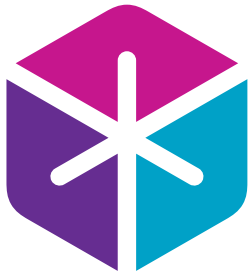
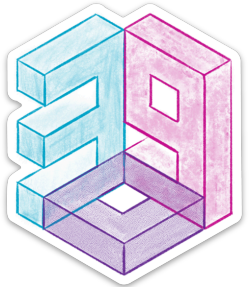




https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 12: Non-parametric Models

Shen Shen

May 4, 2026

3pm, Room 10-250

Slides and Lecture Recording

Outline

- Non-parametric models overview
- Similarity-based methods
  - k-nearest neighbor
  - k-means clustering
- Tree-based methods
  - Decision Tree
  - Bagging and ensembles

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

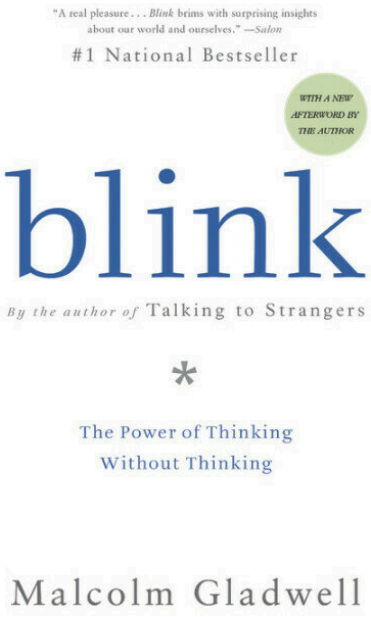
Yin Lou
LinkedIn Corporation
ylou@linkedin.com

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



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

kagg

[source]

What non-parametric methods buy you

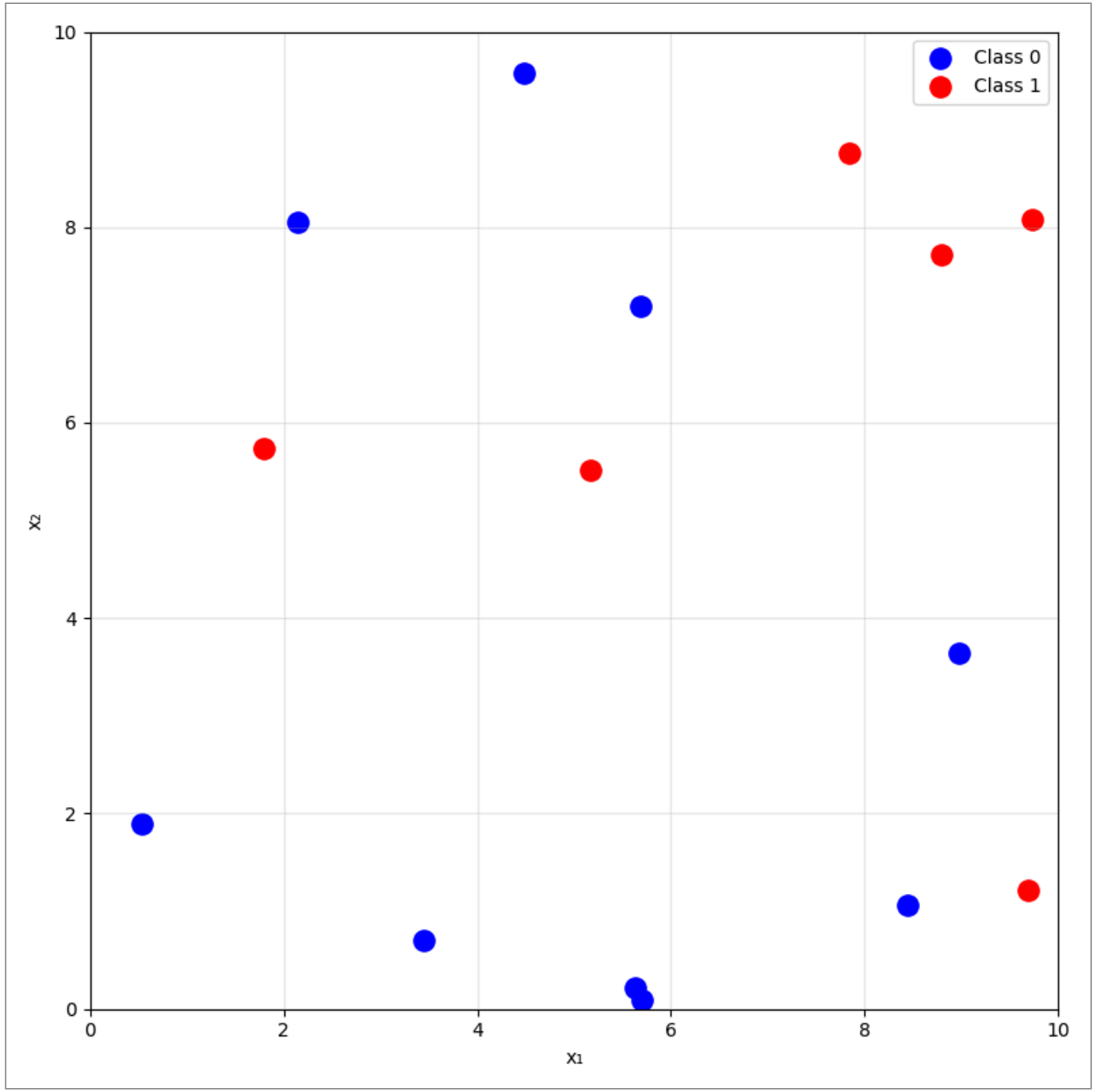
- **Interpretability**: a stakeholder, regulator, or doctor can read the rule. Auditable for medicine, lending, hiring.
- **Insight**: discover structure in the data, not just predictions about it. Especially when there are no labels.
- **Speed**: a strong baseline in minutes, not GPU-hours. The right first reach for sanity-checks.
- **Adaptivity**: complexity grows with the data, no retraining from scratch. Trees still win Kaggle on small or tabular data.

Often the right first reach. Sometimes the only one.

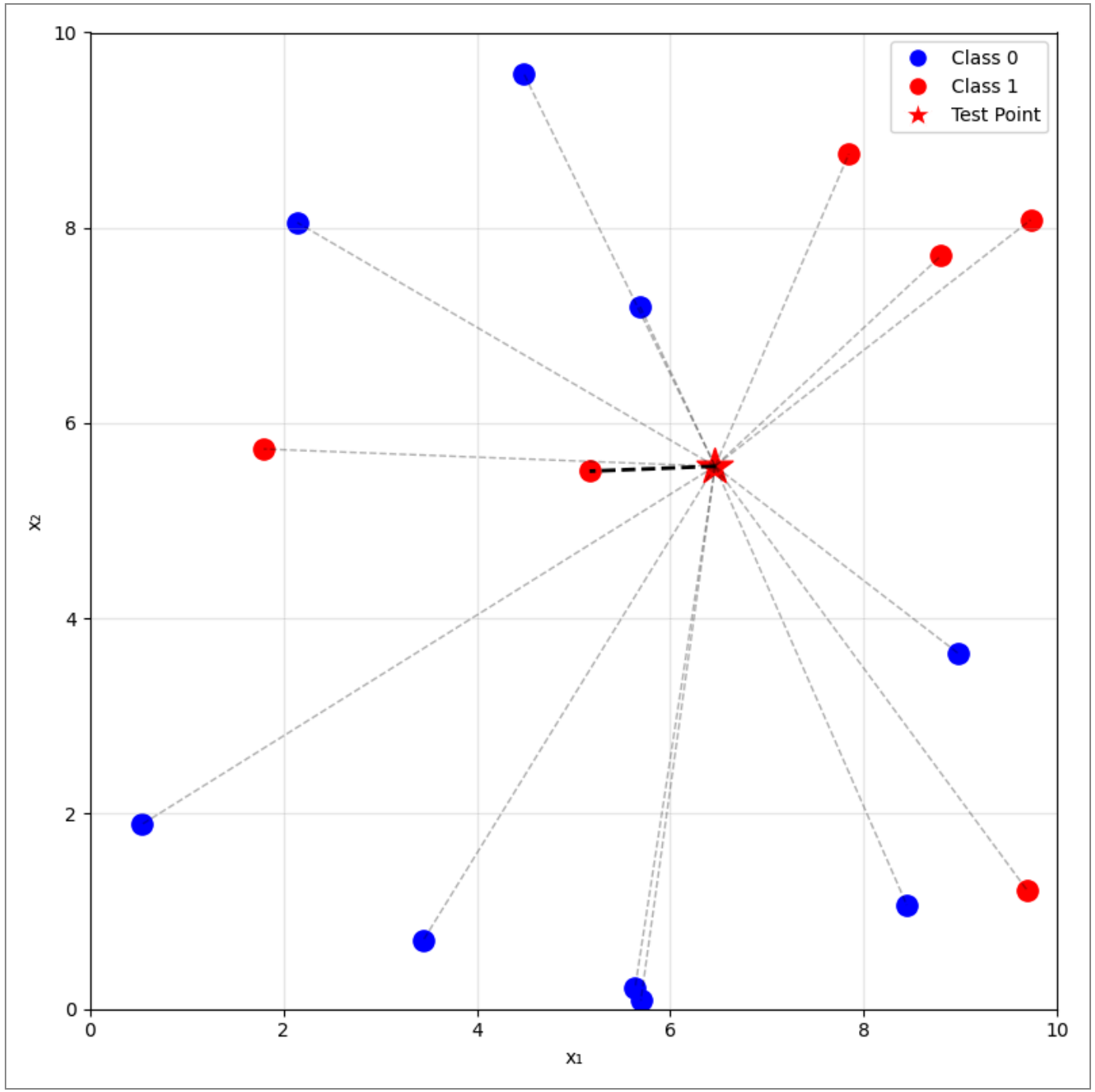
Outline

- Non-parametric models overview
- Similarity-based methods
  - k-nearest neighbor
  - k-means clustering
- Tree-based methods
  - Decision Tree
  - Bagging and ensembles

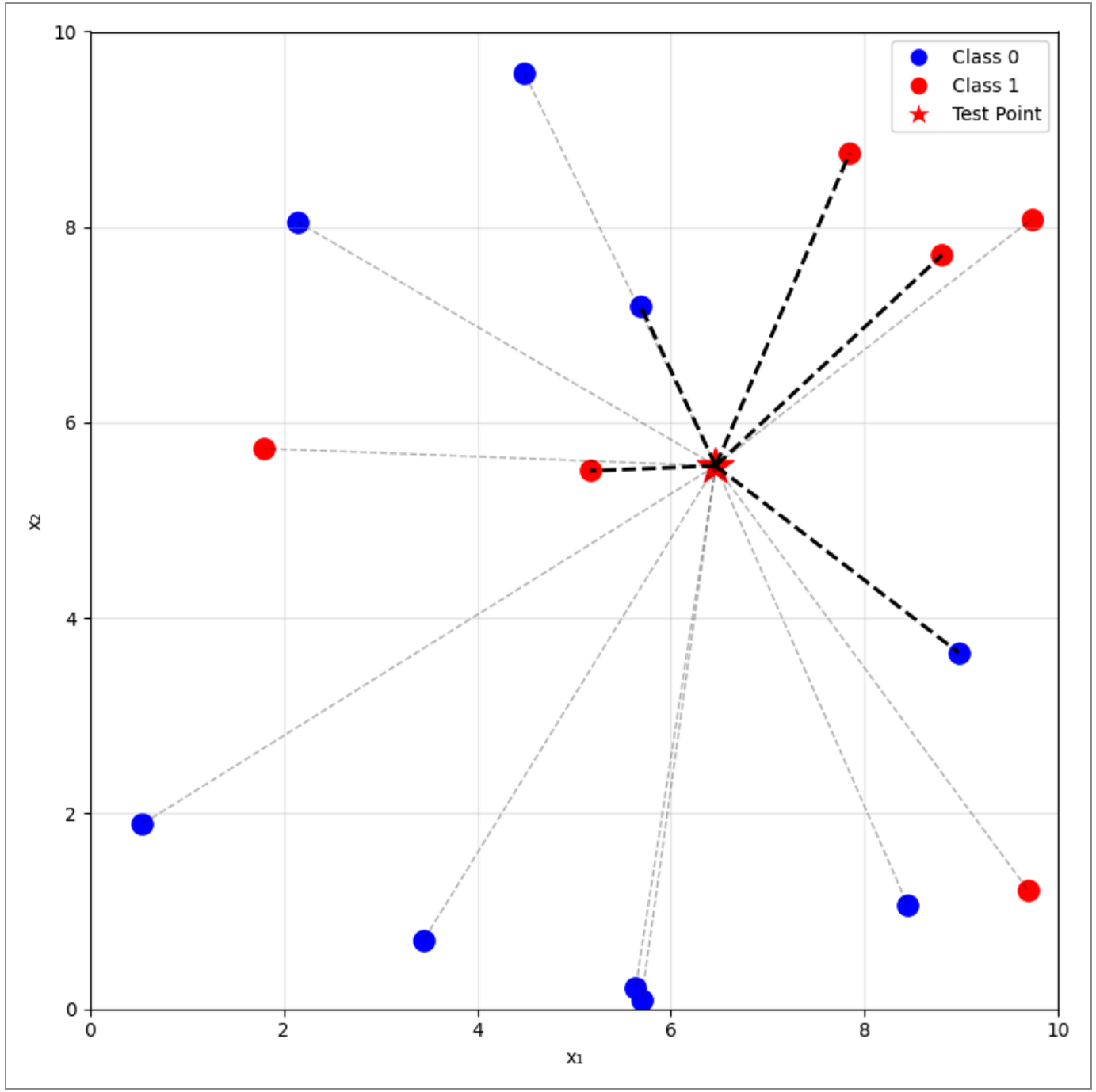
training data

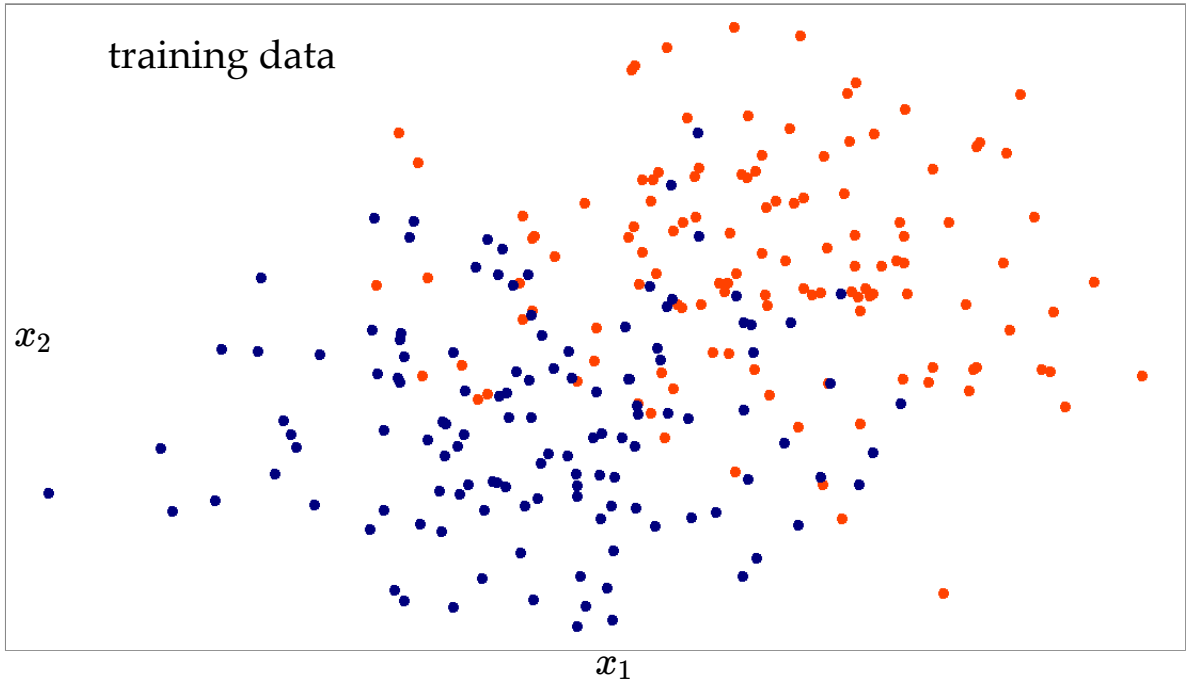


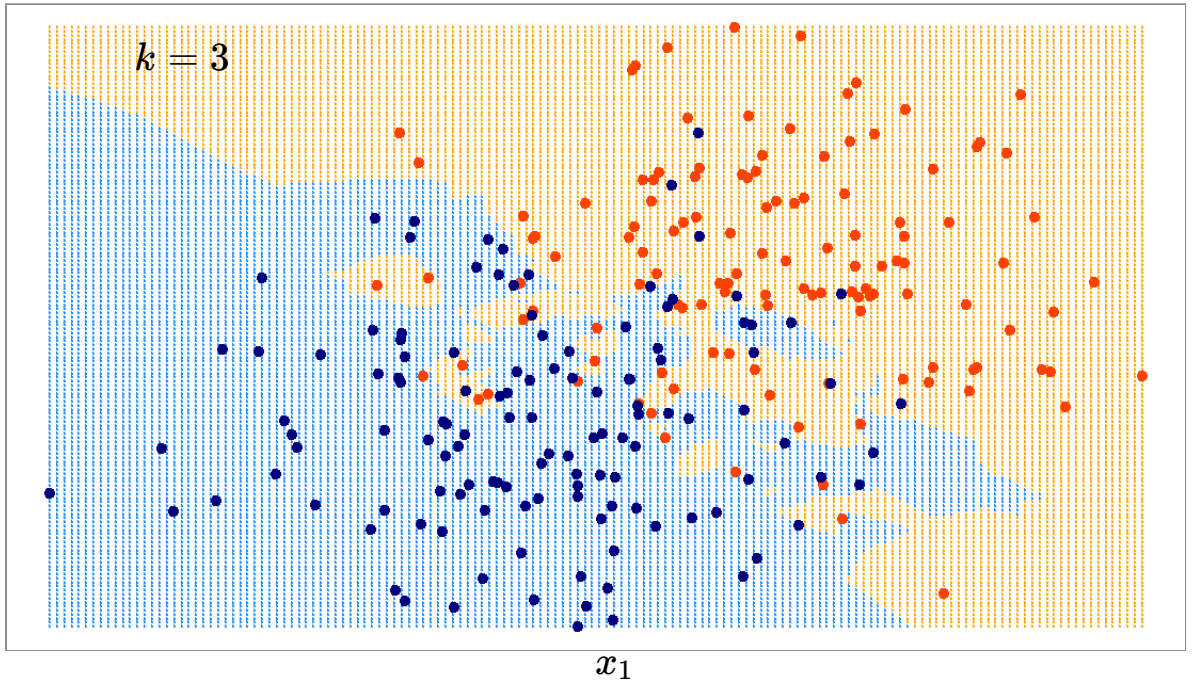
prediction on new data using $k = 1$ nearest neighbor

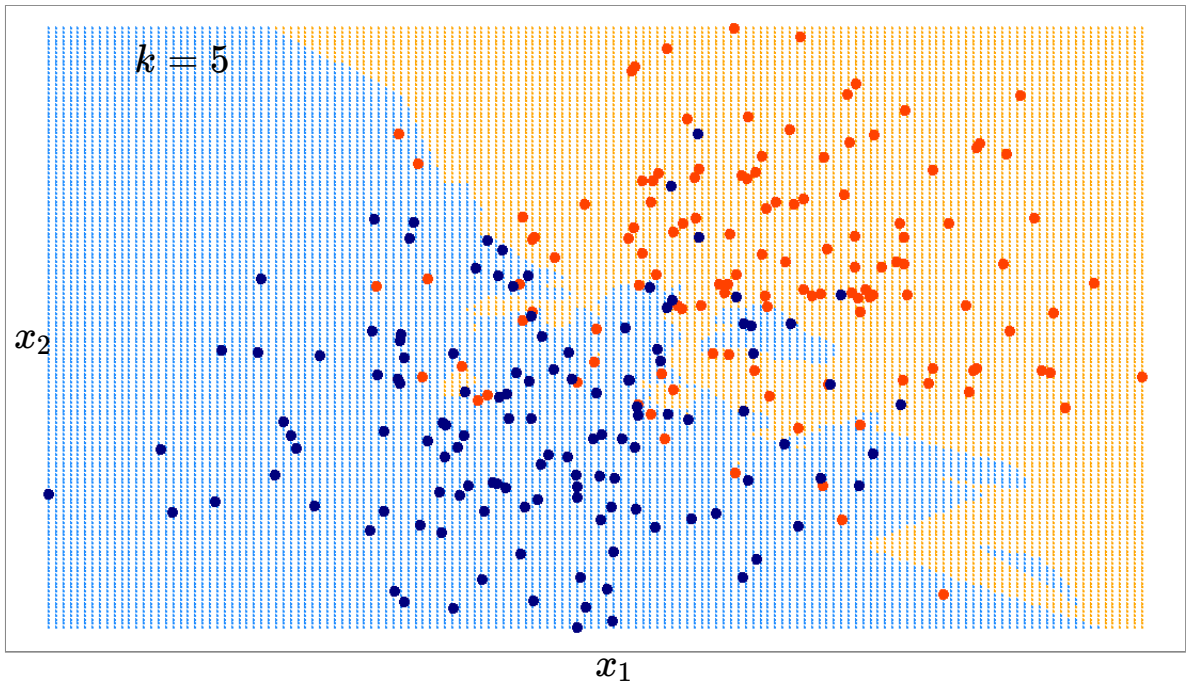


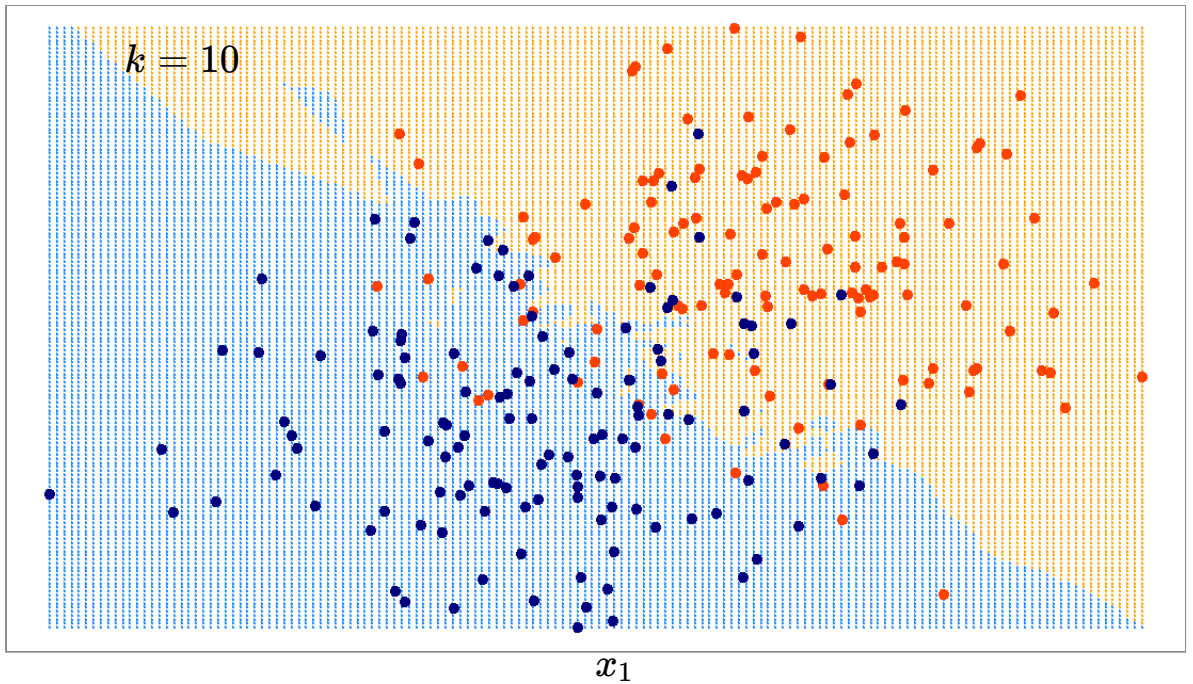
prediction on new data using $k = 5$ nearest neighbor




training data
x_2
x_1

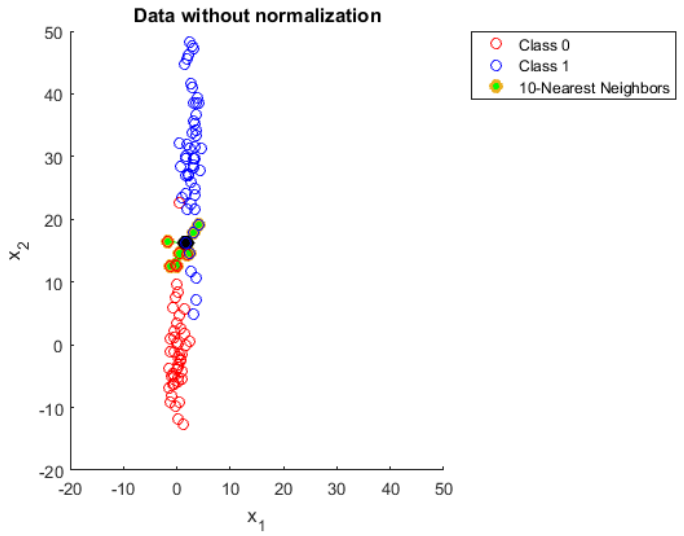

$k = 3$
x_2
x_1


$k = 5$
x_2
x_1

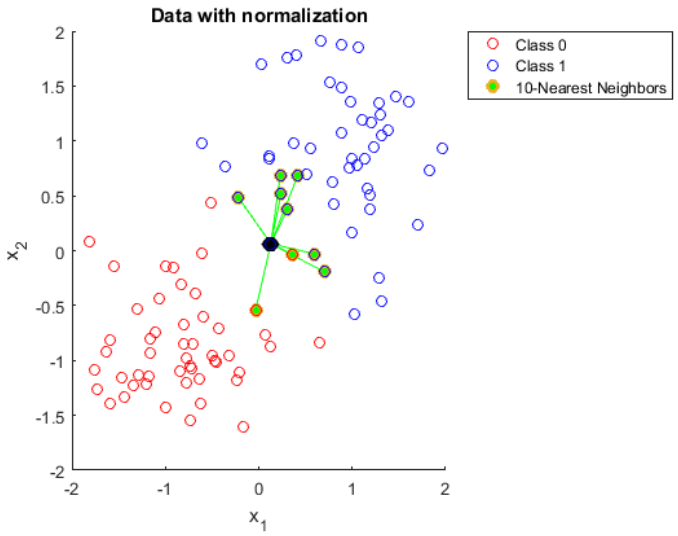

$k = 10$
x_2
x_1

Be cautious when...

- very large n or d; curse of dimensionality.
- many irrelevant or noisy features.
- features on different scales:



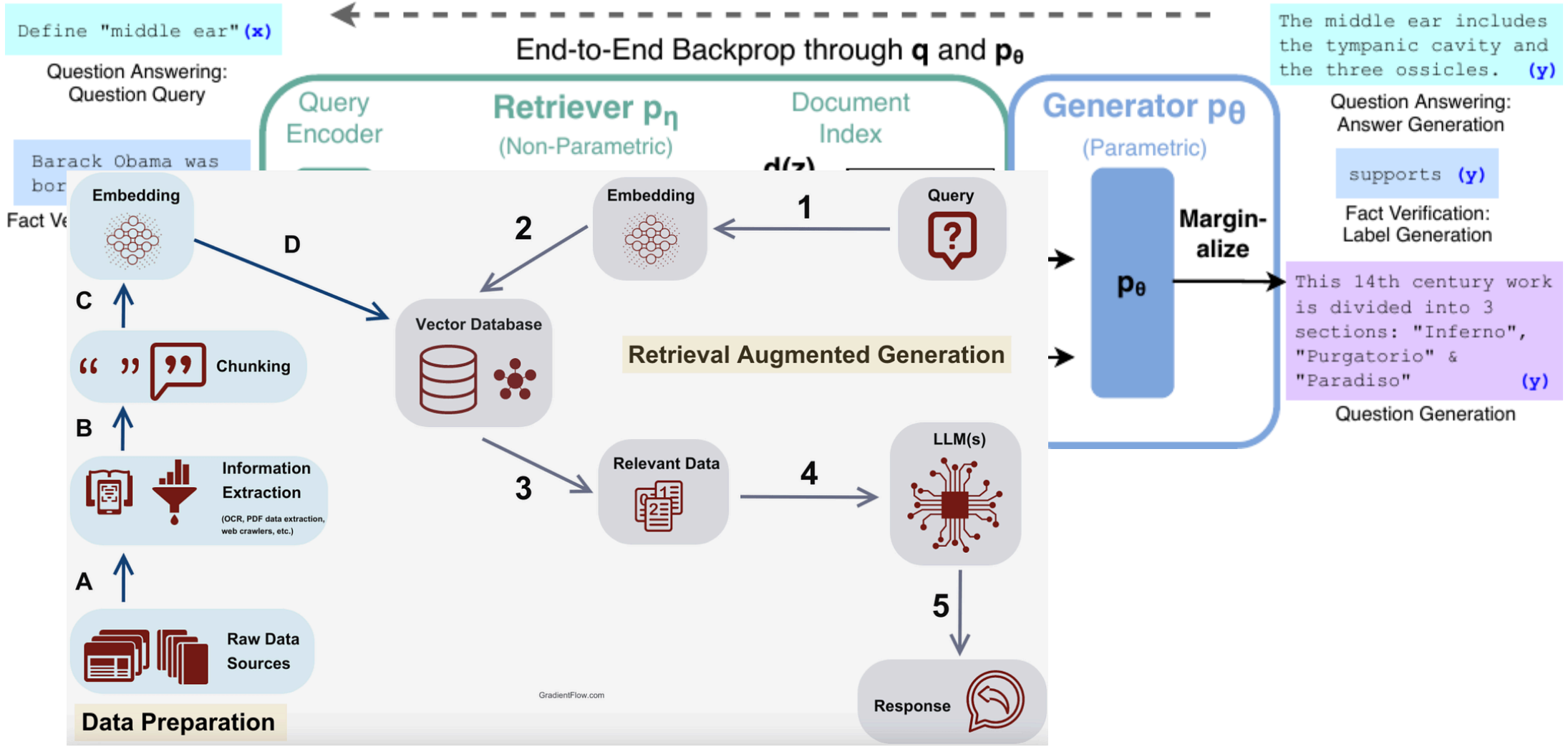
unscaled



scaled

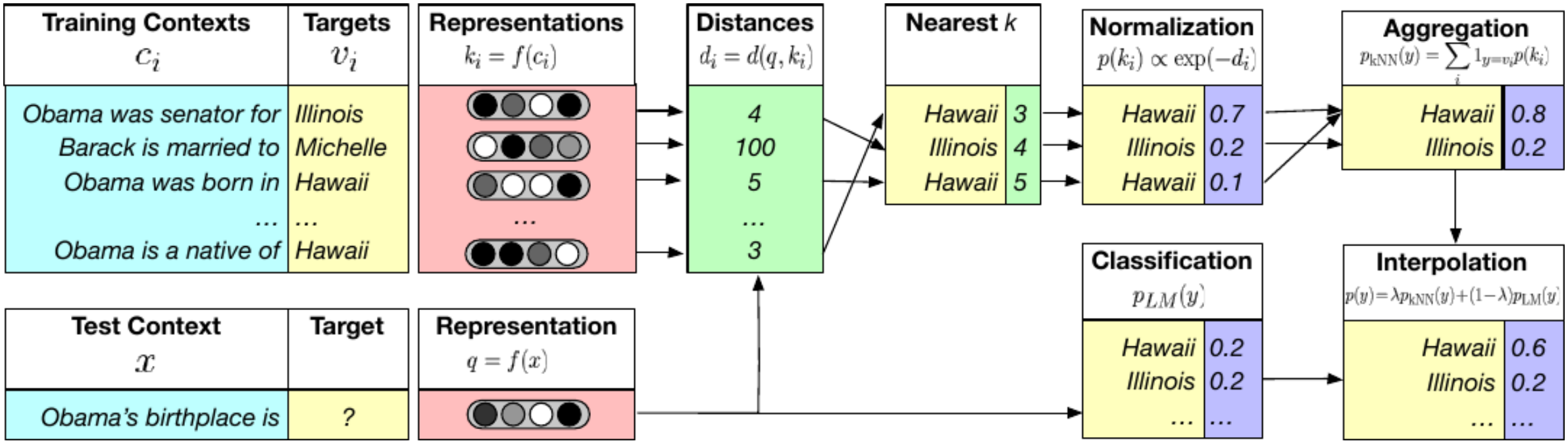
[source]

Equip modern language models with a knowledge base



Lewis, Perez, Piktus, et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* NeurIPS 2020.

Memorization, attached to the model from outside.



- Build a datastore (key and values) by running one. LM over the entire training corpus.
- At decode time, do k-NN of one query against the datastore keys. The k retrieved values give a kNN distribution over next tokens.
- Interpolate the kNN results with the regular next-token results.

Khandelwal, Levy, Jurafsky, Zettlemoyer, Lewis. *Generalization through Memorization: Nearest Neighbor Language Models.* ICLR 2020.

Training: none. Just memorize the training set.

Predicting: for a new x_{new}, take the majority (class) or mean (regression) label of its k nearest neighbors.

Parameters learned: the entire training set (its features and labels).

Hyperparameters:

- k: number of neighbors considered.
- distance metric (typically Euclidean or Manhattan).
- tie-breaking scheme (typically at random).

All ML models *learn* parameters; not all are *parametric*.

"Having **parameters**" ≠ "Being **parametric**"

just as

"Having **mechanisms**" ≠ "Being **mechanical**"

e.g. (electronic, chemical, biological) systems also have mechanisms; but they are not mechanical.

Non-parametric: the model doesn't assume a fixed functional form for *h;* complexity grows with the data.

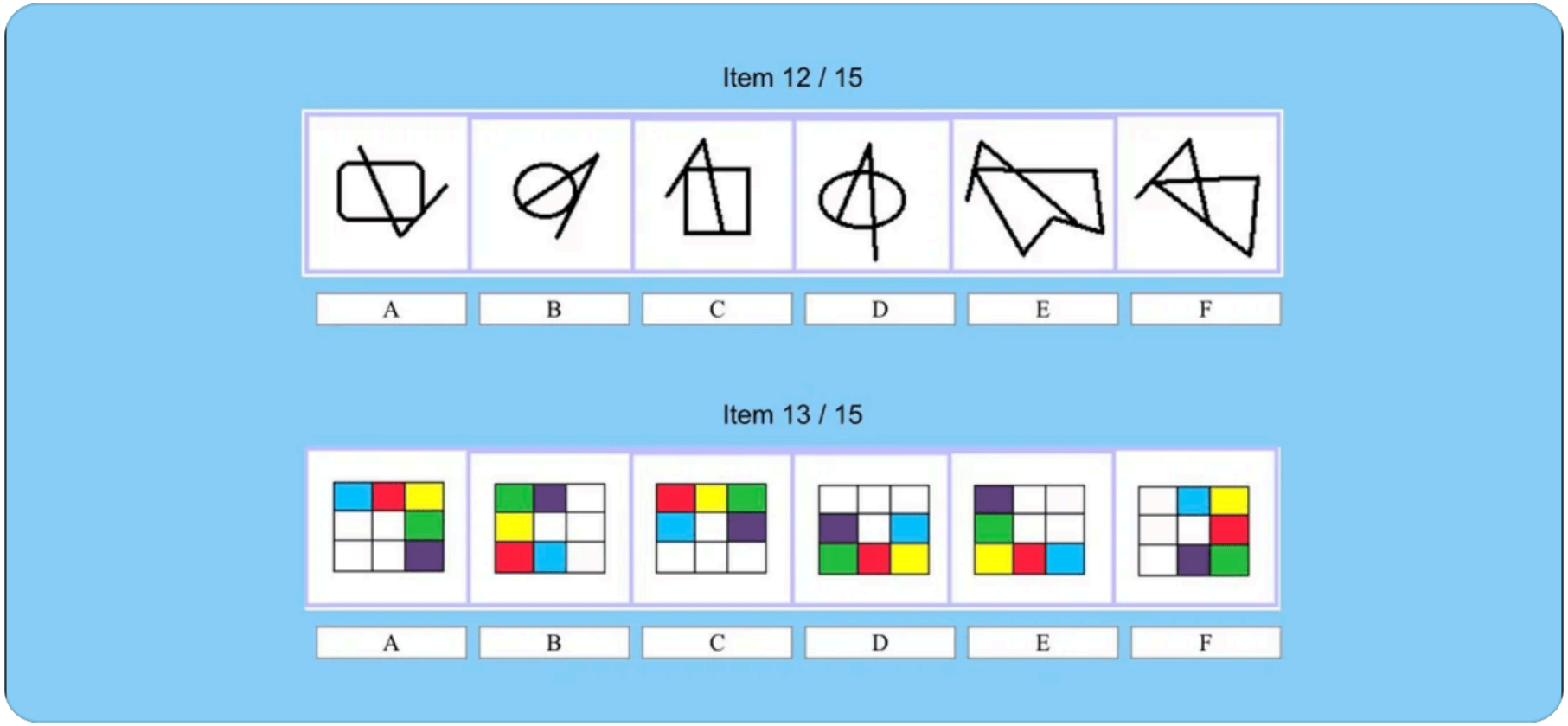
Outline

- Non-parametric models overview
- Similarity-based methods
  - k-nearest neighbor
  - k-means clustering
- Tree-based methods
  - Decision Tree
  - Bagging and ensembles

No labeled categories, we can still group by similarities.

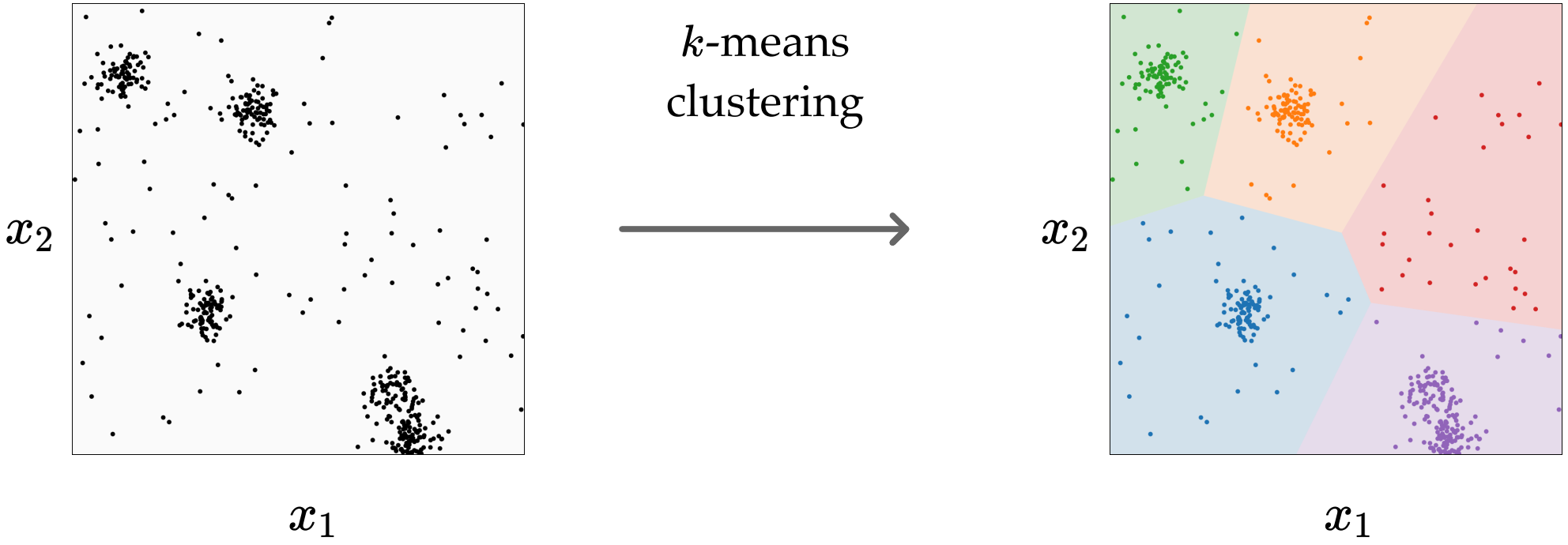
r/mensa • 6y ago
Spare-Research

How would you solve these; which two don't belong?

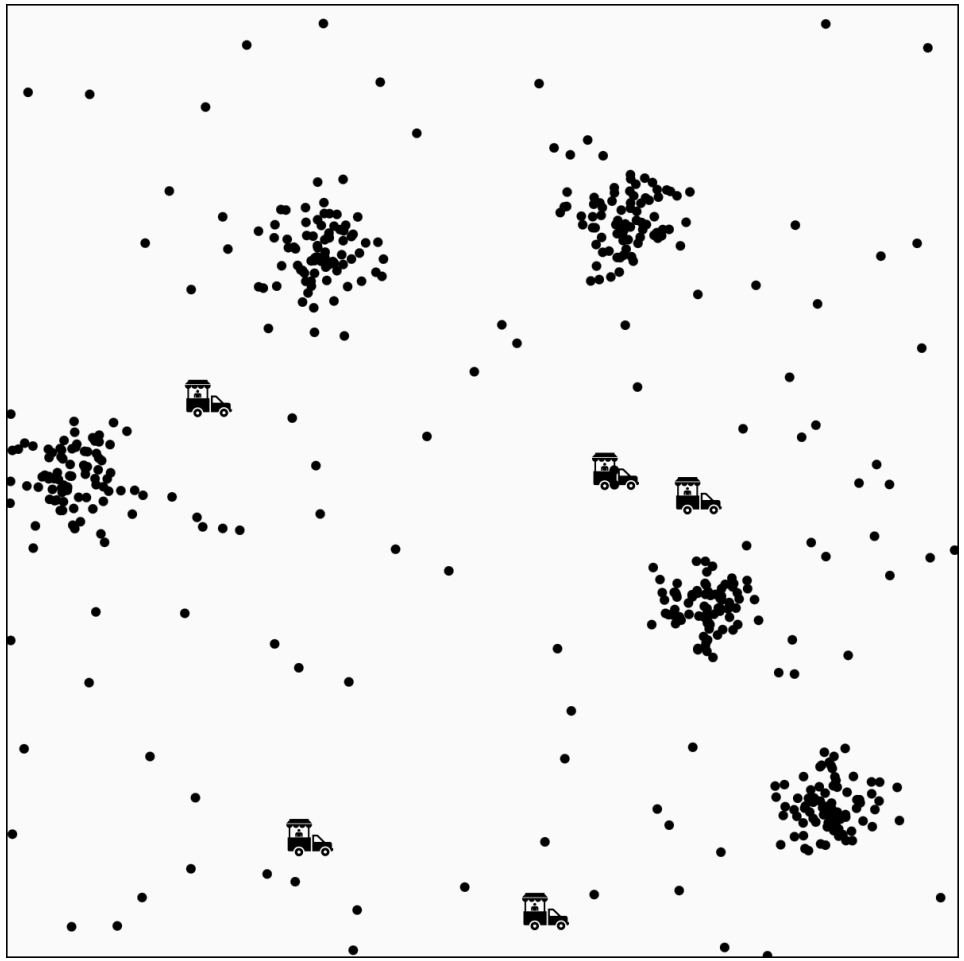


(I'm actually not sure of the answers..)

structure discovery without labels

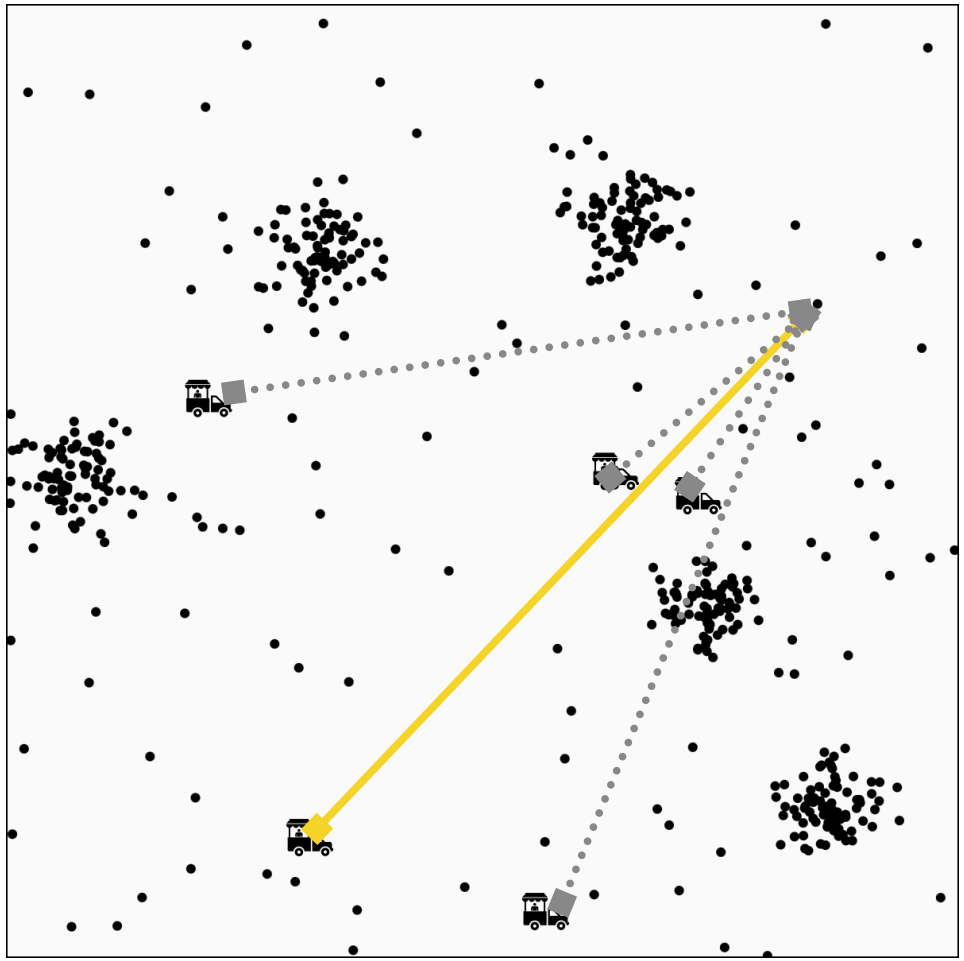


Food-truck placement



- x_1: longitude, x_2: latitude
- n people, person i at location $x^{(i)}$
- k food trucks (say $k = 5$)
- truck j at location $\mu^{(j)}$

Per-person loss



Cost for person i to walk to truck j:

$$\left\|x^{(i)} - \mu^{(j)}\right\|^2$$

Let $y^{(i)}$ be the truck person i walks to.

$\mathbf{1}\{y^{(i)} = j\}$ is 1 if assigned, 0 otherwise.

Person i's loss:

$$\sum_{j=1}^{k} \mathbf{1}\{y^{(i)} = j\} \left\|x^{(i)} - \mu^{(j)}\right\|^2$$

k-means objective

cluster membership

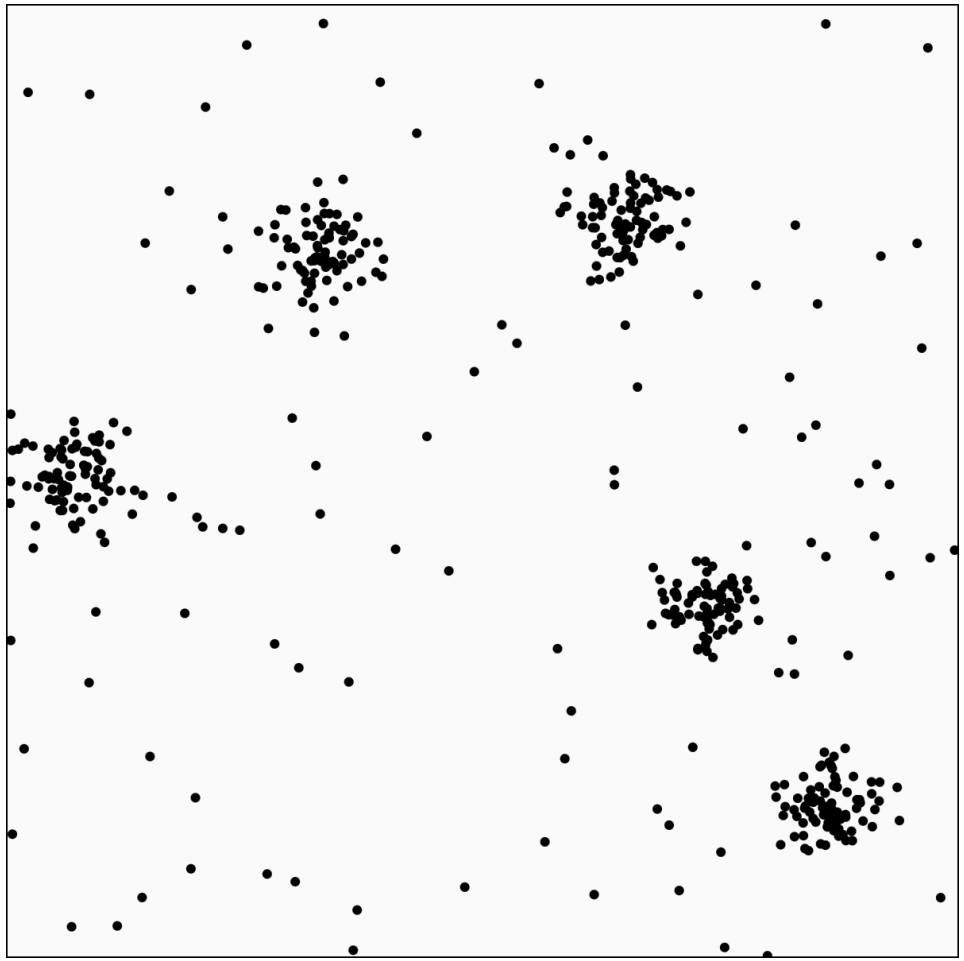
centroid location

$$\sum_{i=1}^{n} \sum_{j=1}^{k} \mathbf{1}\left\{y^{(i)} = j\right\} \left\|x^{(i)} - \mu^{(j)}\right\|^2$$

sum over data points

sum over cluster

what we learn



K-MEANS$(k, \tau, \{x^{(i)}\}_{i=1}^{n})$

$\mu, y =$ random initialization

for $t = 1$ to τ

    $y_{\text{old}} = y$

    for $i = 1$ to n

        $y^{(i)} = \arg\min_j \left\| x^{(i)} - \mu^{(j)} \right\|^2$

    for $j = 1$ to k

        $\mu^{(j)} = \frac{1}{N_j} \sum_{i=1}^{n} \mathbf{1}\left(y^{(i)} = \mathrm{j}\right) x^{(i)}$

    if $y == y_{\text{old}}$

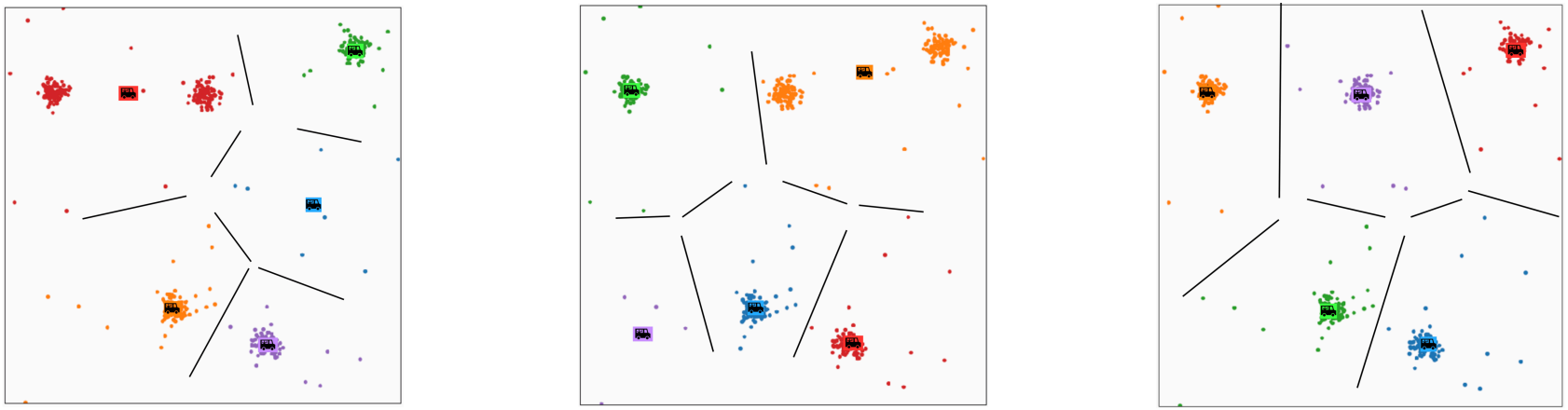
        break

return μ, y

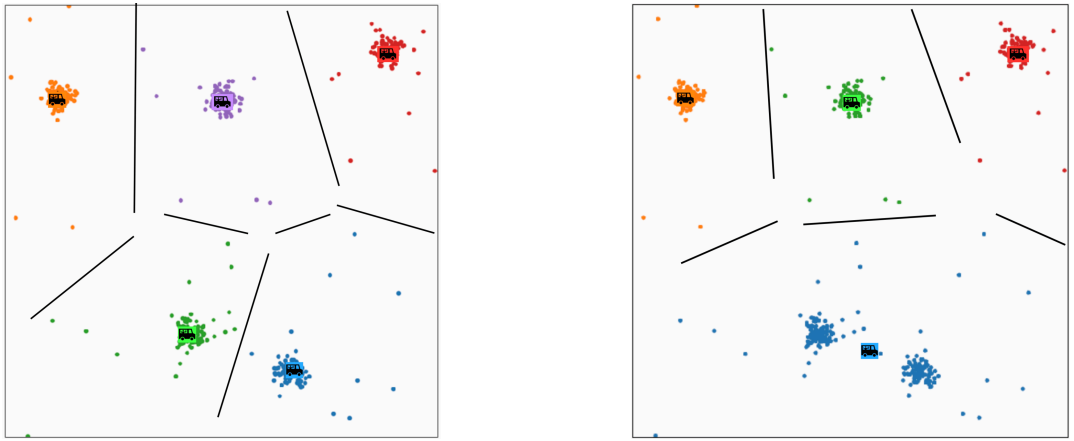
https://shenshen.mit.edu/demos/kmeans/kmeans-stepped.html

k-means is sensitive to initialization and to k

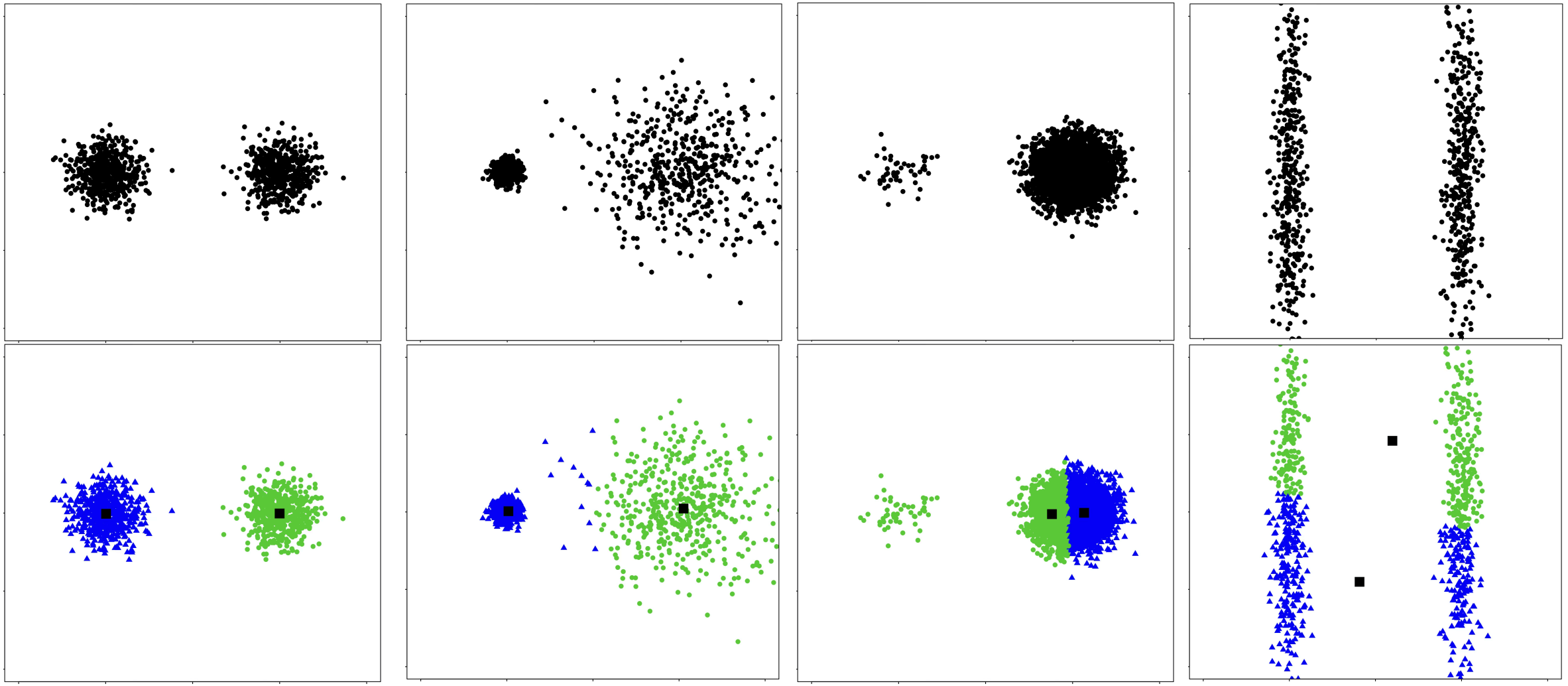
same data, three different initializations:



same data, two different choices of k:

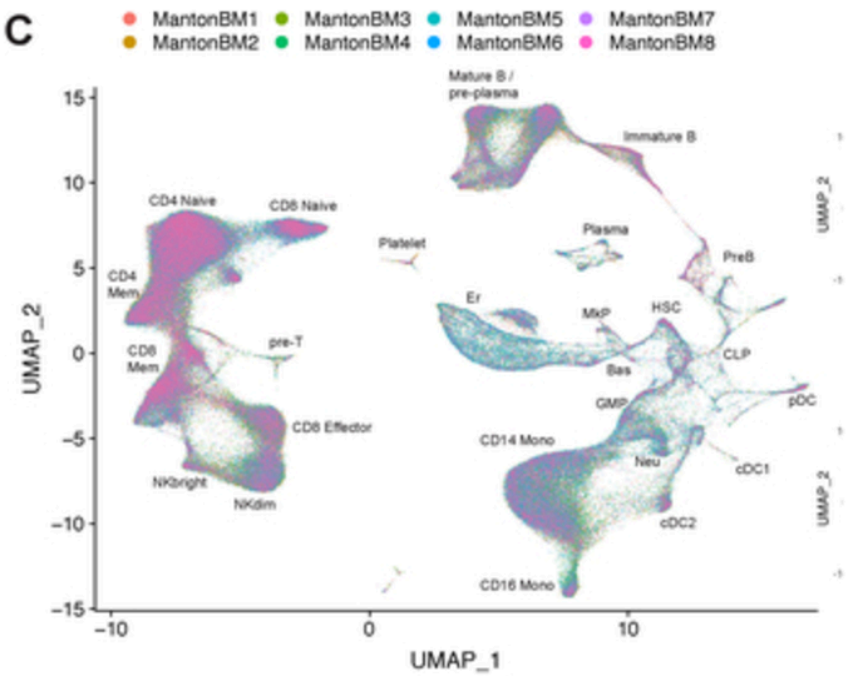


(k-means works well for well-separated circular clusters of the same size)



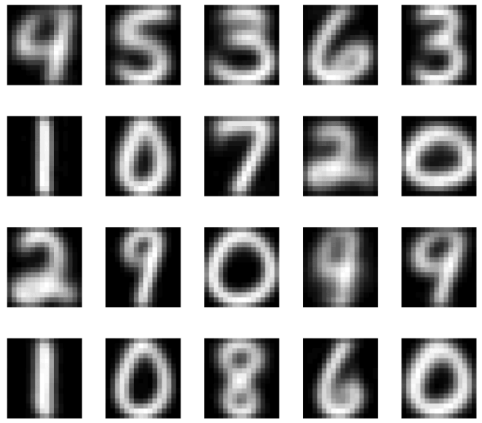
An unsupervised microscope, run in feature space.

- Each cell becomes a high-dim RNA-expression vector.
- Clustering those vectors uncovers cell types, **including** ones we didn't know existed.
- Standard tools builds this in. Routine in tissue atlases (Human Cell Atlas, Tabula Muris).



Stuart, Butler, Hoffman, et al. *Comprehensive Integration of Single-Cell Data.* Cell, 2019 (Seurat v3).

k-means interpreted as a discrete autoencoder

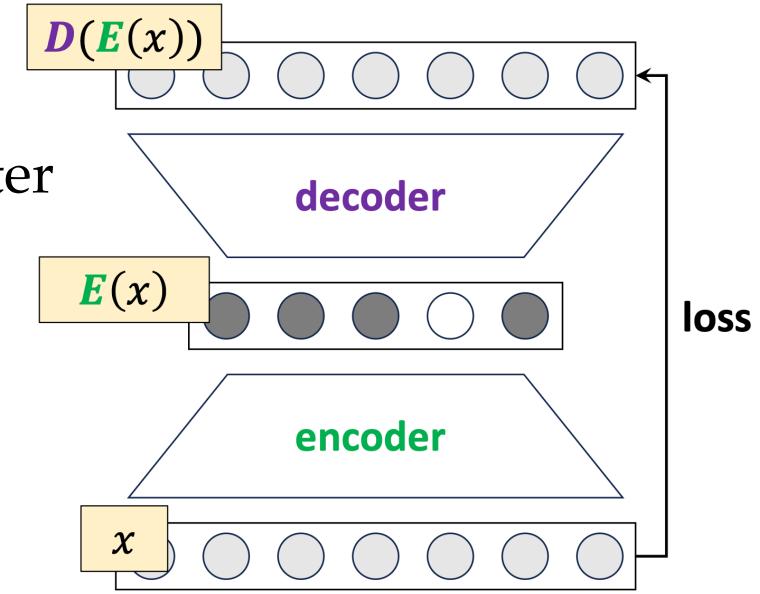


cluster center:

- not actual data points in MNIST
- learned representation and prototypes

map the one-hot to a center

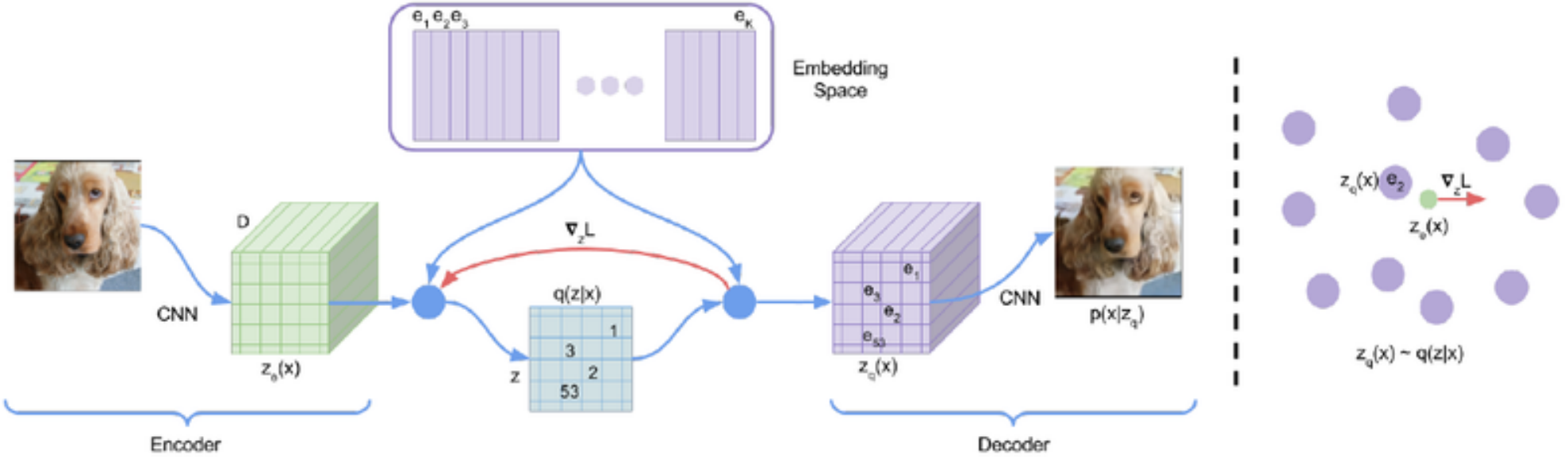
map x to a one-hot



$$\min_{D,E} \sum_x \|x - D(E(x))\|^2$$

Codebooks inside generative models

- **VQ-VAE** quantizes each encoder output to its nearest codebook vector. The codebook is exactly a set of k centroids learned from the data.
- Same idea elsewhere: **VQGAN**, **DALL-E** for images; **SoundStream**, **EnCodec** (used in **MusicGen**) for audio.



van den Oord, Vinyals, Kavukcuoglu. *Neural Discrete Representation Learning*. NeurIPS 2017.

Group similar things, even when "similar" is a model's internal state.

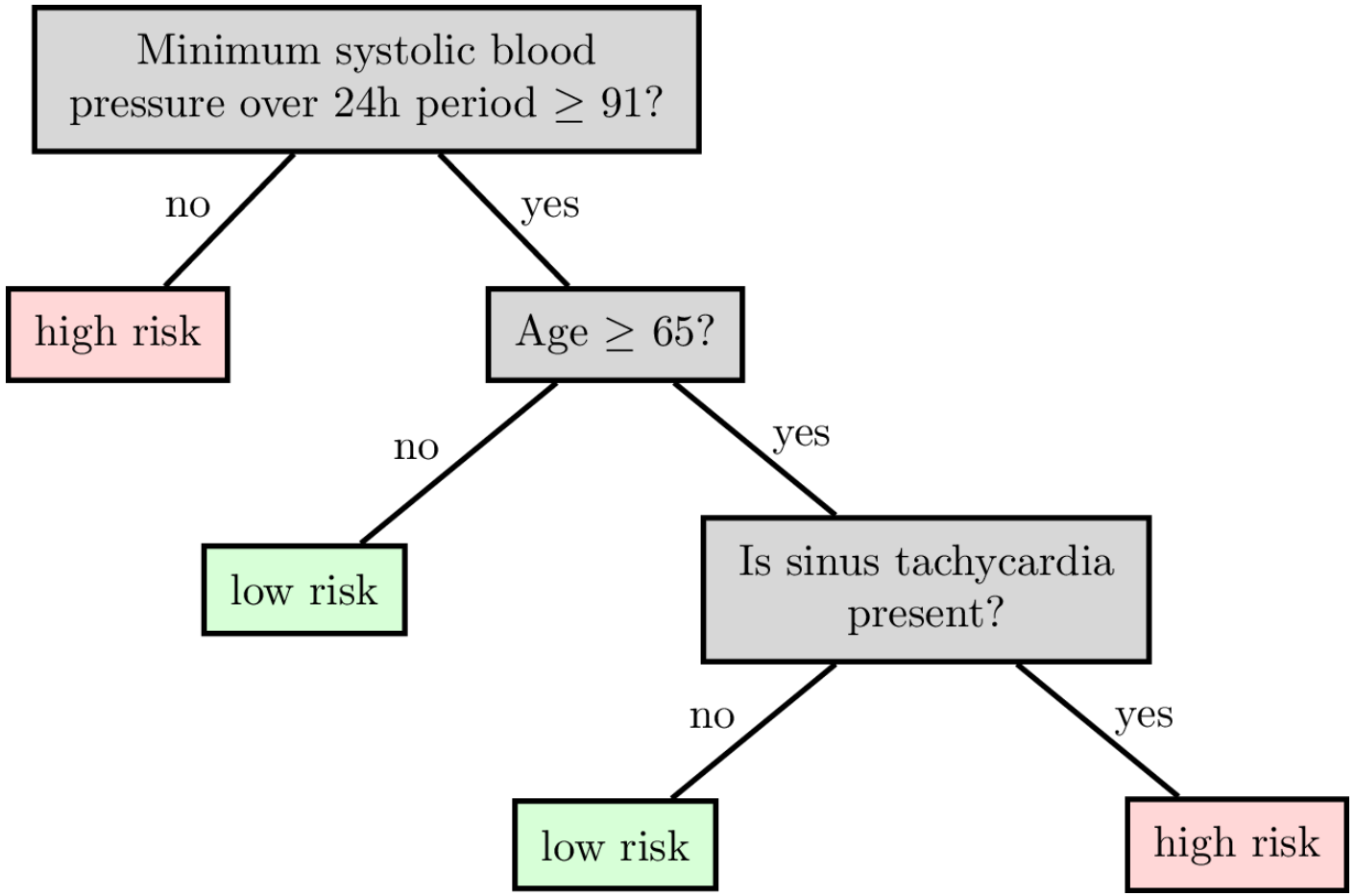
- Train a sparse autoencoder on a transformer's hidden states; sparsity makes features behave as discrete prototypes.
- Concepts emerge: Arabic-script tokens, base64 strings, code patterns, locations, emotions.
- Anthropic scaled this to Claude 3. Shares the same high-level instinct as k-means (store small set of meaningful prototypes).

Bricken, Templeton, Batson, et al. *Towards Monosemanticity*. Anthropic, 2023.

Outline

- Non-parametric models overview
- Similarity-based methods
  - k-nearest neighbor
  - k-means clustering
- Tree-based methods
  - Decision Tree
  - Bagging and ensembles

Reading a classification 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

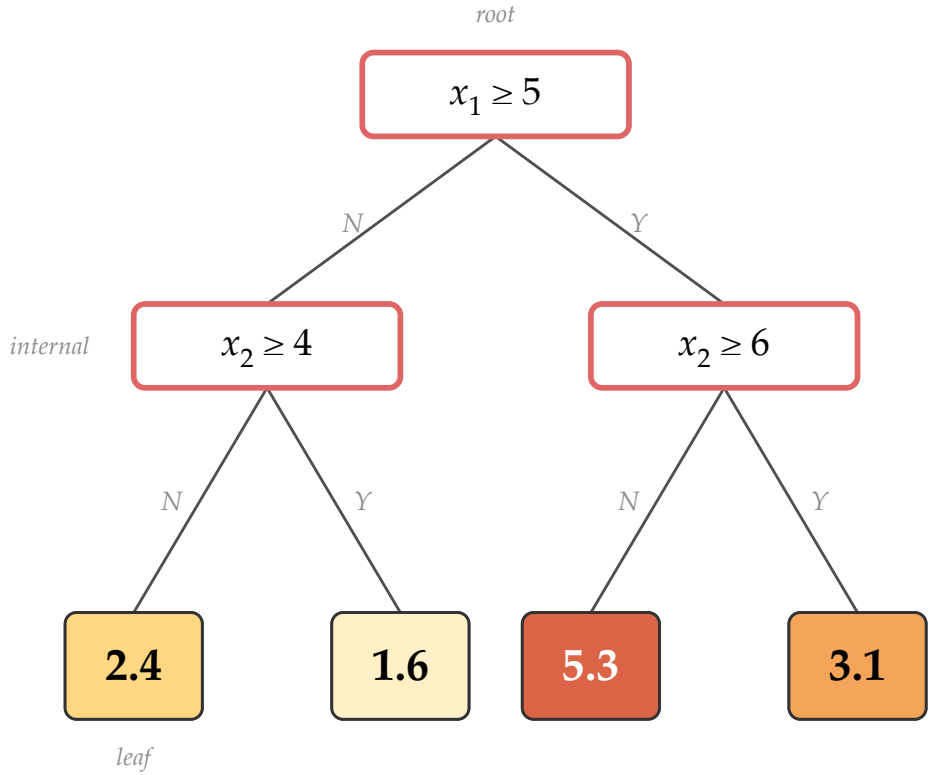
x_7 : latest diastolic bp

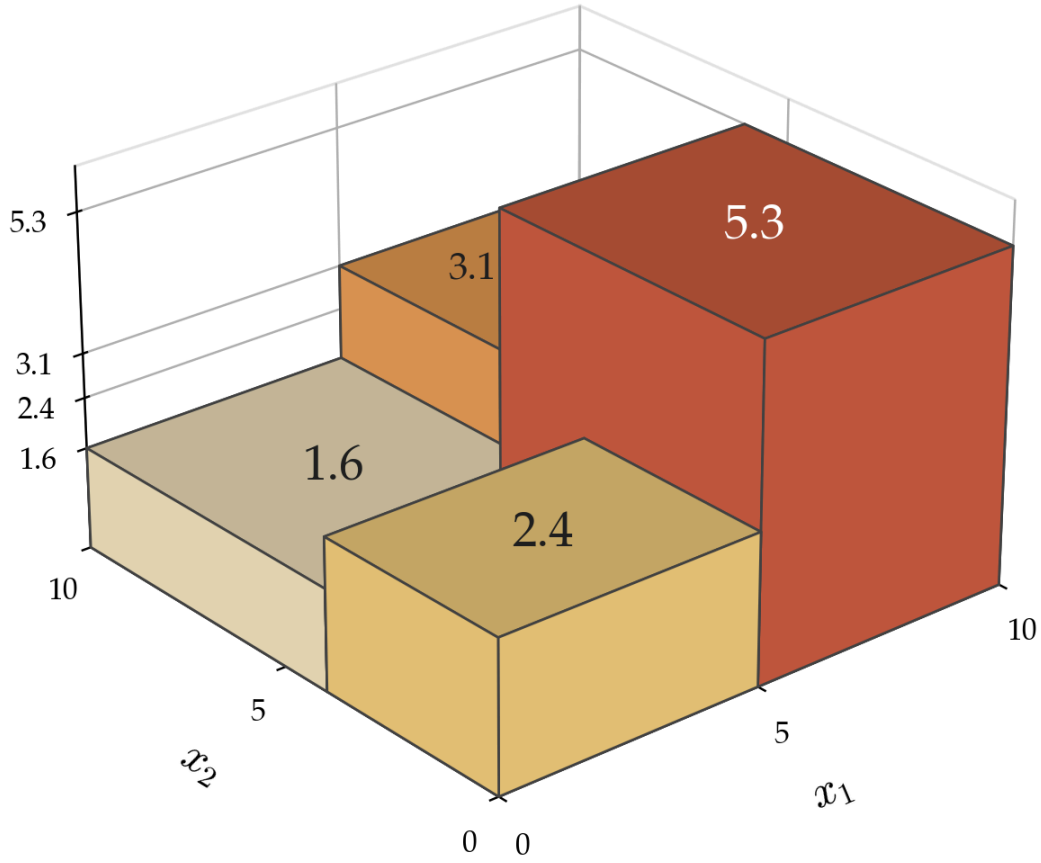
labels y :

1: high risk

0: low risk

Tree root node, internal node, leaf, and depth





Each internal node carries a split dimension and split value ($x_j \geq \theta$).

Each leaf corresponds to an axis-aligned region; the tree predicts the leaf's value there.

How to Grow a Regression Tree

1. Grow — start with all data at the root.

   Try candidate splits along each feature.

2. Split — pick the split that best reduces prediction error

   (lowest weighted variance of target values).

3. Recurse — treat each child region as a whole new dataset and run the

   algorithm again.

4. Stop — if a region is small or already consistent

   (data within $\leq$ leaf size, or variance below a threshold), make it a leaf.

set of indices

hyper-parameter: largest leaf size

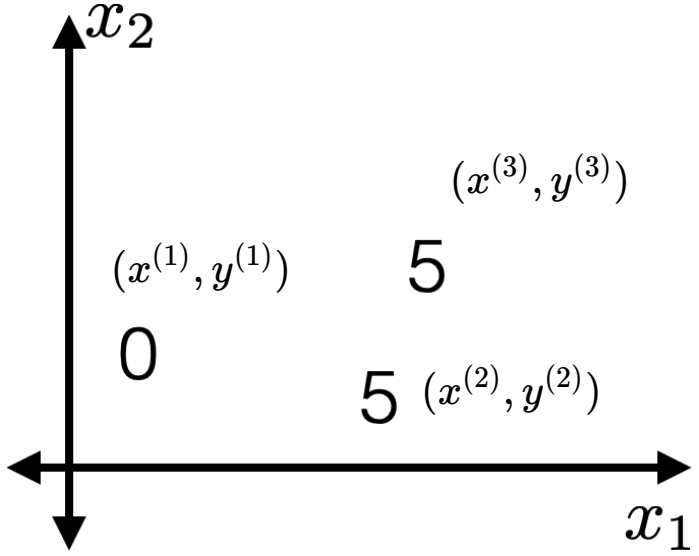
maximum number of training datapoints allowed in a leaf

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

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

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

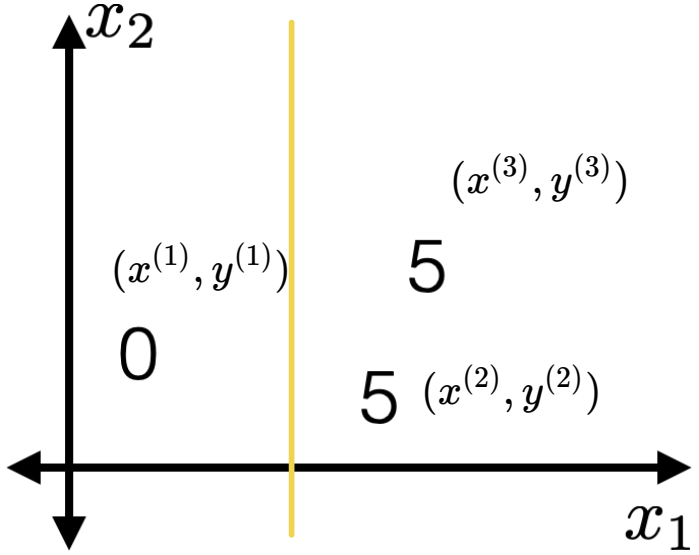
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\ i \in I_{j,s}^{+}}\ y^{(i)}$
6. Set $\hat{y}_{j,s}^{-}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



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

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

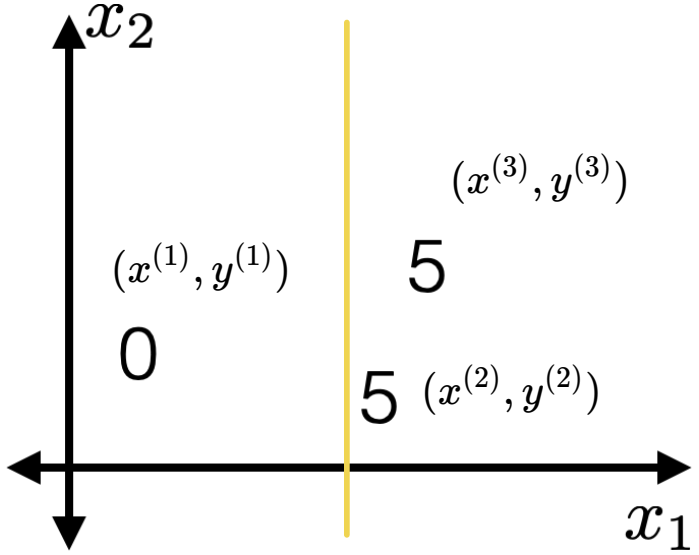
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\ i \in I_{j,s}^{+}}\ y^{(i)}$
6. Set $\hat{y}_{j,s}^{-}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



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

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

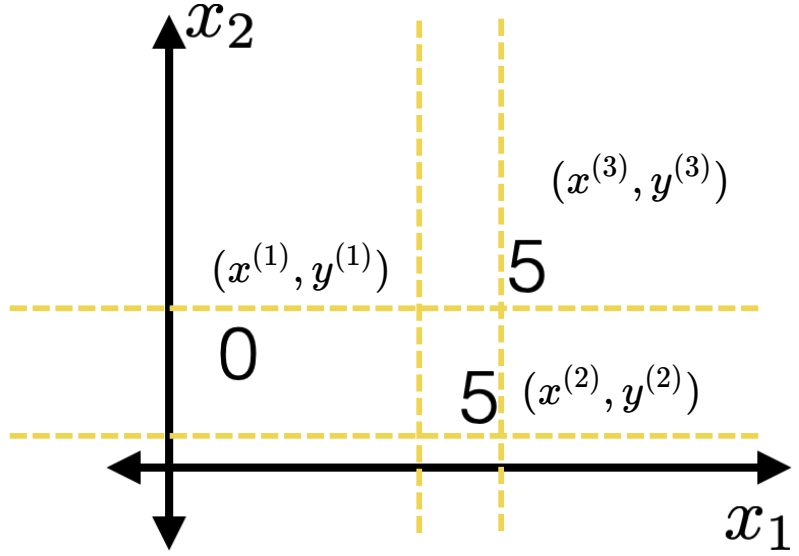
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{average}_{\ i \in I_{j,s}^{+}}\ y^{(i)}$
6. Set $\hat{y}_{j,s}^{-}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



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

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

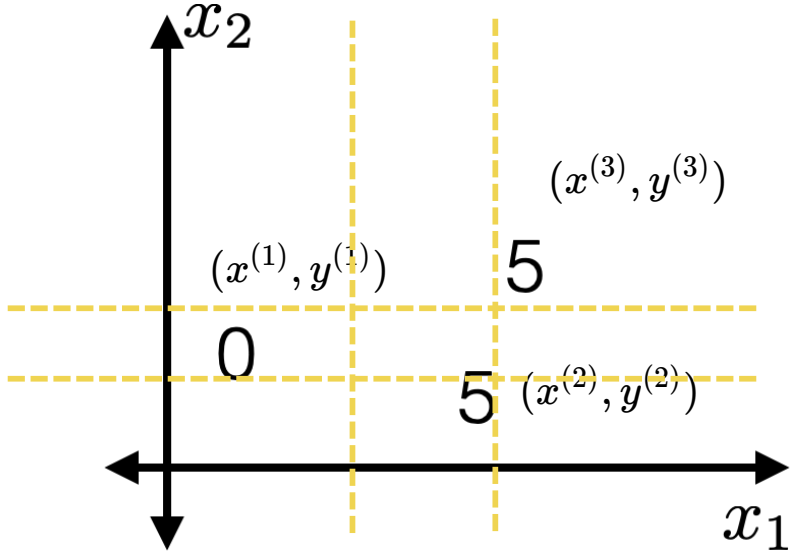
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I^+_{j,s} = \left\{ i \in I \mid x^{(i)}_j \geq s \right\}$
4. Set $I^-_{j,s} = \left\{ i \in I \mid x^{(i)}_j < s \right\}$
5. Set $\hat{y}^+_{j,s} = \text{average}_{\ i \in I^+_{j,s}}\ y^{(i)}$
6. Set $\hat{y}^-_{j,s} = \text{average}_{\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



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

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

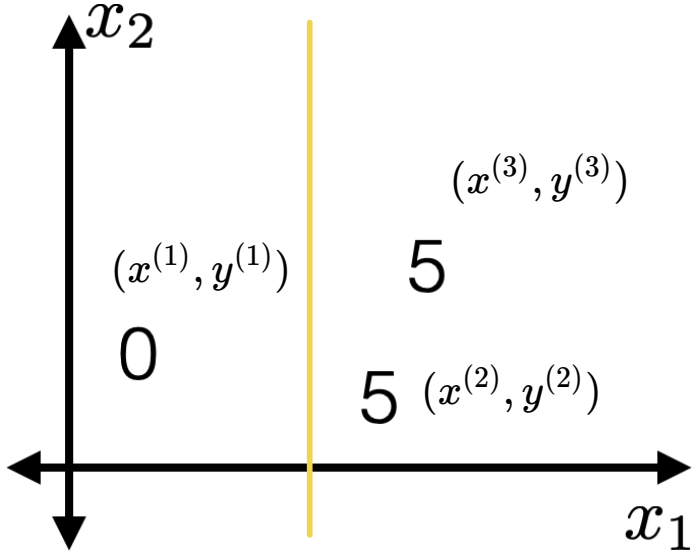
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I^+_{j,s} = \left\{ i \in I \mid x^{(i)}_j \geq s \right\}$
4. Set $I^-_{j,s} = \left\{ i \in I \mid x^{(i)}_j < s \right\}$
5. Set $\hat{y}^+_{j,s} = \text{average}_{\ i \in I^+_{j,s}}\ y^{(i)}$
6. Set $\hat{y}^-_{j,s} = \text{average}_{\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



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

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

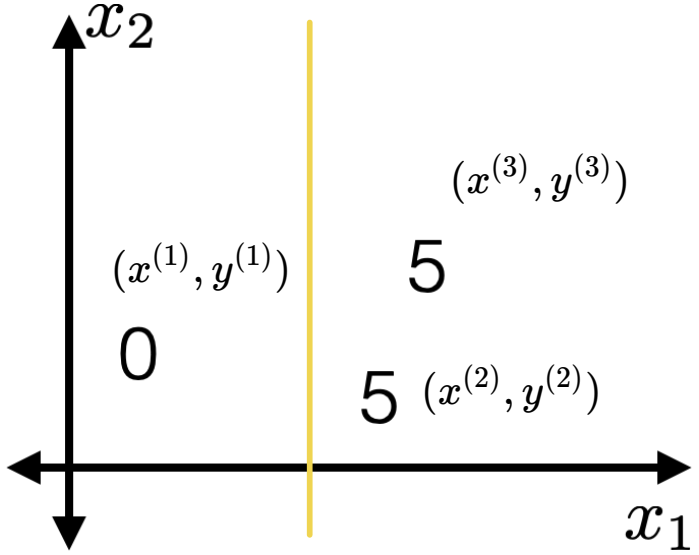
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I^+_{j,s} = \left\{ i \in I \mid x^{(i)}_j \geq s \right\}$
4. Set $I^-_{j,s} = \left\{ i \in I \mid x^{(i)}_j < s \right\}$
5. Set $\hat{y}^+_{j,s} = \text{average}_{\ i \in I^+_{j,s}}\ y^{(i)}$
6. Set $\hat{y}^-_{j,s} = \text{average}_{\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



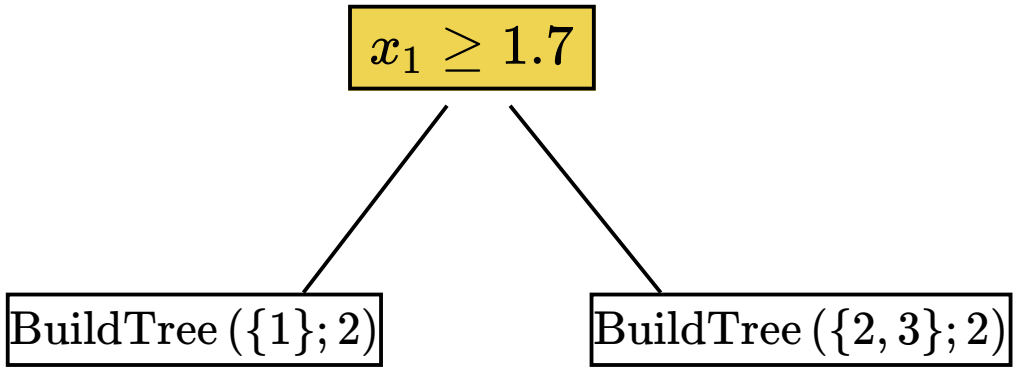
Suppose line 8 sets this (j^*, s^*),
say $(j^*, s^*) = (1, 1.7)$

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

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

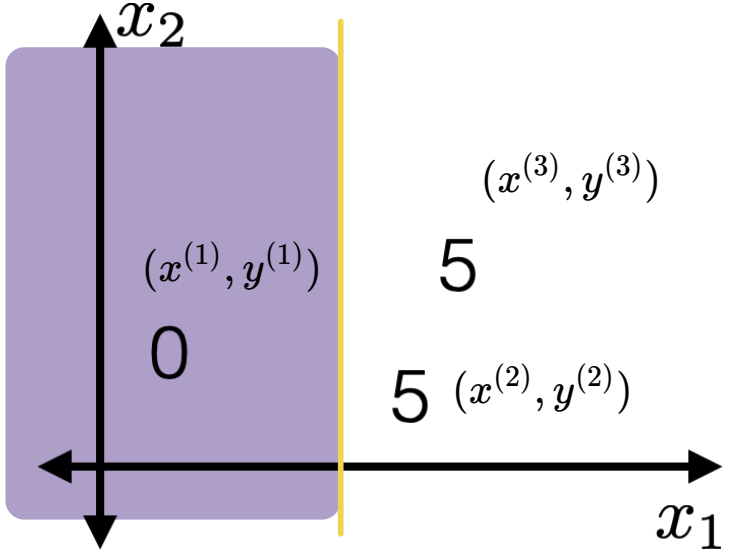


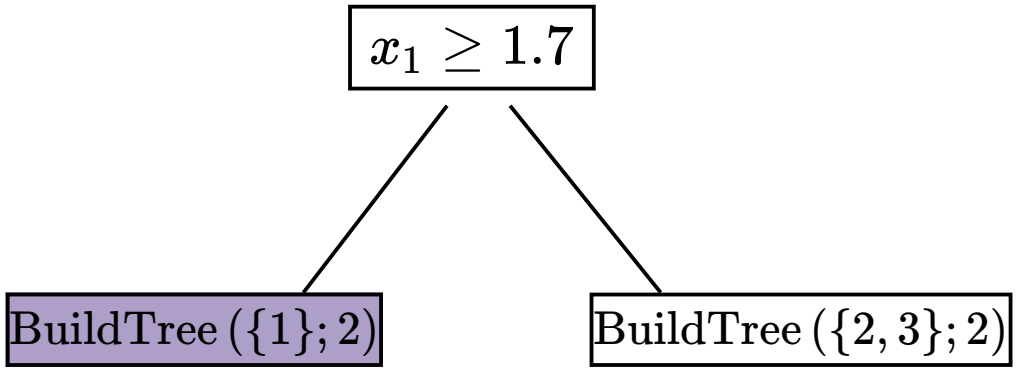
Line 12, do recursion



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

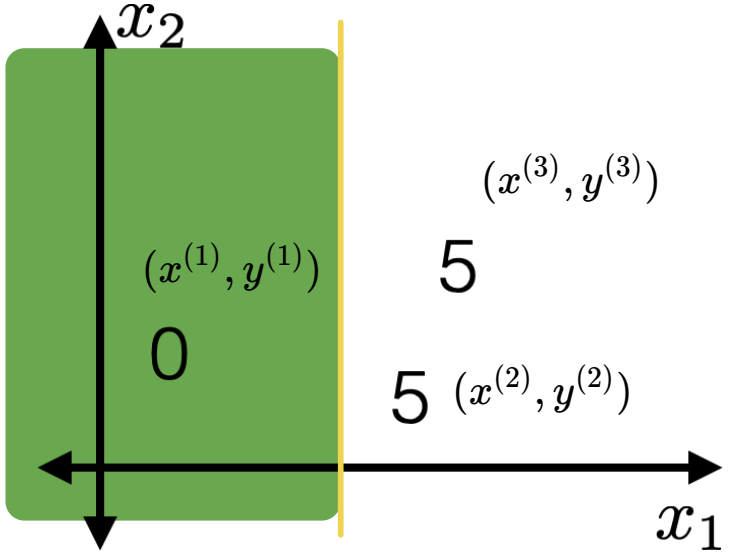
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I^+_{j,s} = \left\{ i \in I \mid x^{(i)}_j \geq s \right\}$
4. Set $I^-_{j,s} = \left\{ i \in I \mid x^{(i)}_j < s \right\}$
5. Set $\hat{y}^+_{j,s} = \text{average}_{\ i \in I^+_{j,s}}\ y^{(i)}$
6. Set $\hat{y}^-_{j,s} = \text{average}_{\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

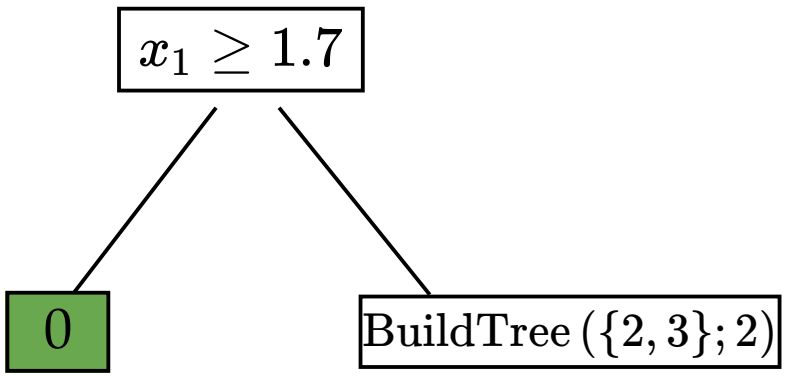




BuildTree$(I, k, \mathcal{D})$

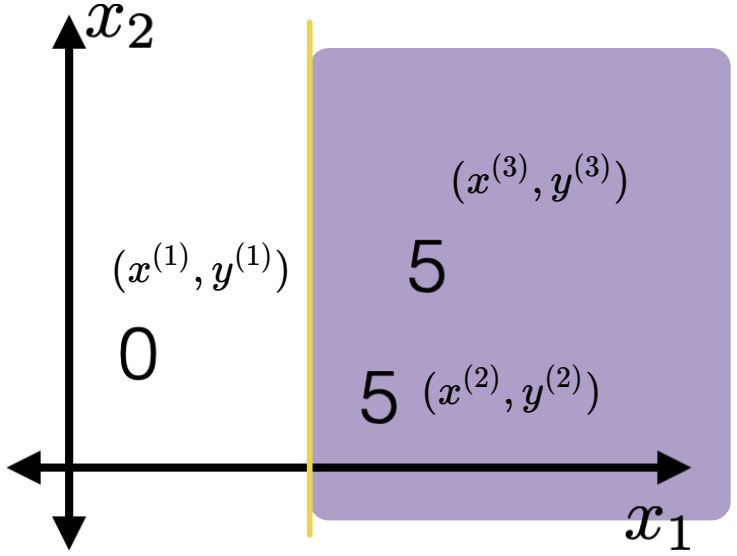
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I^+_{j,s} = \left\{ i \in I \mid x^{(i)}_j \geq s \right\}$
4. Set $I^-_{j,s} = \left\{ i \in I \mid x^{(i)}_j < s \right\}$
5. Set $\hat{y}^+_{j,s} = \text{average}_{\ i \in I^+_{j,s}}\ y^{(i)}$
6. Set $\hat{y}^-_{j,s} = \text{average}_{\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

