

---

# Binary Prediction: From the Goal to KNN

ECE 57000 — September 23, 2026

David I. Inouye

## | We will revisit the Arc 3 opening as a concise recap

We will go through the revised Arc 3 opening again, but more quickly this time.

The slides are unchanged from the new arc sequence. Because the ideas should now be review, we will focus on their connections and move faster through details.

---

# Binary Prediction: Goals and Historical Data

David I. Inouye

Last updated: September 22, 2026

What would justify trusting a prediction for someone we have never seen?

# | Different settings ask **the same kind of prediction question**

| Setting              | Prediction question with two possible answers   |
|----------------------|-------------------------------------------------|
| Bank phone campaign  | Will this customer open a term-deposit account? |
| Wine screening       | Will this wine receive a high sensory rating?   |
| