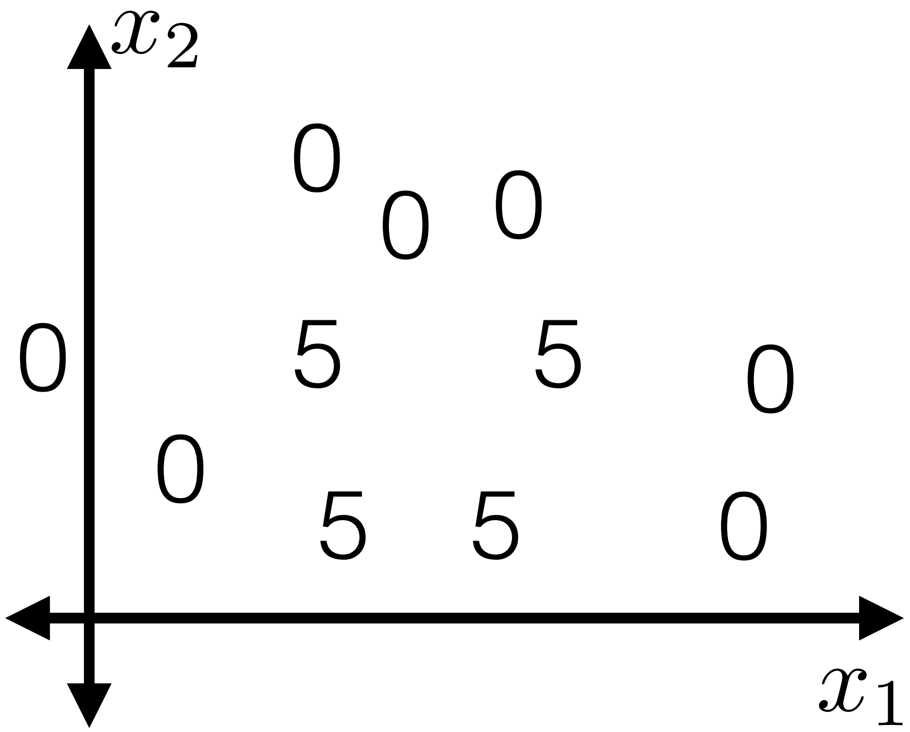
```
BuildTree({1,…,n};2)
```

```
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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*$ ,left-child,right-child)
```

Building a decision tree

- Regression tree with squared error loss

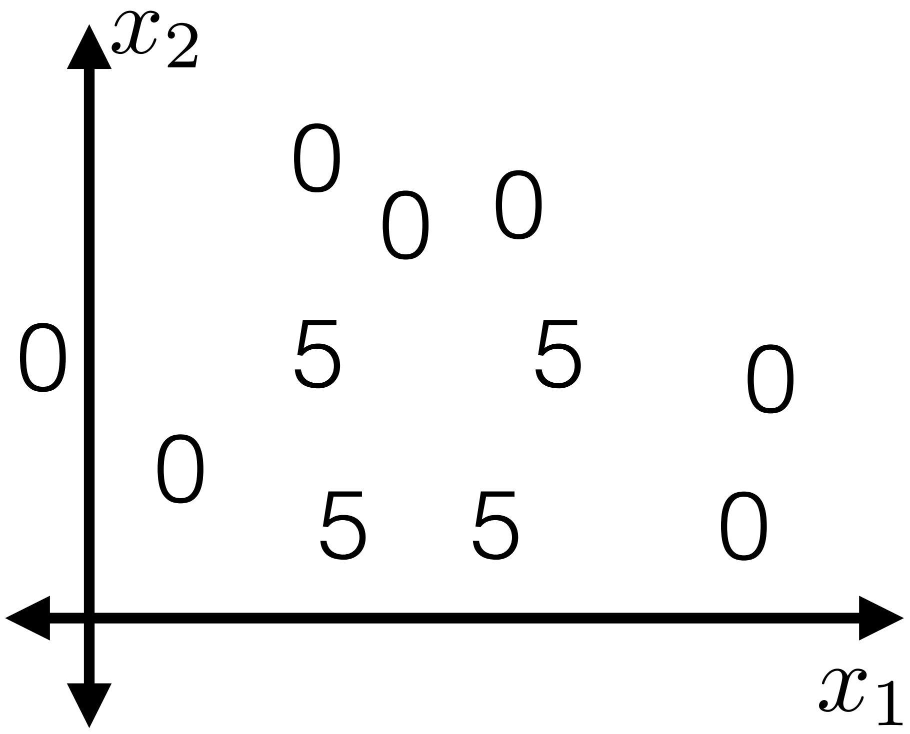
```
BuildTree($I$;$k$)
  if |$I$| ≤ k
    Set $\hat{y}$ = $\text{average}_{i \in I} y^{(i)}$
    return Leaf(label = $\hat{y}$)
  else
    for each split dim $j$ & value $s$
      Set $I^+_{j,s} = \{i \in I | x_j^{(i)} \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x_j^{(i)} < s\}$
      Set $\hat{y}^+_{j,s} = \text{average}_{i \in I^+_{j,s}} y^{(i)}$
      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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k)$, right-child)
```



BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

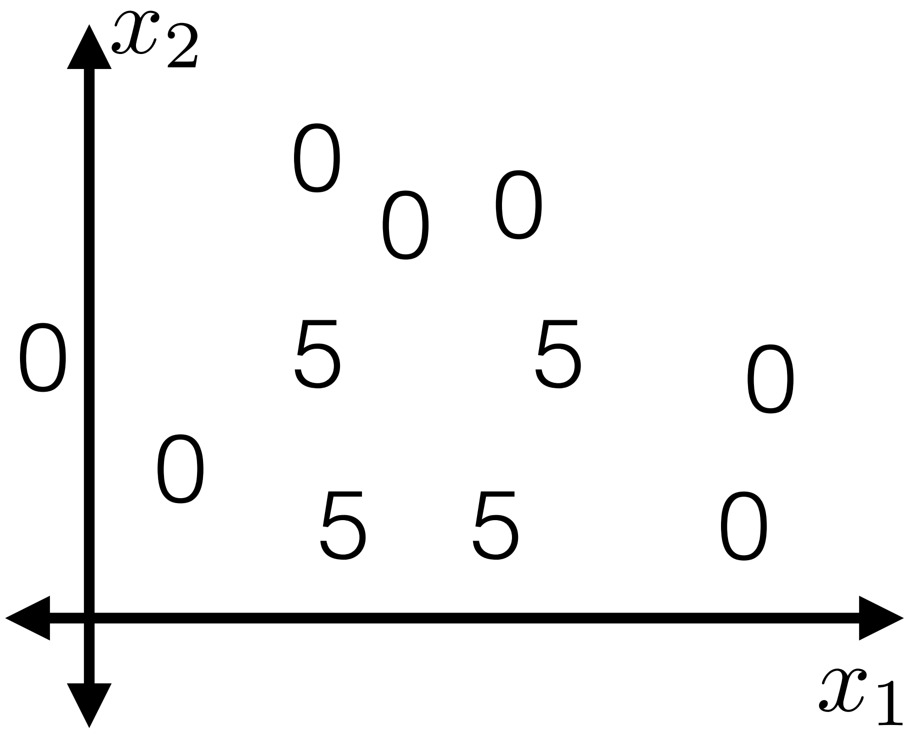


BuildTree({1,…,n};2)

```
BuildTree($I$; $k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k)$, right-child)
```

Building a decision tree

- Regression tree with squared error loss

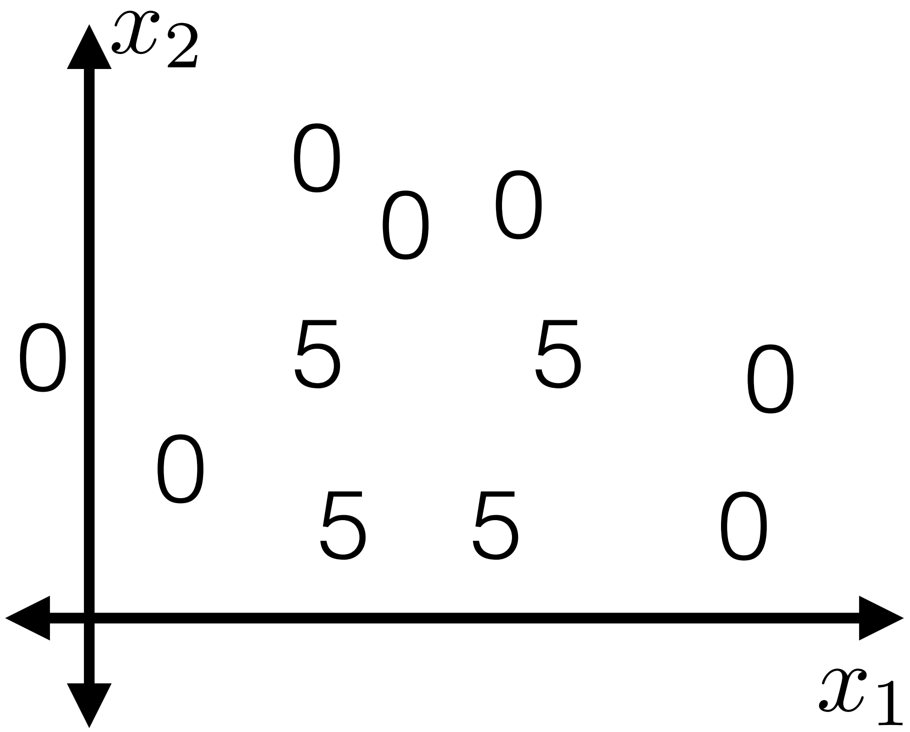


BuildTree({1,…,n};2)

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k)$, $\text{BuildTree}(I^+_{j^*,s^*}, k)$)
```

Building a decision tree

- Regression tree with squared error loss



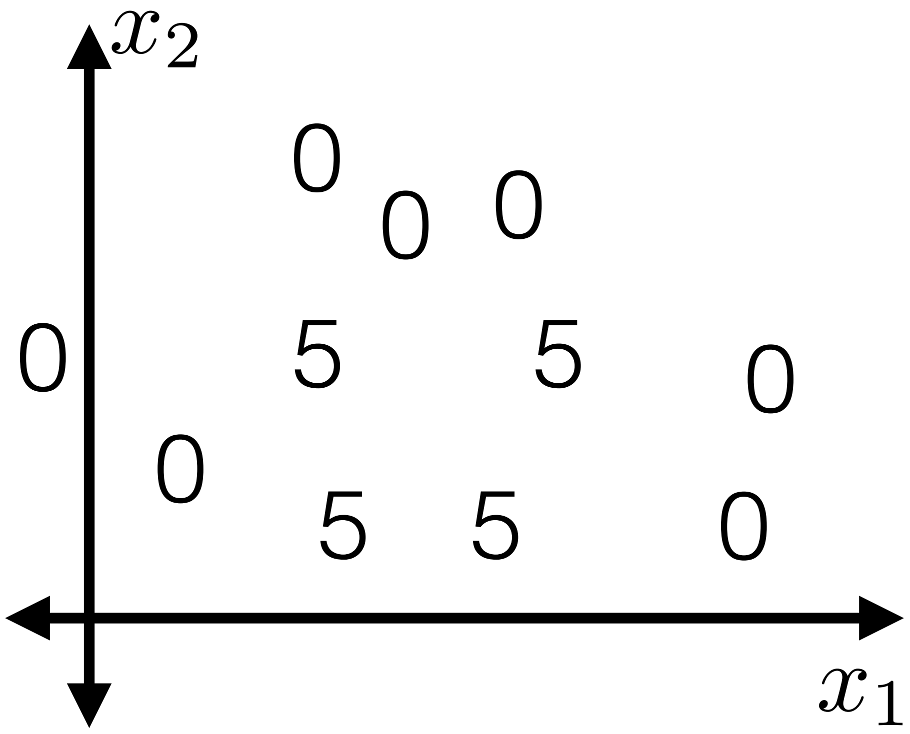
BuildTree({1,…,n};2)

```
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 $I^{+}_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^{-}_{j^*,s^*}, k), \text{BuildTree}(I^{+}_{j^*,s^*}, k))$
```

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k)$, $\text{BuildTree}(I^+_{j^*,s^*}, k)$)
```

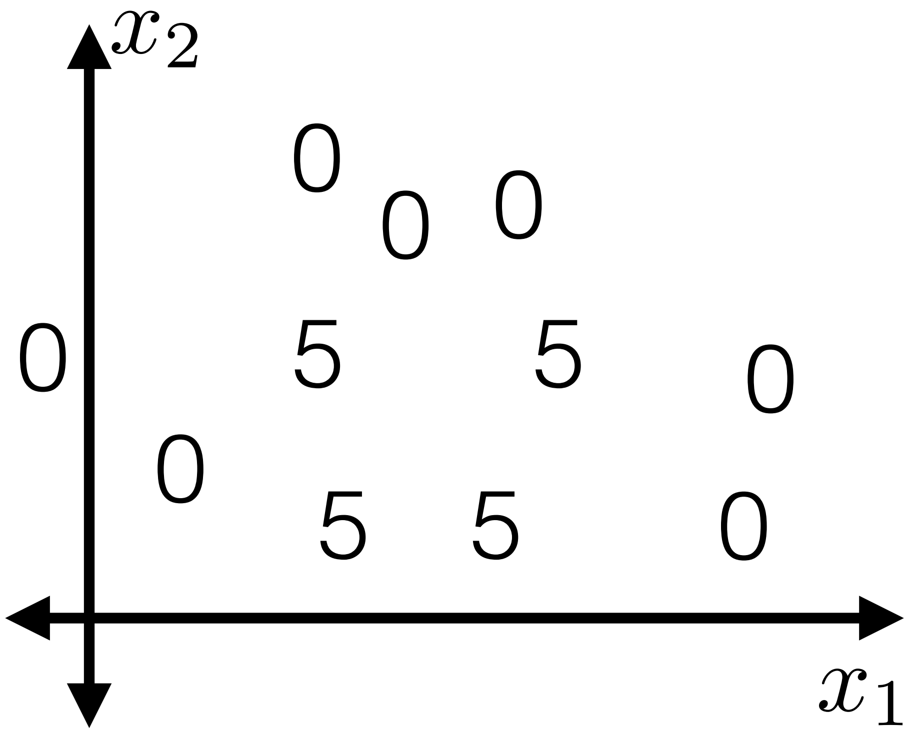


BuildTree({1,…,n};2)

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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```

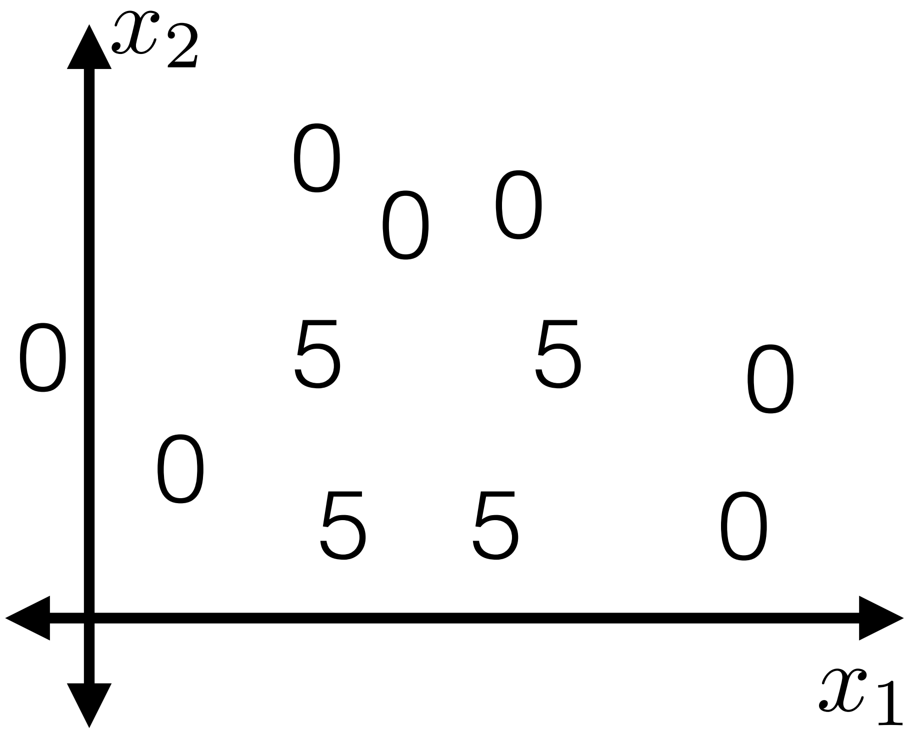


BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$; $k$)
  if |$I$| ≤ 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 $I^{+}_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 ($j^*, s^*, \text{BuildTree}(I^{-}_{j^*,s^*}, k), \text{BuildTree}(I^{+}_{j^*,s^*}, k)$)
```

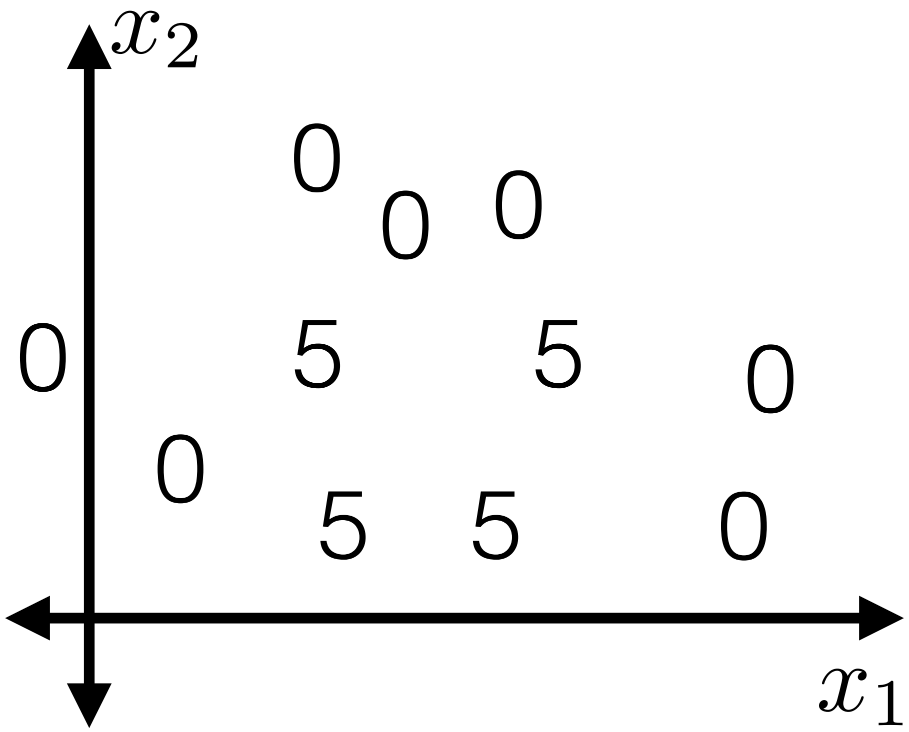


BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

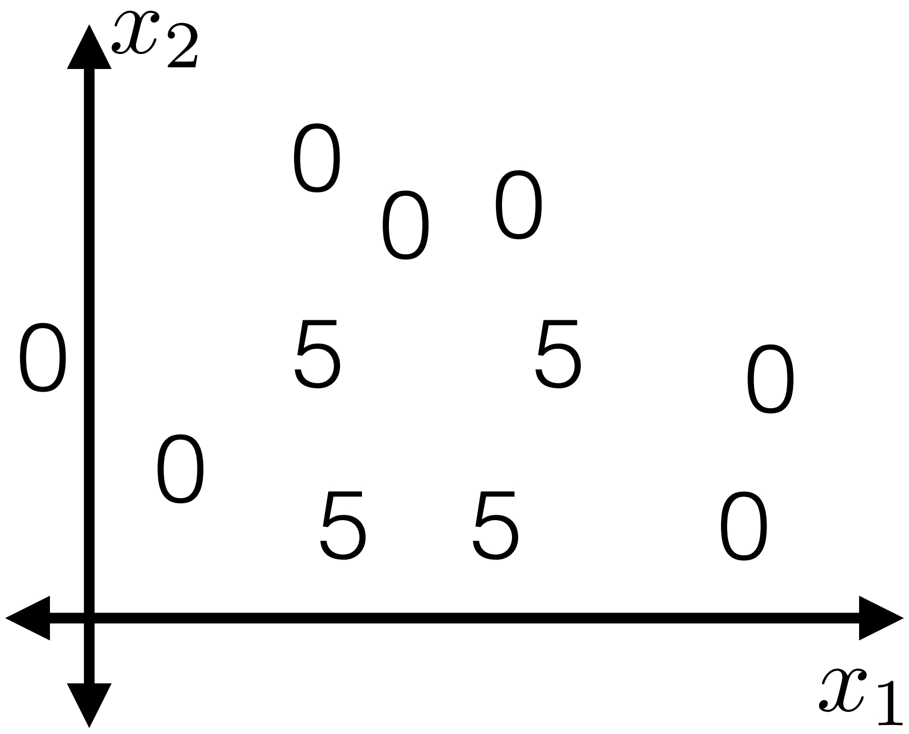
```
BuildTree($I$; $k$)
  if |$I$| ≤ k
    Set $\hat{y}$ = $\text{average}_{i \in I} y^{(i)}$
    return Leaf(label = $\hat{y}$)
  else
    for each split dim $j$ & value $s$
      Set $I_{j,s}^{+} = \{i \in I | x_j^{(i)} \geq s\}$
      Set $I_{j,s}^{-} = \{i \in I | x_j^{(i)} < s\}$
      Set $\hat{y}_{j,s}^{+} = \text{average}_{i \in I_{j,s}^{+}} y^{(i)}$
      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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node($j^*$, $s^*$, $\text{BuildTree}(I_{j^*,s^*}^{-}, k)$, $\text{BuildTree}(I_{j^*,s^*}^{+}, k)$)
```



BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss



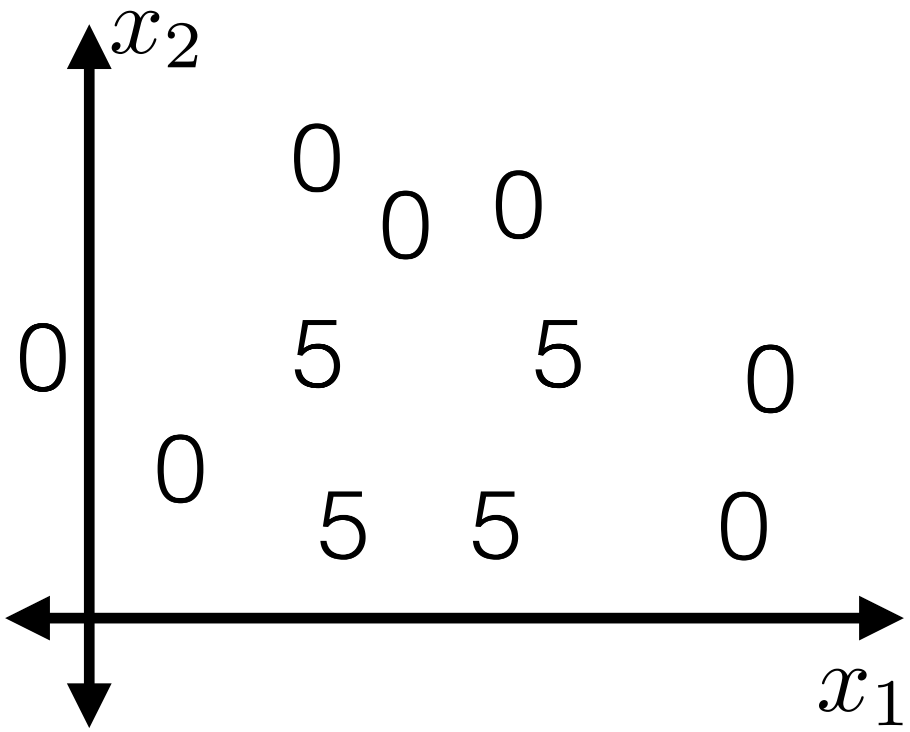
BuildTree({1,…,n};2)

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$; $k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($I^+_{j^*,s^*}, k$))
```

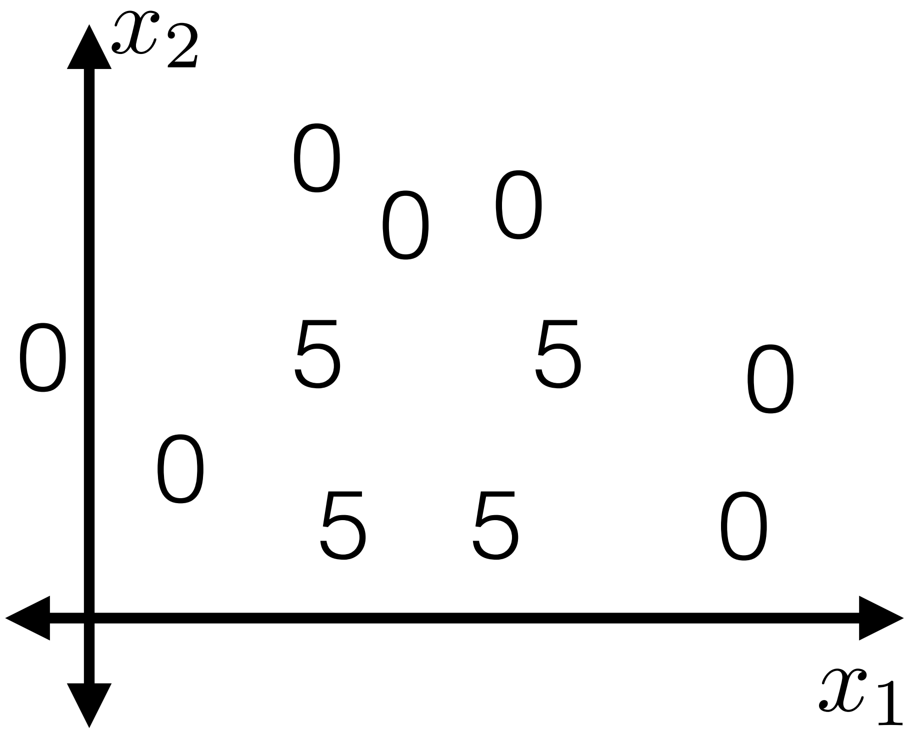


BuildTree({1,…,n};2)

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 $I^+_{j,s} = \{i \in I | x_j^{(i)} \geq s\}$
      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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```

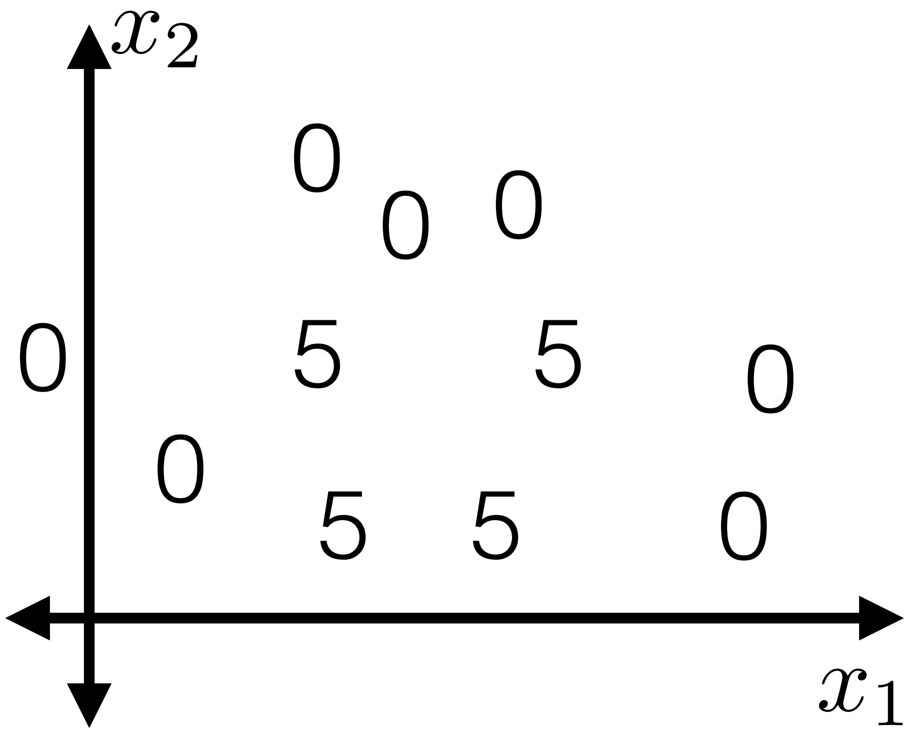


BuildTree({1,…,n};2)

Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I; k$)
  if |$I$| ≤ 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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```

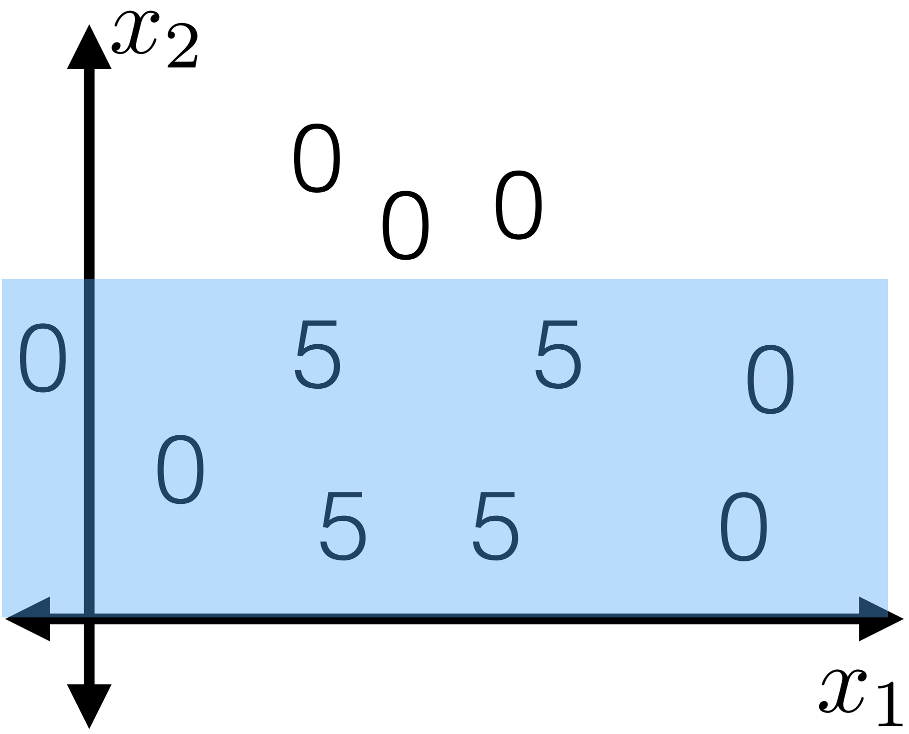


`BuildTree({1,…,n};2)`

$x_2 \geq 0.28$

Building a decision tree

- Regression tree with squared error loss



BuildTree({1,…,n};2)

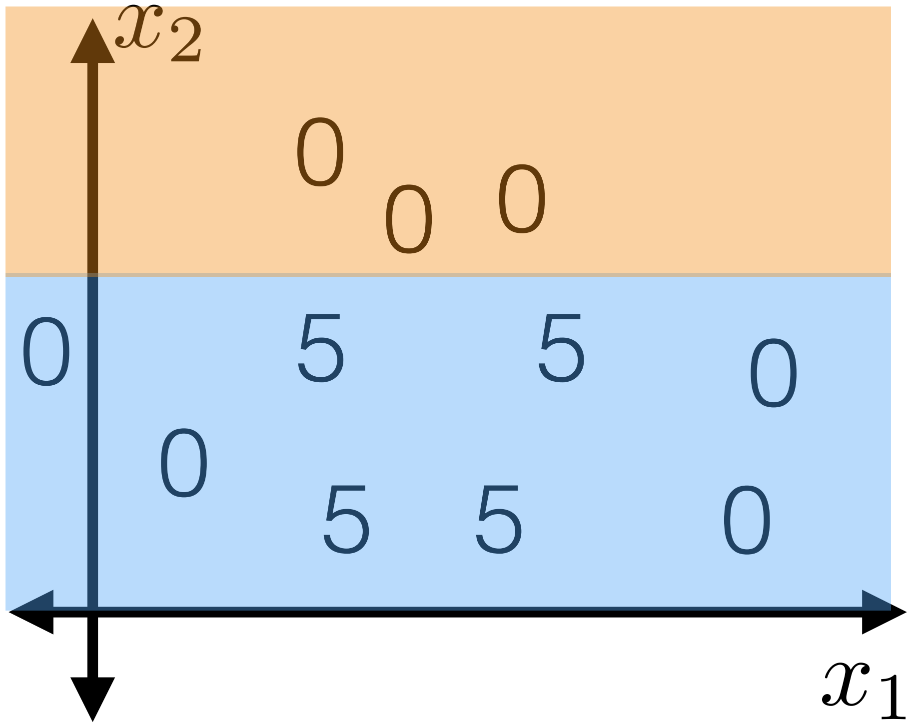
$x_2 \geq 0.28$

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $\text{Node}(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```

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 $I^{+}_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($I^{+}_{j^*,s^*}, k$))
```



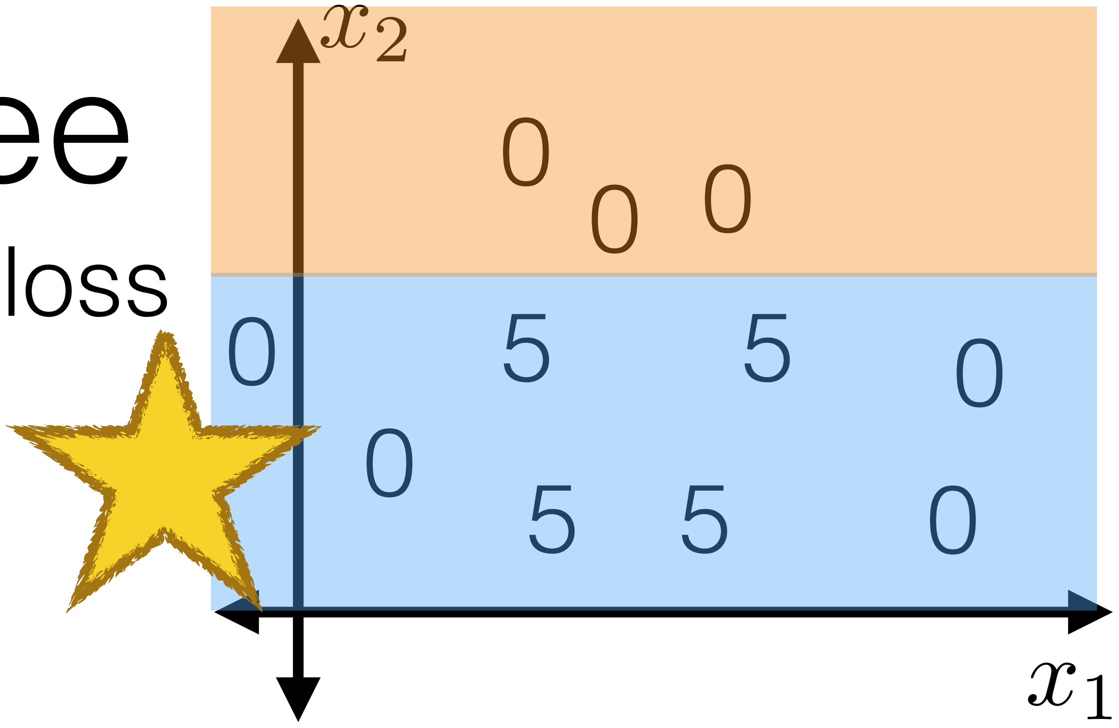
BuildTree({1,…,n};2)

$x_2 \geq 0.28$

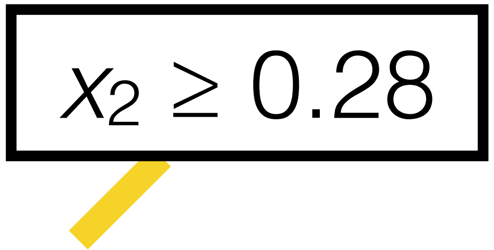
Building a decision tree

- 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*$, $s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k)$, $\text{BuildTree}(I^+_{j^*,s^*}, k)$)
```



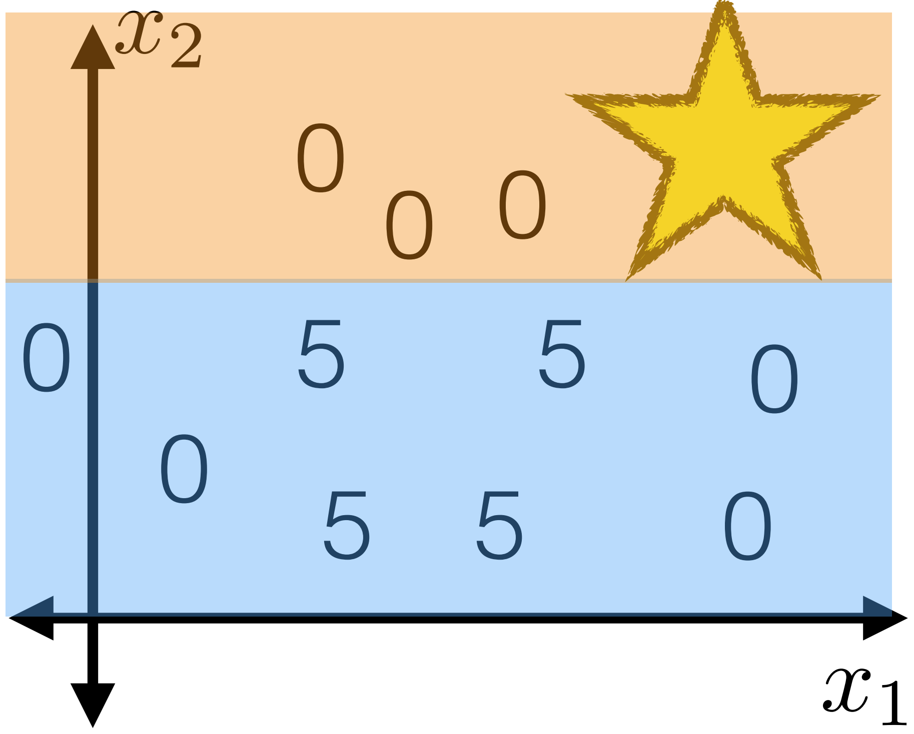
BuildTree({1,…,n};2)



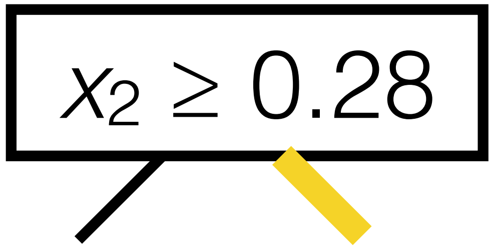
Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$; $k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 ($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k)$, $\text{BuildTree}(I^+_{j^*,s^*}, k)$)
```



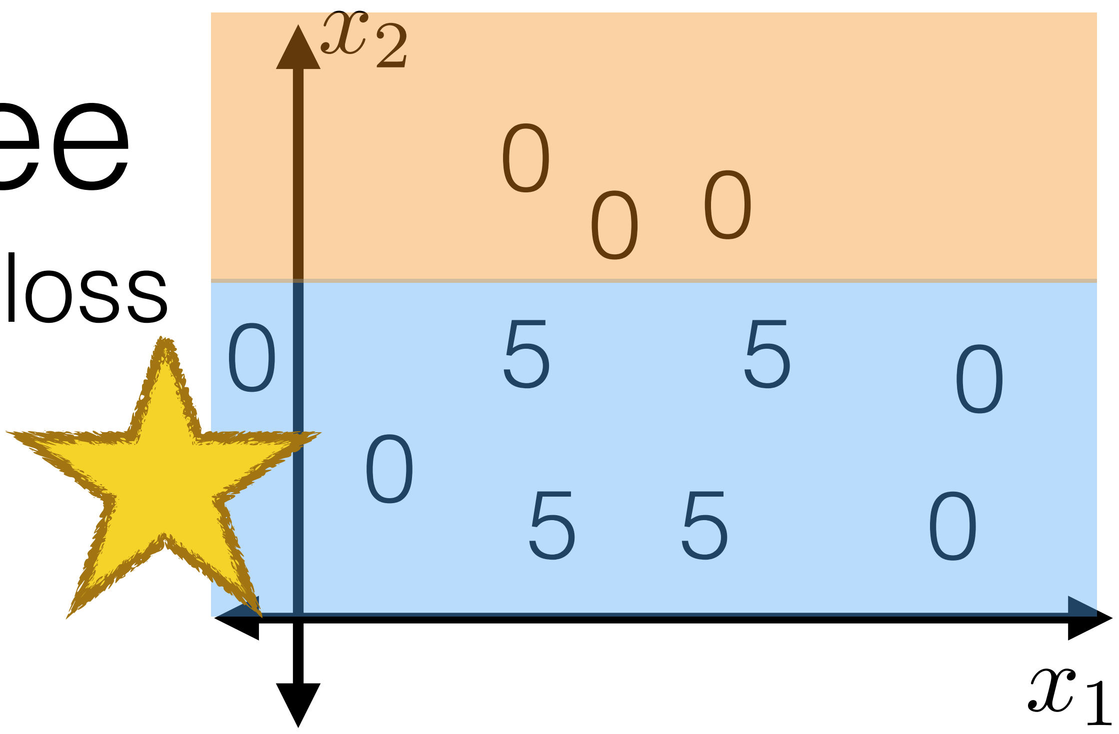
BuildTree({1,…,n};2)



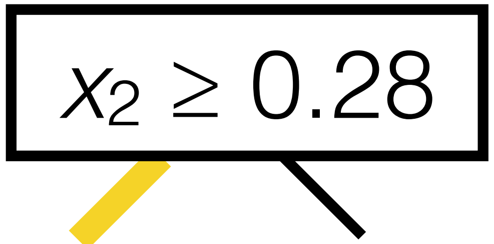
Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$; $k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*$, $s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k)$, $\text{BuildTree}(I^+_{j^*,s^*}, k)$)
```



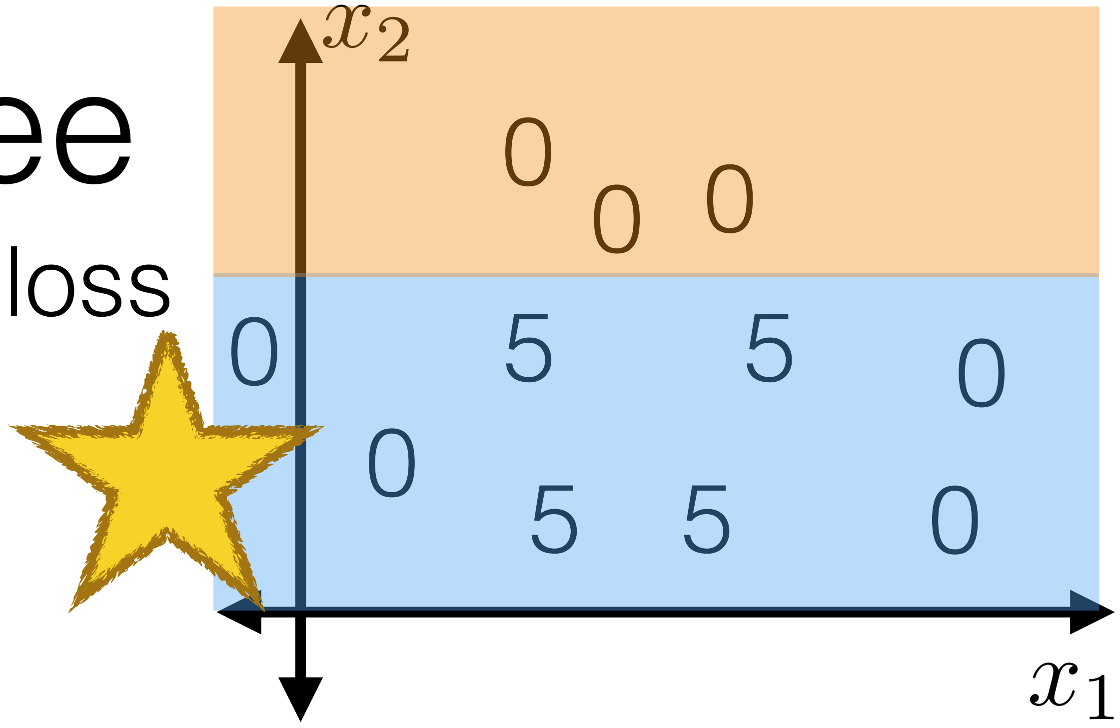
BuildTree({1,…,n};2)



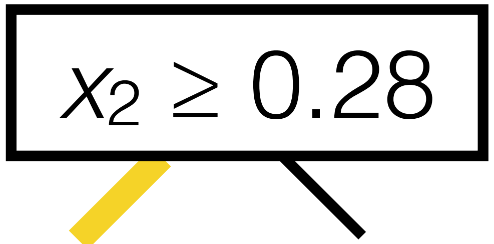
Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I; k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 ($j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```



BuildTree({1,…,n};2)

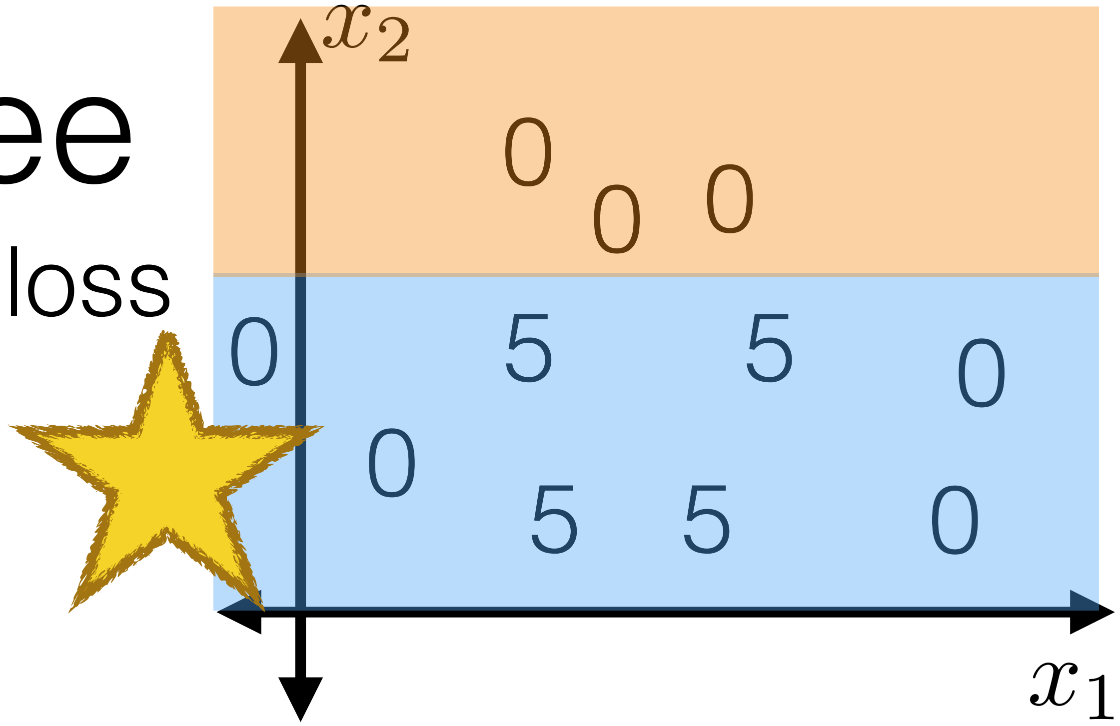


$x_2 \geq 0.28$

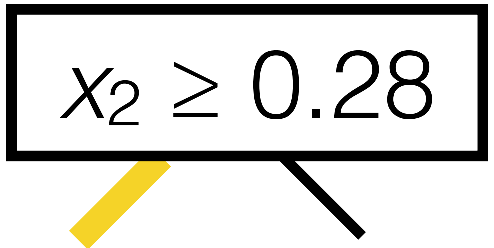
Building a decision tree

- 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 ($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```



BuildTree({1,…,n};2)

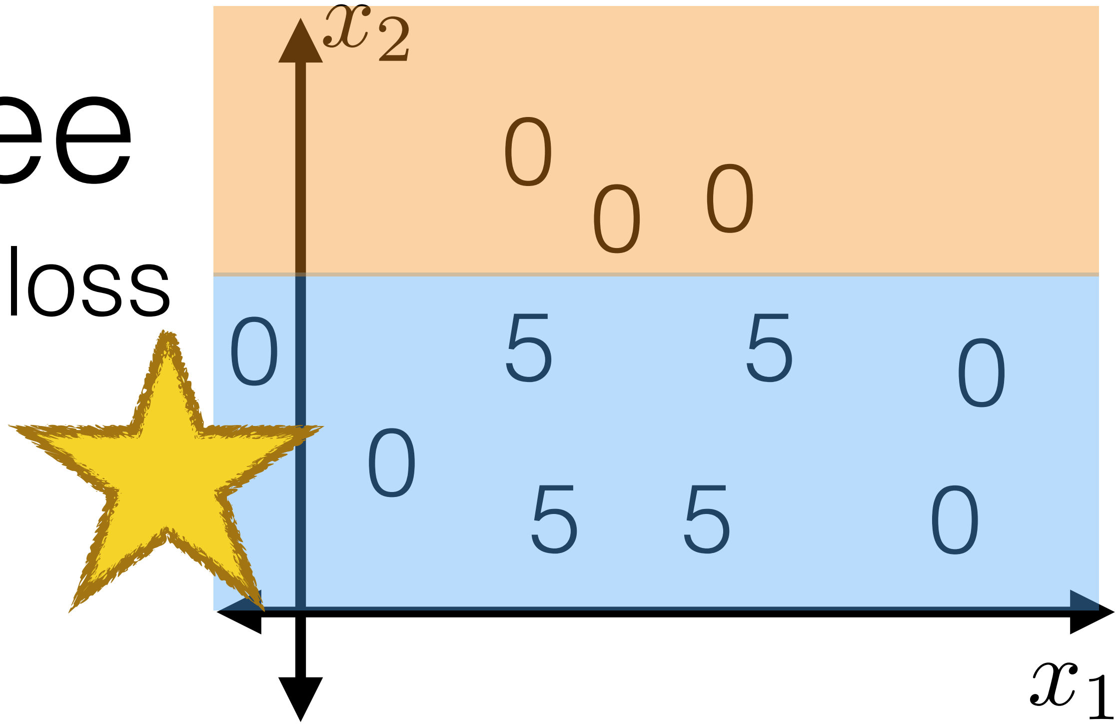


$x_2 \geq 0.28$

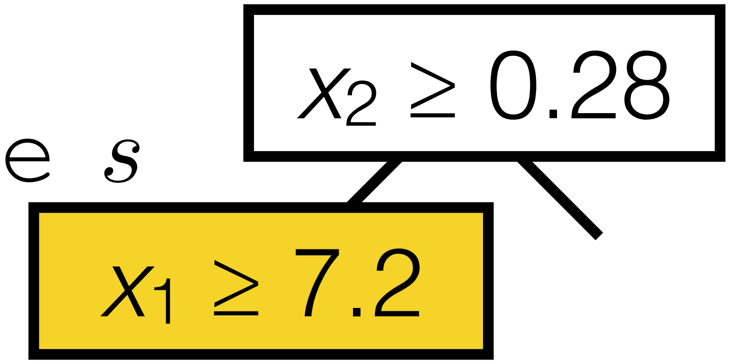
Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I; k$)
  if $|I| \le$ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \ge s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```



BuildTree({1,…,n};2)

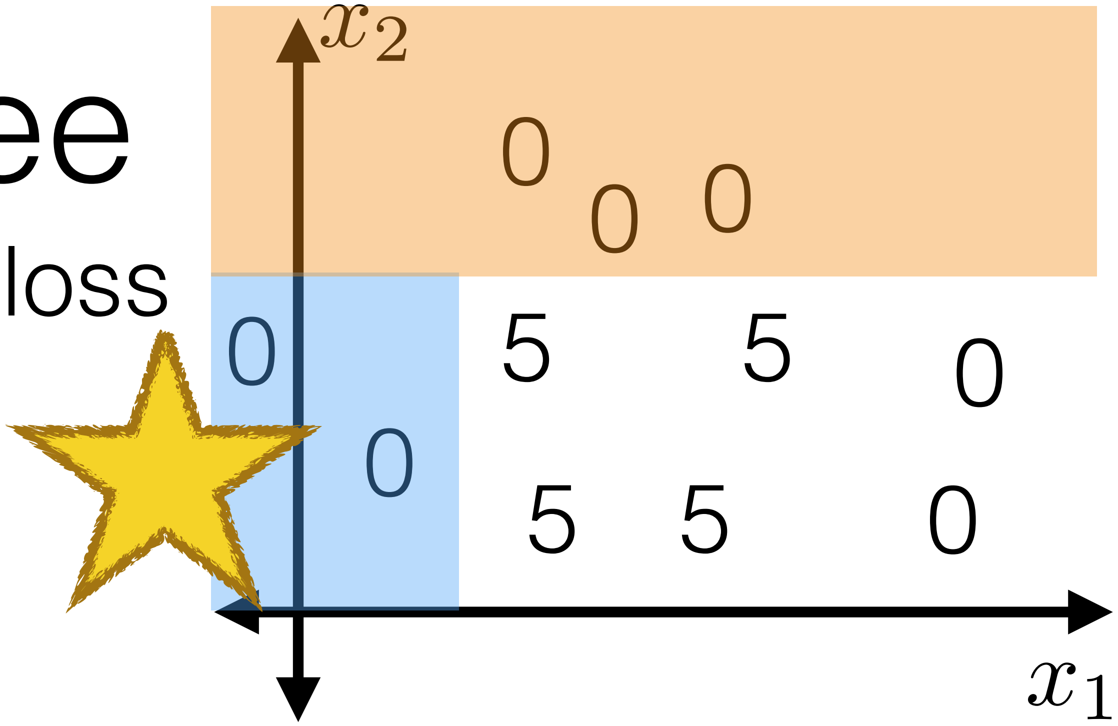


$x_2 \ge 0.28$

$x_1 \ge 7.2$

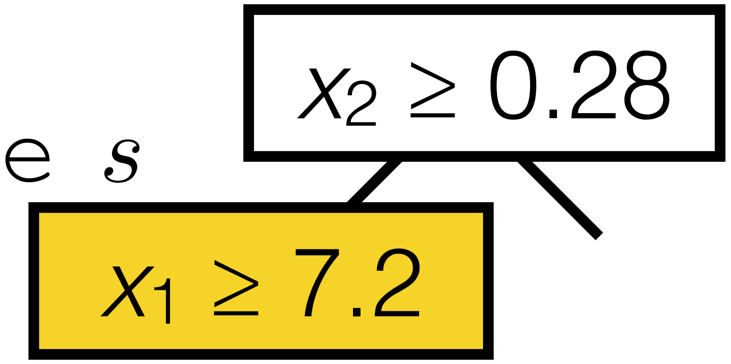
Building a decision tree

- 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```

BuildTree({1,…,n};2)

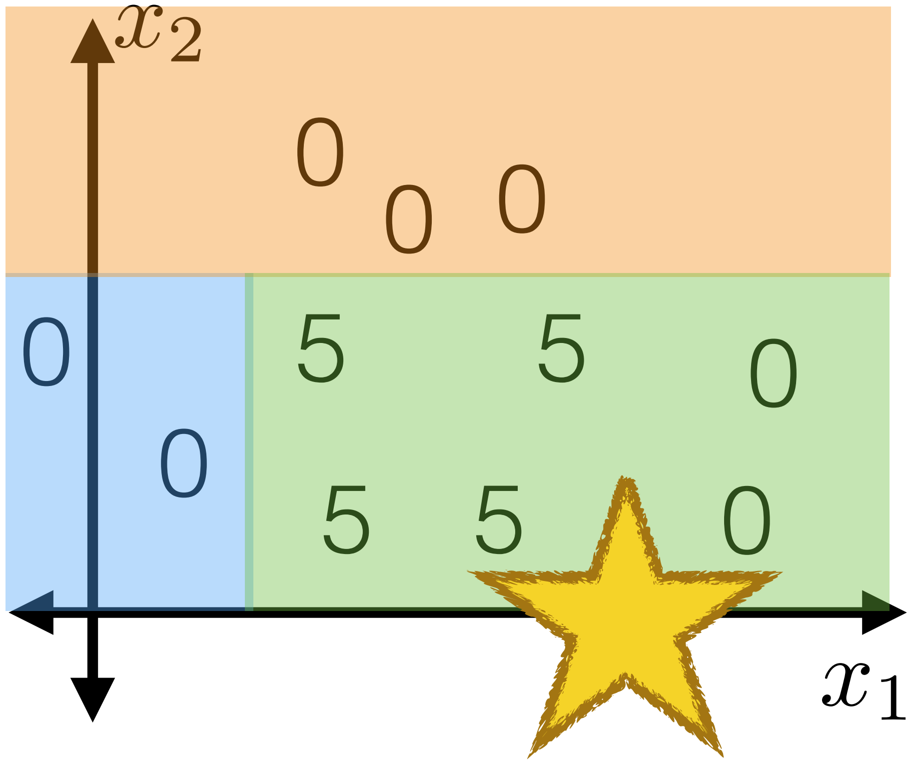

$x_2 \geq 0.28$

$x_1 \geq 7.2$

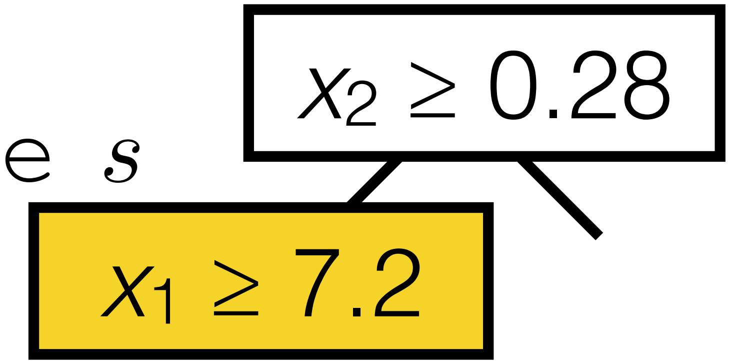
Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$; $k$)
  if |$I$| ≤ 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 $I^{+}_{j,s} = \{i \in I | x^{(i)}_{j} \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_{j} < s\}$
      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$(j^*, s^*$, $\text{BuildTree}(I^{-}_{j^*,s^*}, k), \text{BuildTree}(I^{+}_{j^*,s^*}, k))$
```

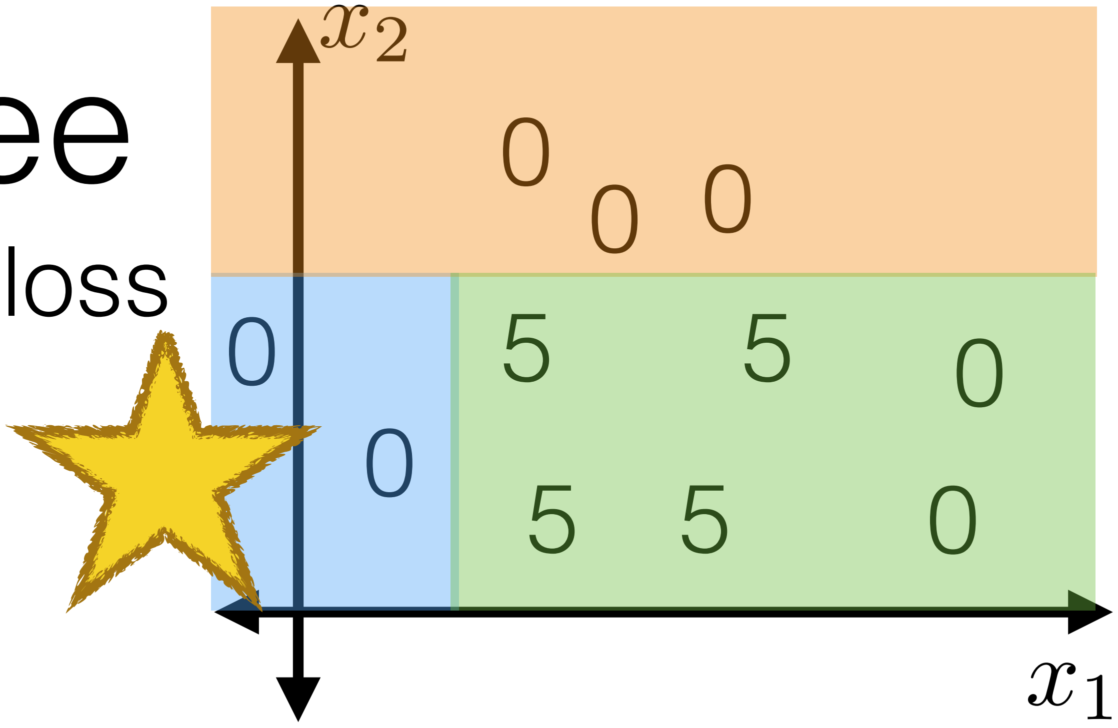


BuildTree({1,…,n};2)

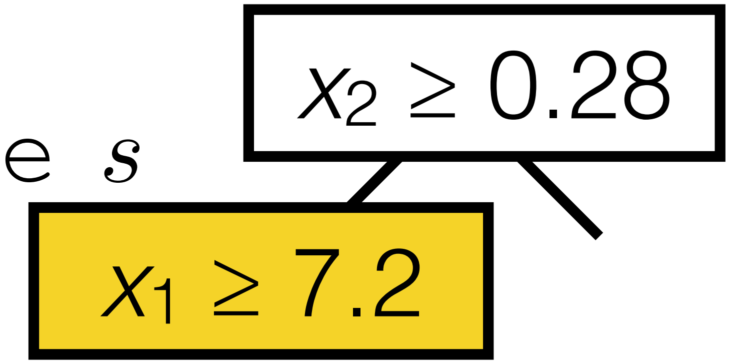


Building a decision tree

- Regression tree with squared error loss



BuildTree({1,…,n};2)



$x_2 \geq 0.28$

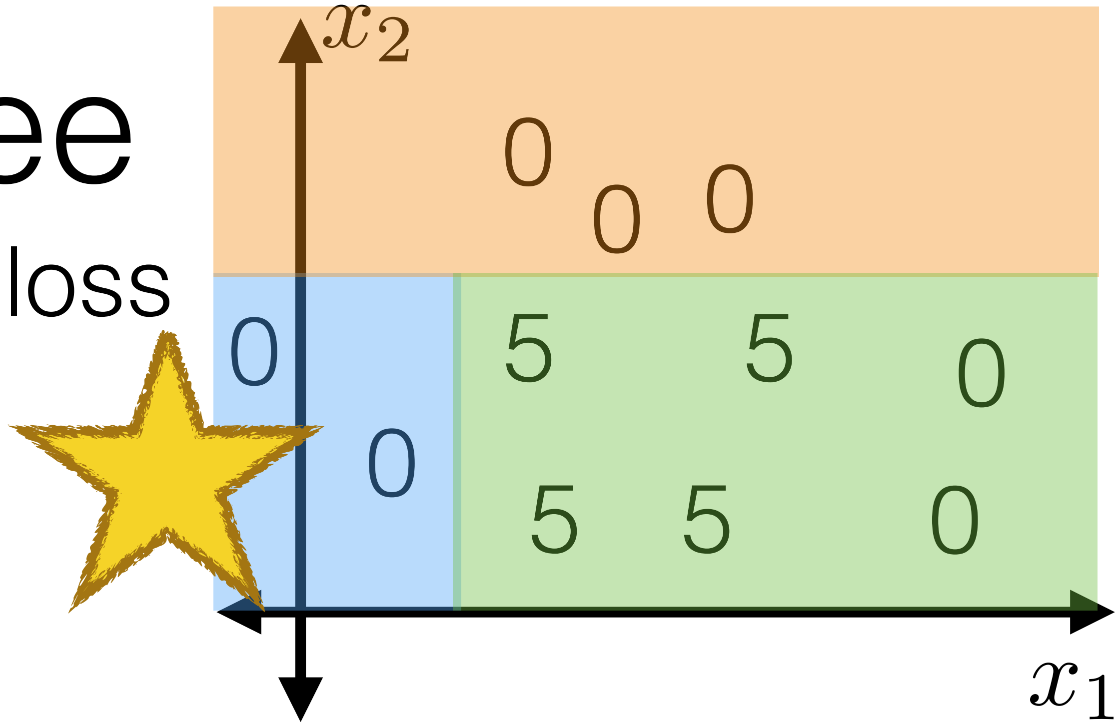
$x_1 \geq 7.2$

```
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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```

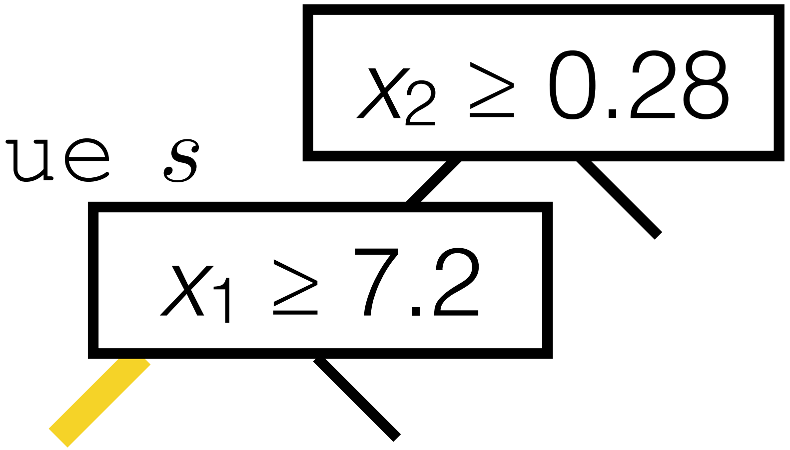
Building a decision tree

- 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k)$, $\text{BuildTree}(I^+_{j^*,s^*}, k))$
```



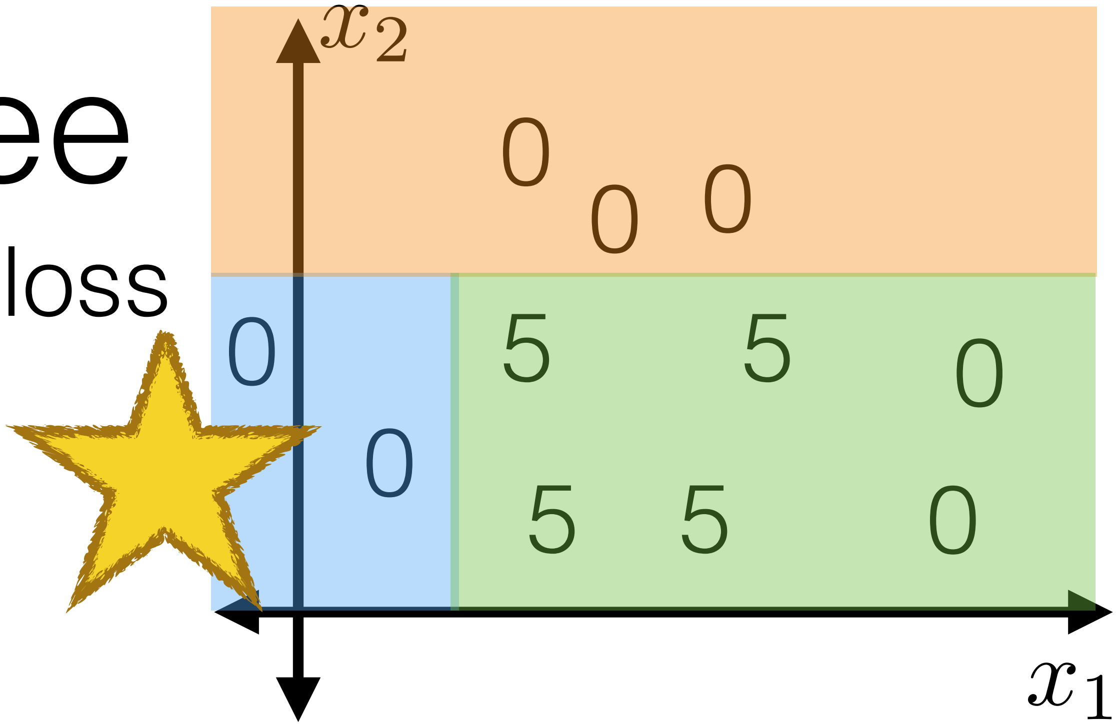
BuildTree({1,…,n};2)



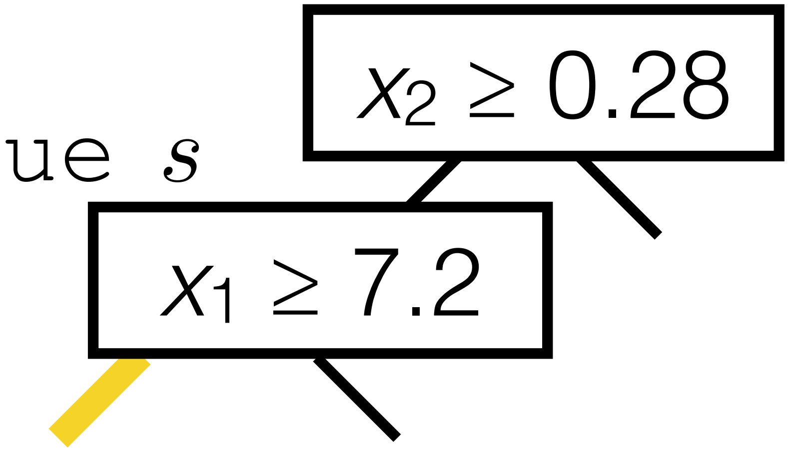
Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$; $k$)
  if $|I| \le$ 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 $I^{+}_{j,s} = \{i \in I | x^{(i)}_j \ge s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^{-}_{j^*,s^*}, k), \text{BuildTree}(I^{+}_{j^*,s^*}, k))$
```



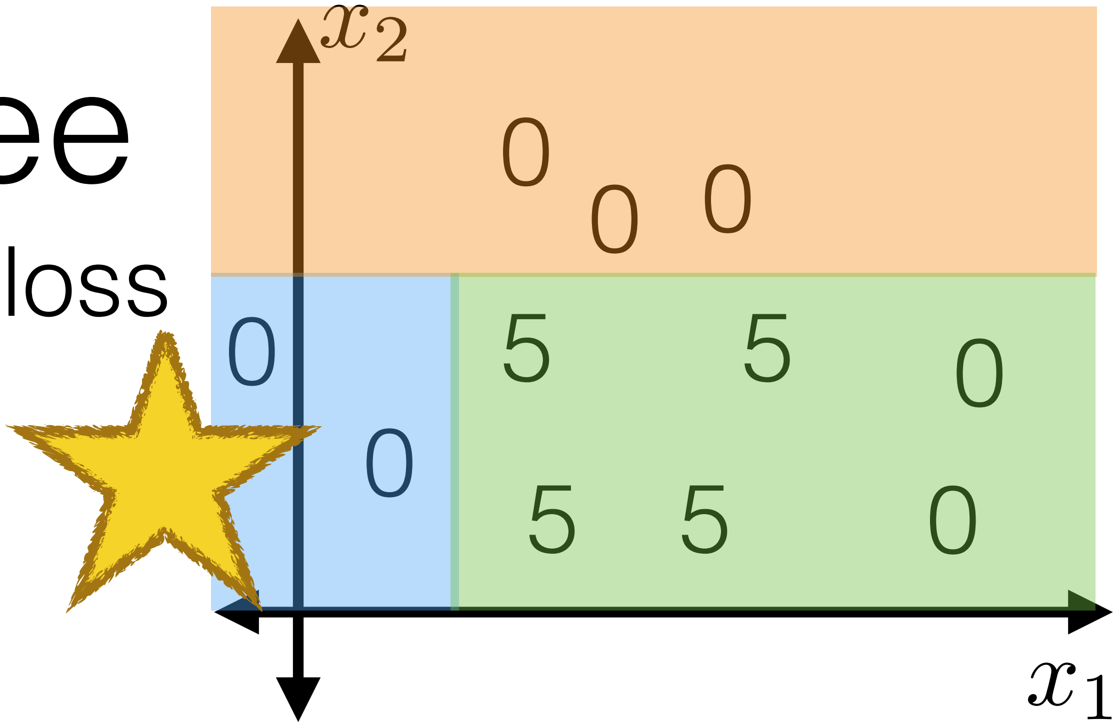
BuildTree({1,…,n};2)



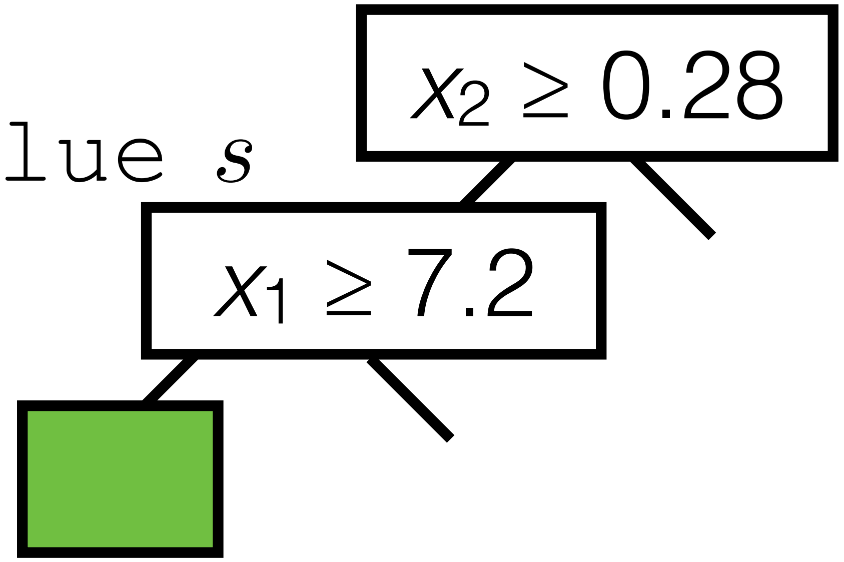
Building a decision tree

- 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```

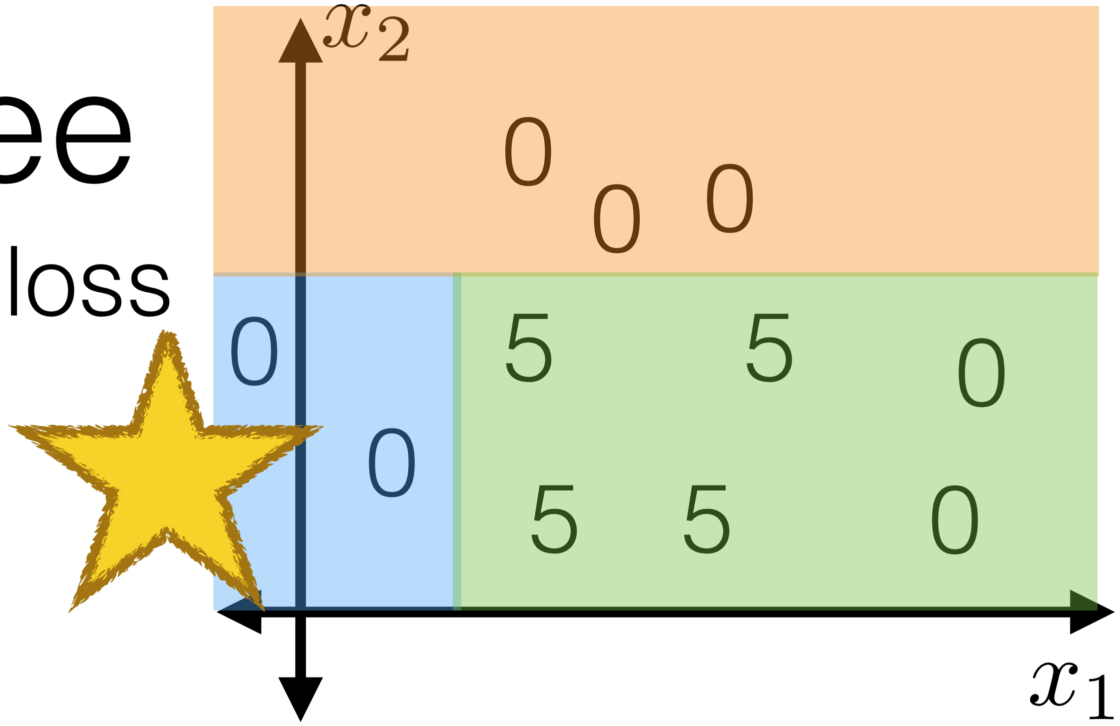


`BuildTree({1,…,n};2)`


$x_2 \geq 0.28$
$x_1 \geq 7.2$

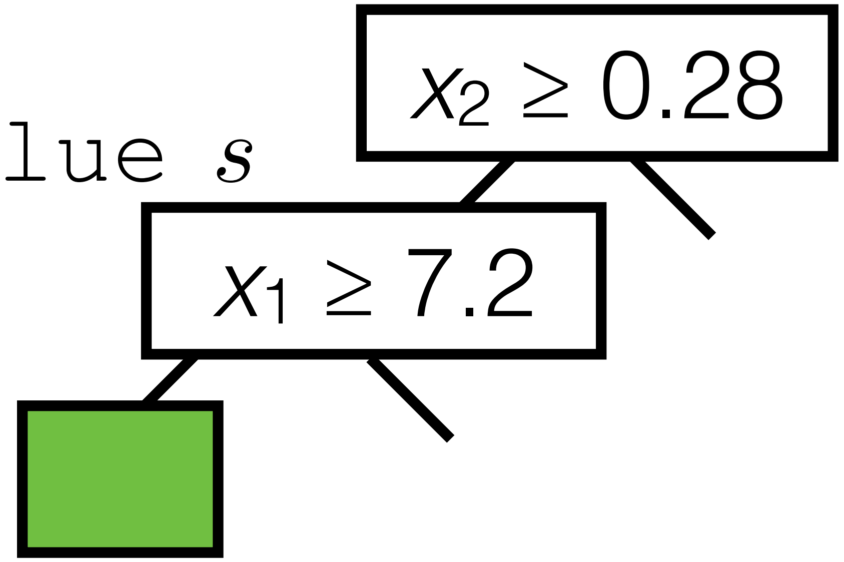
Building a decision tree

- Regression tree with squared error loss



```
BuildTree($I; k$)
  if |$I$| ≤ 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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node($j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```

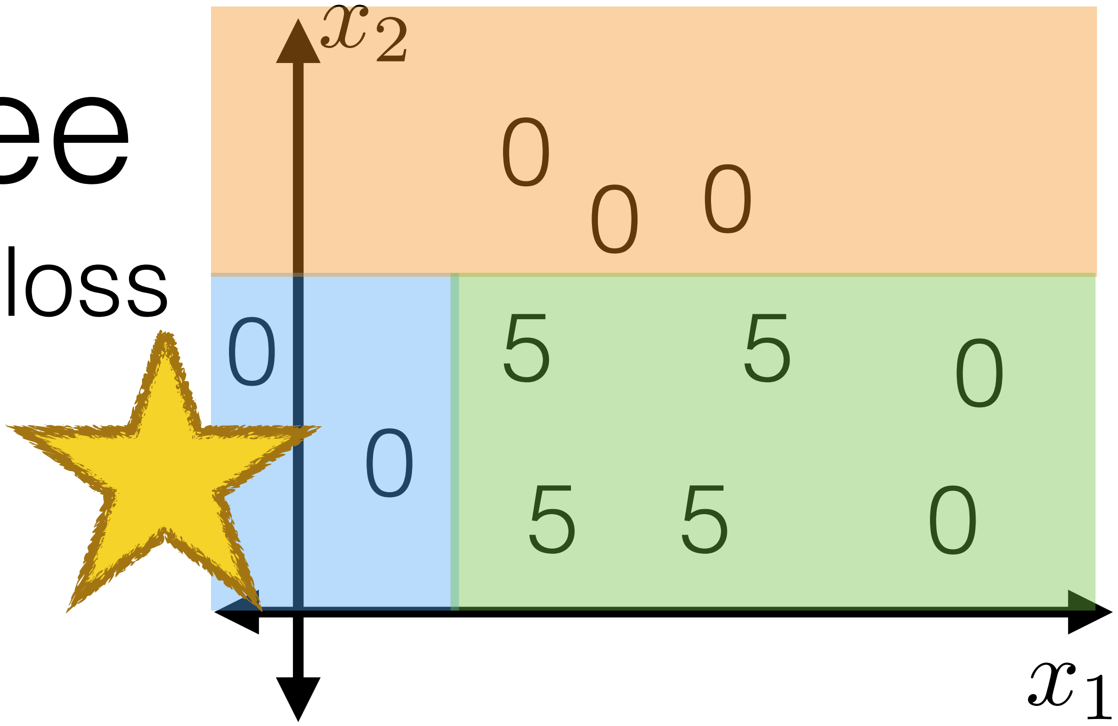
BuildTree({1,…,n};2)



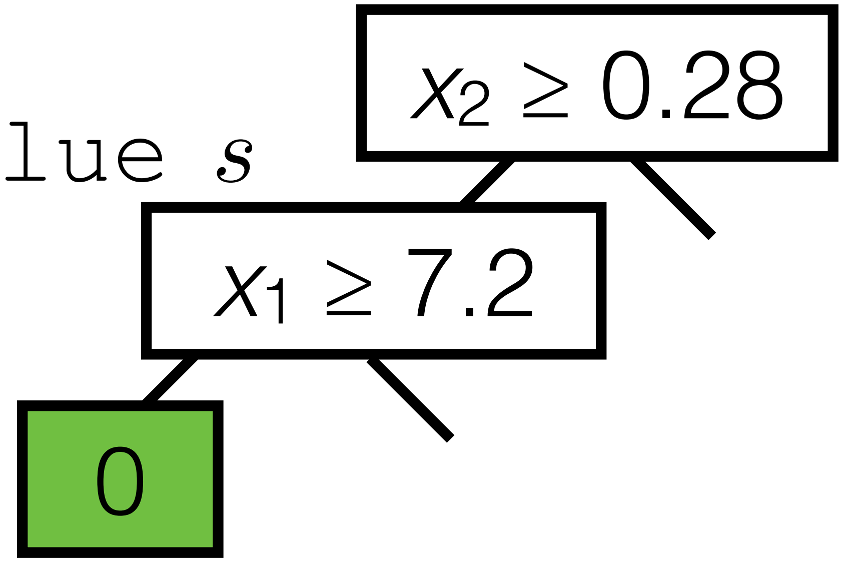
Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I; k$)
  if |$I$| ≤ 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 $I^{+}_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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$(j^*, s^*, \text{BuildTree}(I^{-}_{j^*,s^*}, k), \text{BuildTree}(I^{+}_{j^*,s^*}, k))$
```



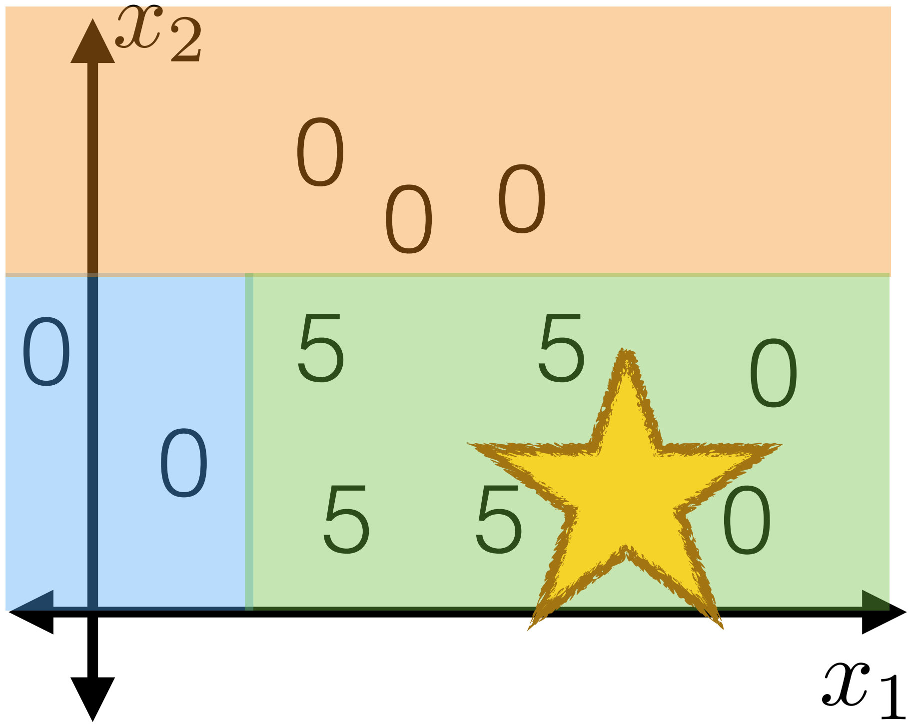
BuildTree({1,…,n};2)



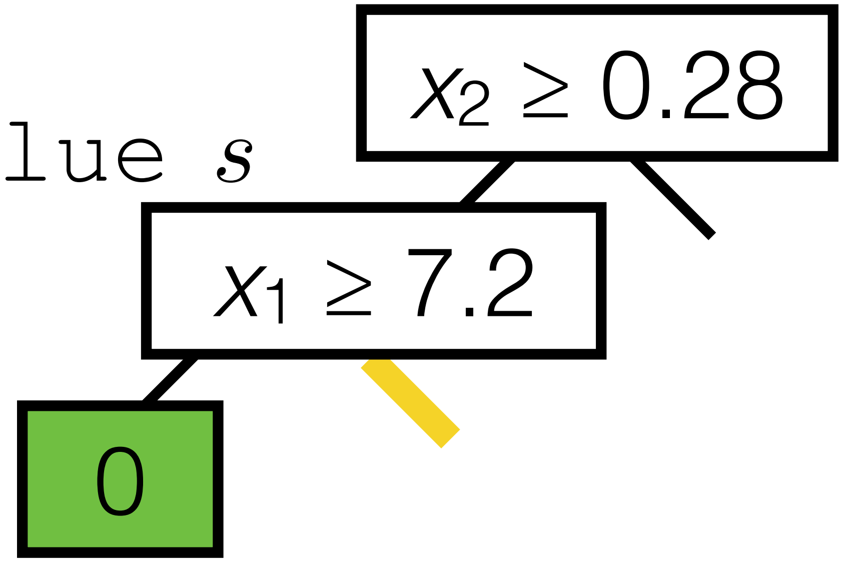
Building a decision tree

- 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```



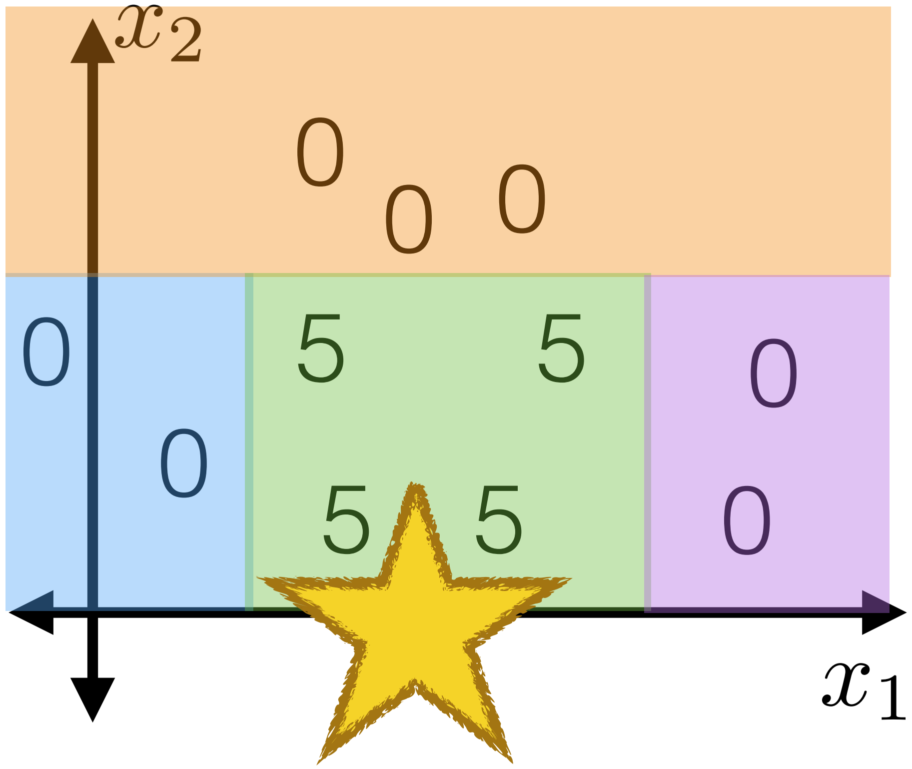
BuildTree({1,…,n};2)



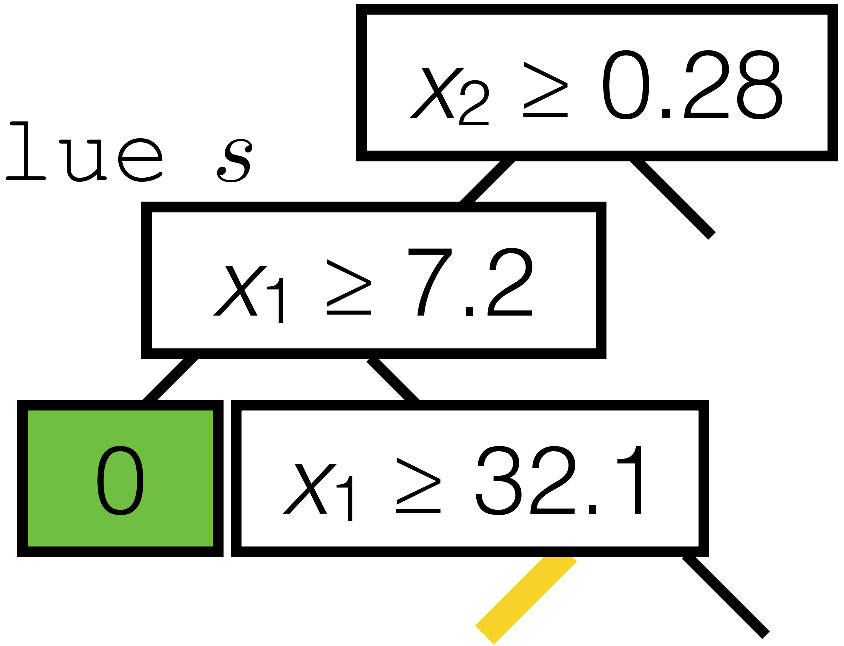
Building a decision tree

- 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 $I^{+}_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^{-}_{j^*,s^*}, k), \text{BuildTree}(I^{+}_{j^*,s^*}, k))$
```



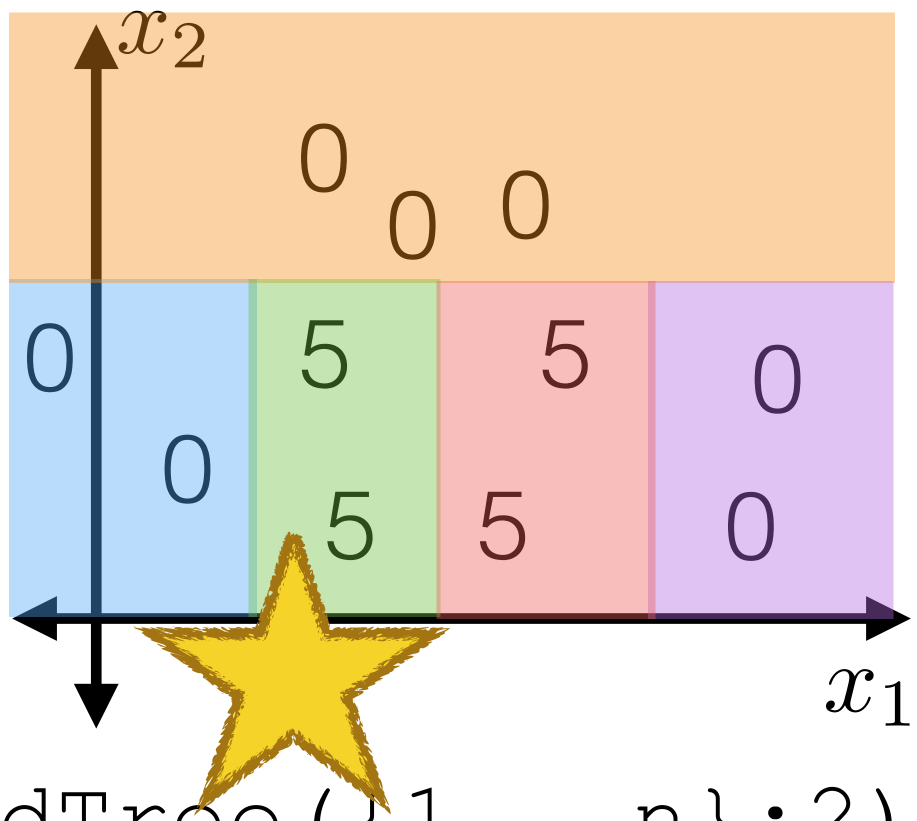
BuildTree({1,…,n};2)



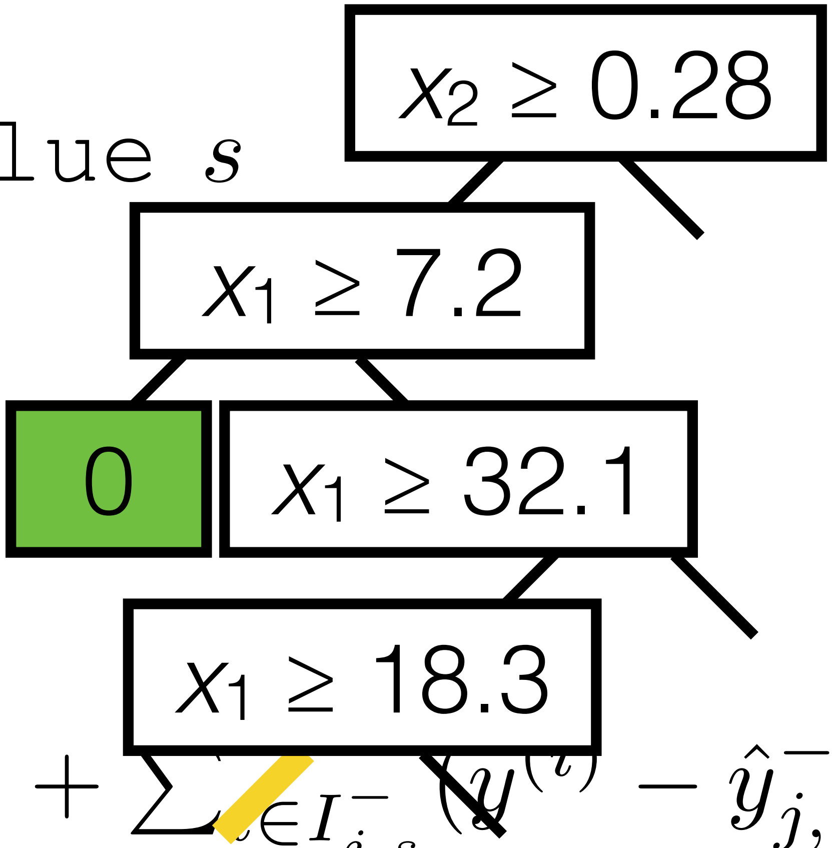
Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I; k$)
  if $|I| \le$ 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 $I^+_{j,s} = \{i \in I | x_j^{(i)} \ge s\}$
      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 $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node($j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```



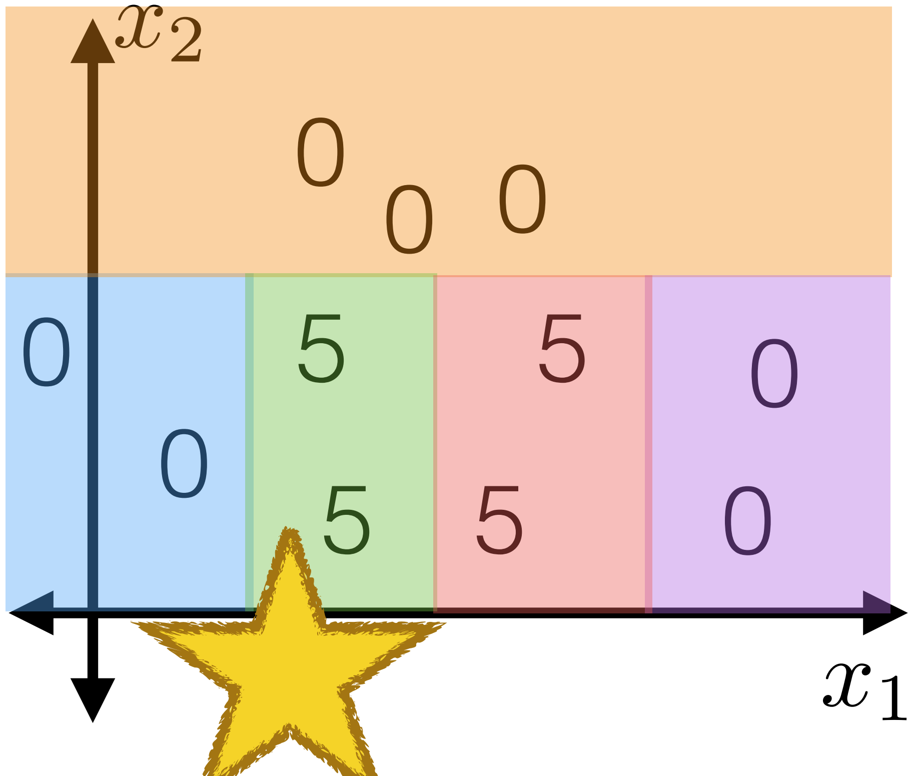
BuildTree({1,…,n};2)



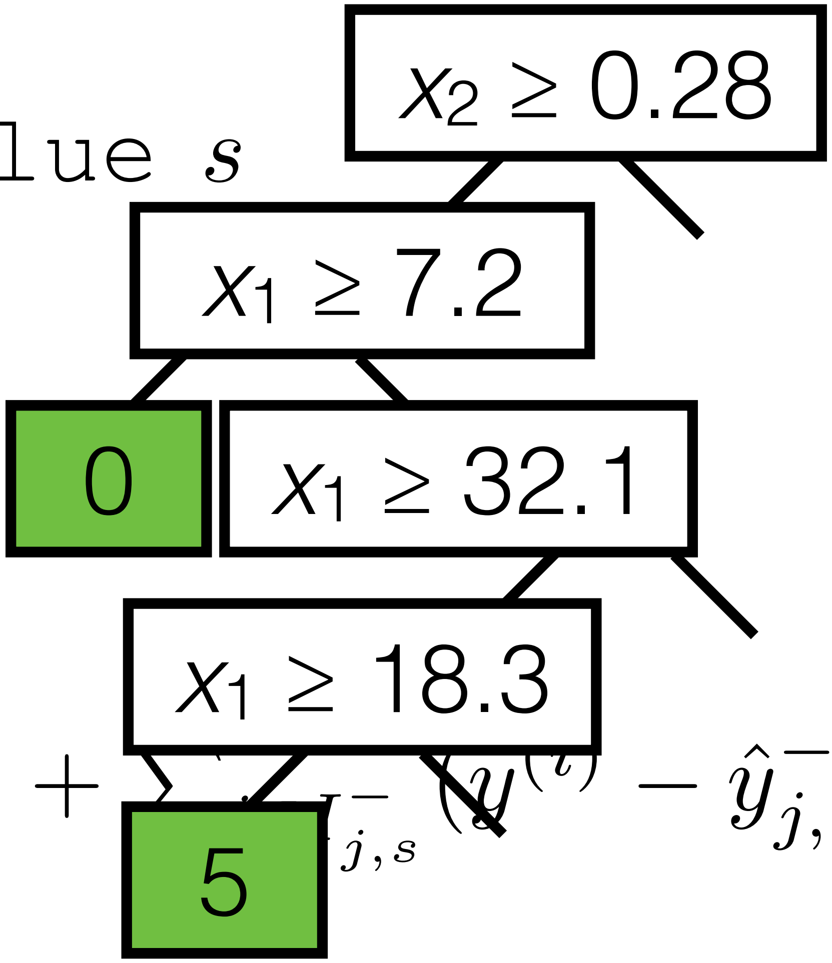
Building a decision tree

- Regression tree with squared error loss

```
BuildTree($I$; $k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*$, $\text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```



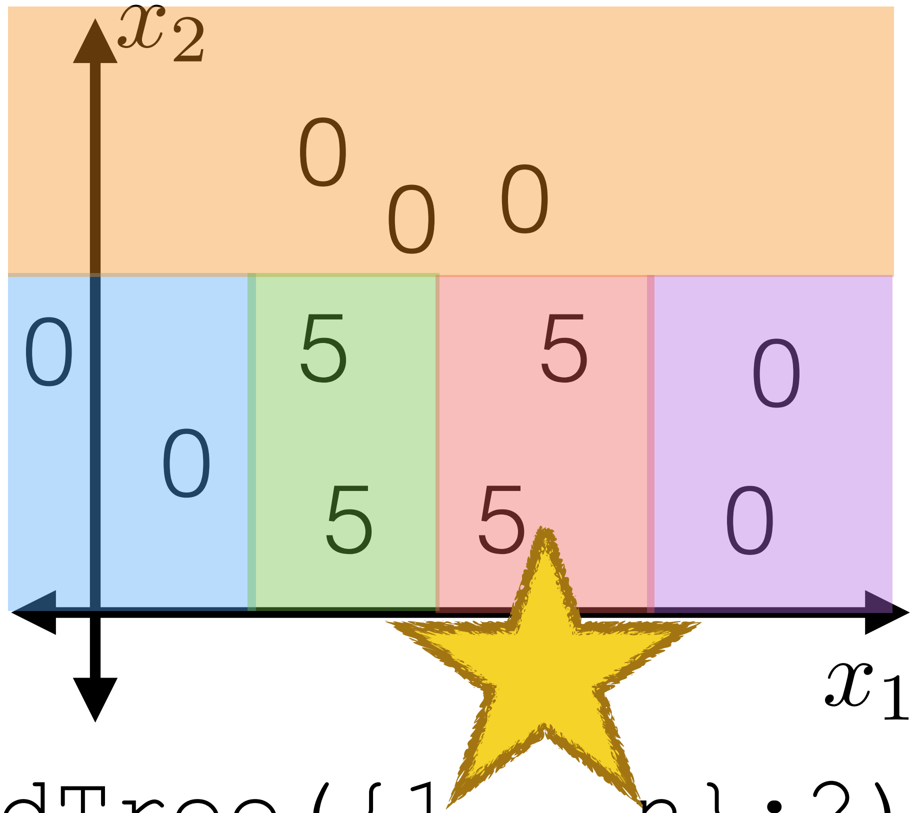
BuildTree({1,…,n};2)



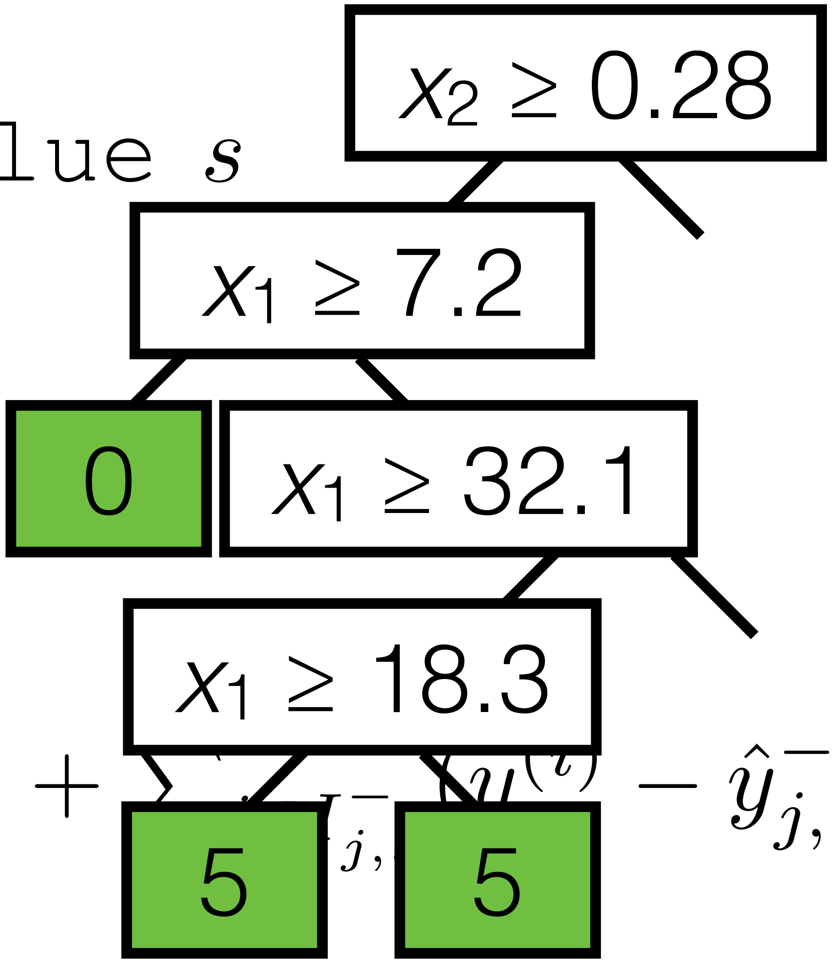
Building a decision tree

- 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```



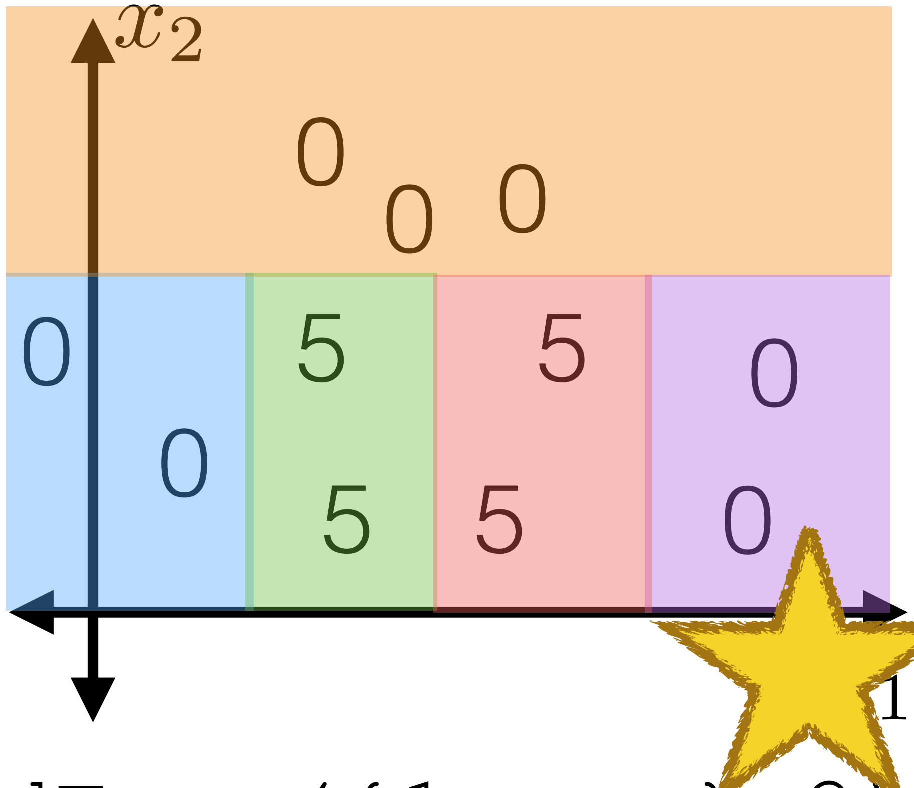
BuildTree({1,…,n};2)



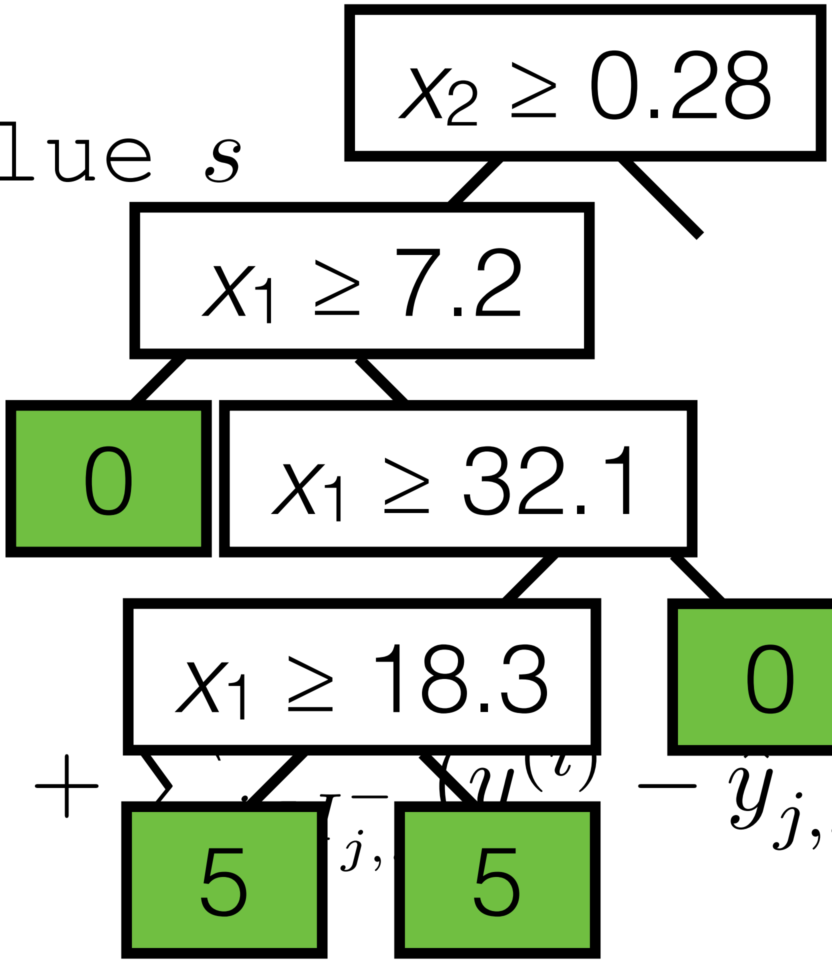
Building a decision tree

- 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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($j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k)$)
```



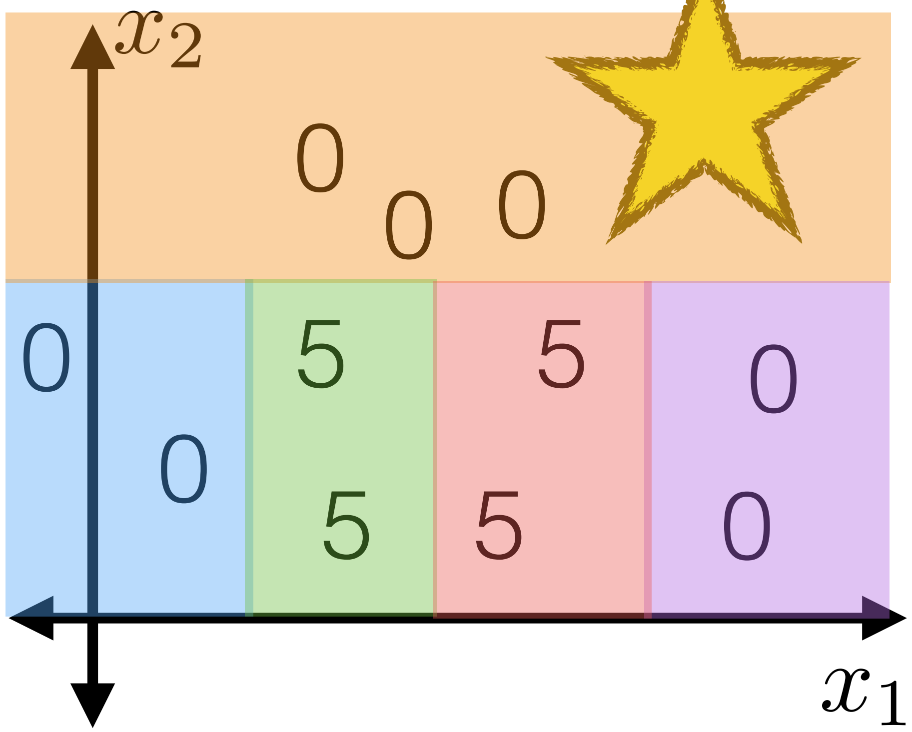
BuildTree({1,…,n};2)



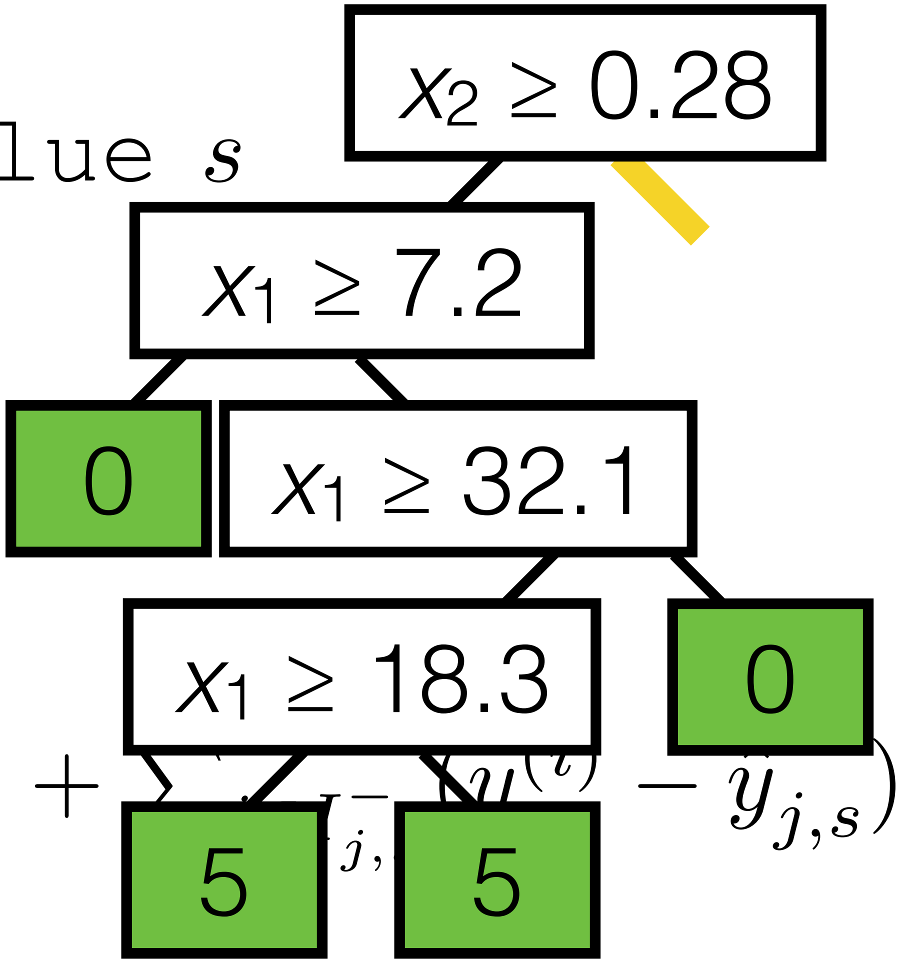
Building a decision tree

- Regression tree with squared error loss

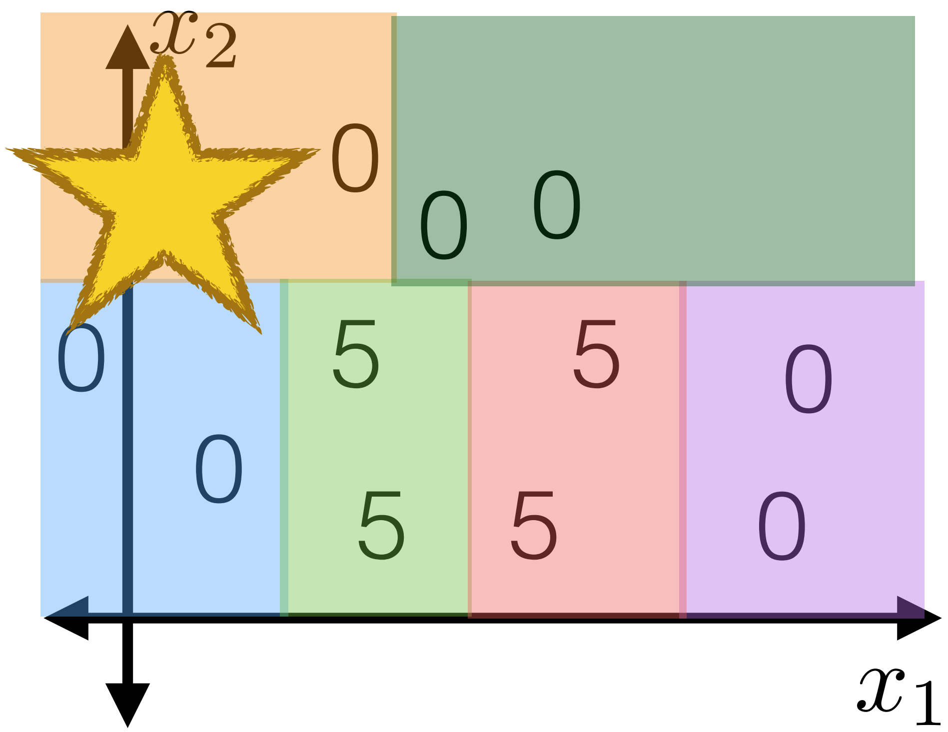
```
BuildTree($I$; $k$)
  if $|I| \leq k$
    Set $\hat{y}$ = $\text{average}_{i \in I} y^{(i)}$
    return Leaf(label = $\hat{y}$)
  else
    for each split dim $j$ & value $s$
      Set $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      Set $\hat{y}^+_{j,s} = \text{average}_{i \in I^+_{j,s}} y^{(i)}$
      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 $(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```



BuildTree({1,…,n};2)



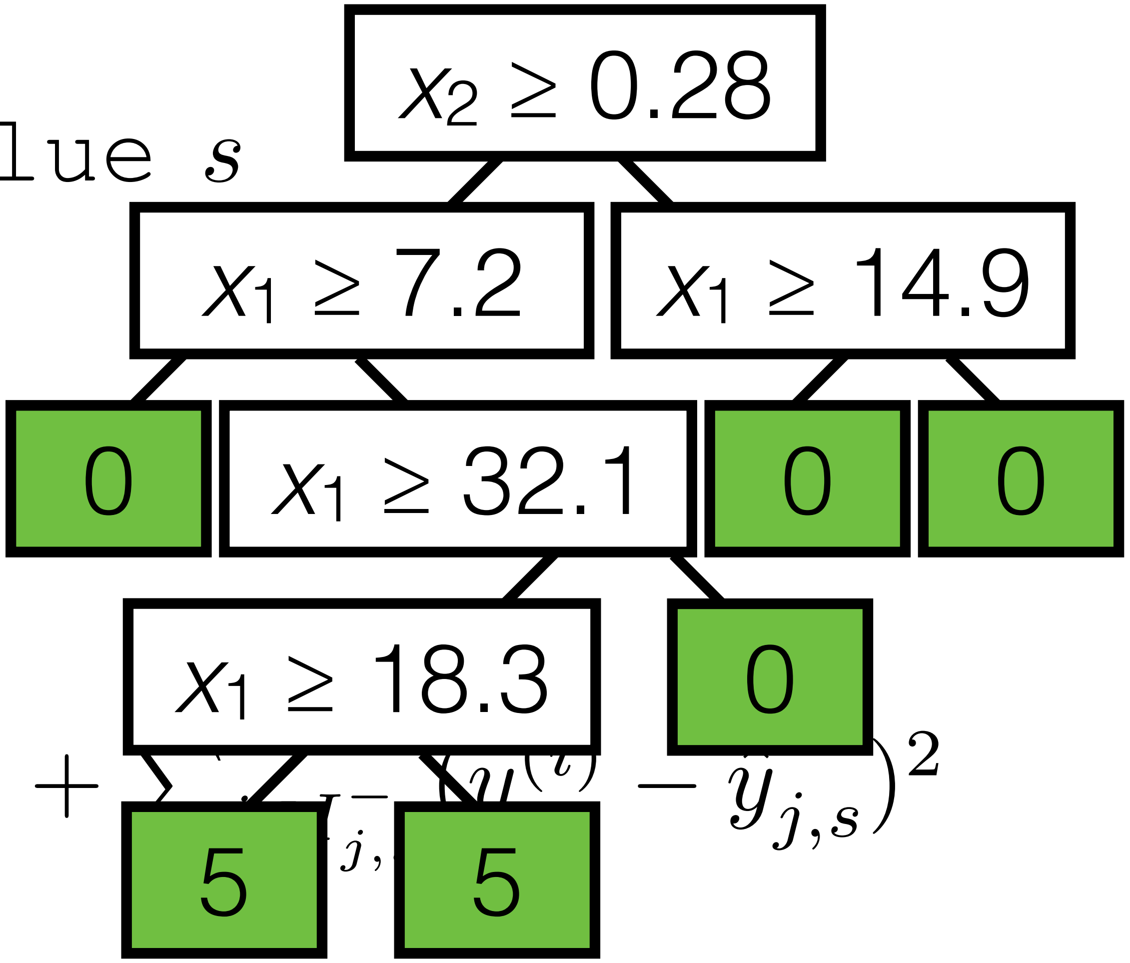
Building a decision tree



- Regression tree with squared error loss

```
BuildTree($I$;$k$)
  if |$I$| ≤ 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```

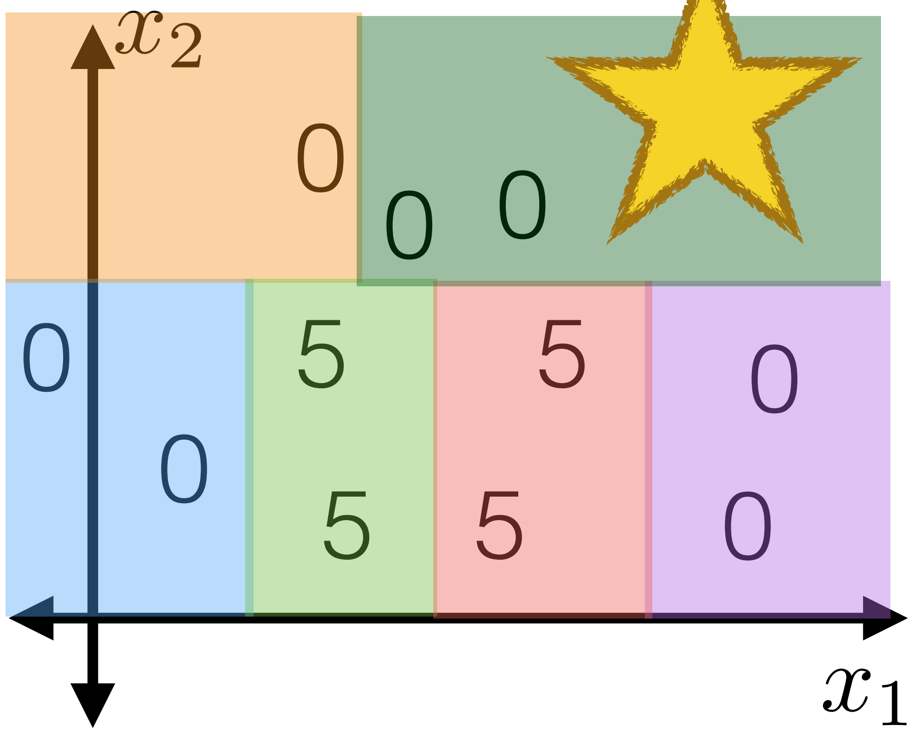
BuildTree({1,…,n};2)


$x_2 \geq 0.28$
$x_1 \geq 7.2$
$x_1 \geq 14.9$
0
$x_1 \geq 32.1$
0
0
$x_1 \geq 18.3$
0
5
5

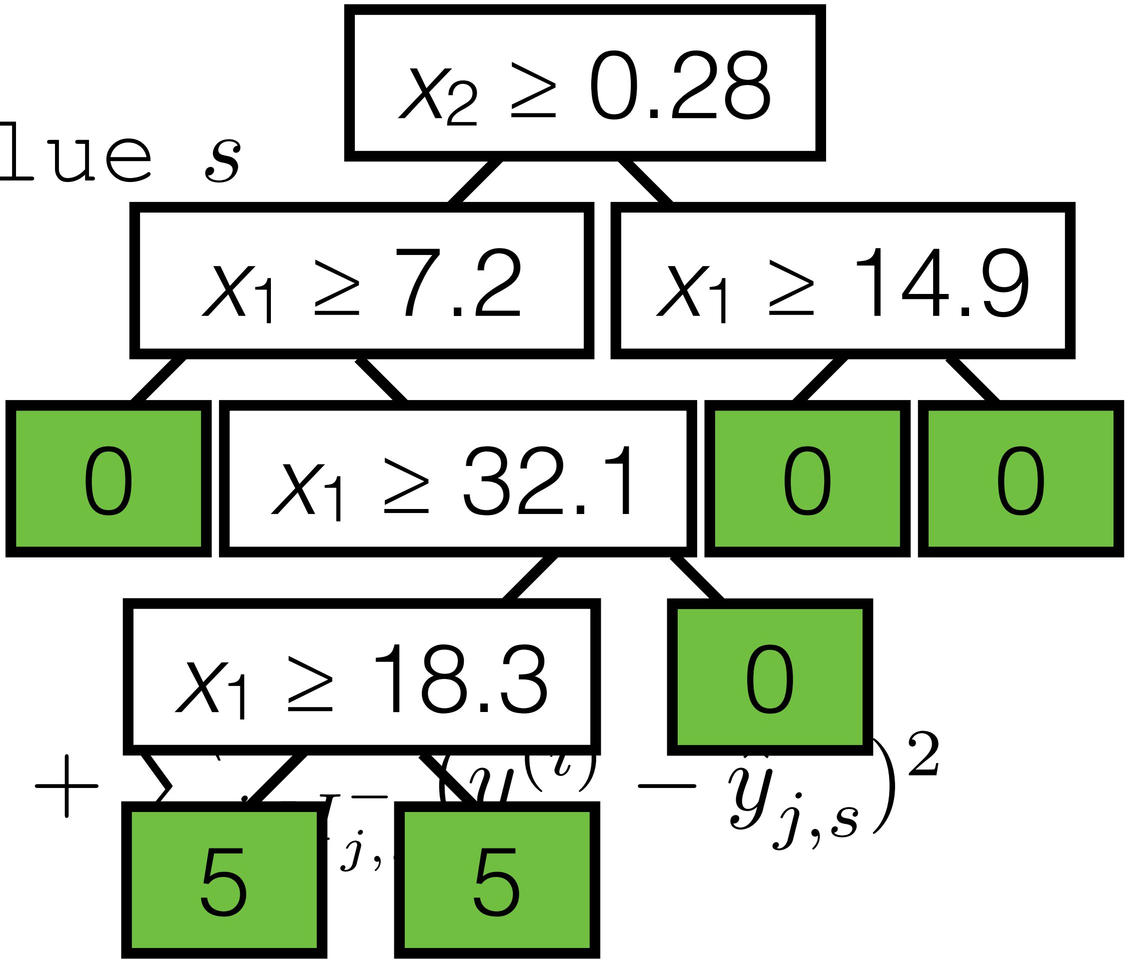
Building a decision tree

- 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 $I^+_{j,s} = \{i \in I | x^{(i)}_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```



BuildTree({1,…,n};2)

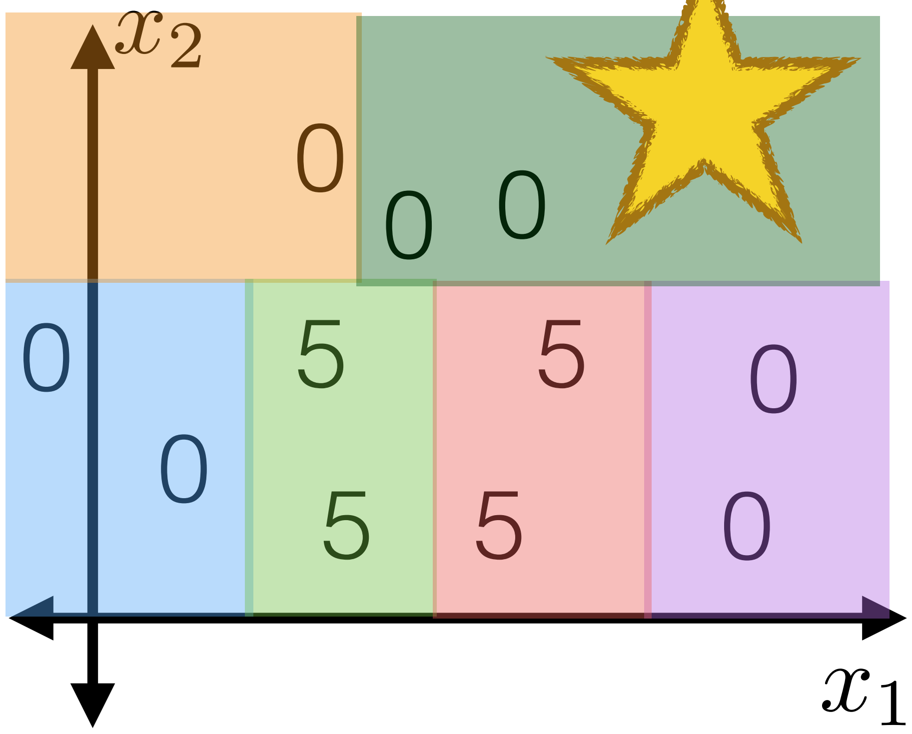


Building a decision tree

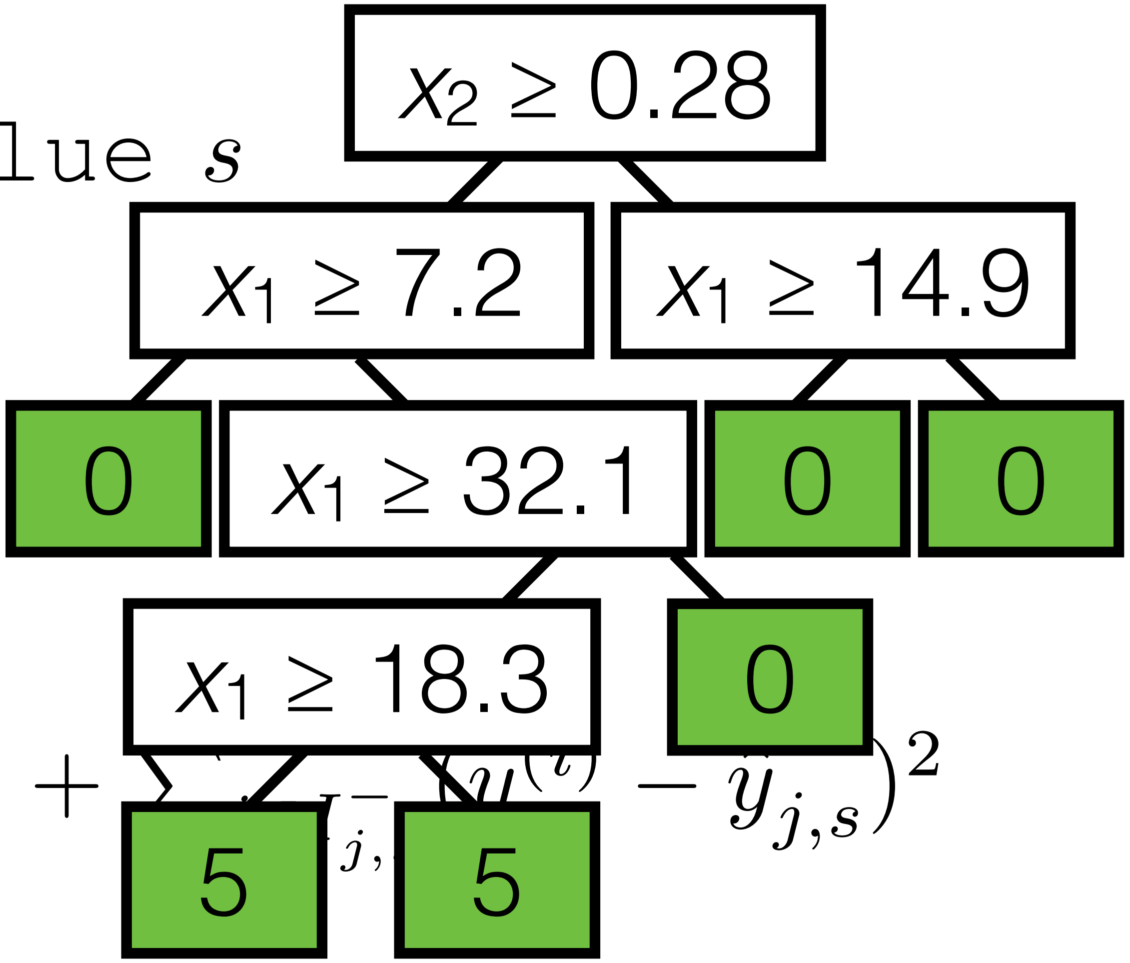
- Regression tree with squared error loss

```
BuildTree($I; k$)
  if $|I| <$
      (label = $\hat{y}$ )
  else
    for each split dim $j$ & value $s$
      Set $I^{+}_{j,s} = \{i \in I | x^{(i)}_{j} \geq s\}$
      Set $I^{-}_{j,s} = \{i \in I | x^{(i)}_{j} < s\}$
      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 $(j^*, s^*, \text{BuildTree}(I^{-}_{j^*,s^*}, k), \text{BuildTree}(I^{+}_{j^*,s^*}, k))$
```

Why keep splitting after our error is low?

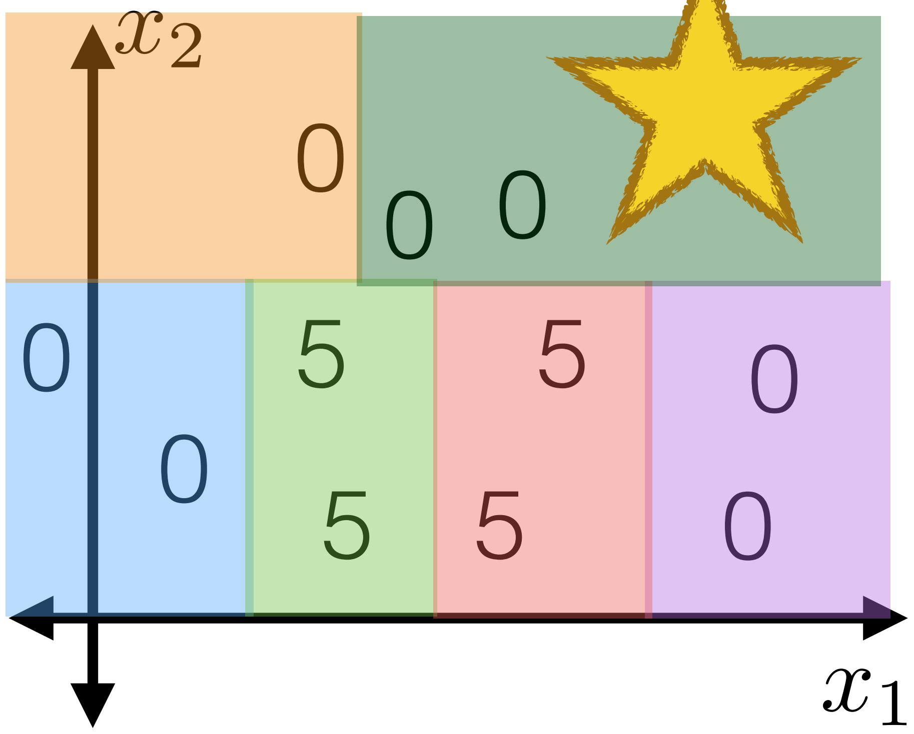


BuildTree({1,…,n};2)



Building a decision tree

- Regression tree with squared error loss

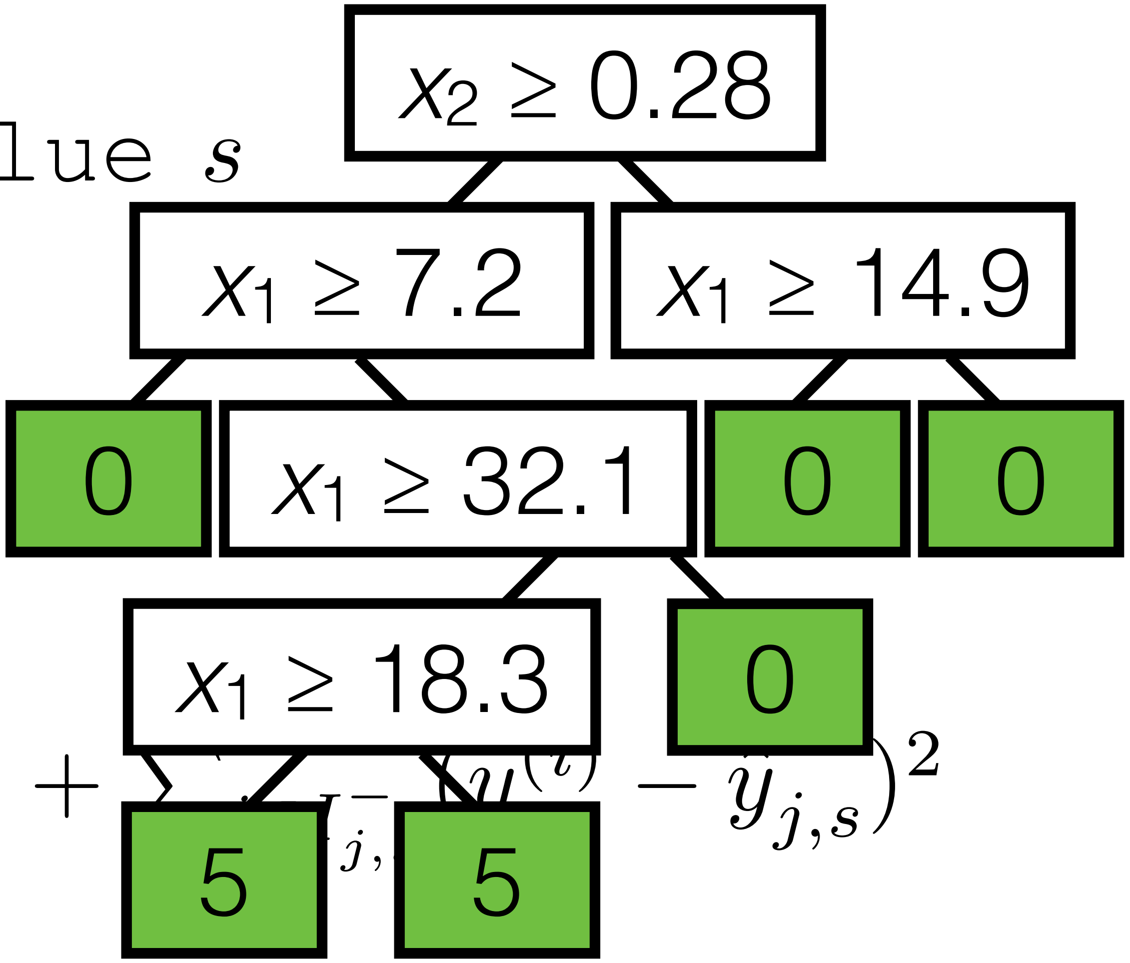


```
BuildTree($I; k$)
  if $|I| <$
                                    )
            (label = $\hat{y}$ )
  else
    f                                m $j$ & value $s$
                              $\geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      Set $\hat{y}^+_{j,s} = \text{average}_{i \in I^+_{j,s}} y^{(i)}$
      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($I^+_{j^*,s^*}, k$))
```

Why keep splitting after our error is low?

Is overfitting an issue?

BuildTree({1,…,n};2)



Building a decision tree

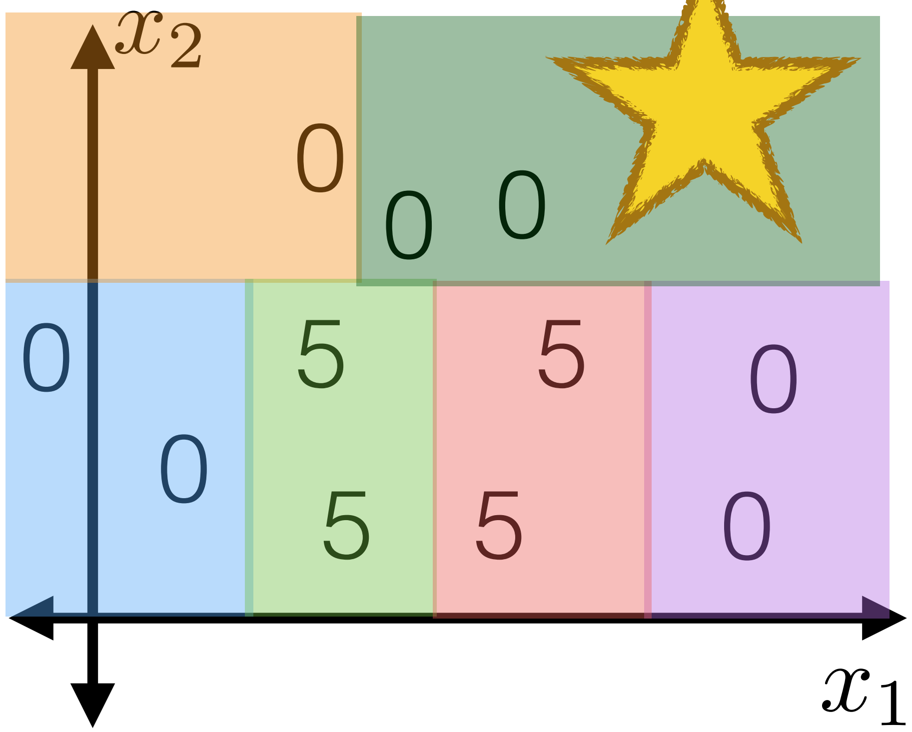
- Regression tree with squared error loss

```
BuildTree($I; k$)
  if $|I| \leq$
                                    )
                  n(label = $\hat{y}$ )
  else
    f                     m $j$ & value $s$
                       $\{x_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      Set $\hat{y}$                               $y^{(i)}$
                                                  $y^{(i)}$
                           $i \in I_{j,s}$
      $E_{j,s} = \sum_{i \in I^+_{j,s}} (y^{(i)} - \hat{y}^+_{j,s})^2 + \sum_{i \in I^-_{j,s}} (y^{(i)} - \hat{y}^-_{j,s})^2$
    Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node $(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```

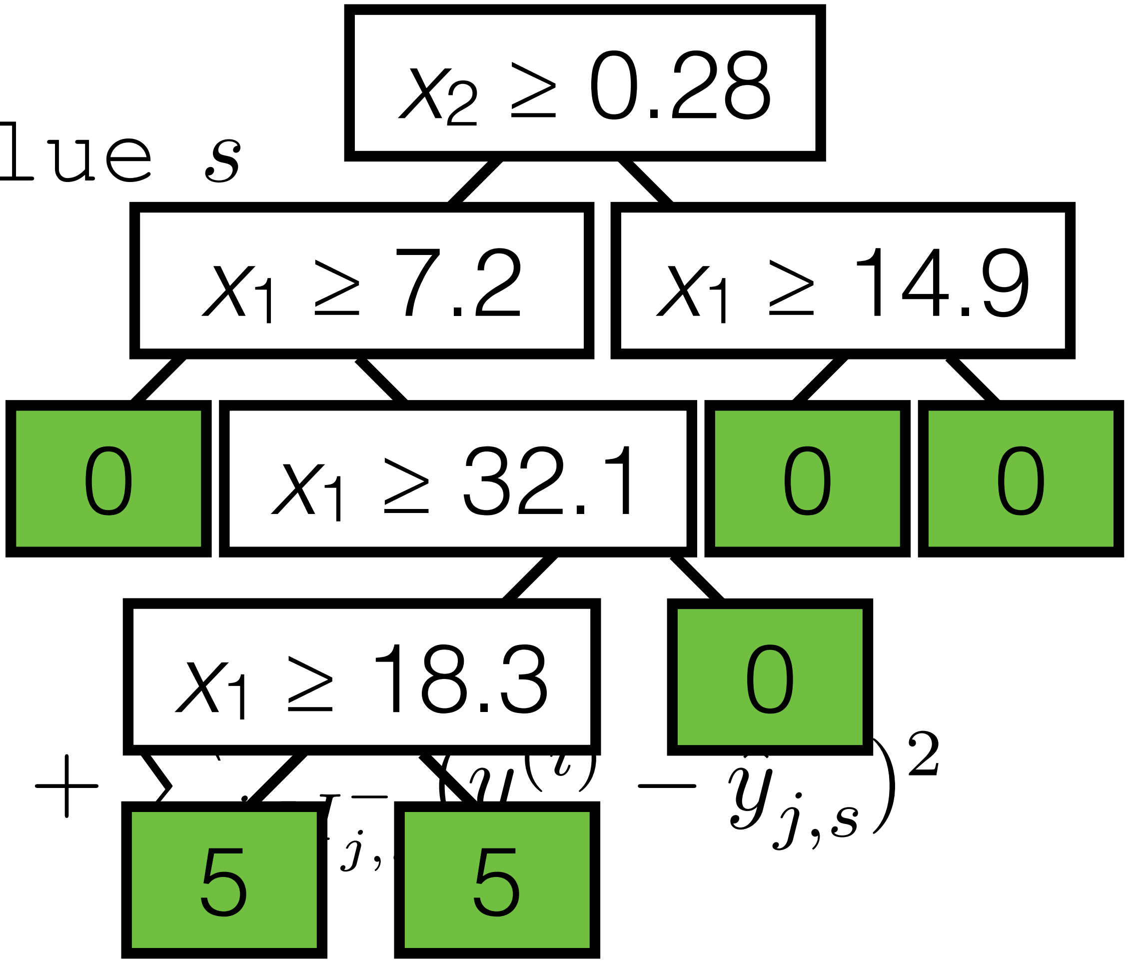
Why keep splitting after our error is low?

Is overfitting an issue?

Idea: stop when no/ small change in loss



BuildTree({1,…,n};2)



Building a decision tree

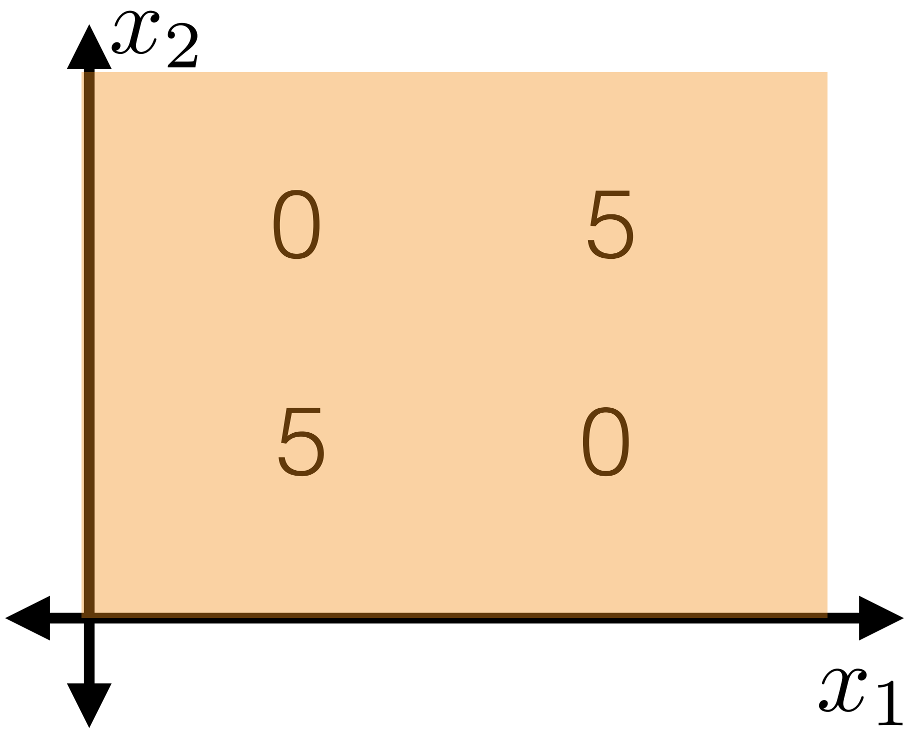
- Regression tree with squared error loss

```
BuildTree($I; k$)
  if $|I| \leq$
      )
      (label $= \hat{y}$ )
  else
    f                  m $j$ & value $s$
                  $| x_j \geq s\}$
      Set $I^-_{j,s} = \{i \in I | x^{(i)}_j < s\}$
      Se                     $y^{(i)}$
                             $y^{(i)}$
        $E_{j,s} = \sum_{i \in I^+_{j,s}} (y^{(i)} - \hat{y}^+_{j,s})^2 + \sum_{i \in I^-_{j,s}} (y^{(i)} - \hat{y}^-_{j,s})^2$
    Set $(j^*, s^*) = \arg\min_{j,s} E_{j,s}$
    return Node $(j^*, s^*, \text{BuildTree}(I^-_{j^*,s^*}, k), \text{BuildTree}(I^+_{j^*,s^*}, k))$
```

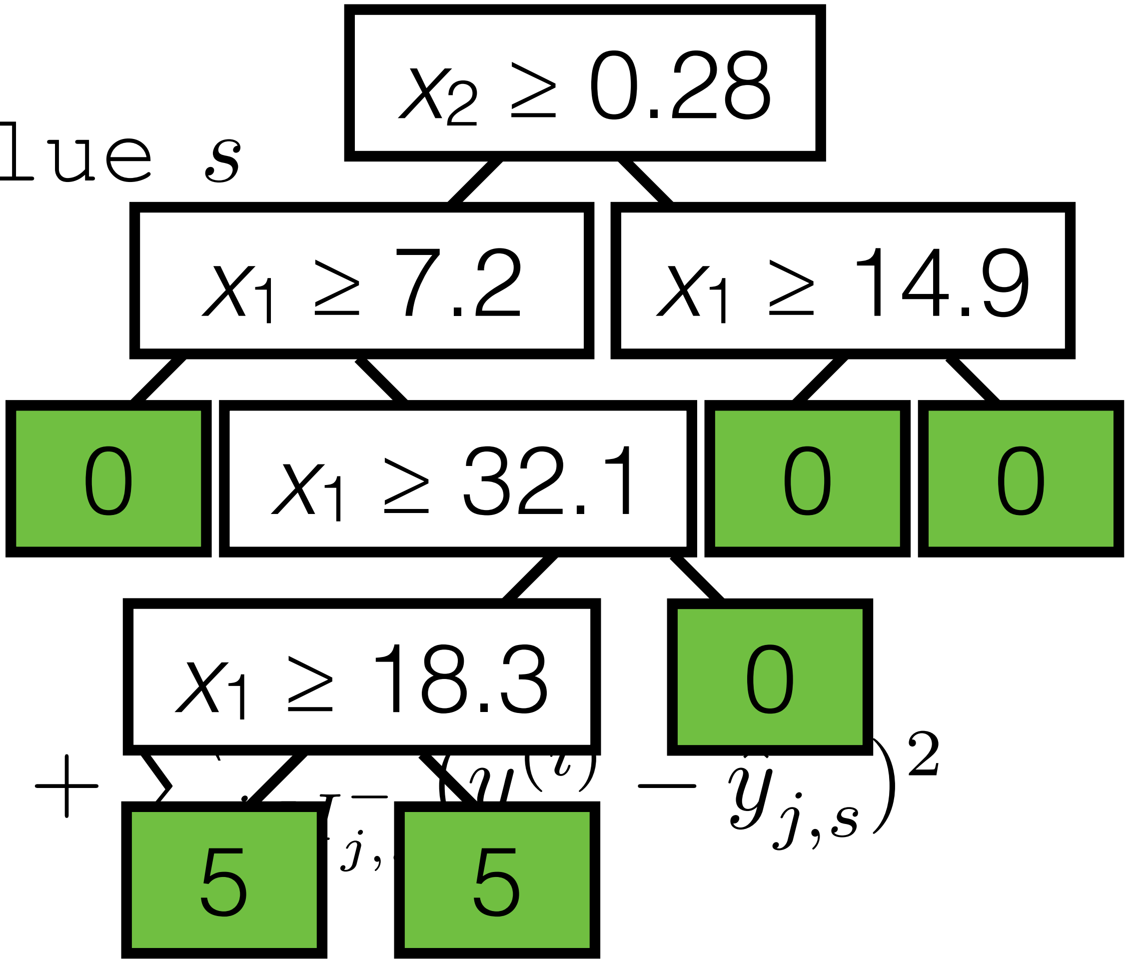
Why keep splitting after our error is low?

Is overfitting an issue?

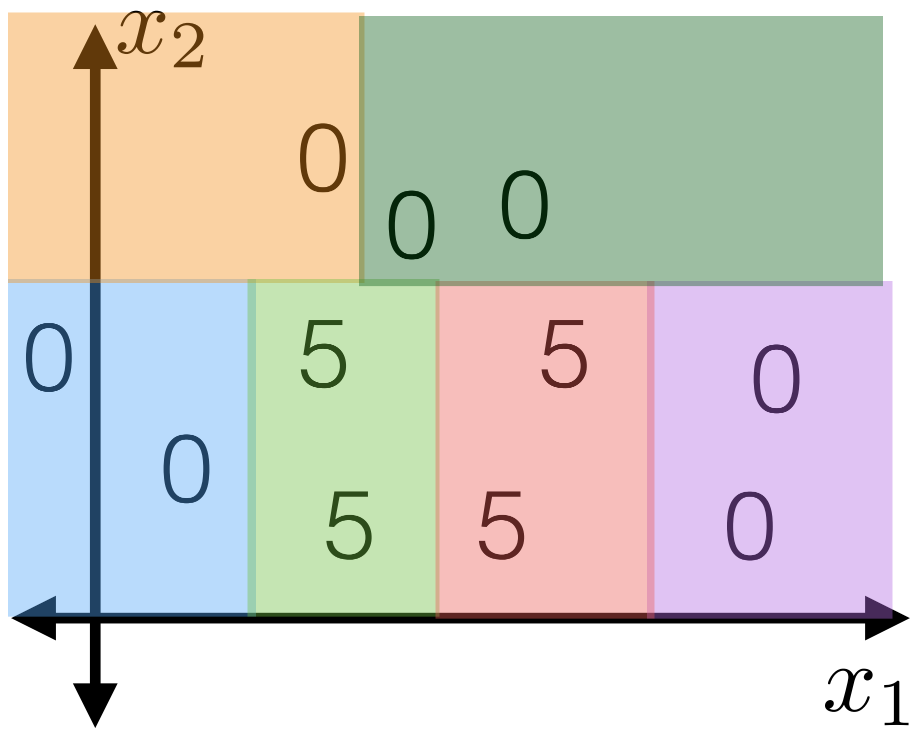
Idea: stop when no/ small change in loss



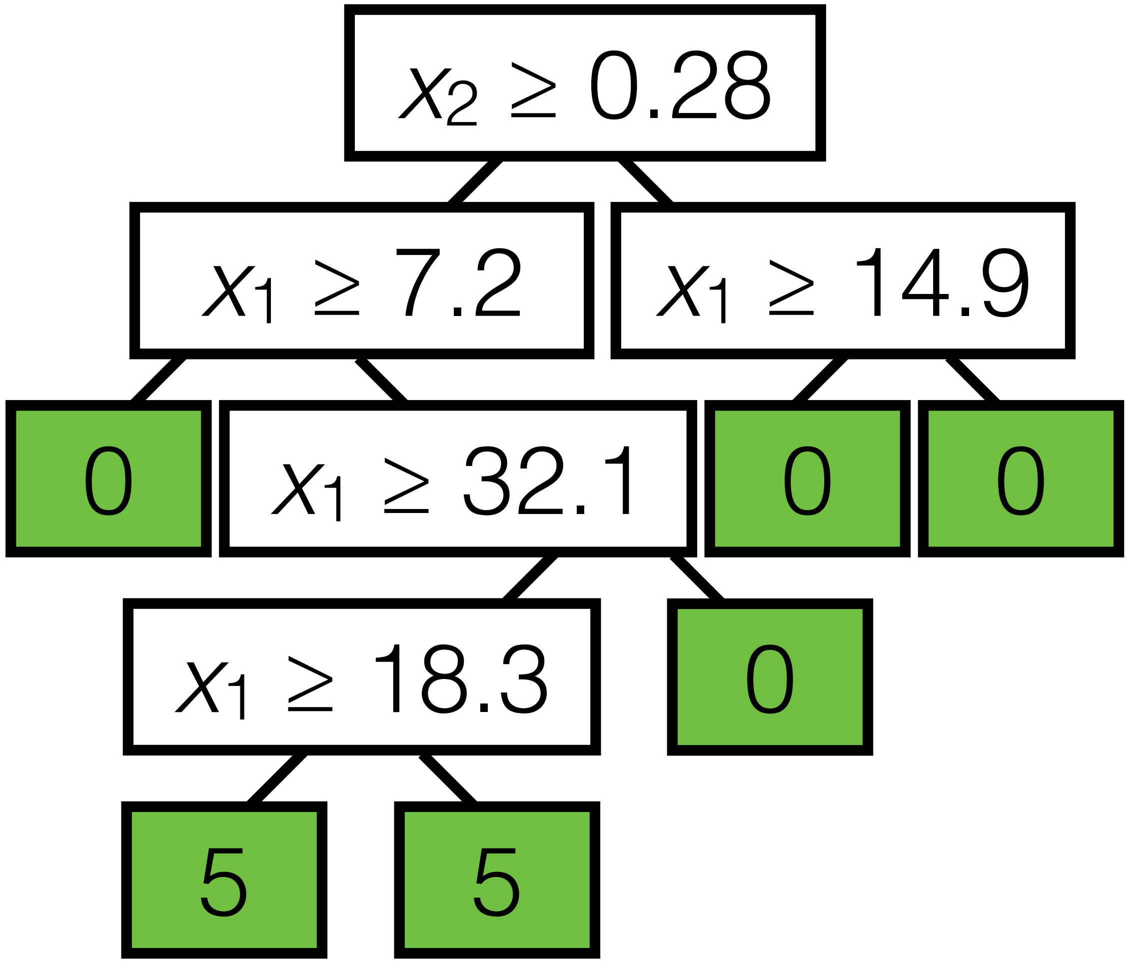
BuildTree({1,…,n};2)



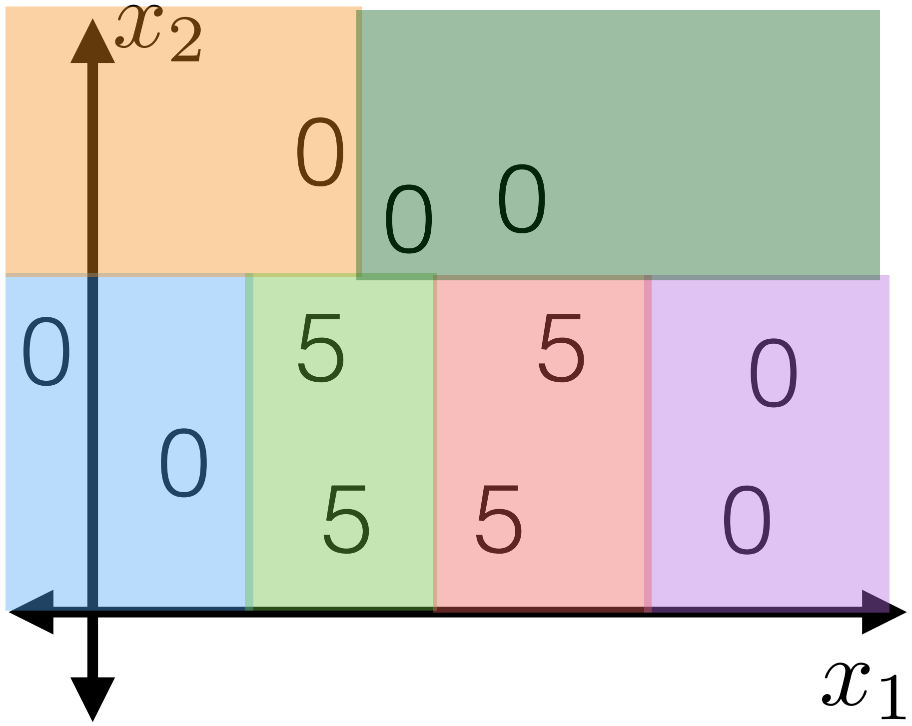
How to regularize?



`BuildTree({1,…,n};2)`

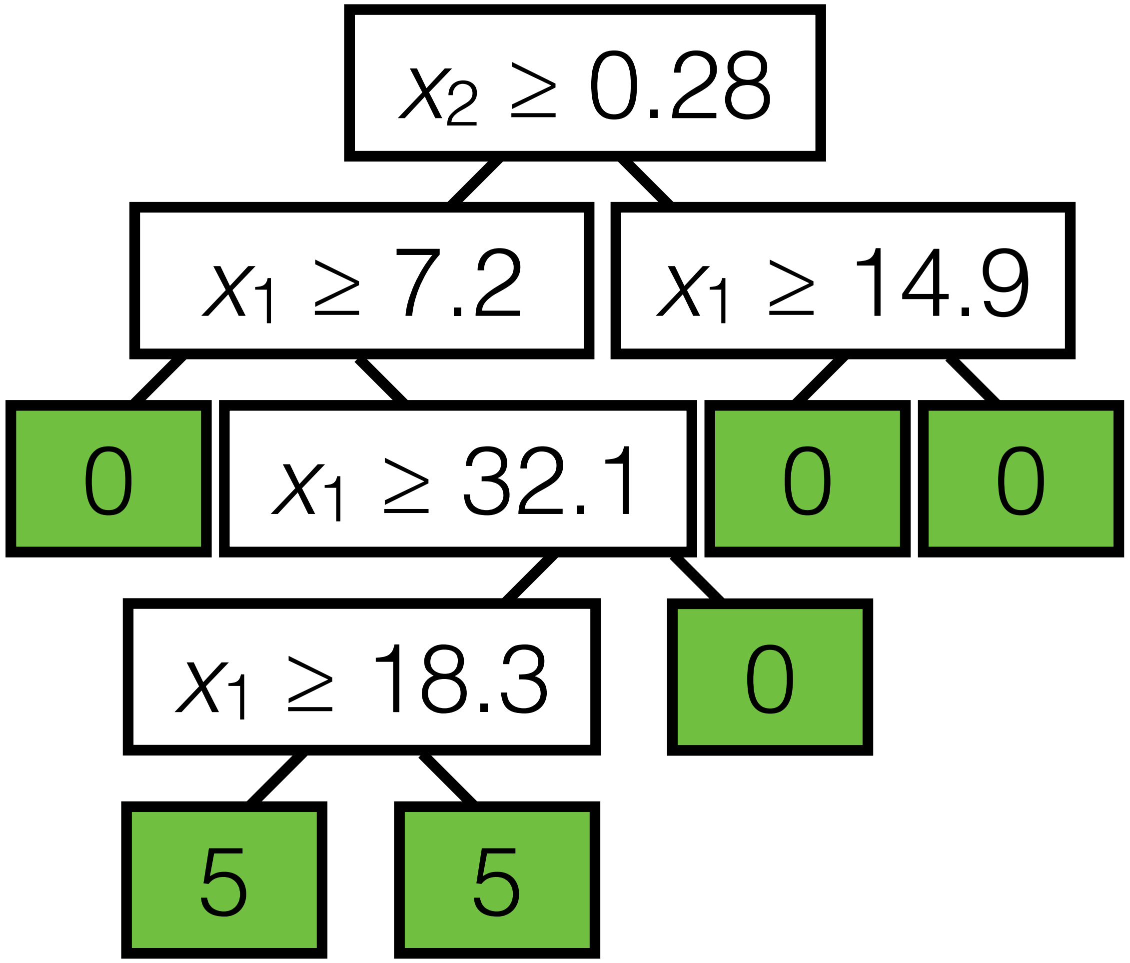


How to regularize?



- Objective = training loss + constant * regularizer

`BuildTree({1,…,n};2)`


$x_2 \geq 0.28$

$x_1 \geq 7.2$

$x_1 \geq 14.9$

0

$x_1 \geq 32.1$

0

0

$x_1 \geq 18.3$

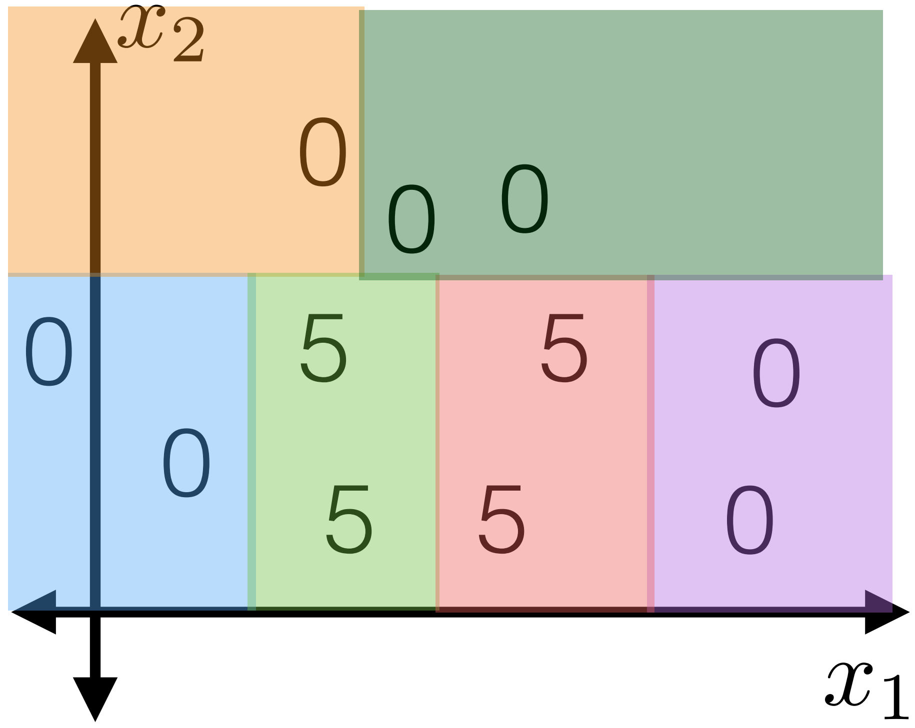
0

5

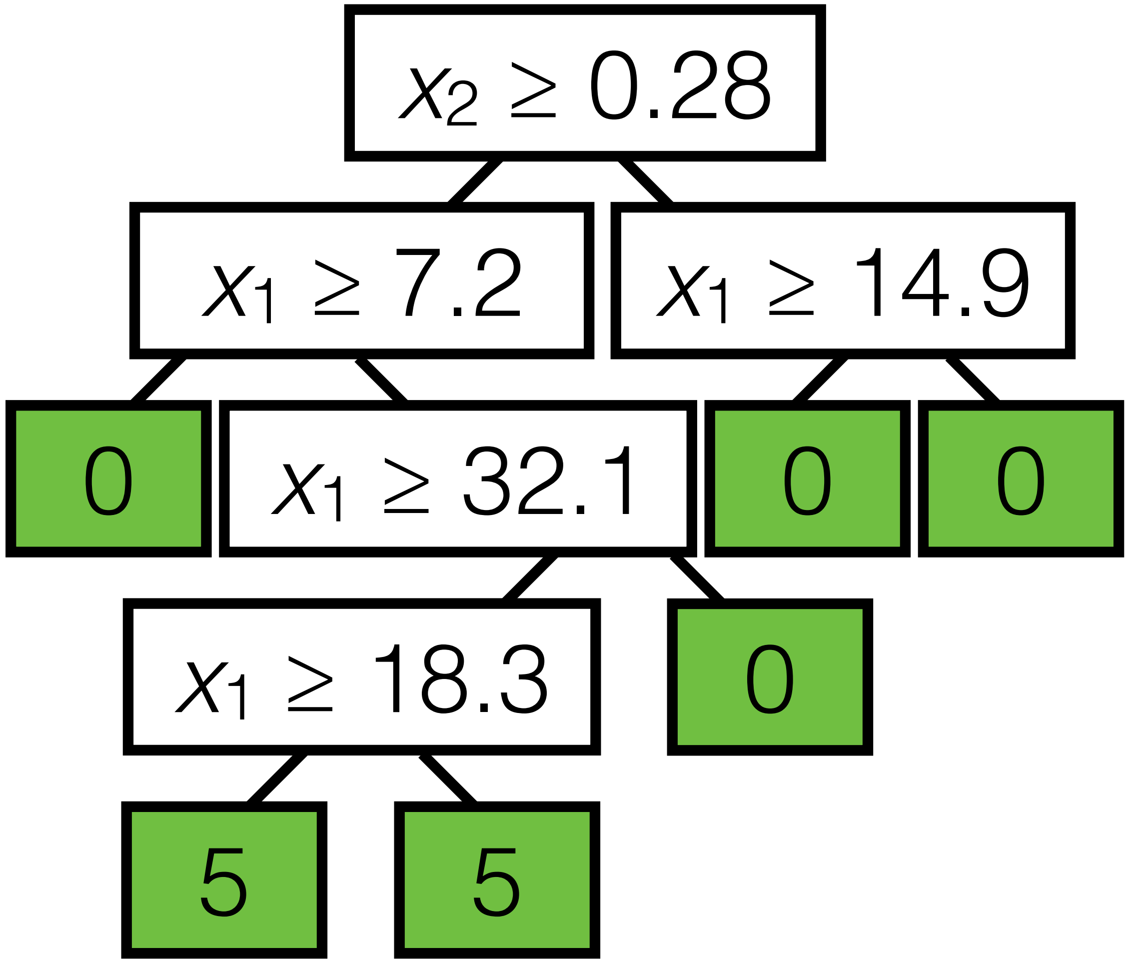
5

How to regularize?

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

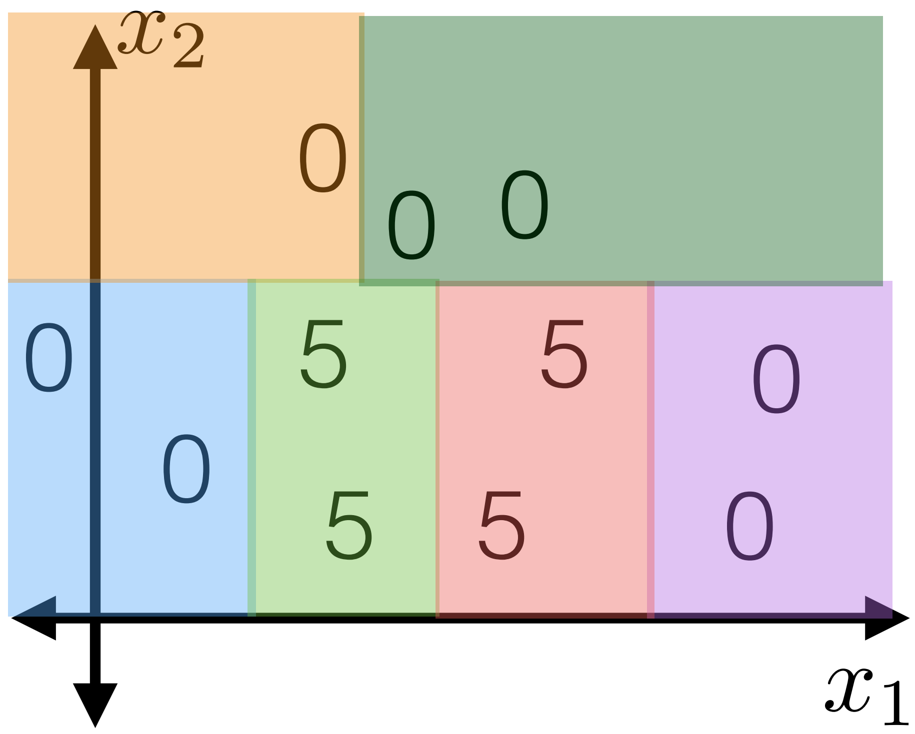


`BuildTree({1,…,n};2)`

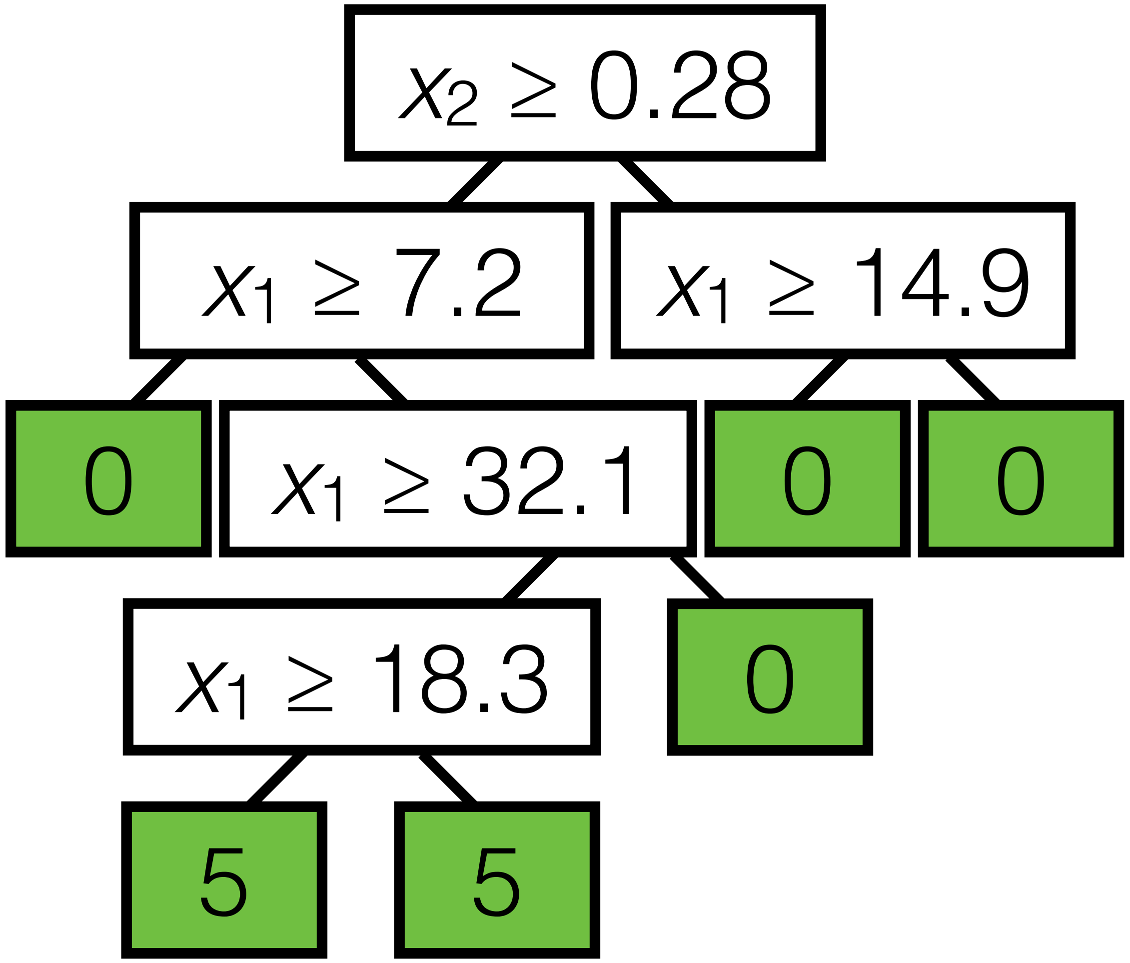


How to regularize?

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

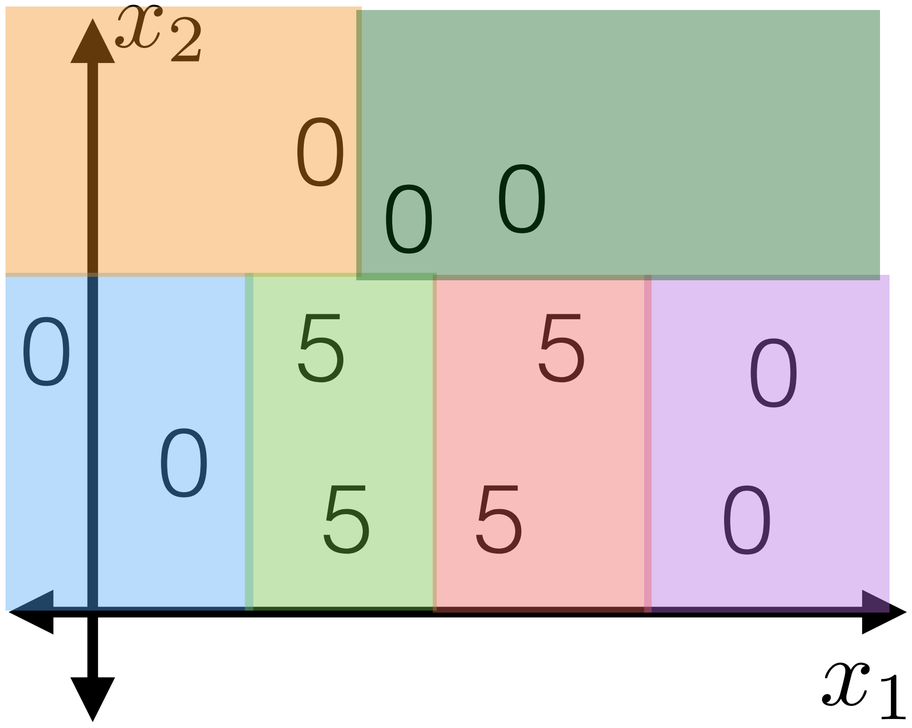


`BuildTree({1,…,n};2)`

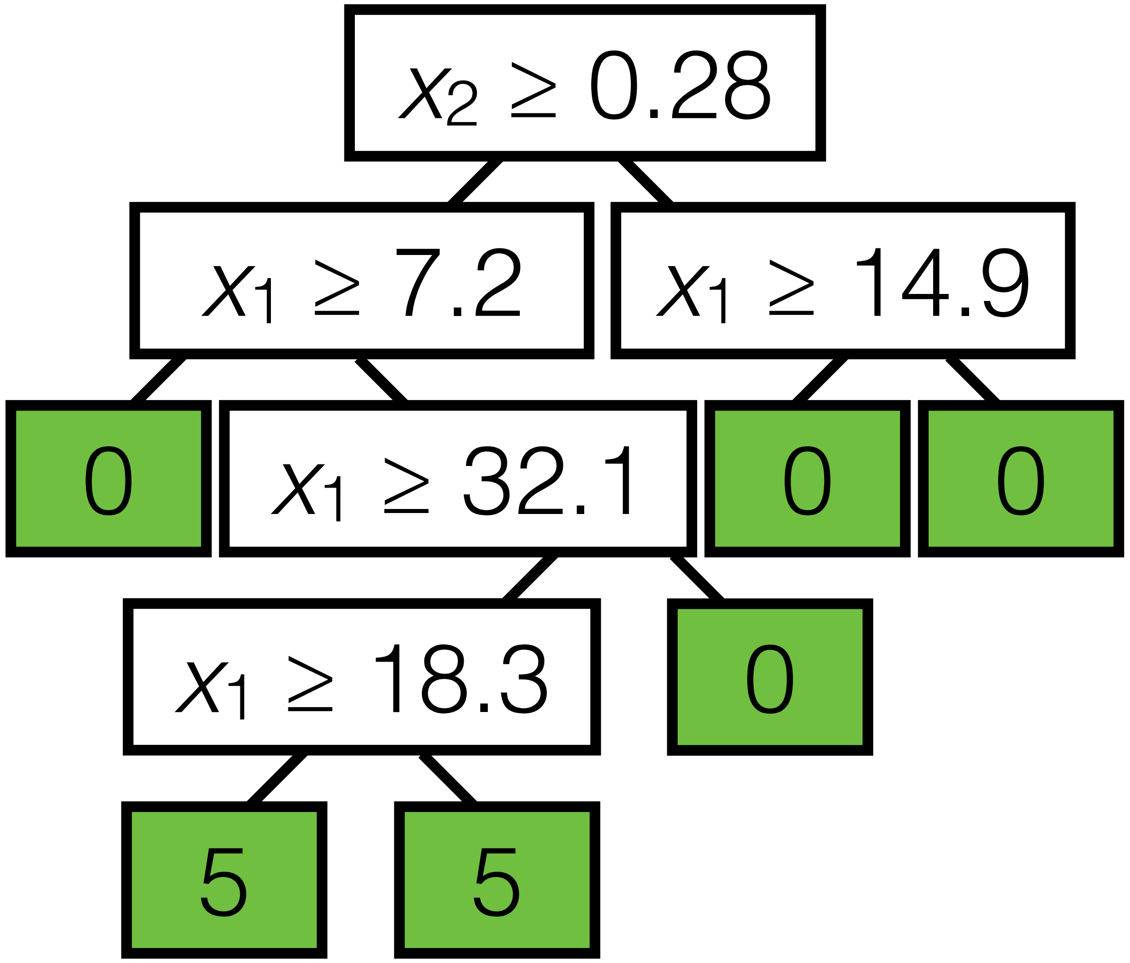


How to regularize?

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

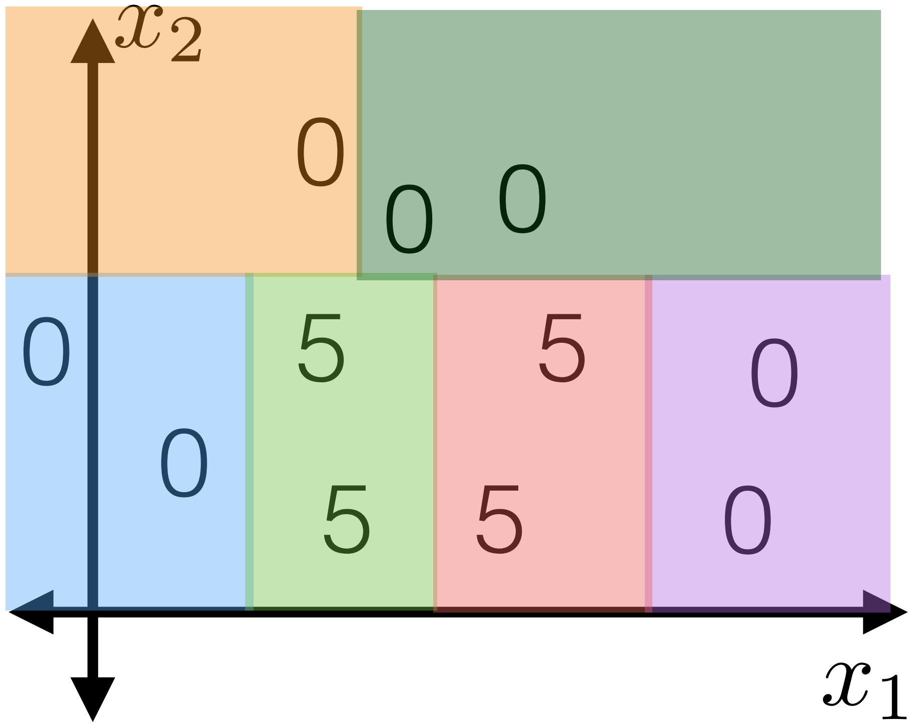


`BuildTree({1,…,n};2)`

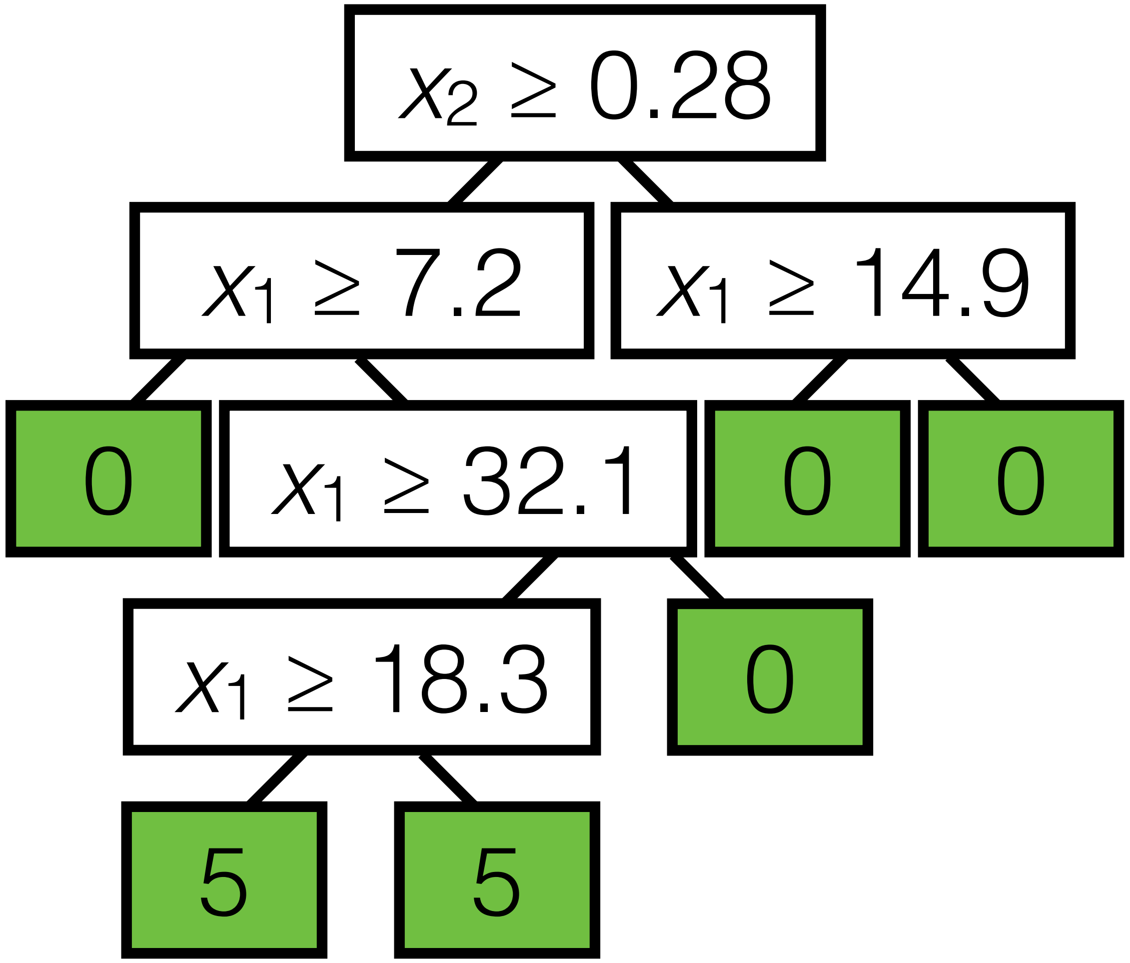


How to regularize?

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

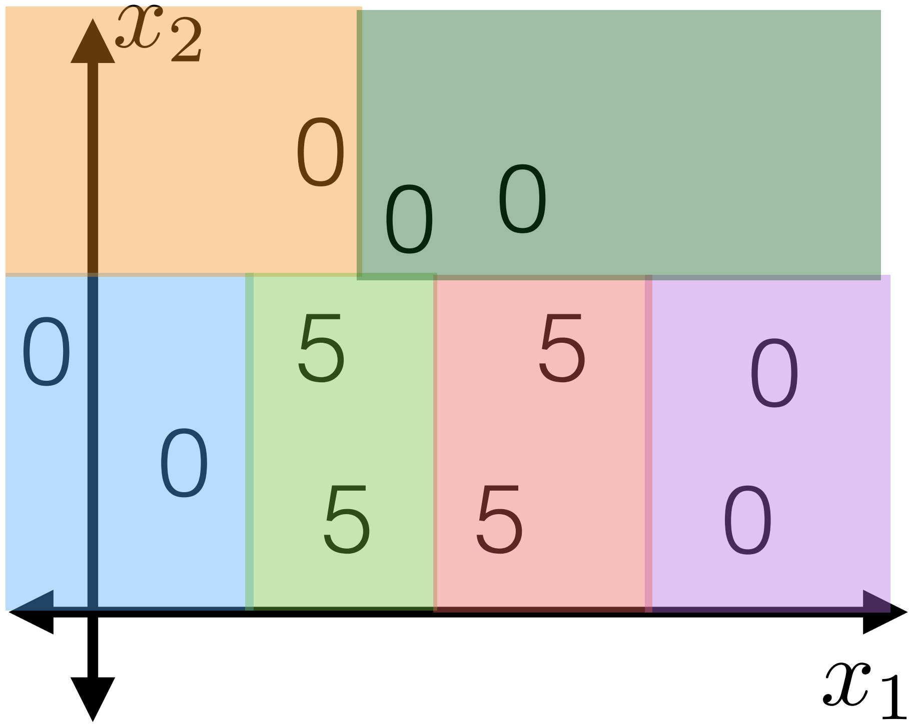


`BuildTree({1,…,n};2)`

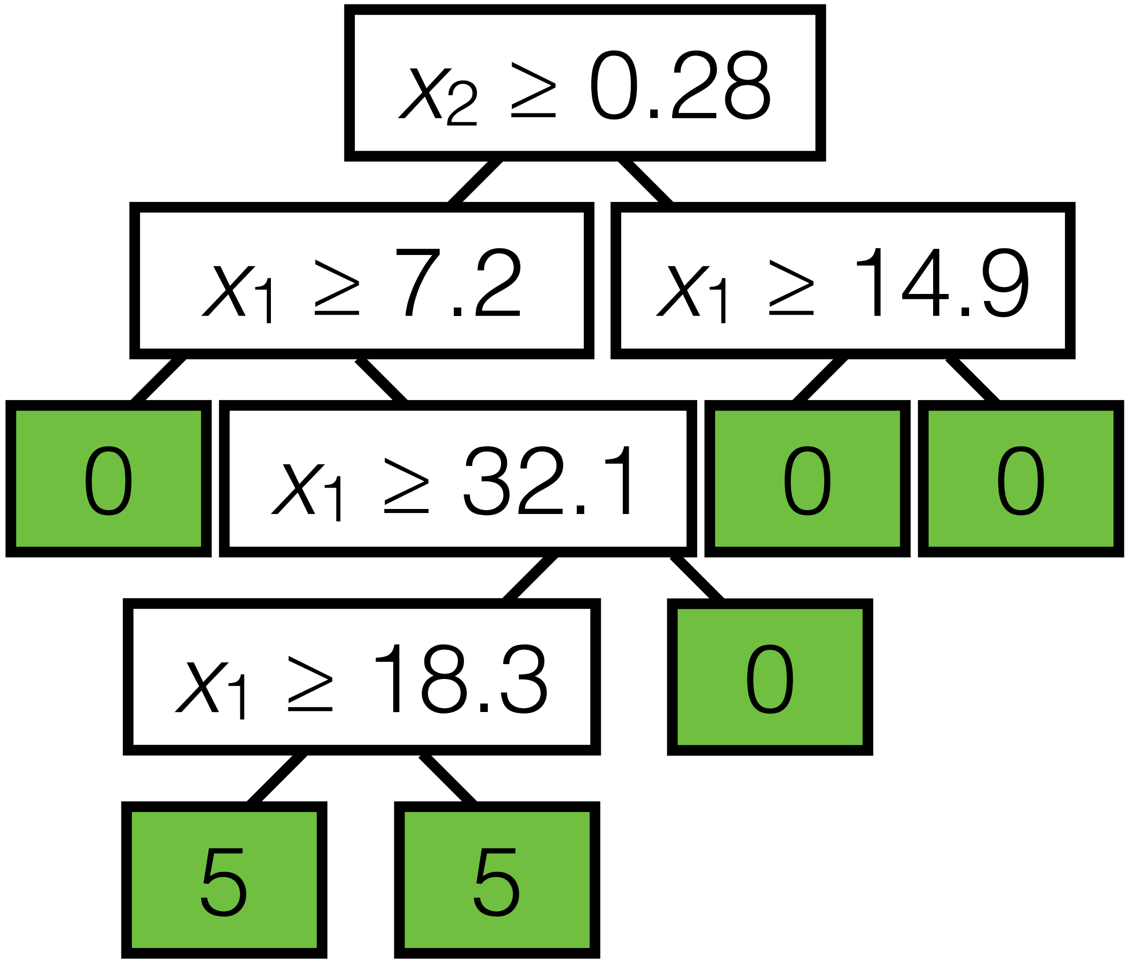


How to regularize?

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



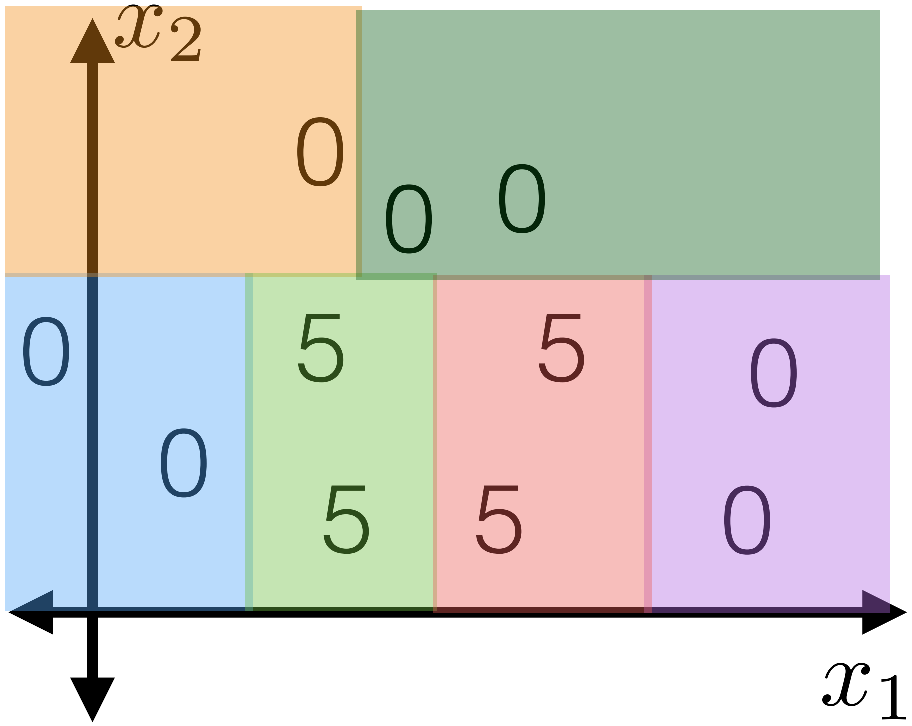
`BuildTree({1,…,n};2)`



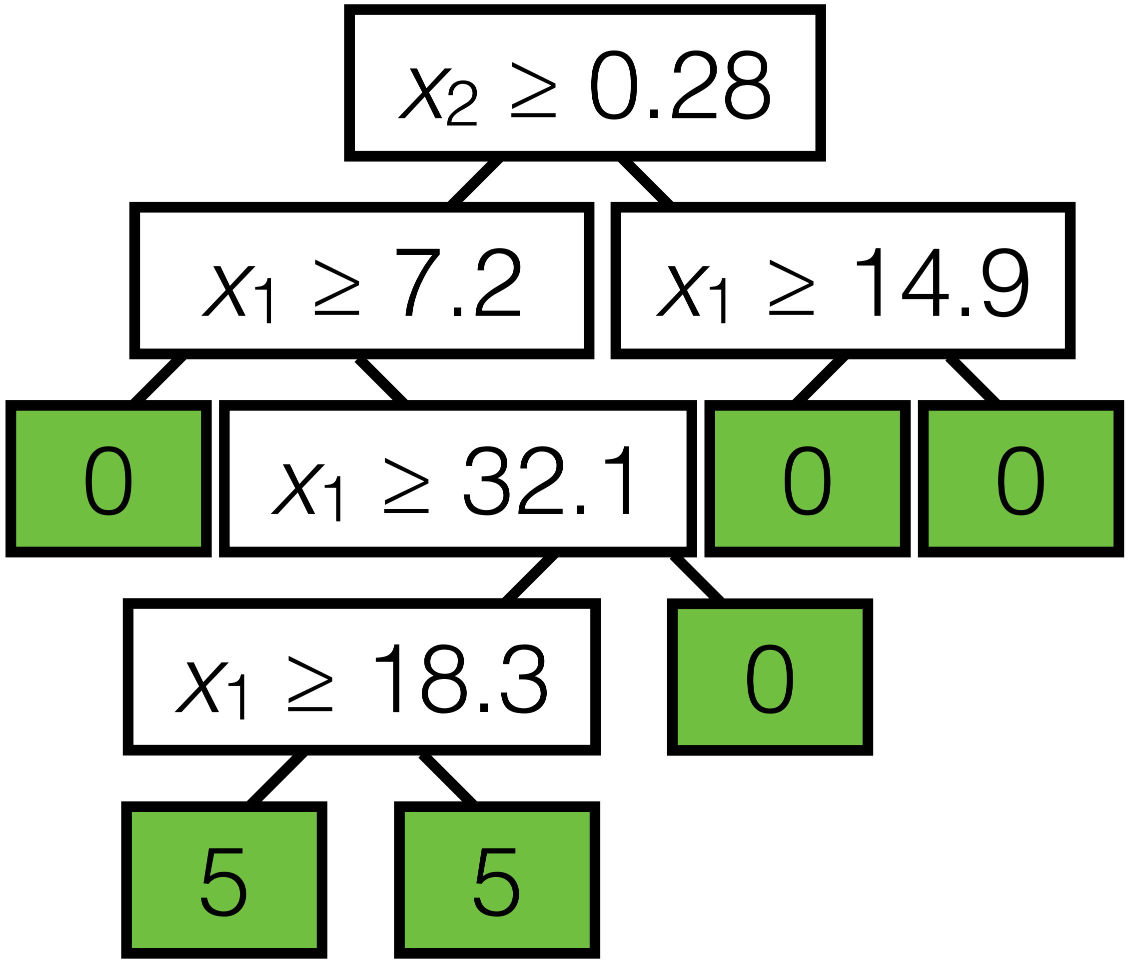
How to regularize?

- "Cost complexity" of a tree *T*

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



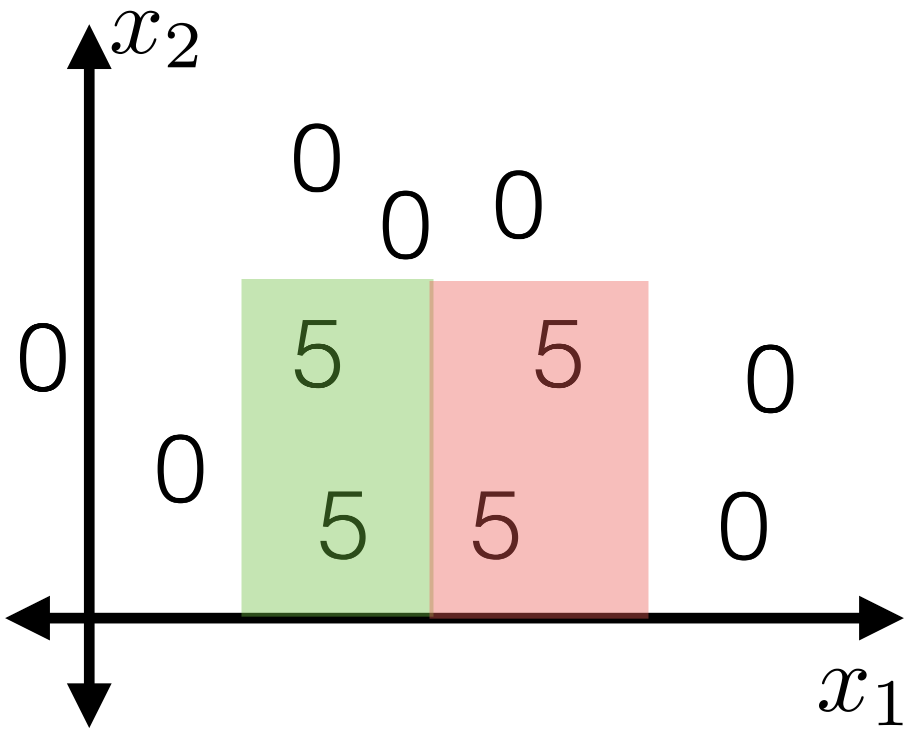
`BuildTree({1,…,n};2)`



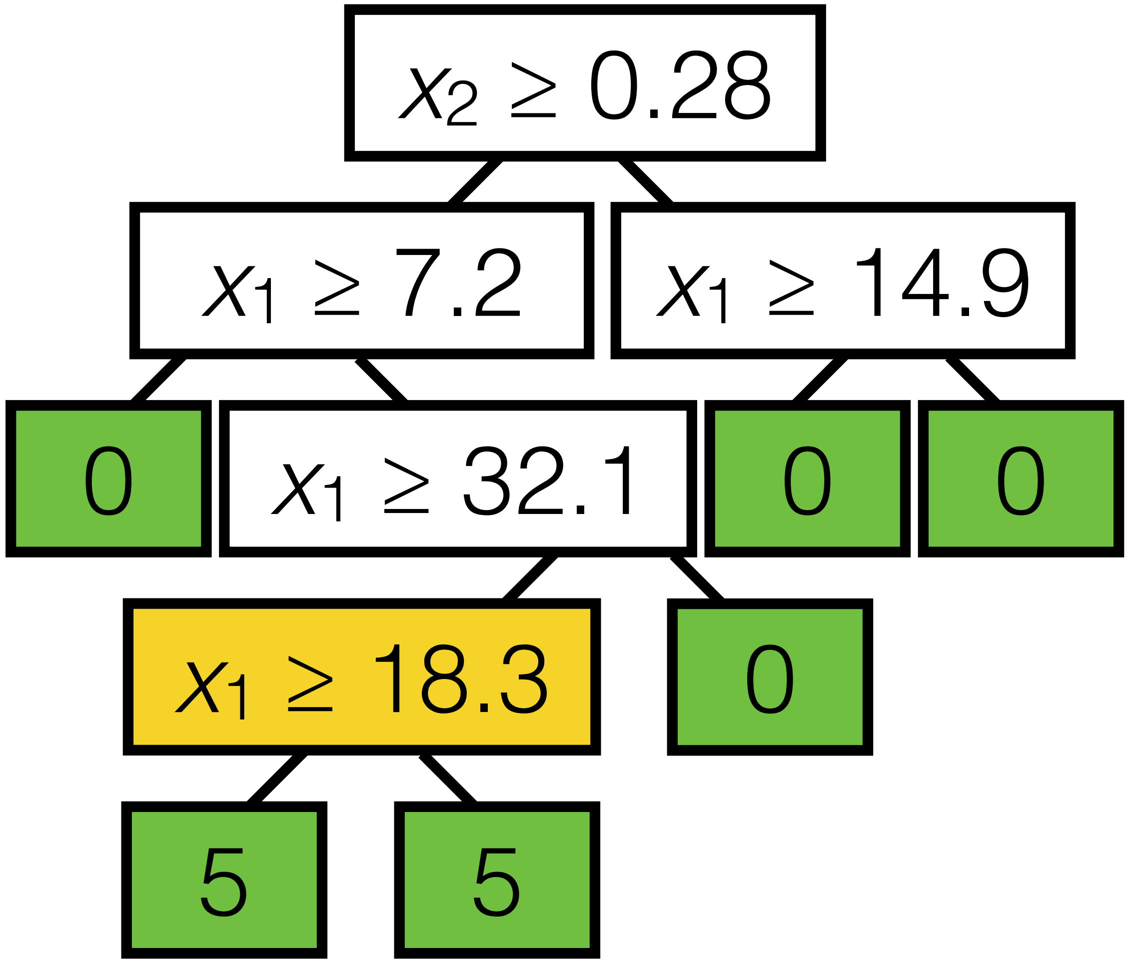
How to regularize?

- “Cost complexity” of a tree *T*

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



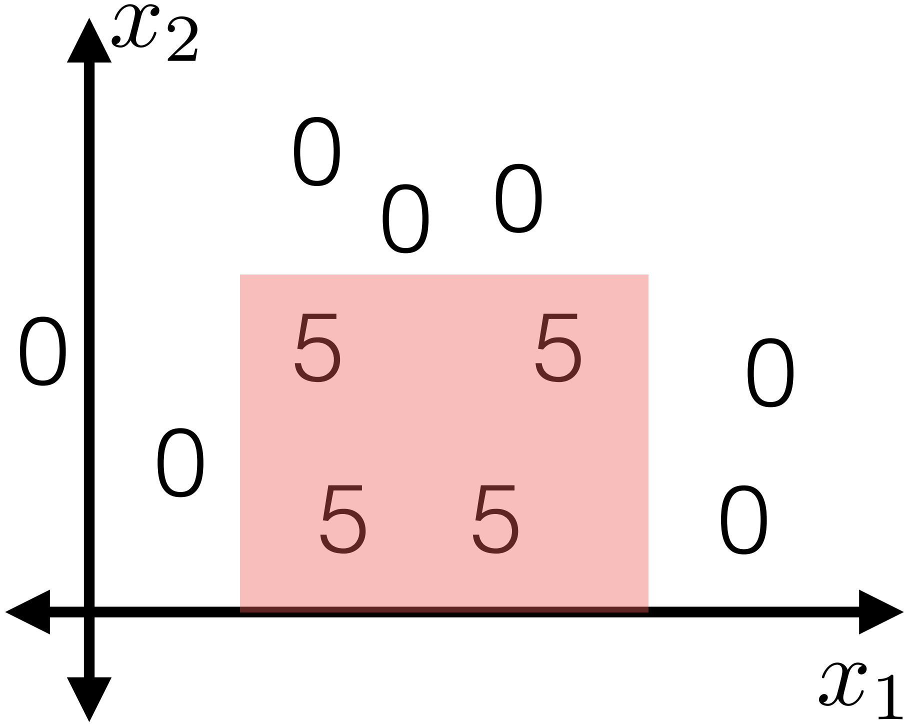
`BuildTree({1,…,n};2)`



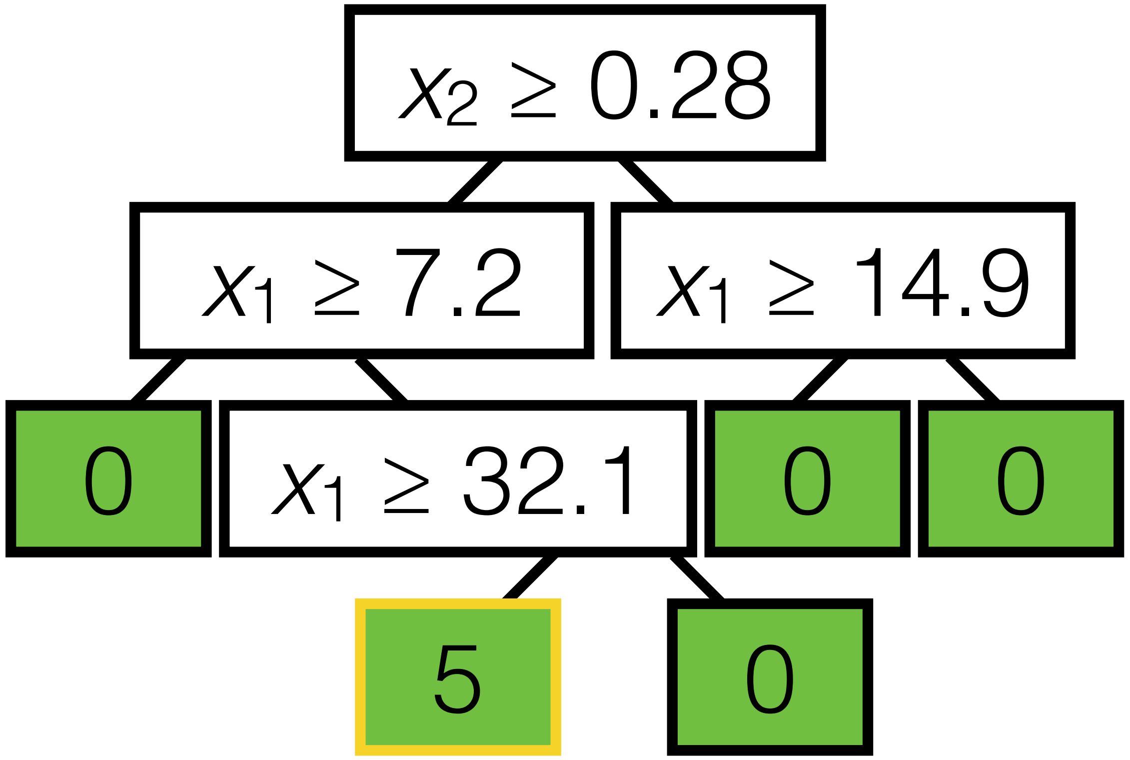
How to regularize?

- “Cost complexity” of a tree *T*

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



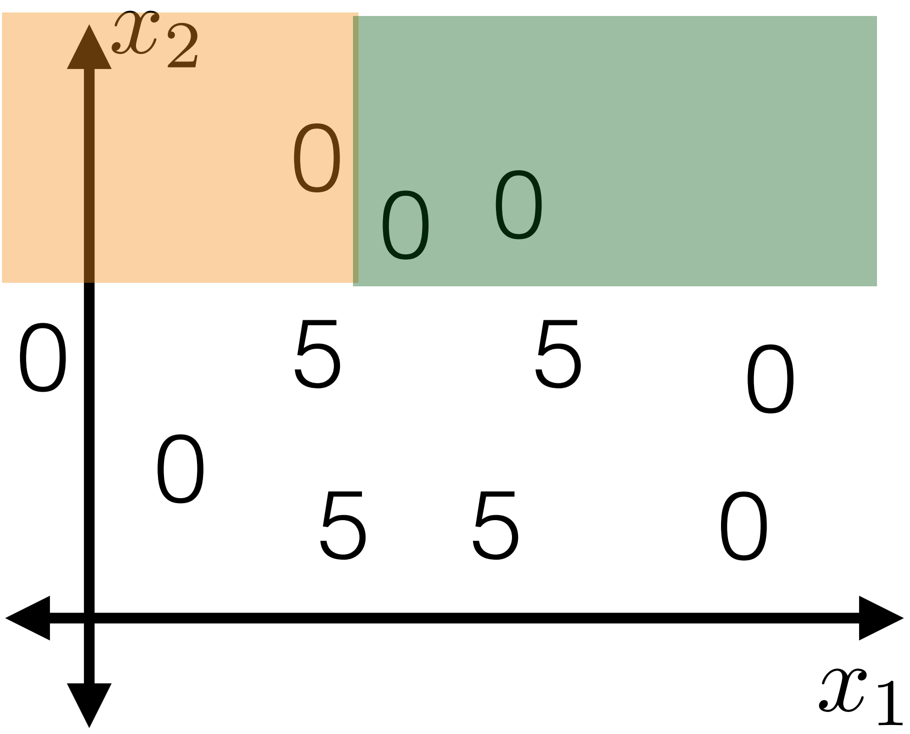
`BuildTree({1,…,n};2)`



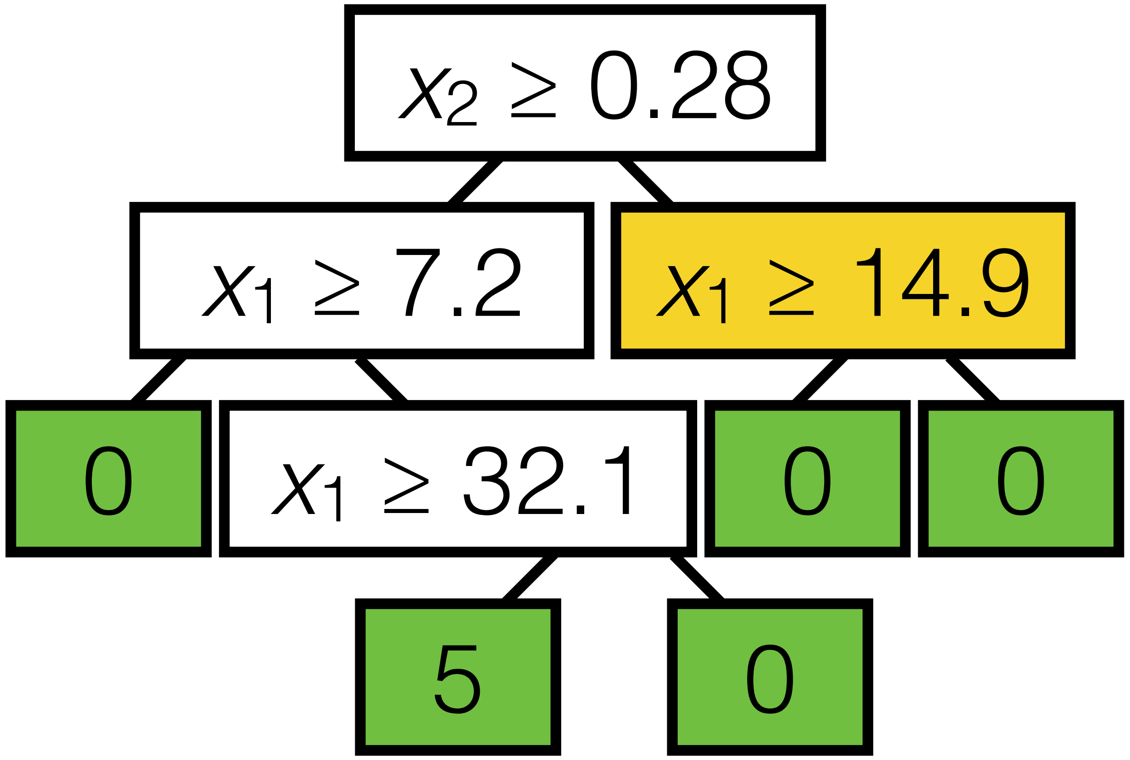
How to regularize?

- “Cost complexity” of a tree *T*

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



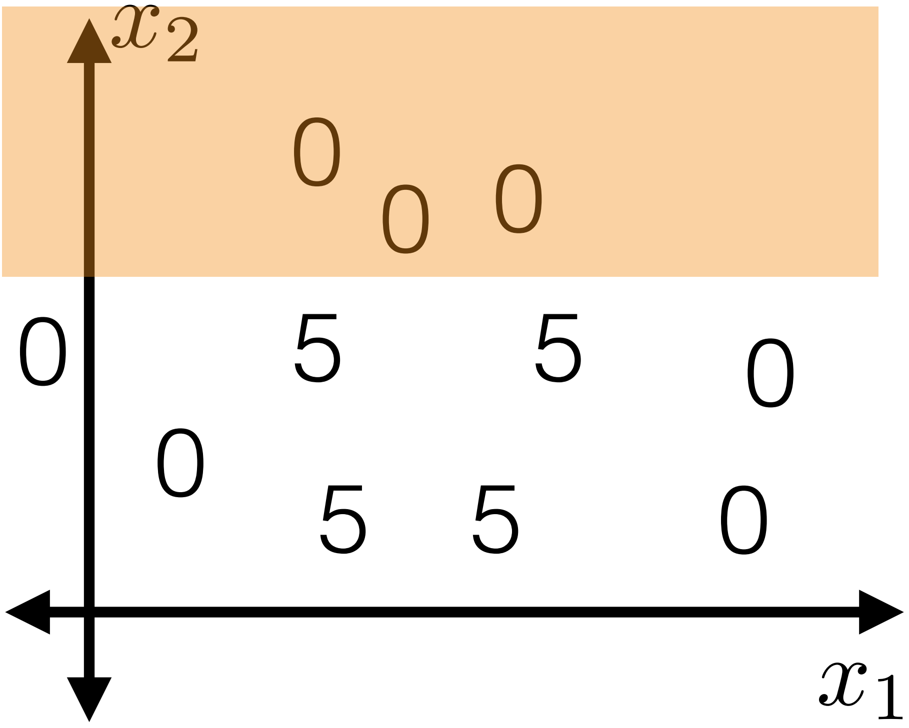
`BuildTree({1,…,n};2)`



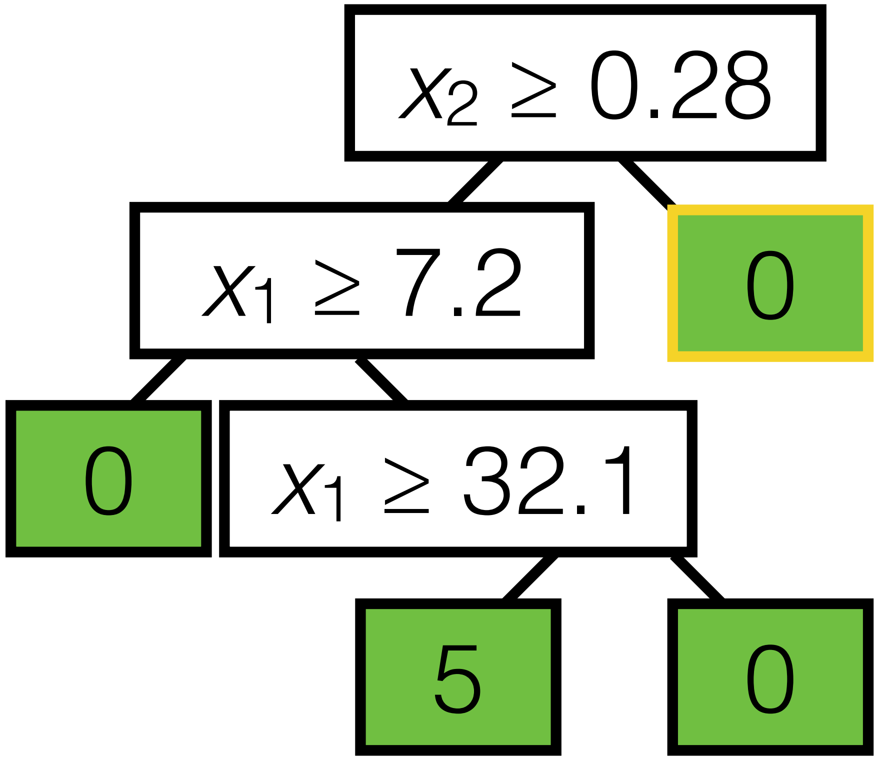
How to regularize?

- “Cost complexity” of a tree *T*

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



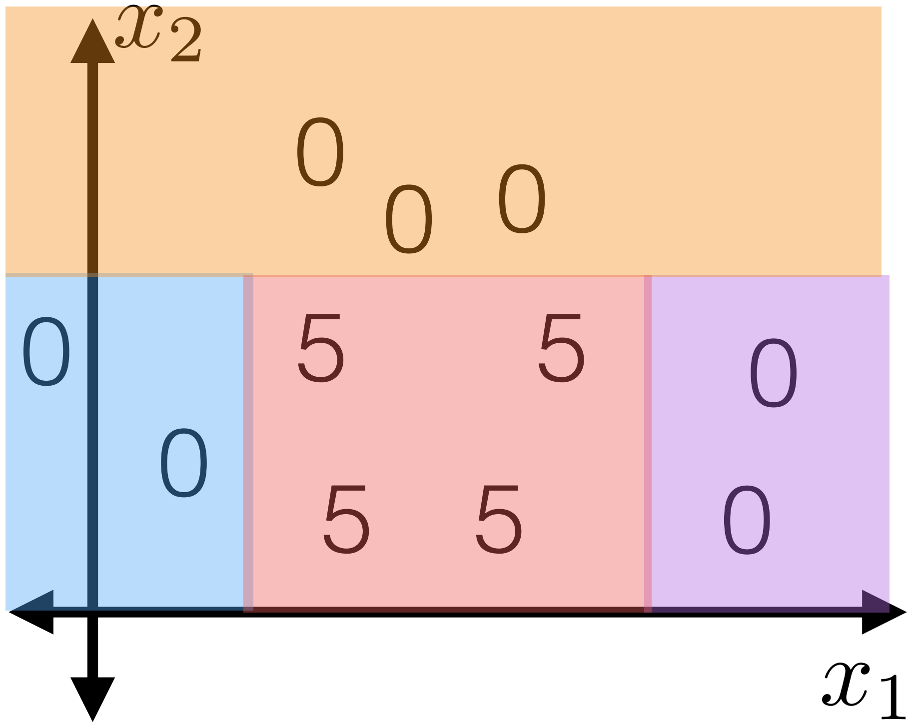
`BuildTree({1,…,n};2)`



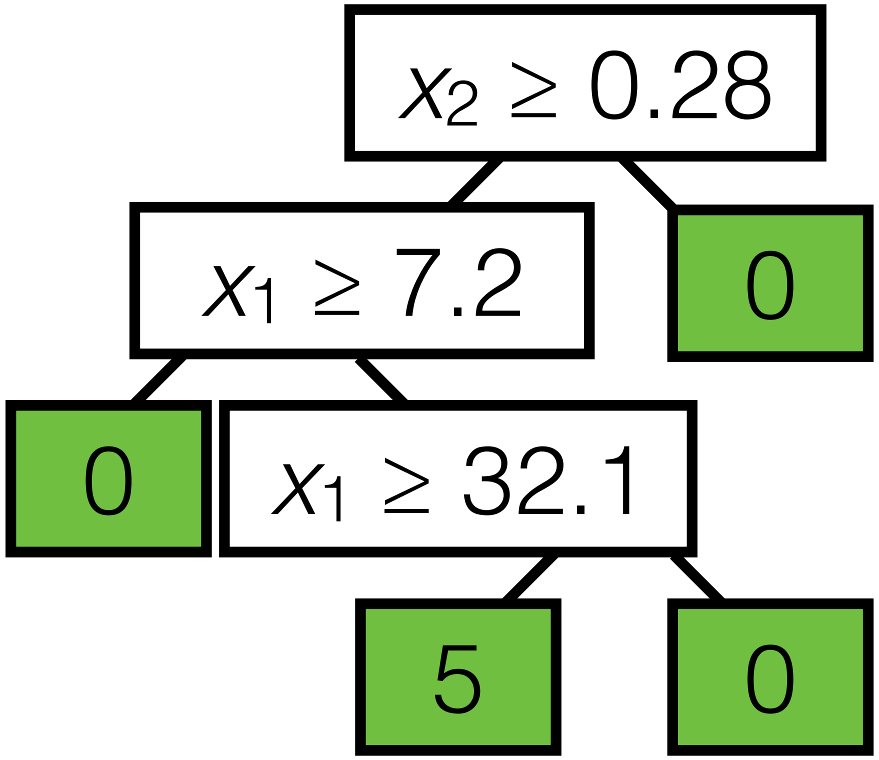
How to regularize?

- "Cost complexity" of a tree T

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



`BuildTree({1,…,n};2)`

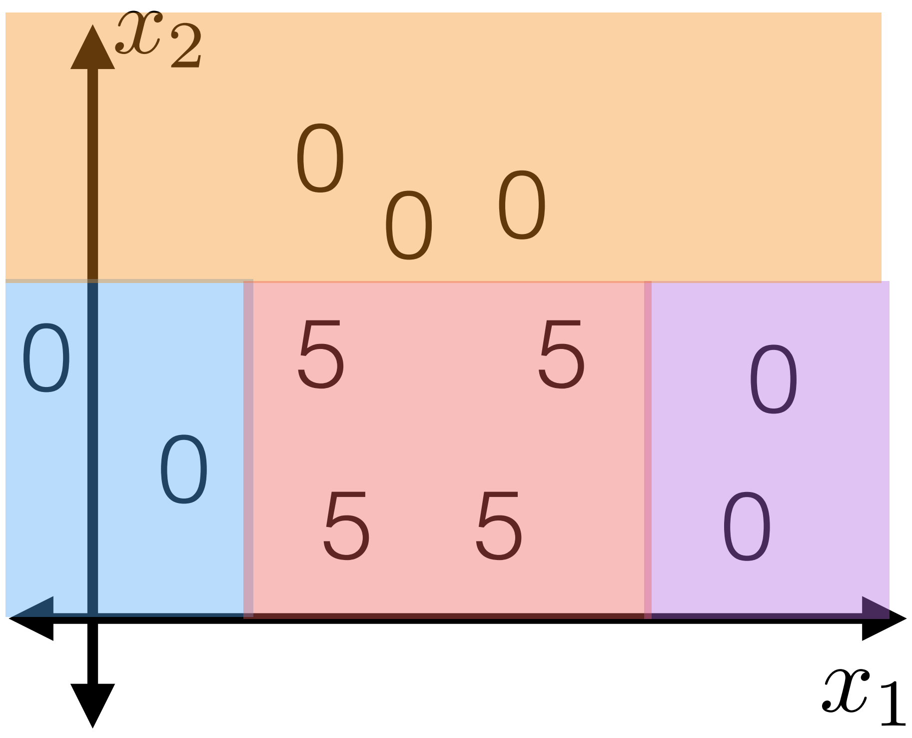


How to regularize?

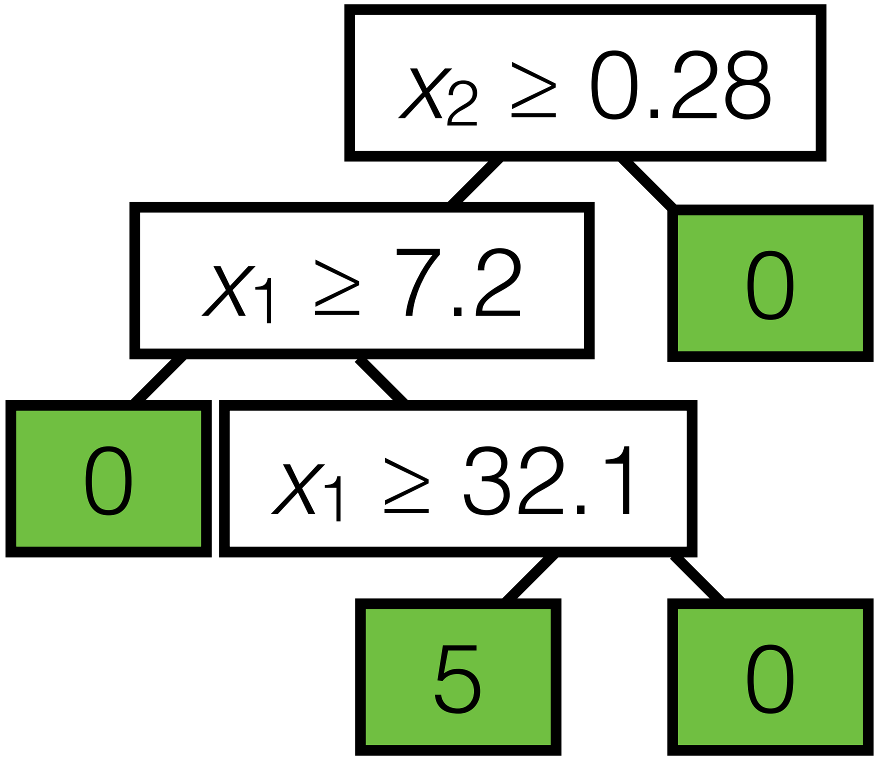
- "Cost complexity" of a tree T

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

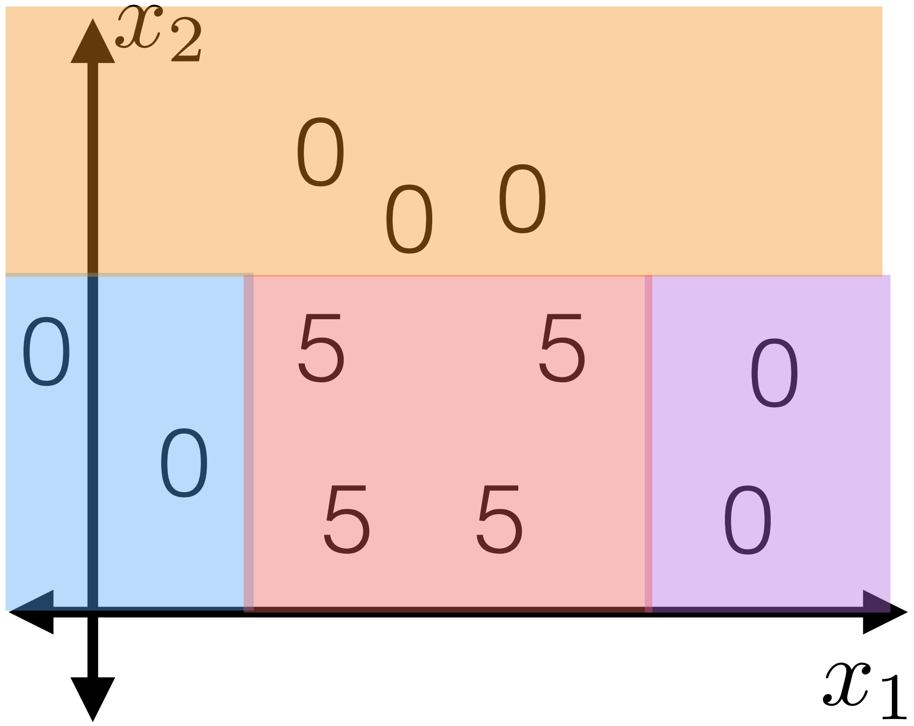
- Pruning



`BuildTree({1,…,n};2)`



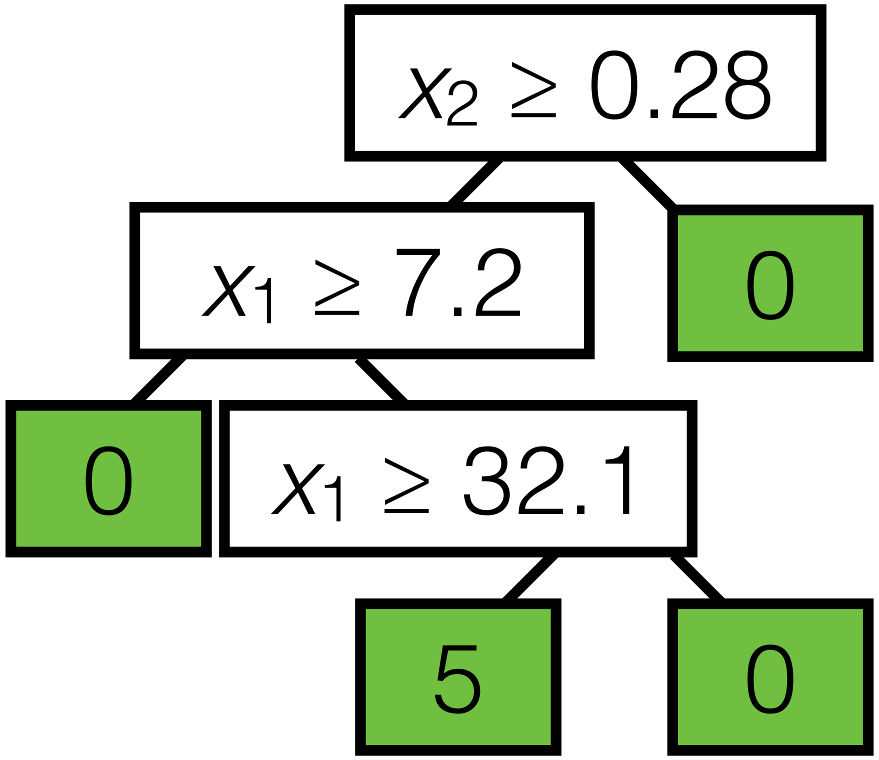
How to regularize?



- “Cost complexity” of a tree T

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

`BuildTree({1,…,n};2)`



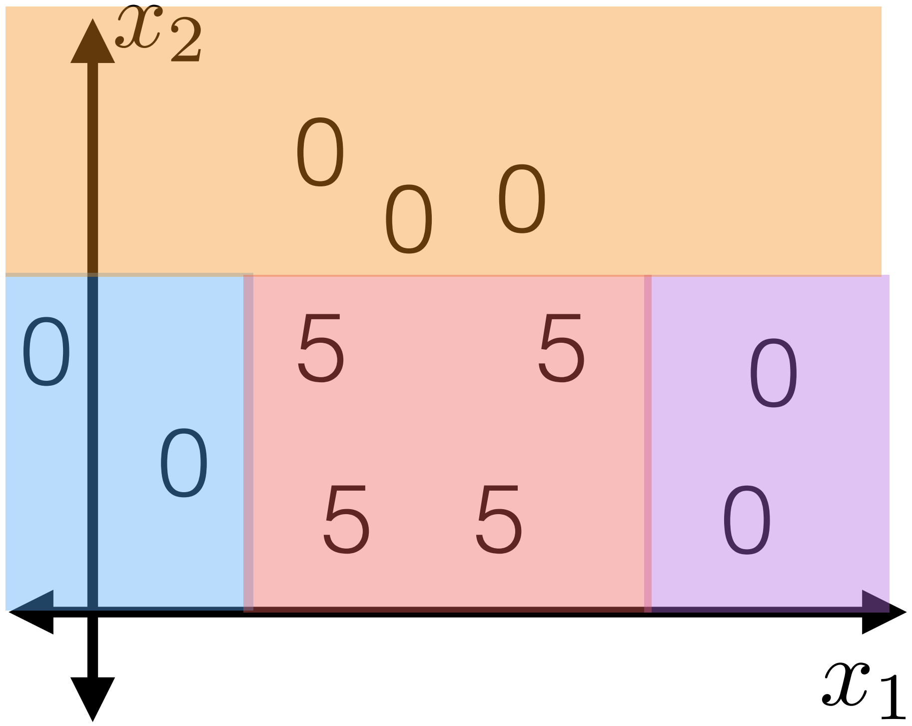
- Pruning
  - For each α, choose T_α by pruning subtrees until it’s not worthwhile

How to regularize?

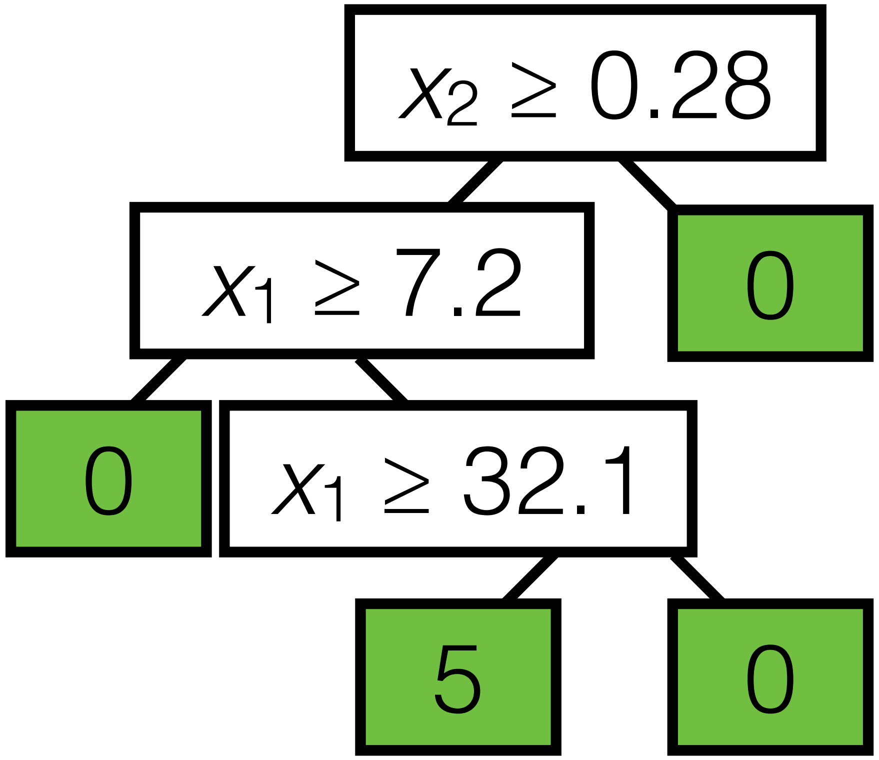
- “Cost complexity” of a tree *T*

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

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



`BuildTree({1,…,n};2)`



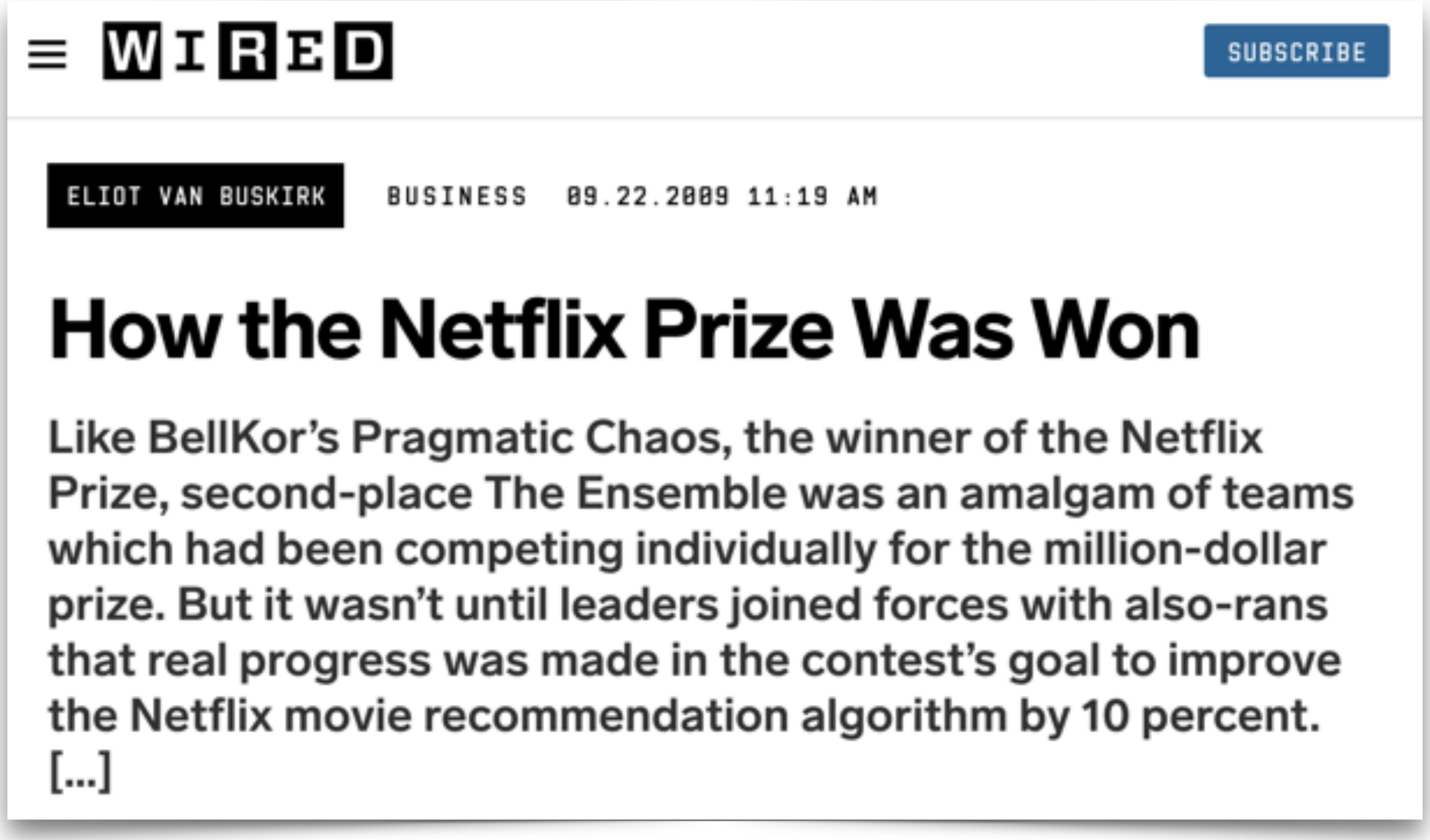
Ensembling

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



ELIOT VAN BUSKIRK BUSINESS 09.22.2009 11:19 AM

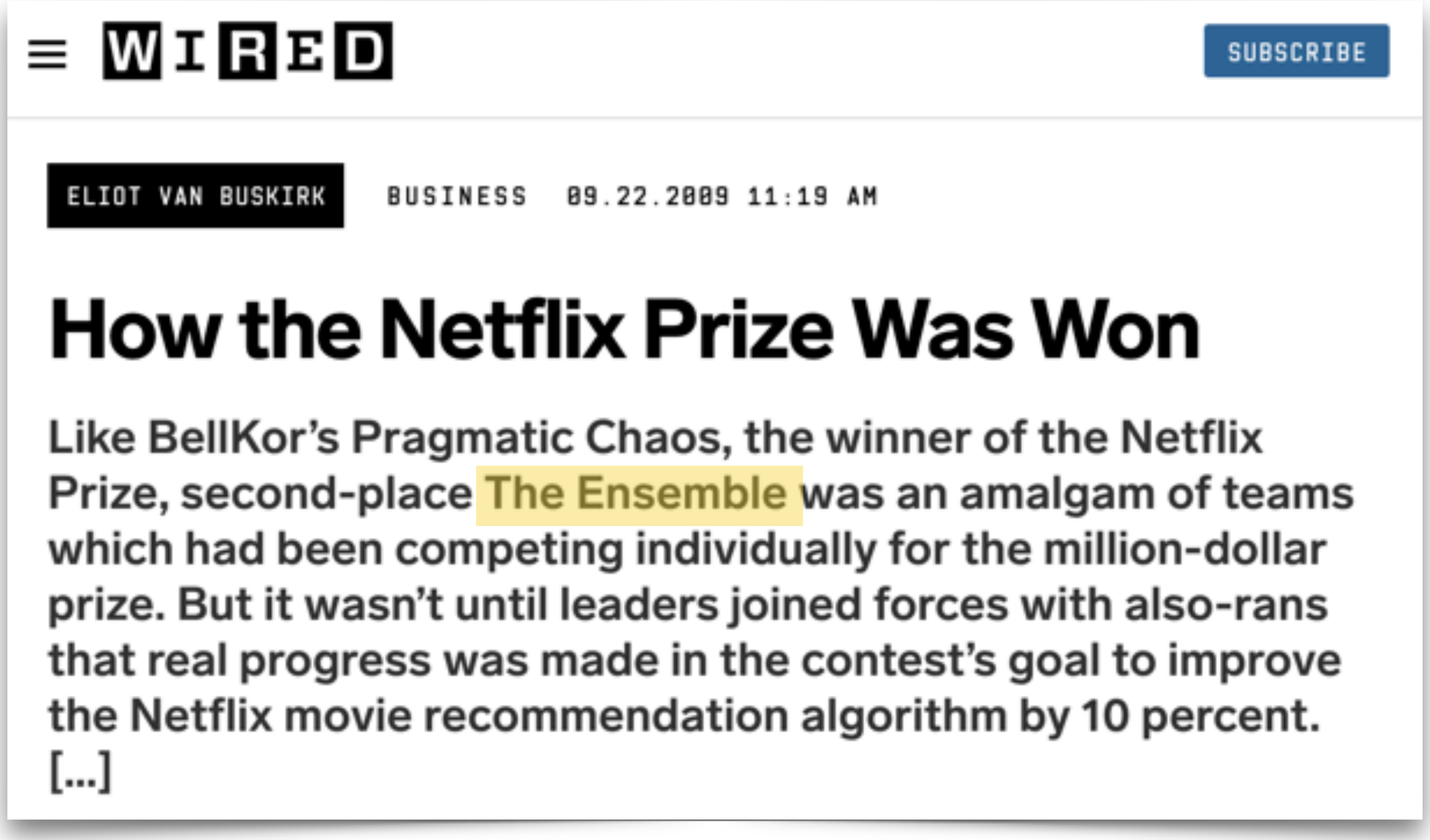
How the Netflix Prize Was Won

Like BellKor's Pragmatic Chaos, the winner of the Netflix Prize, second-place The Ensemble was an amalgam of teams which had been competing individually for the million-dollar prize. But it wasn't until leaders joined forces with also-rans that real progress was made in the contest's goal to improve the Netflix movie recommendation algorithm by 10 percent. [...]

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

Ensembling

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



ELIOT VAN BUSKIRK BUSINESS 09.22.2009 11:19 AM

How the Netflix Prize Was Won

Like BellKor's Pragmatic Chaos, the winner of the Netflix Prize, second-place The Ensemble was an amalgam of teams which had been competing individually for the million-dollar prize. But it wasn't until leaders joined forces with also-rans that real progress was made in the contest's goal to improve the Netflix movie recommendation algorithm by 10 percent. [...]

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

Ensembling

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

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
ercent.

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]

[https://www.wired.com/2009/09/how-the-netflix-prize-was-won/; https://www.sciencedirect.com/science/article/pii/S0169207019301128]

Ensembling

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

ELIOT VAN BUSKIRK BUSINESS 09.22.2009 11:19 AM

How the Netflix Prize Was W

Like BellKor's Pragmatic Chaos, the winner of the Prize, second-place The Ensemble was an amalga which had been competing individually for the milli

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]

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/; 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

ELIOT VAN BUSKIRK BUSINESS 09.22.2009 11:19 AM

How the Netflix Prize Was W

Like BellKor's Pragmatic Chaos, the winner of the
Prize, second-place The Ensemble was an amalga
which had been competing individually for the milli

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.

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]

[https://www.wired.com/2009/09/how-the-netflix-prize-was-won/; 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**regat**ing**

Bagging

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

Bagging

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

Bagging

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

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

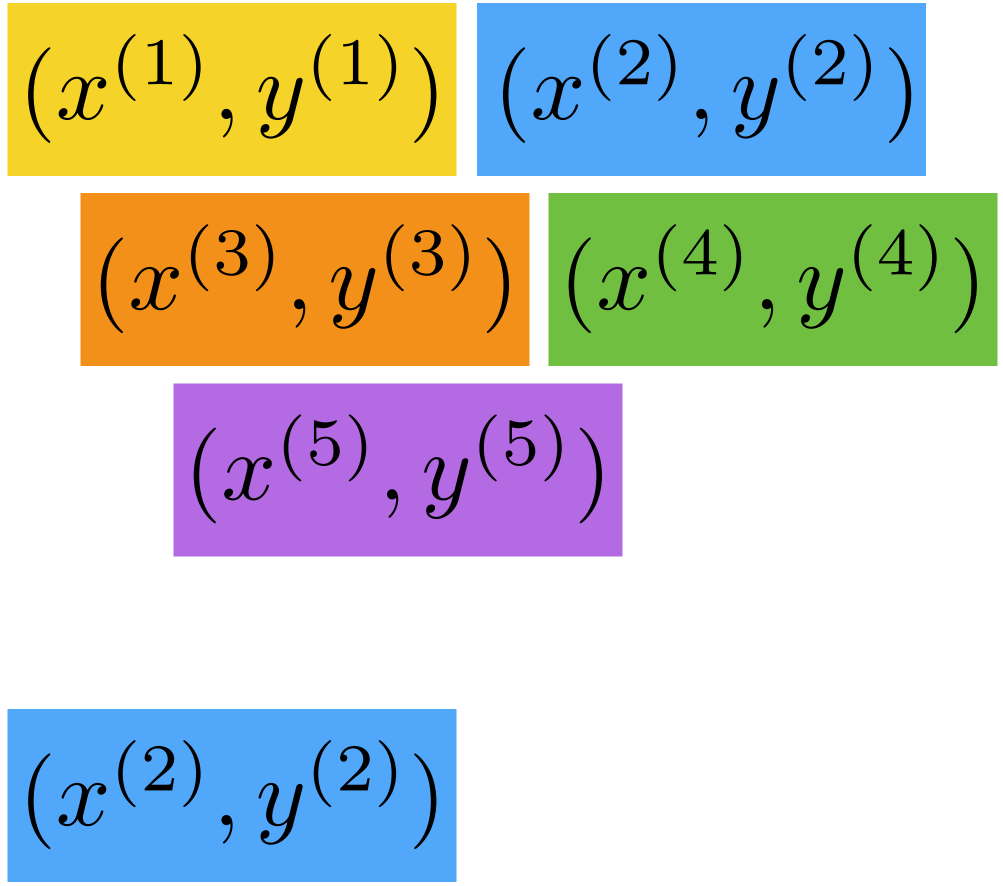
$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

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

Bagging

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



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

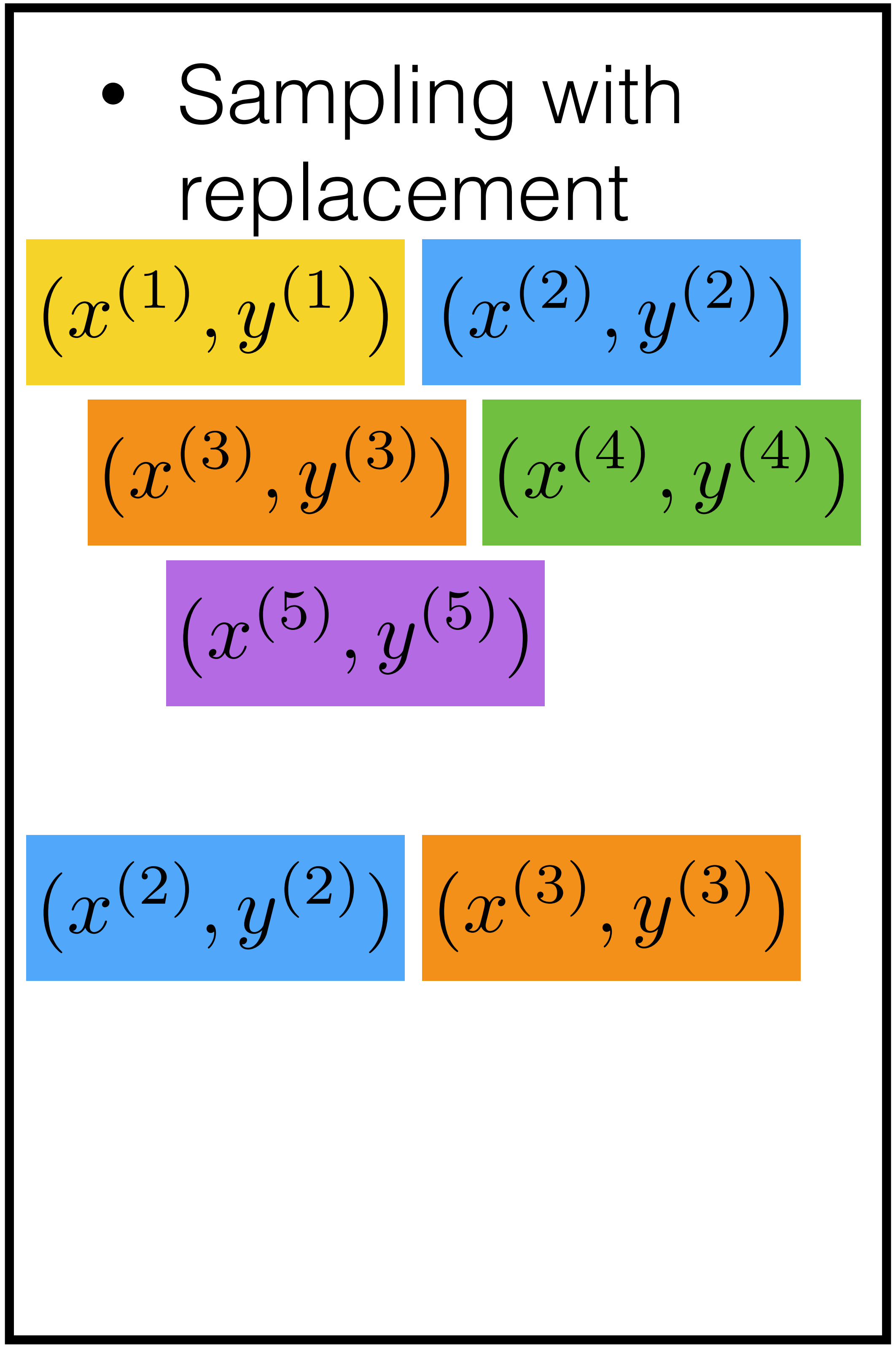
$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$

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

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

Bagging

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

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

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

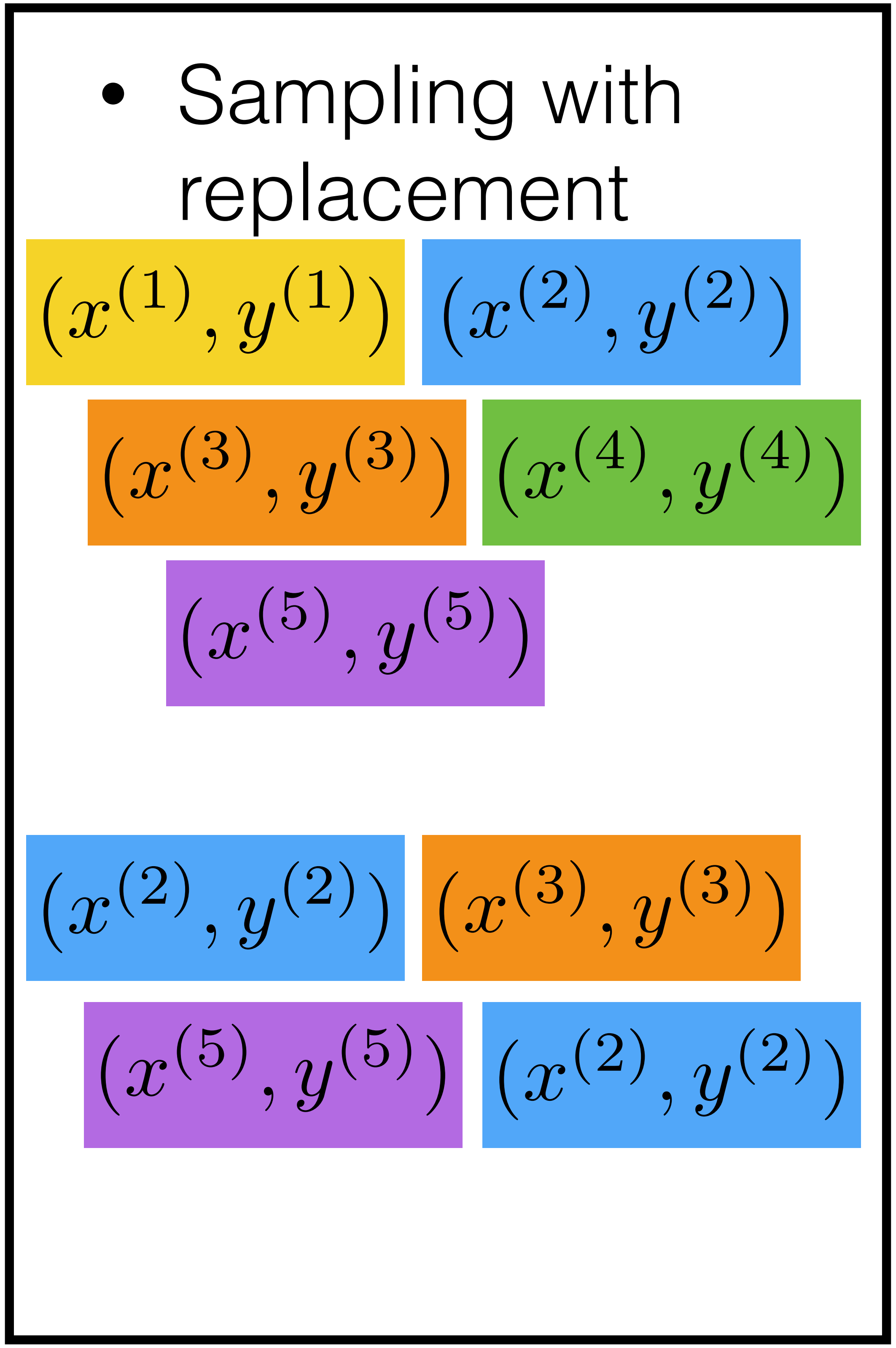
$(x^{(5)}, y^{(5)})$

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

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

Bagging

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

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

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

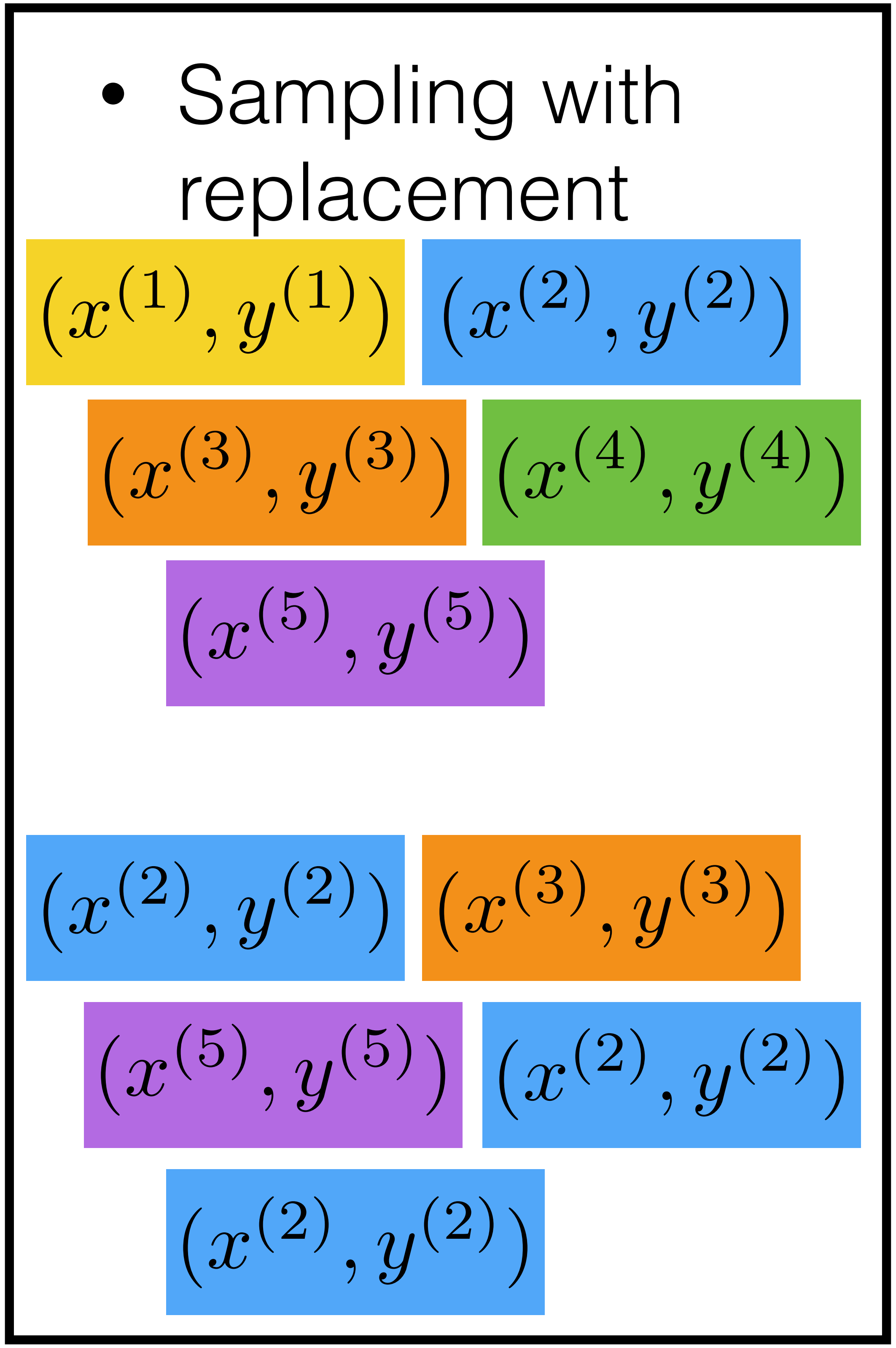
$(x^{(5)}, y^{(5)})$

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

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

Bagging

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

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

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

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

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

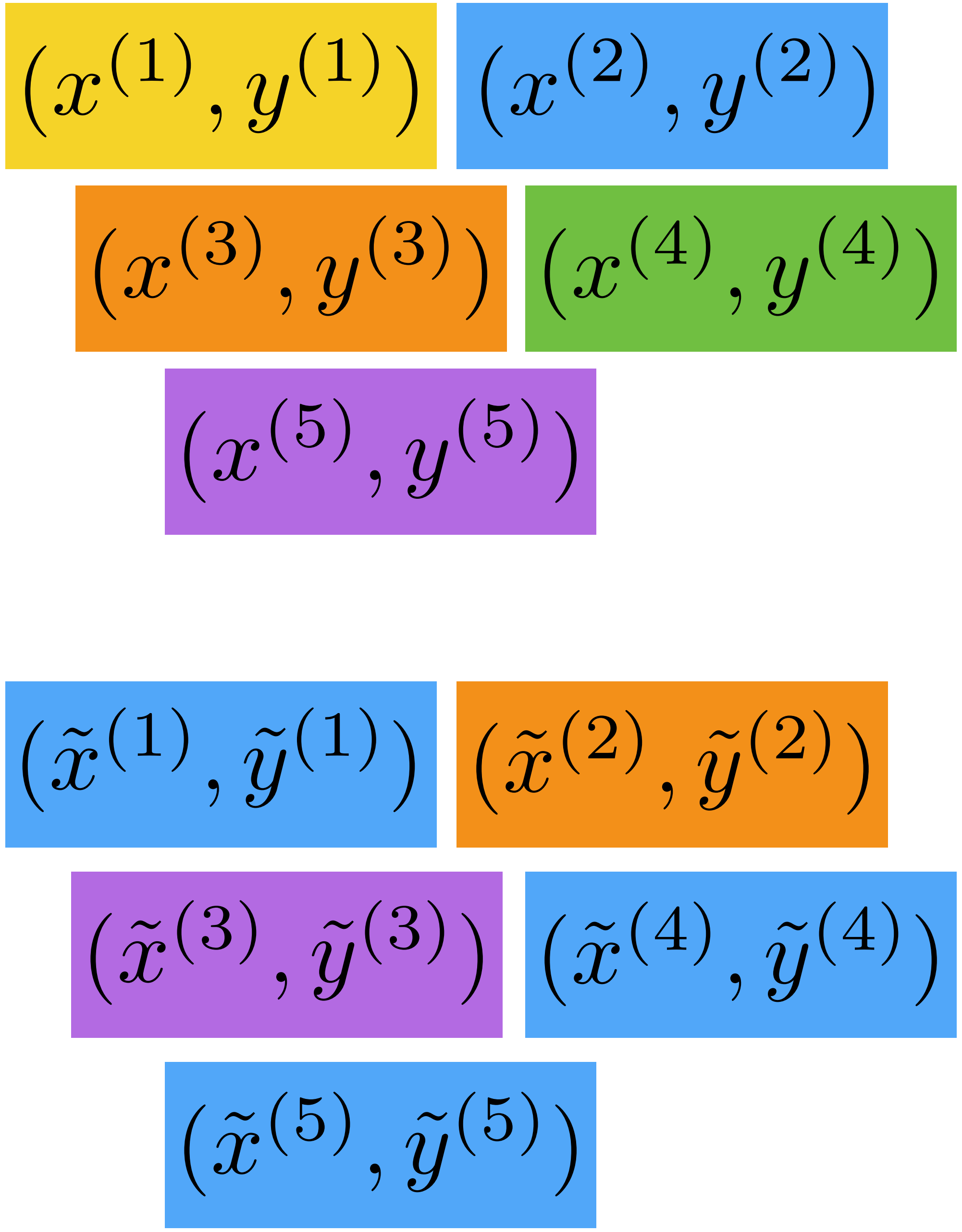
$(x^{(5)}, y^{(5)})$ $(x^{(2)}, y^{(2)})$

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

Bagging

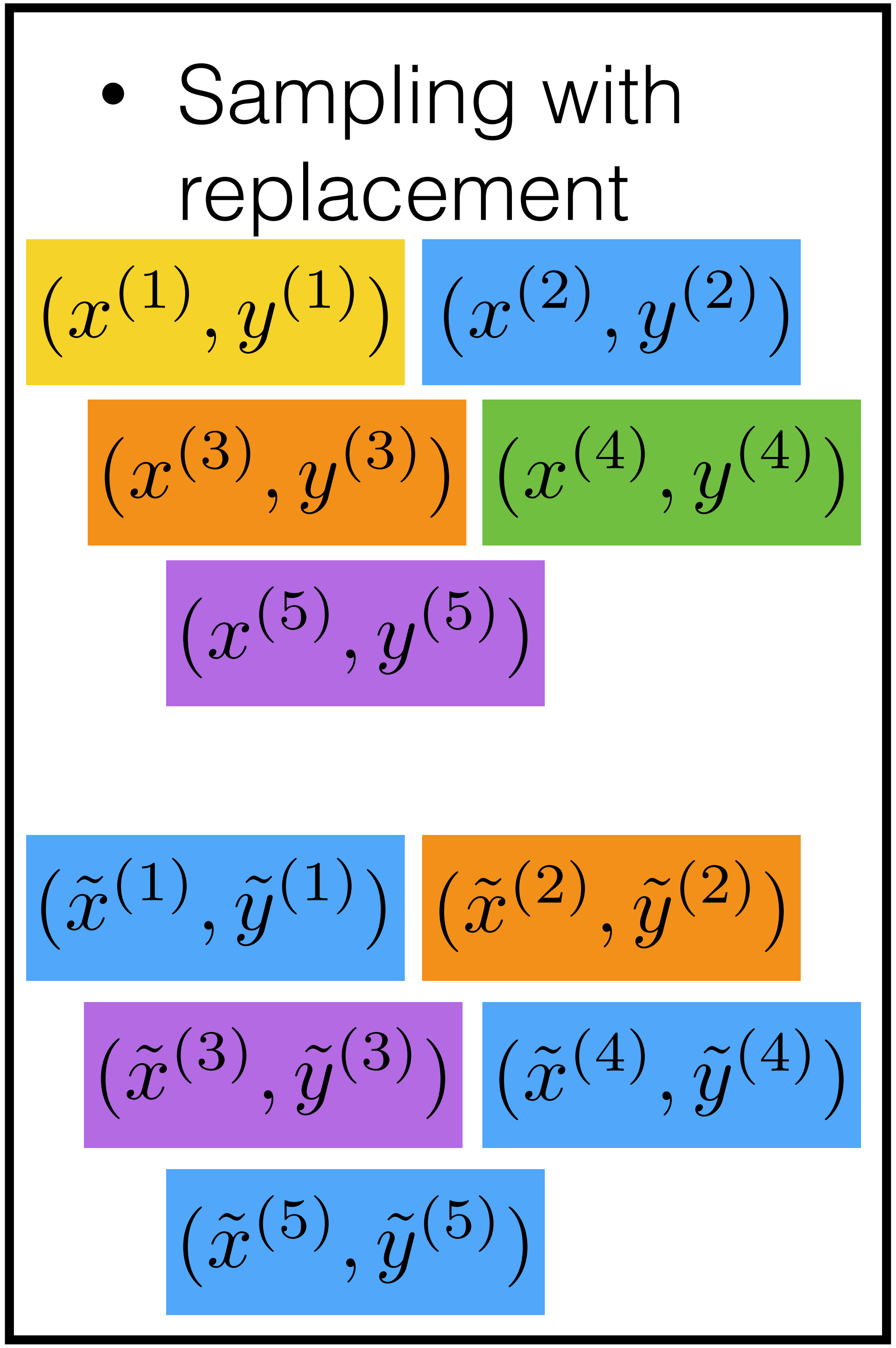
- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regat**ing**
  - Training data $\mathcal{D}_n$
  - For *b* = 1, …, *B*
    - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$

- Sampling with replacement



Bagging

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

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

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

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

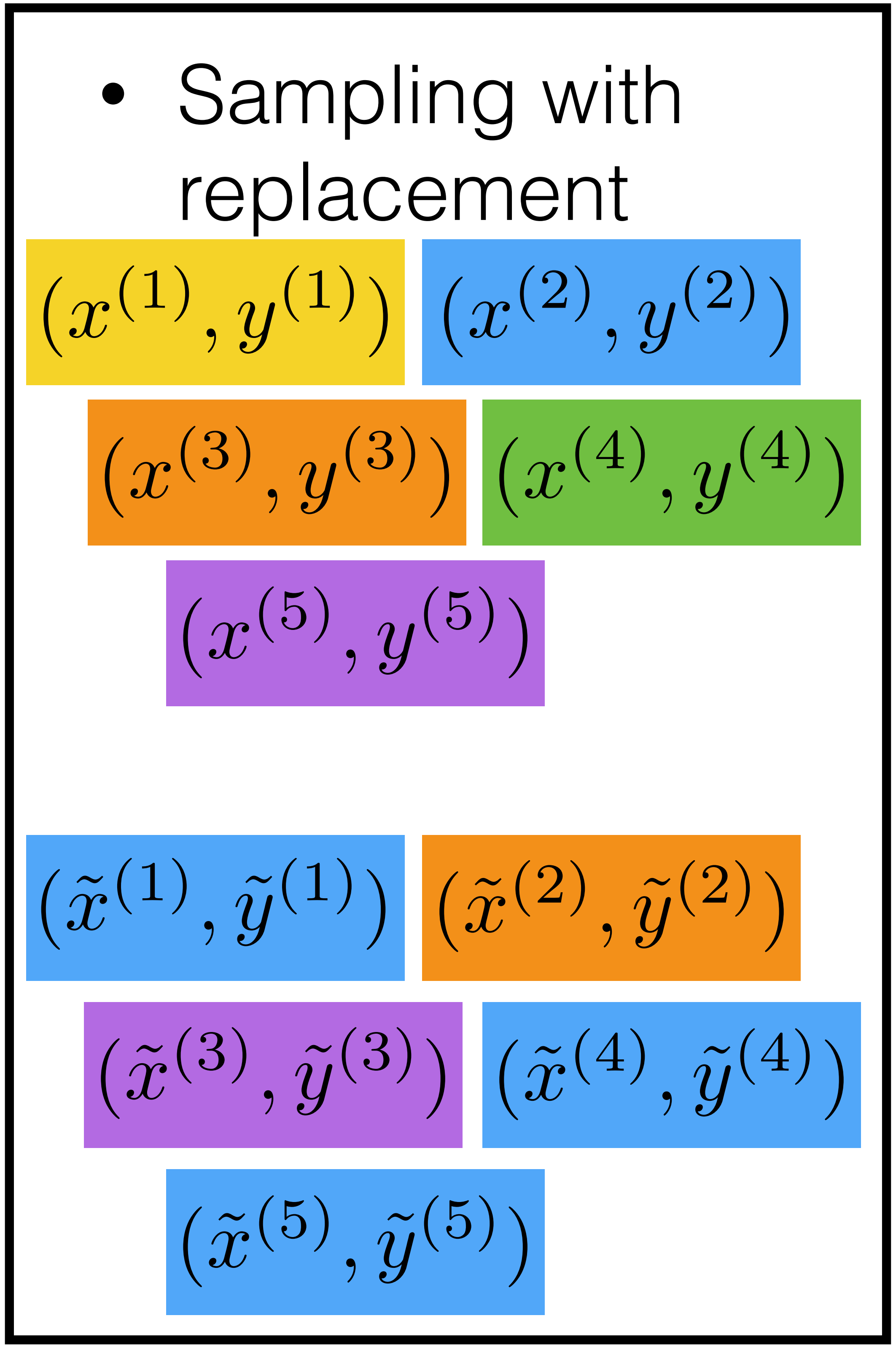
$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$

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

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

Bagging

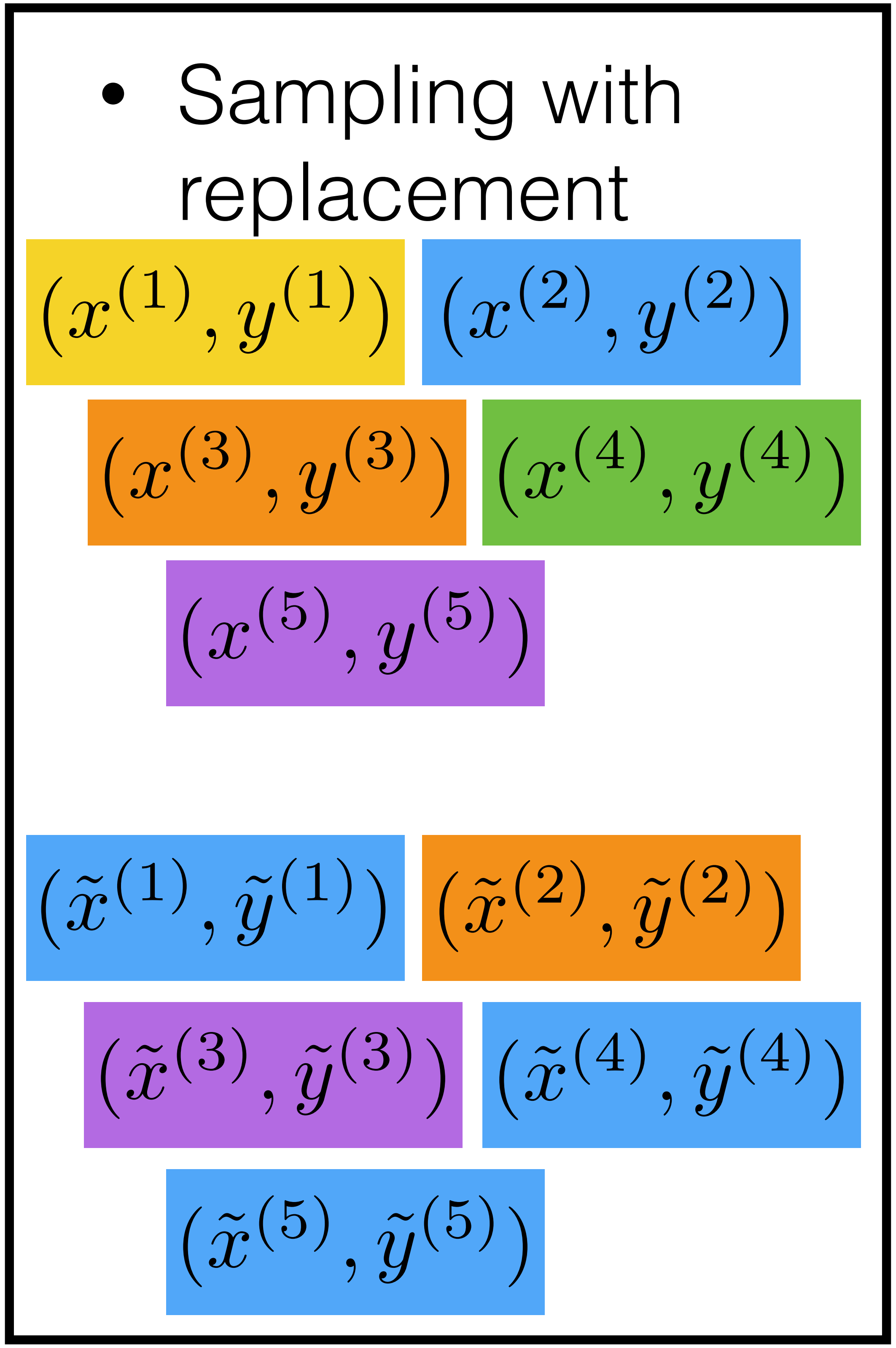
- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regat**ing**
  - Training data $\mathcal{D}_n$
  - For *b* = 1, …, *B*
    - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$
    - Train a predictor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



- Sampling with replacement

Bagging

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

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

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

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

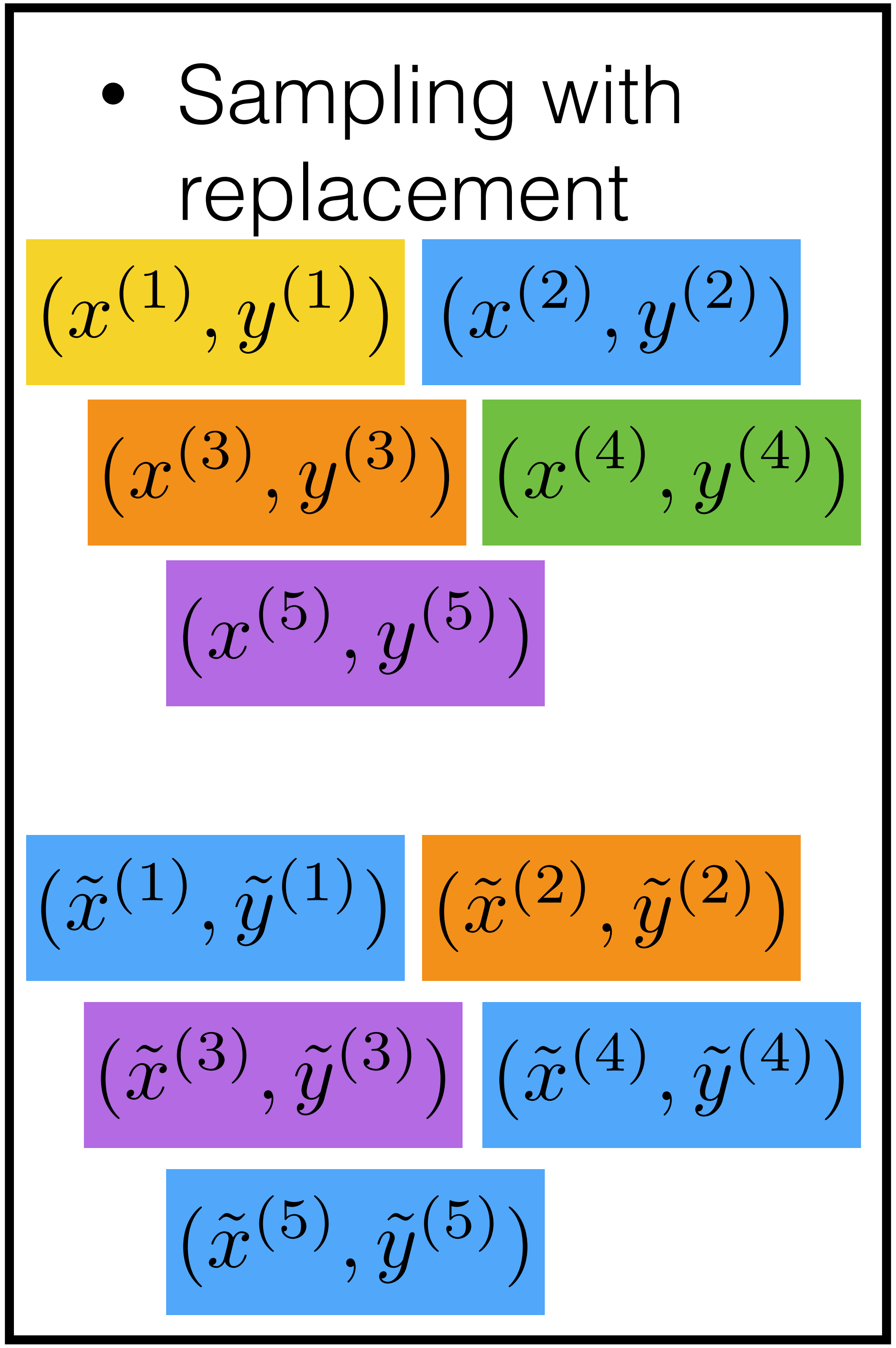
$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$

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

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

Bagging

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



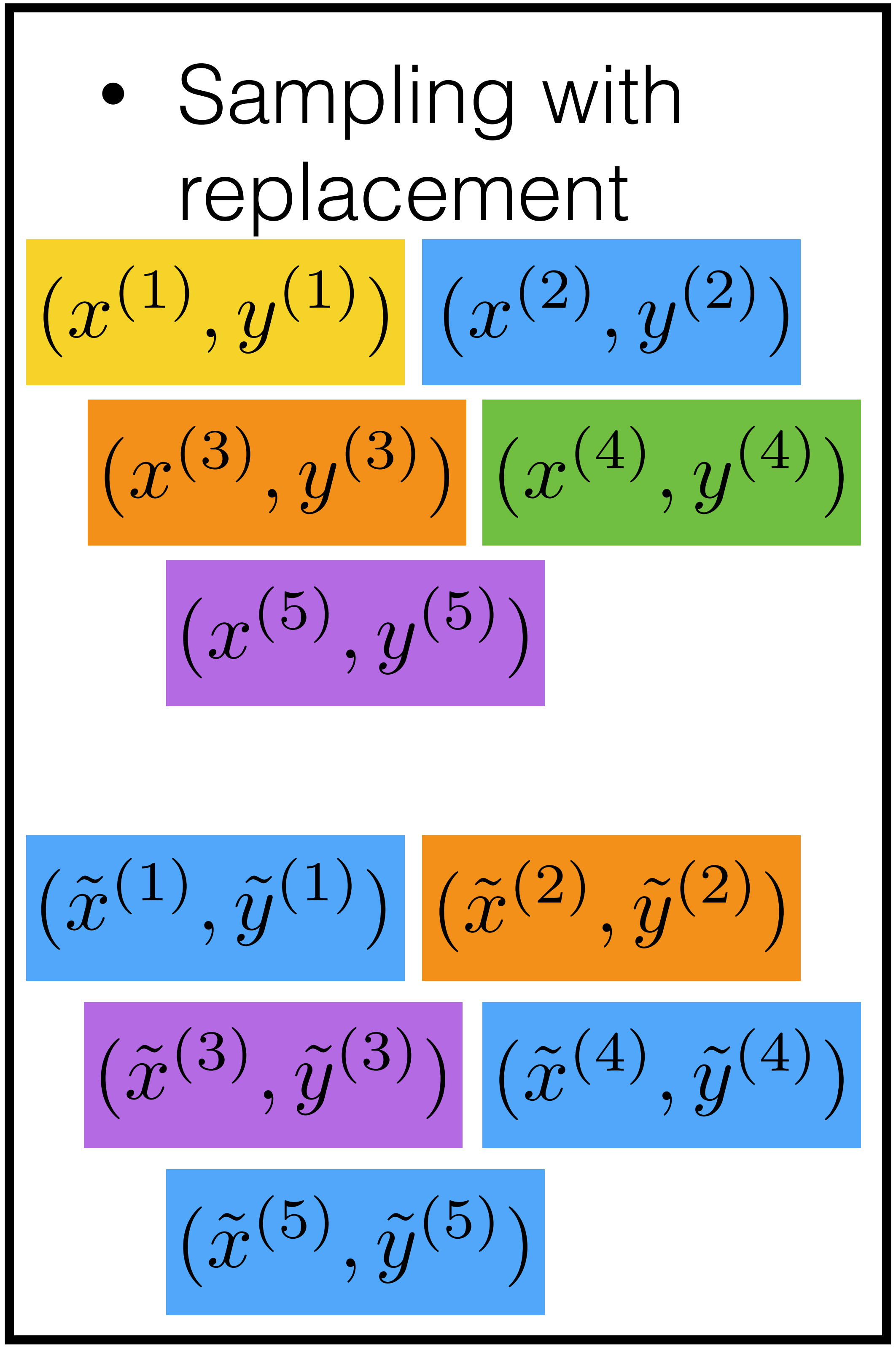
- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$ $(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$ $(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$ $(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$ $(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Bagging

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



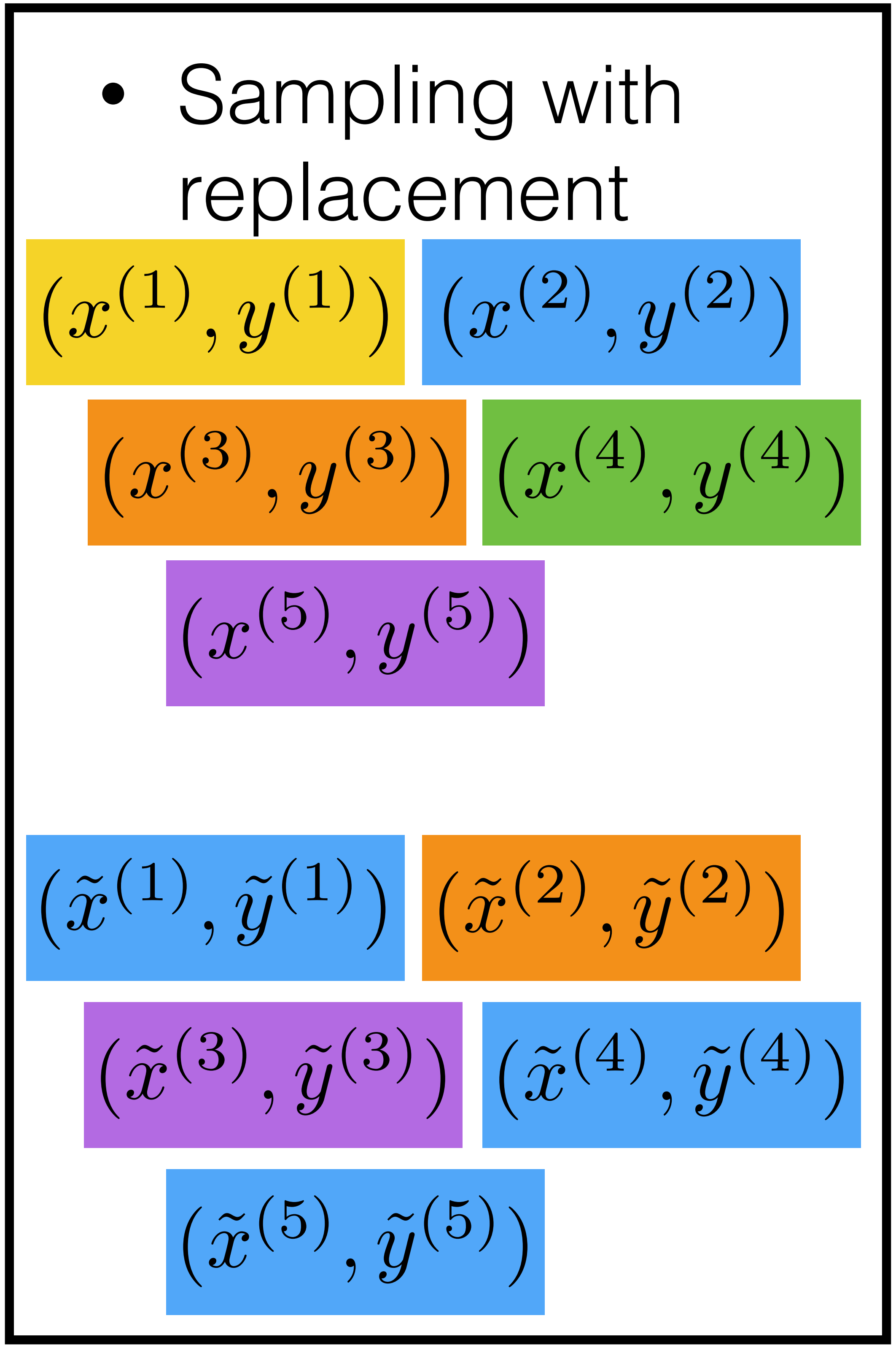
- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$ $(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$ $(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$ $(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$ $(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Bagging