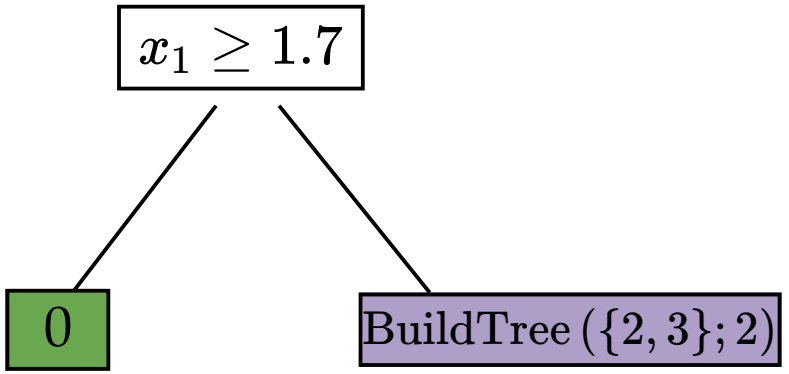




BuildTree$(I, k, \mathcal{D})$

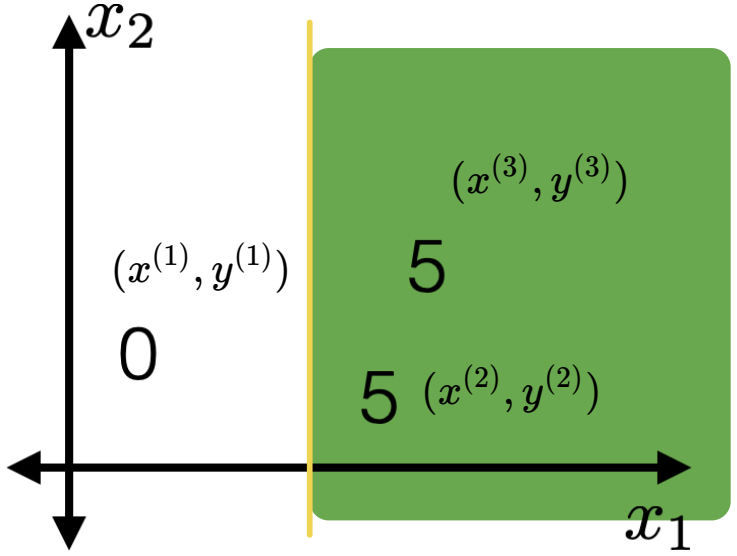
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I^+_{j,s} = \left\{ i \in I \mid x^{(i)}_j \geq s \right\}$
4. Set $I^-_{j,s} = \left\{ i \in I \mid x^{(i)}_j < s \right\}$
5. Set $\hat{y}^+_{j,s} = \text{average}_{\ i \in I^+_{j,s}}\ y^{(i)}$
6. Set $\hat{y}^-_{j,s} = \text{average}_{\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

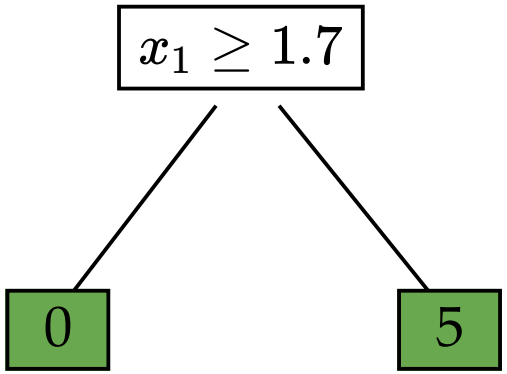




BuildTree$(I, k, \mathcal{D})$

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





How to Grow a Classification Tree

1. Grow — start with all data at the root.

   Try candidate splits along each feature.

2. Split — pick the split that best separates the classes

   (lowest weighted entropy or most "homogeneous").

3. Recurse — treat each child region as a whole new dataset and run the algorithm again.

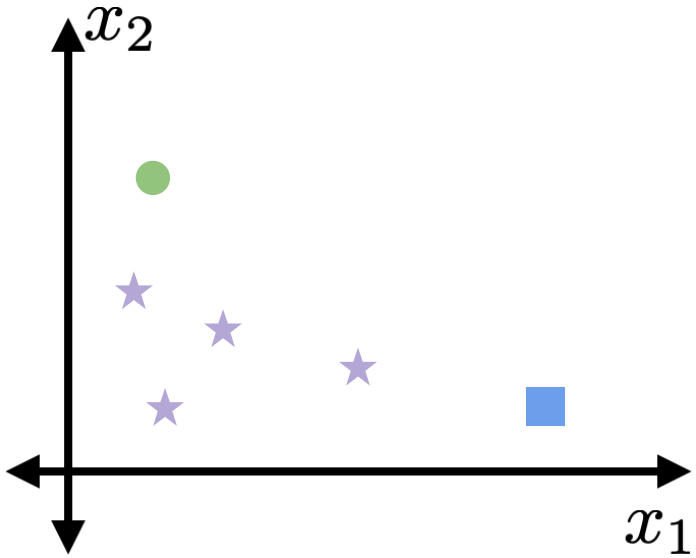
4. Stop — if a region is small or already pure

   (data within ≤ leaf size, or most data share the same label), make it a leaf.

entropy $H := -\sum_{\text{class } c} \hat{P}_c(\log_2 \hat{P}_c)$

empirical probability

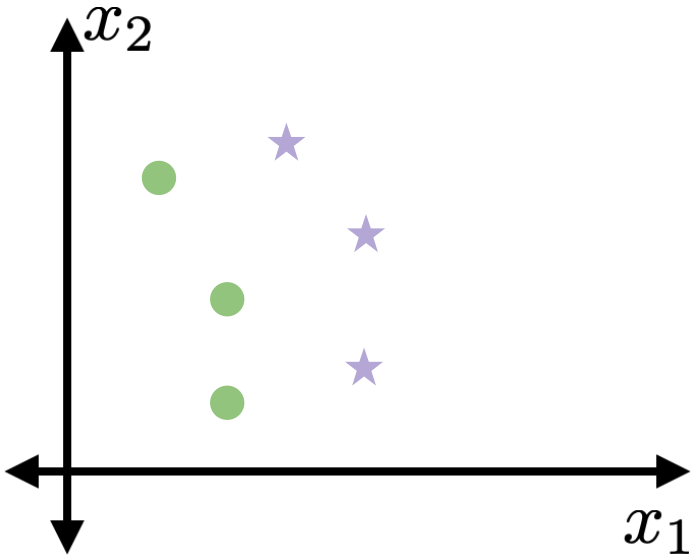
for example: c iterates over 3 classes ★ ● ■



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

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

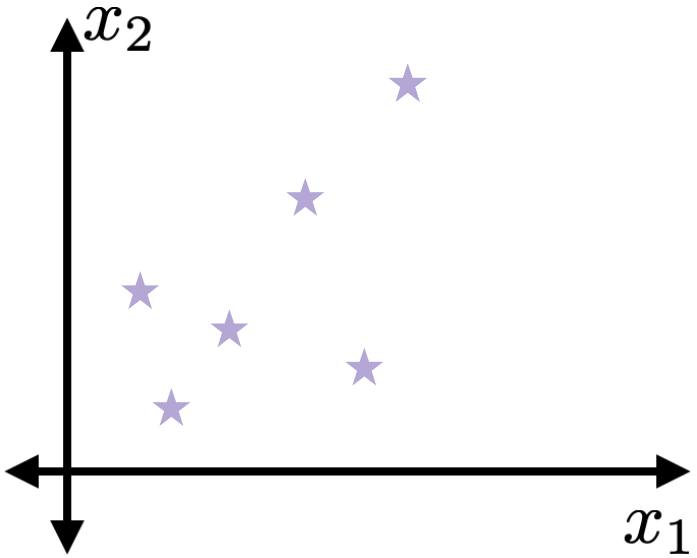
(about 1.252)



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

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

(about 1.1)



$\hat{P}_c$: ★ $\frac{6}{6}$ ● 0 ■ 0

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

($= 0$)

For classification

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

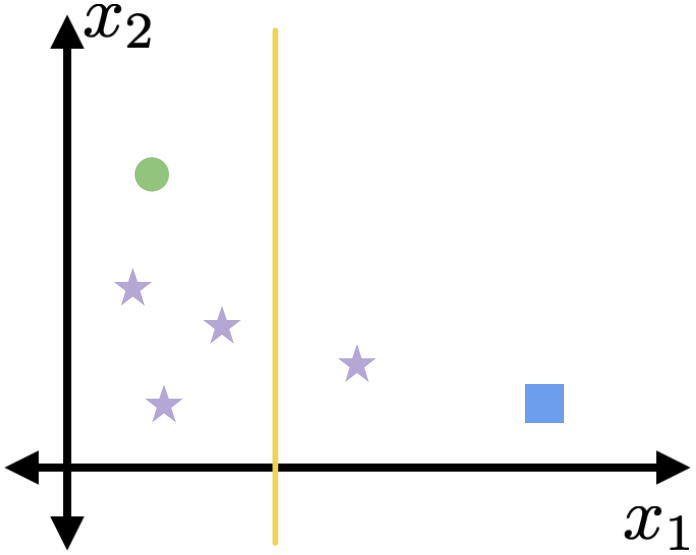
1. **if** $|I| > k$
2. **for** each split dim j and split value s
3. Set $I_{j,s}^{+} = \left\{ i \in I \mid x_j^{(i)} \geq s \right\}$
4. Set $I_{j,s}^{-} = \left\{ i \in I \mid x_j^{(i)} < s \right\}$
5. Set $\hat{y}_{j,s}^{+} = \text{majority}_{\ i \in I_{j,s}^{+}}\ y^{(i)}$
6. Set $\hat{y}_{j,s}^{-}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

majority vote as regional prediction

weighted average entropy (WAE) as performance metric

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

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



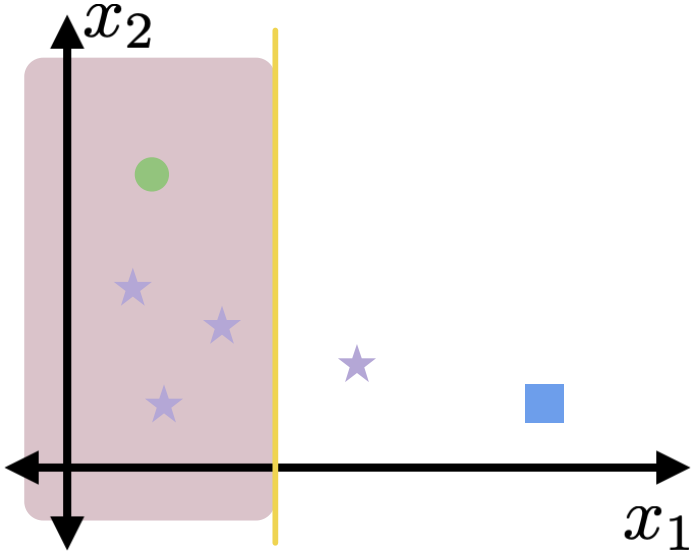
$$E_{j,s} = \frac{4}{6} \cdot H\left(I_{j,s}^{-}\right) + \frac{2}{6} \cdot H\left(I_{j,s}^{+}\right)$$

fraction of points to the left of the split

fraction of points to the right of the split

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

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



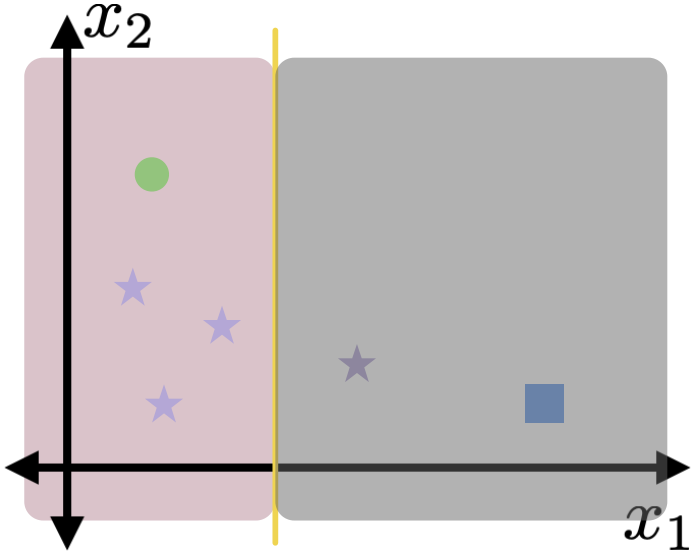
$$\frac{4}{6} \cdot H\left(I_{j,s}^{-}\right) + \frac{2}{6} \cdot H\left(I_{j,s}^{+}\right)$$

$\hat{P}_c$: ★ $\frac{3}{4}$ ● $\frac{1}{4}$ ■ $\frac{0}{4}$

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

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

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



≈ 0.811

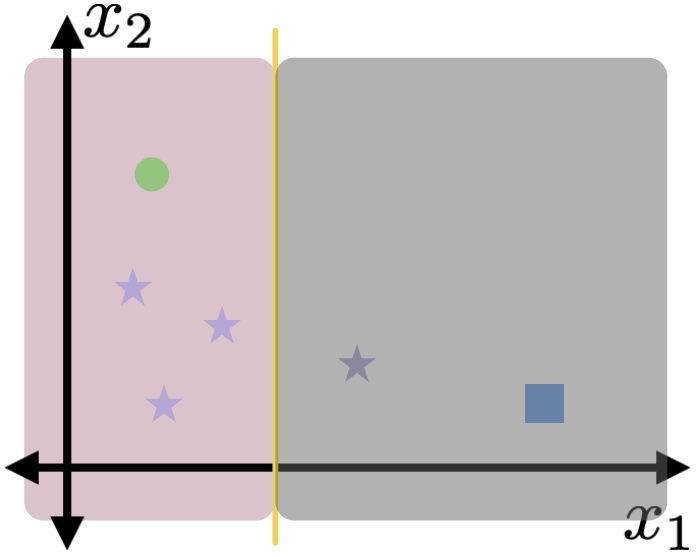
$$\frac{4}{6} \cdot H\left(I_{j,s}^{-}\right) + \frac{2}{6} \cdot H\left(I_{j,s}^{+}\right)$$

$$\hat{P}_c : \ \bigstar \ \frac{1}{2} \quad \bullet \ \frac{0}{2} \quad \blacksquare \ \frac{1}{2}$$

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

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

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



≈ 0.811 $= 1$

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

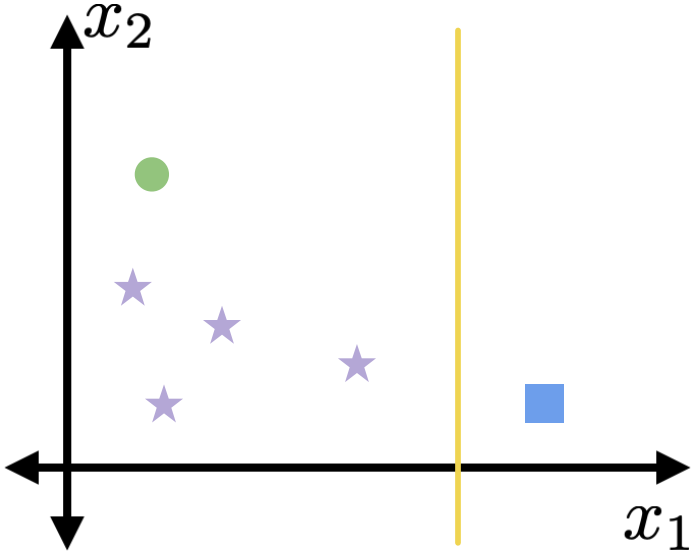
$$\approx \frac{4}{6} \cdot 0.811 + \frac{2}{6} \cdot 1$$

$$= 0.874$$

(for this split choice, line 7 $E_{j,s} \approx 0.874$)

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

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



$$-\left[\tfrac{4}{5}\log_2\left(\tfrac{4}{5}\right) + \tfrac{1}{5}\log_2\left(\tfrac{1}{5}\right) + 0\right] \approx 0.722$$

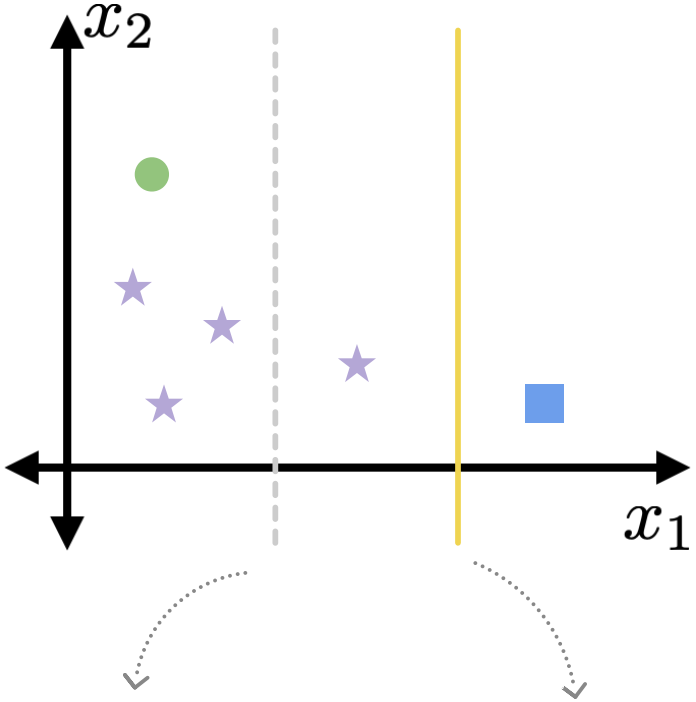
$$\tfrac{5}{6} \cdot H\left(I_{j,s}^{-}\right) + \tfrac{1}{6} \cdot H\left(I_{j,s}^{+}\right)$$

$$-\left[1\log_2\left(1\right) + 0 + 0\right] = 0$$

(line 7, overall $E_{j,s} \approx$ 0.602)

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

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



line 8, set the better (j, s)

Ensemble: Train multiple (diverse) models, combine their predictions

WIRED

SUBSCRIBE

ELIOT VAN BUSKIRK BUSINESS 09.22.2009 11:19 AM

How the Netflix Prize Was Won

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

Coronavirus Disease MENU >

CASES, DATA & SURVEILLANCE

Forecasts of COVID-19 Deaths

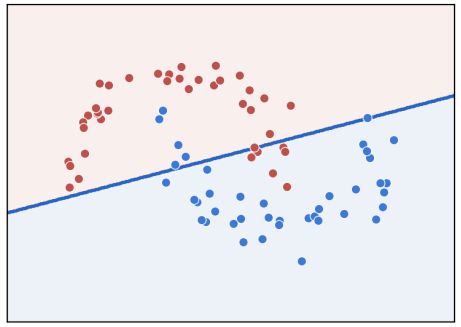
Updated Nov. 12, 2020

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

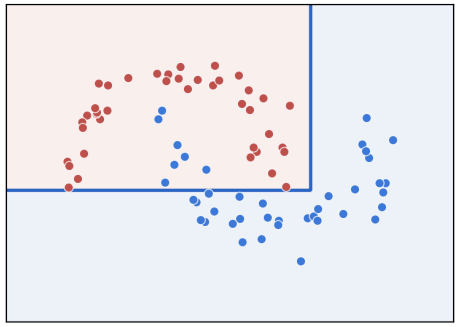
Interpretation of Forecasts of New and Total Deaths

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

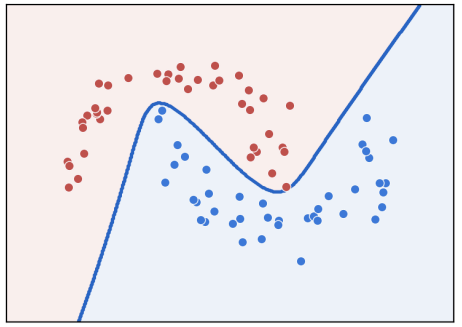
Logistic Regression



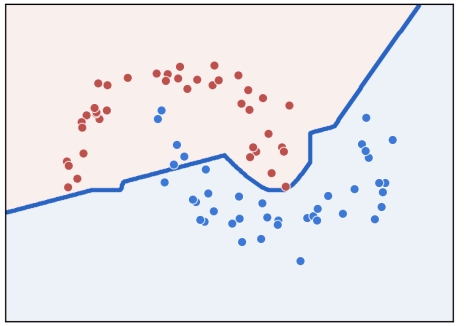
Decision Tree



Neural Net

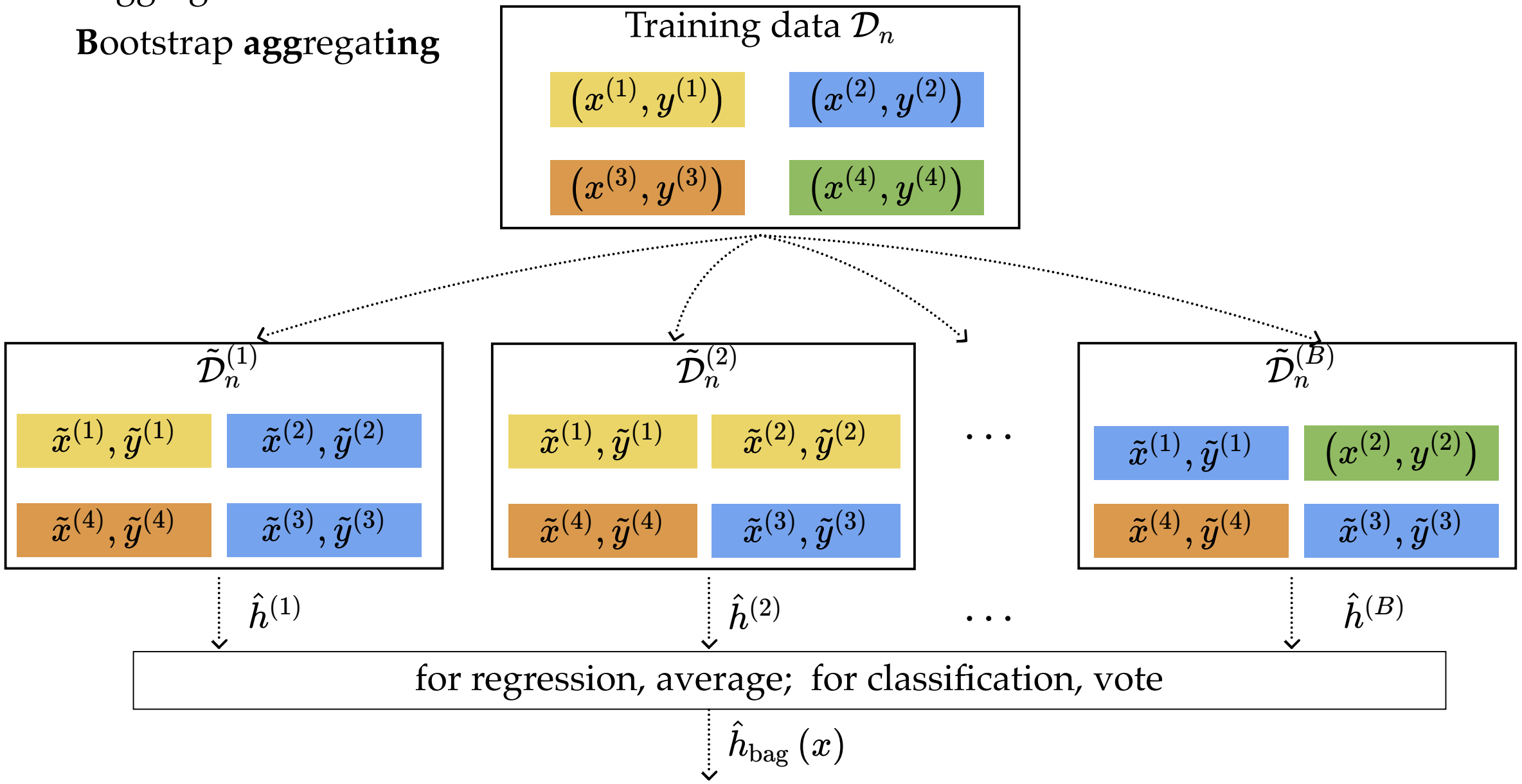


Ensemble (Majority Vote)



Bagging =

Bootstrap **agg**regat**ing**



Trees still win on tabular data

- NNs prefer *smooth* functions; tabular targets are *discrete*.
- Each column is a feature, not a coordinate — rotations mix meaning.
- Trees auto-prune uninformative features; NNs have to learn to ignore them.

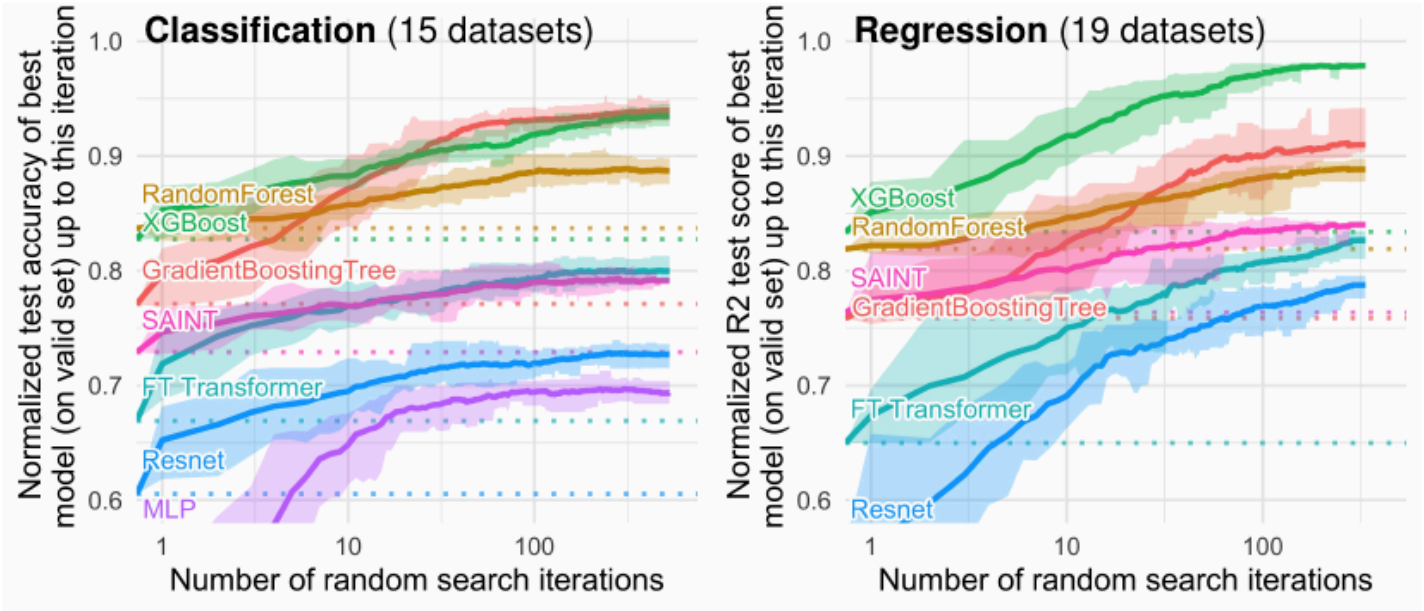


Figure 1: **Benchmark on medium-sized datasets, with only numerical features**. Dotted lines correspond to the score of the default hyperparameters, which is also the first random search iteration. Each value corresponds to the test score of the best model (on the validation set) after a specific number of random search iterations, averaged on 15 shuffles of the random search order. The ribbon

Grinsztajn, Oyallon, Varoquaux. *Why do tree-based models still outperform deep learning on tabular data?* NeurIPS 2022.

Every Kinect-using game shipped a random forest inside.

Real-time skeletal tracking from a depth image.

A random forest classifies each pixel into a body part at 30 fps.

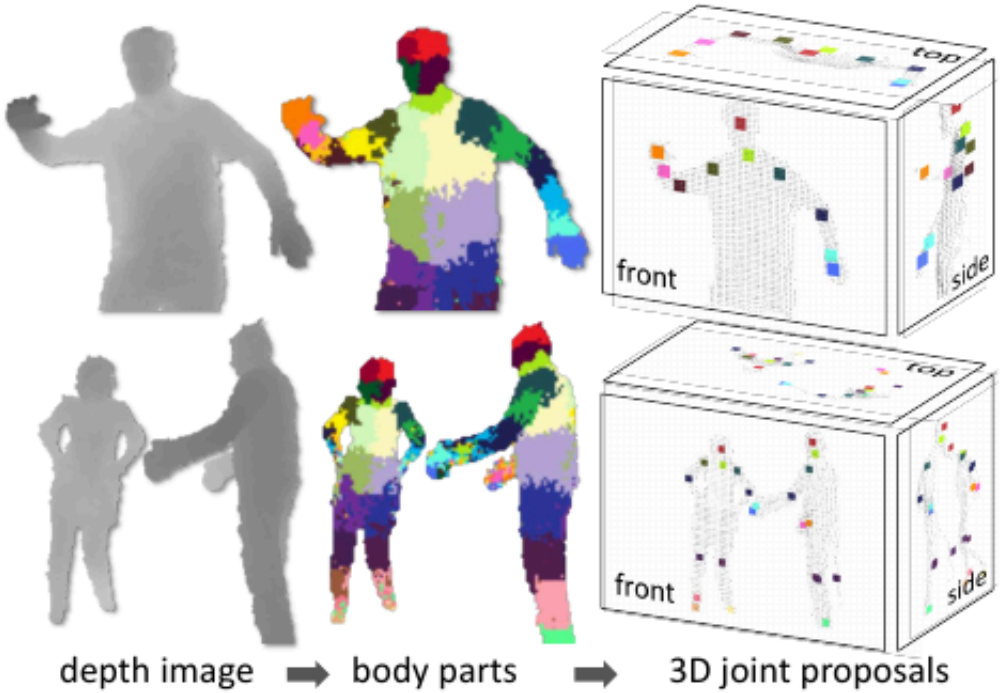


Figure 1. **Overview.** From an single input depth image, a per-pixel body part distribution is inferred. (Colors indicate the most likely part labels at each pixel, and correspond in the joint proposals). Local modes of this signal are estimated to give high-quality proposals for the 3D locations of body joints, even for multiple users.

Shotton et al. *Real-time human pose recognition in parts from single depth images.* CVPR 2011.

Tree structure as inductive bias inside deep models.

- **Soft decision trees**: distill a neural net into an interpretable tree. (Frosst & Hinton, 2017)
- **TabNet**: attention-based feature selection in a tree-like flow. (Arik & Pfister, AAAI 2021)
- **NODE**: differentiable oblivious tree ensembles trained end-to-end. (Popov et al., ICLR 2020)

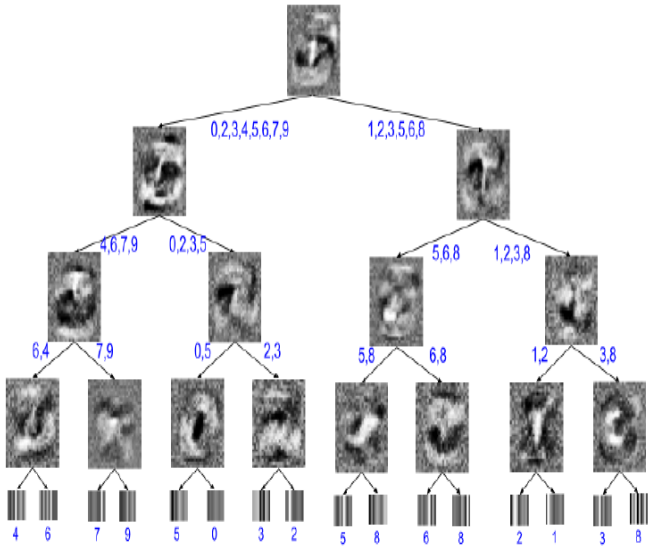


Fig. 2: This is a visualization of a soft decision tree of depth 4 trained on MNIST. The images at the inner nodes are the learned filters, and the images at the leaves are visualizations of the learned probability distribution over classes. The final

Summary

- Non-parametric models let the data define structure, with few assumptions about functional form.
- k-Nearest Neighbors: predict using nearby training points under a chosen distance metric.
- k-Means: unsupervised grouping by distance, sensitive to initialization and the choice of k.
- Decision Trees: recursively split data into interpretable, flow-chart-like rules.
- Ensembles: combine many simple models so their votes become stronger, more stable predictors.