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



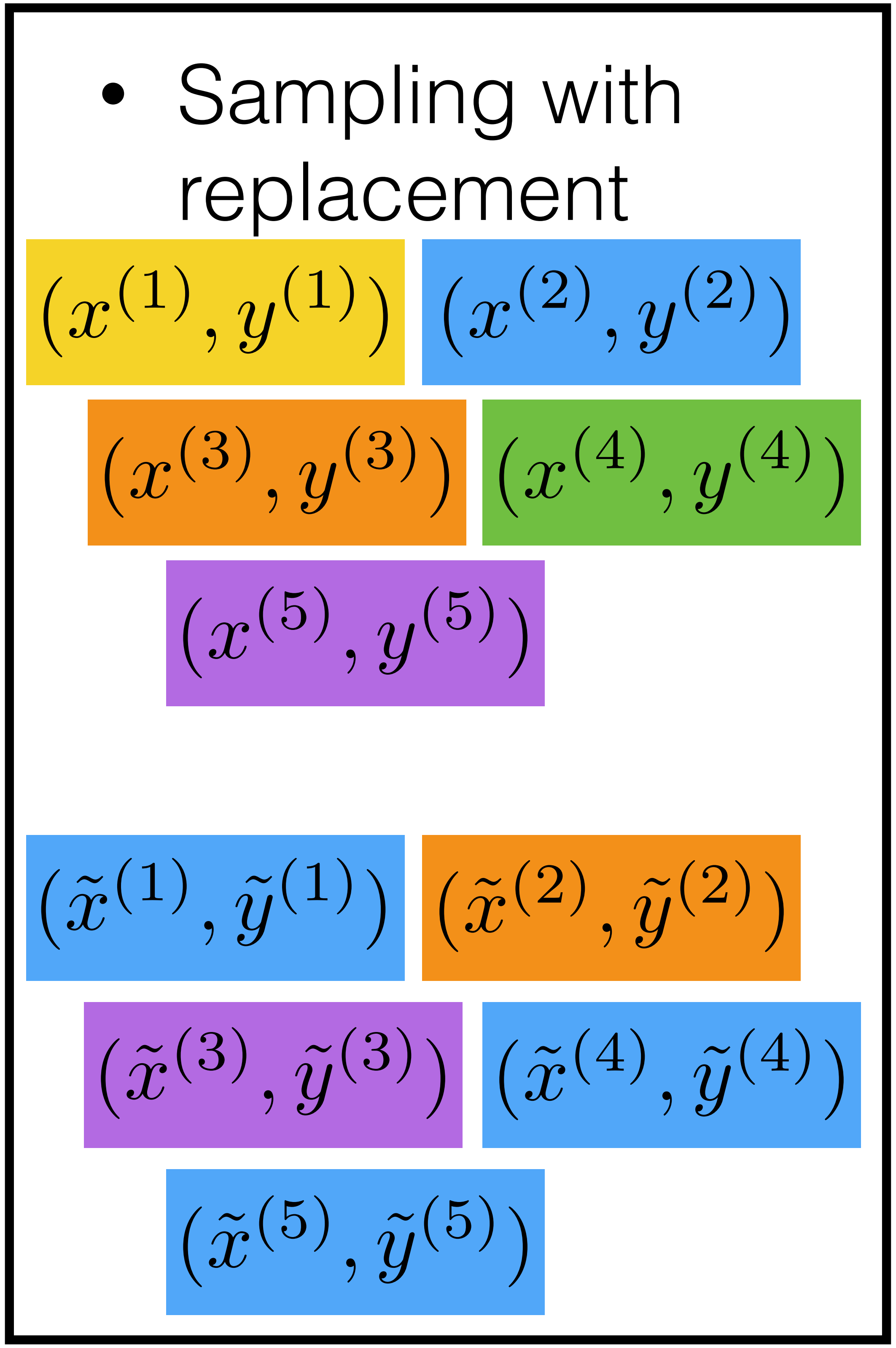
- Sampling with replacement

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$
$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$
$(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$
$(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$
$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Bagging

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

Bagging

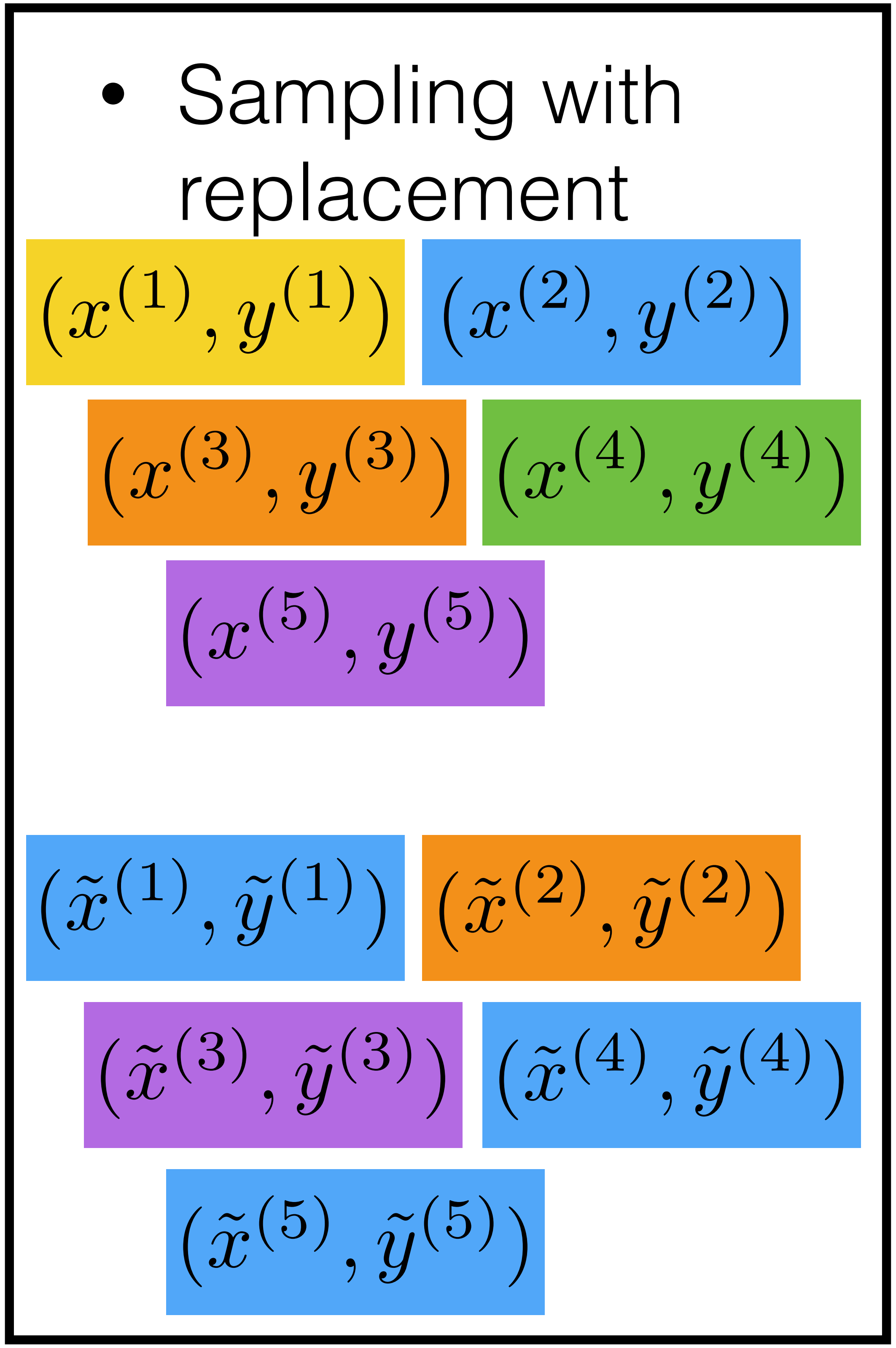
- One of multiple ways to make and use an ensemble
- Bagging = **B**ootstrap **agg**regat**ing**
  - Training data $\mathcal{D}_n$
  - For b = 1, …, B

bootstrap

    - 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

    - 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

$(x^{(1)}, y^{(1)})$ $(x^{(2)}, y^{(2)})$
$(x^{(3)}, y^{(3)})$ $(x^{(4)}, y^{(4)})$
$(x^{(5)}, y^{(5)})$

$(\tilde{x}^{(1)}, \tilde{y}^{(1)})$ $(\tilde{x}^{(2)}, \tilde{y}^{(2)})$
$(\tilde{x}^{(3)}, \tilde{y}^{(3)})$ $(\tilde{x}^{(4)}, \tilde{y}^{(4)})$
$(\tilde{x}^{(5)}, \tilde{y}^{(5)})$

Random forests

Random forests

- Bagging + decision trees + extra randomness

Random forests

- Bagging + decision trees + extra randomness
- Random forest

Random forests

- Bagging + decision trees + extra randomness
- Random forest
    - For b = 1, …, B

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For *b* = 1, …, *B*
    - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For *b* = 1, …, *B*
    - Draw a new "data set" $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$
    - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size *k* is reached:

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For *b* = 1, …, *B*
    - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$
    - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size *k* is reached:
      - Select *m* features uniformly at random, *without* replacement, from the *d* features

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For *b* = 1, …, *B*
    - Draw a new "data set" $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$
    - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size *k* is reached:
      - Select *m* features uniformly at random, *without* replacement, from the *d* features
      - Pick the best split dimension and split value among the *m* features

Random forests

- Bagging + decision trees + extra randomness
- Random forest
    - For *b* = 1, …, *B*
        - Draw a new "data set" $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$
        - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size *k* is reached:
            - Select *m* features uniformly at random, *without* replacement, from the *d* features
            - Pick the best split dimension and split value among the *m* features
            - Build two children

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For *b* = 1, …, *B*
    - Draw a new "data set" $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$
    - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size *k* is reached:
      - Select *m* features uniformly at random, *without* replacement, from the *d* features
      - Pick the best split dimension and split value among the *m* features
      - Build two children
  - Return: average for regression; vote for classification

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For $b = 1, \ldots, B$
    - Draw a new "data set" $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from $\mathcal{D}_n$
    - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size k is reached:
      - Select m features uniformly at random, *without* replacement, from the d features
      - Pick the best split dimension and split value among the m features
      - Build two children
  - Return: average for regression; vote for classification

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For *b* = 1, …, *B*
    - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$
    - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size *k* is reached:
      - Select *m* features uniformly at random, *without* replacement, from the *d* features
      - Pick the best split dimension and split value among the *m* features
      - Build two children
  - Return: average for regression; vote for classification

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For b = 1, …, B

    bagging

    - Draw a new "data set" $\tilde{\mathcal{D}}_n^{(b)}$ of size n by sampling *with* replacement from $\mathcal{D}_n$
    - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size k is reached:
      - Select m features uniformly at random, *without* replacement, from the d features
      - Pick the best split dimension and split value among the m features
      - Build two children

    bagging

  - Return: average for regression; vote for classification

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For *b* = 1, …, *B*

    bagging
    - Draw a new "data set" $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$

    trees
    - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size *k* is reached:
      - Select *m* features uniformly at random, *without* replacement, from the *d* features
      - Pick the best split dimension and split value among the *m* features
      - Build two children

  bagging
  - Return: average for regression; vote for classification

Random forests

- Bagging + decision trees + extra randomness
- Random forest
  - For b = 1, …, B

    bagging

    - Draw a new “data set” $\tilde{\mathcal{D}}_n^{(b)}$ of size *n* by sampling *with* replacement from $\mathcal{D}_n$

    trees

    - Build a tree on $\tilde{\mathcal{D}}_n^{(b)}$ by recursively repeating the following until minimum node size *k* is reached:
      - Select *m* features uniformly at random, *without* replacement, from the *d* features
      - Pick the best split dimension and among the *m* features
      - Build two children

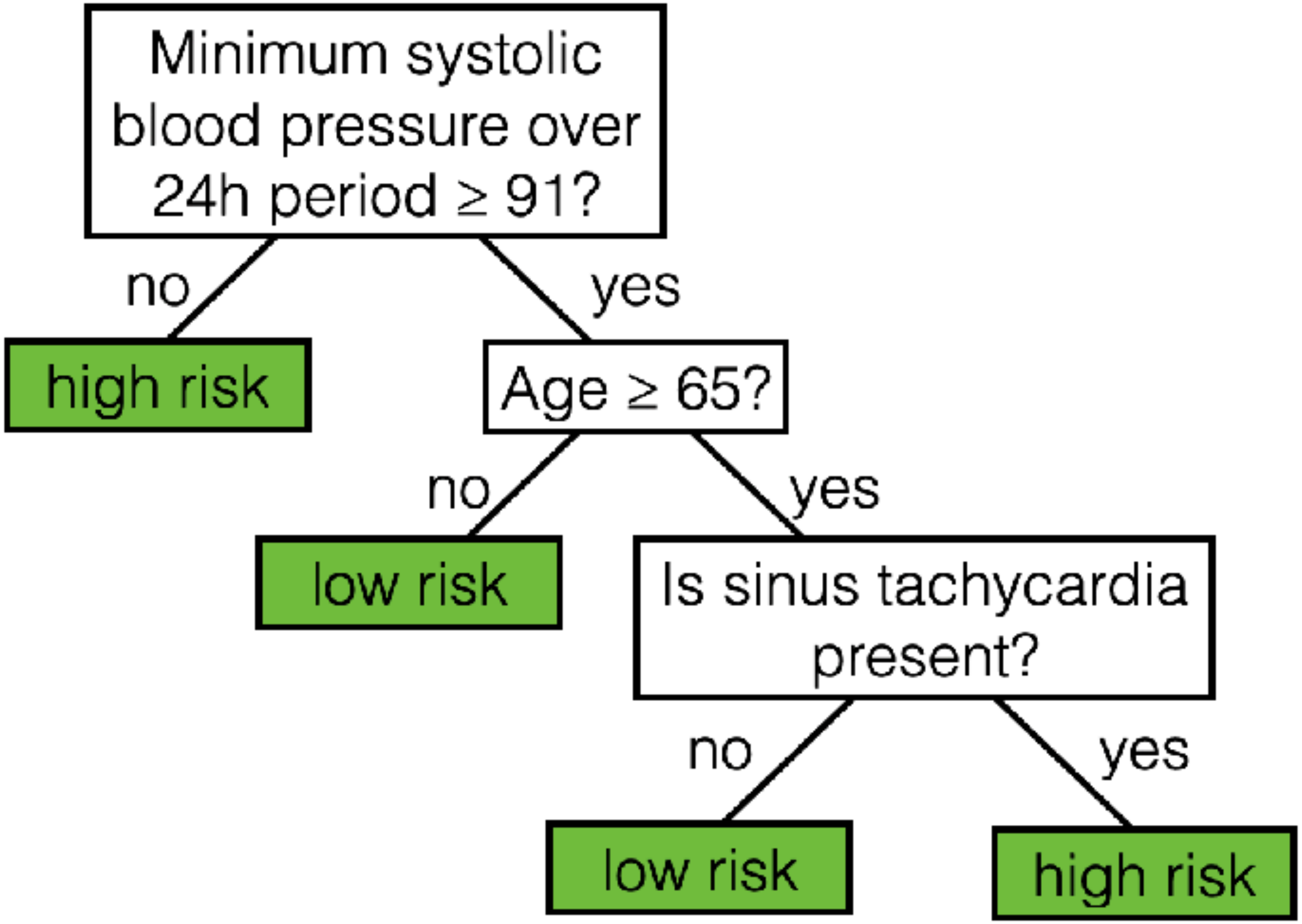
      extra randomness

  bagging

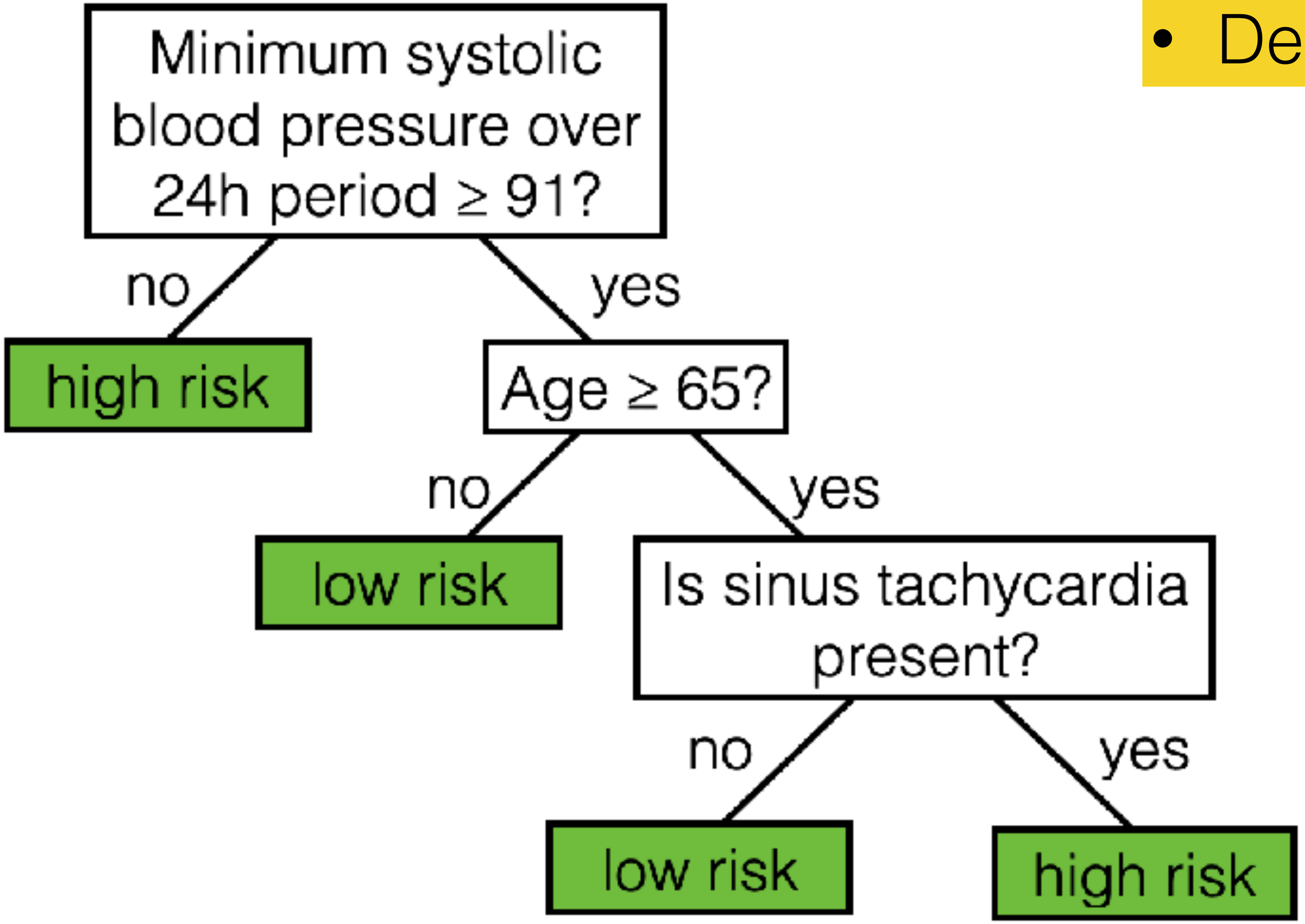
  - Return: average for regression; vote for classification

Decision trees & random forests

Decision trees & random forests

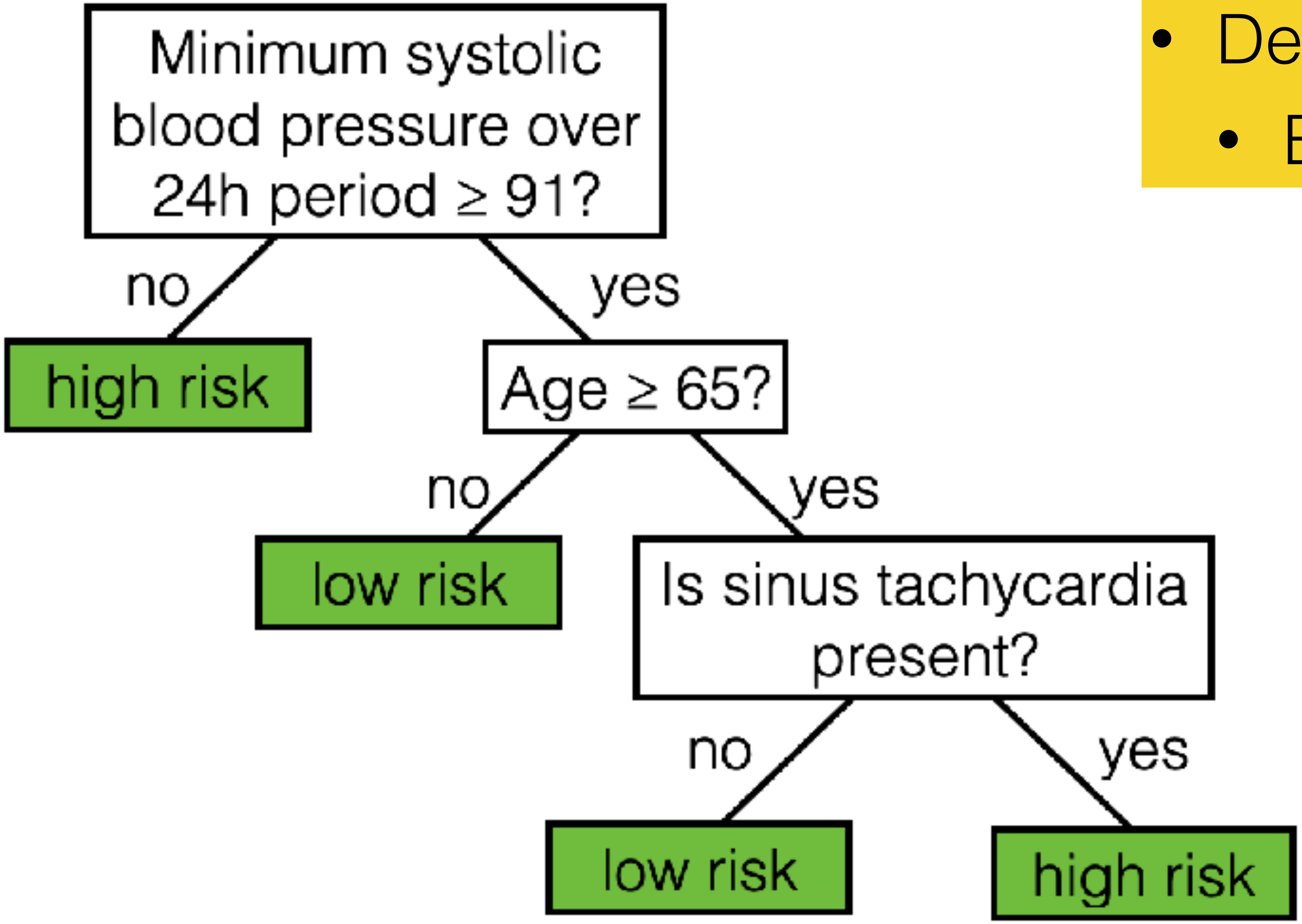


Decision trees & random forests



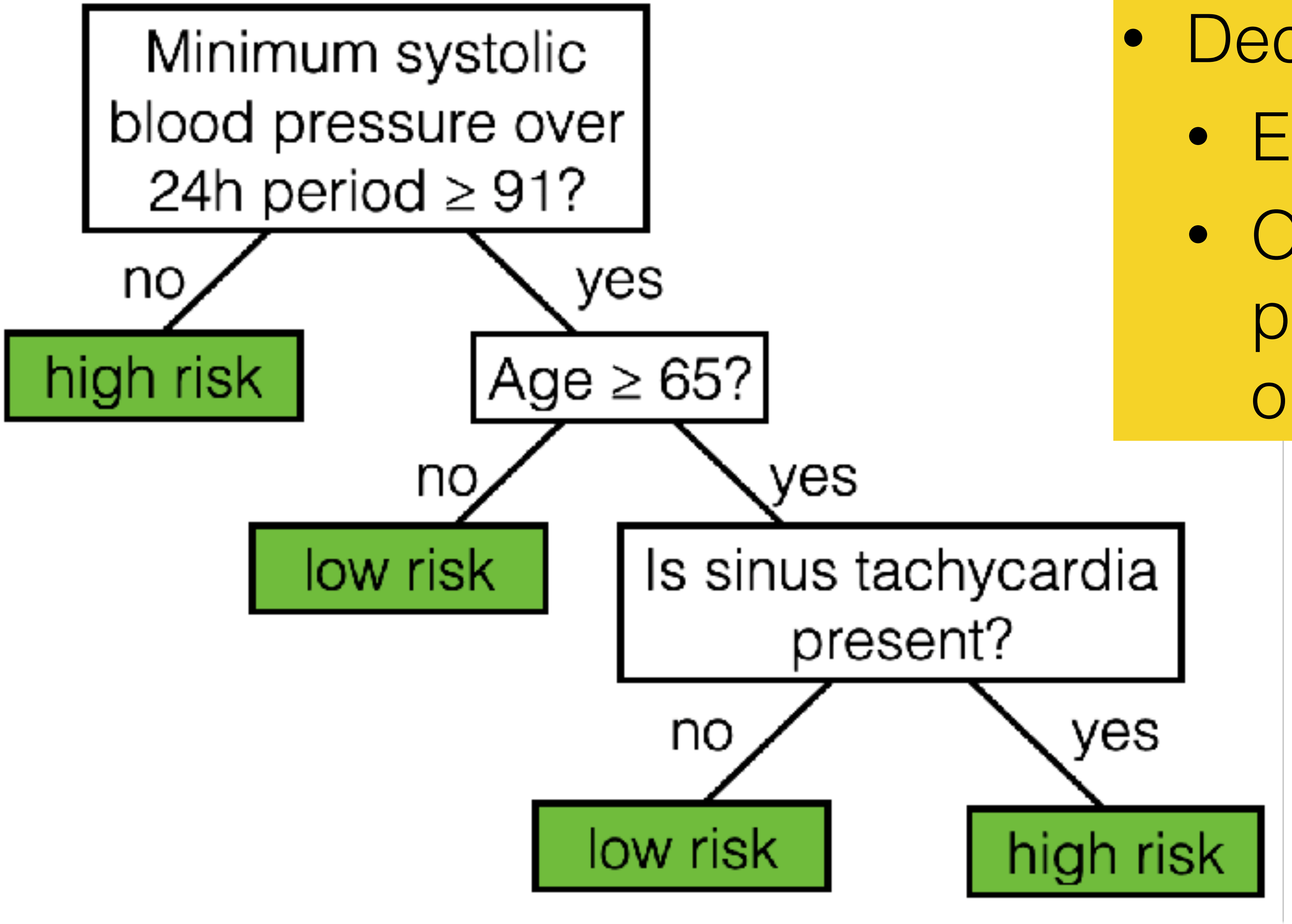
- Decision trees

Decision trees & random forests



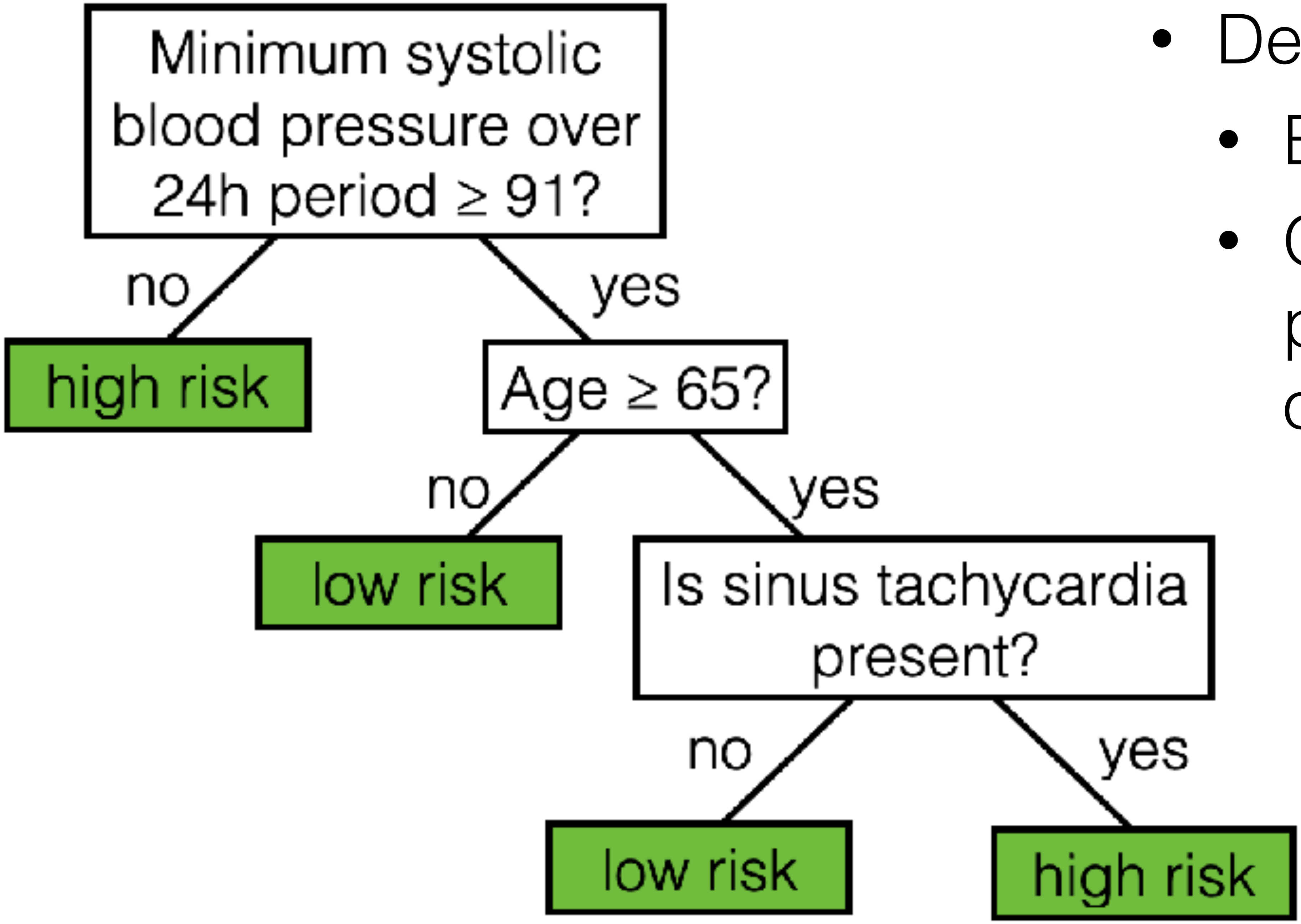
- Decision trees
  - Easy to interpret

Decision trees & random forests



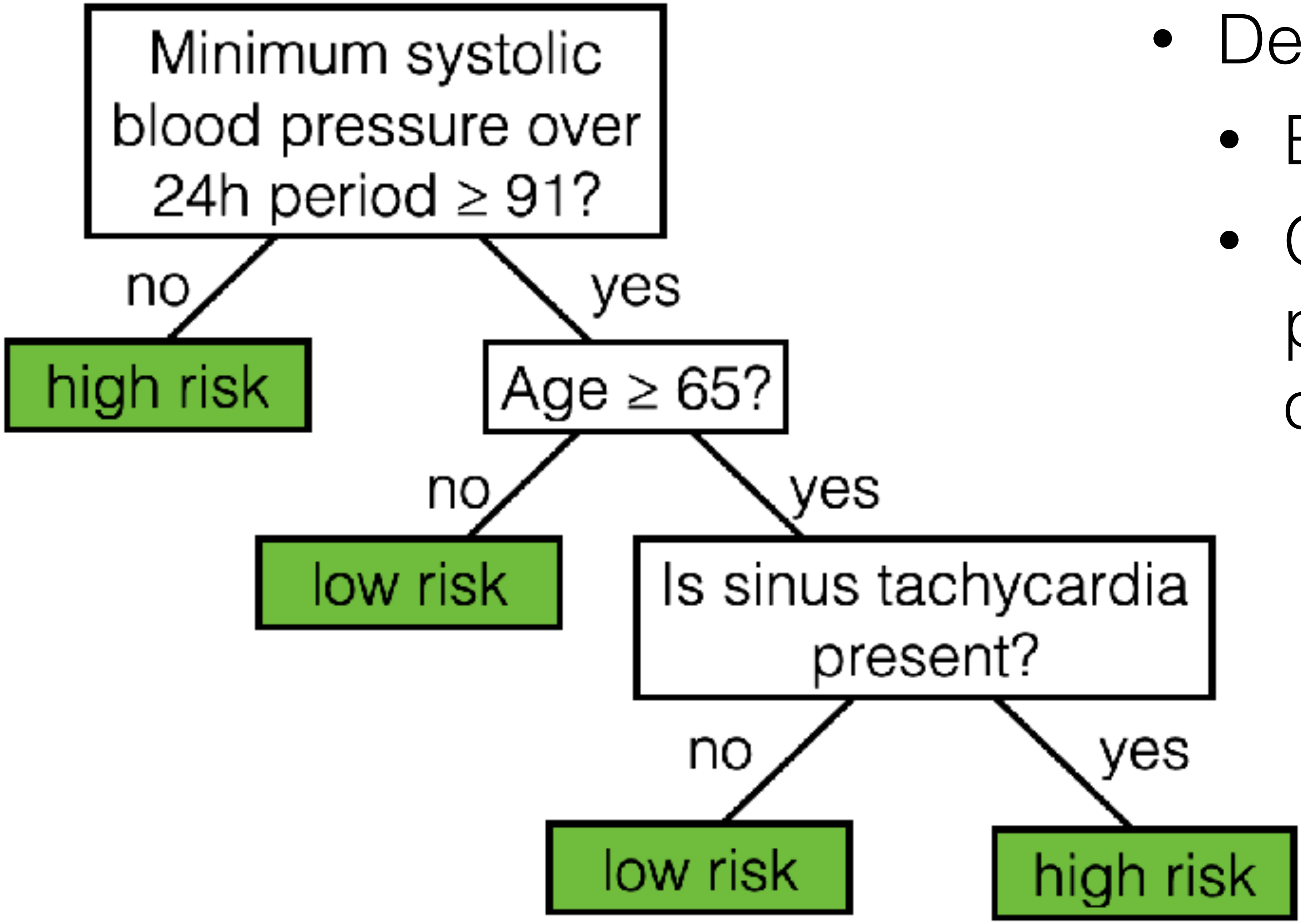
- Decision trees
  - Easy to interpret
  - Often not the best predictions (error on new data)

Decision trees & random forests



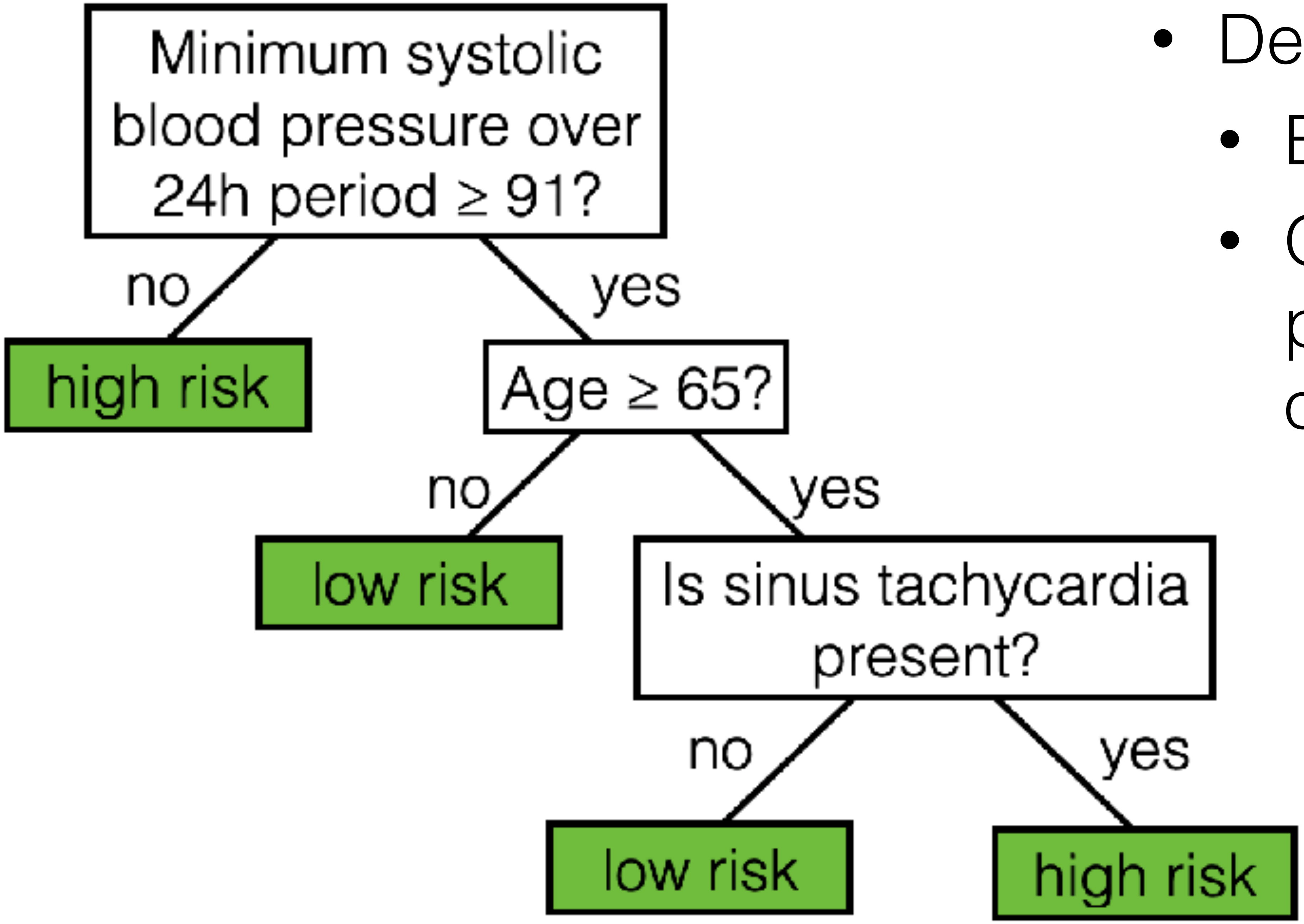
- Decision trees
  - Easy to interpret
  - Often not the best predictions (error on new data)
- Random forests/ensembling

Decision trees & random forests



- Decision trees
  - Easy to interpret
  - Often not the best predictions (error on new data)
- Random forests/ensembling
  - Harder to interpret

Decision trees & random forests



Minimum systolic blood pressure over 24h period ≥ 91?

no

high risk

yes

Age ≥ 65?

no

low risk

yes

Is sinus tachycardia present?

no

low risk

yes

high risk

- Decision trees
    - Easy to interpret
    - Often not the best predictions (error on new data)

- Random forests/ensembling
    - Harder to interpret
    - Often much better predictions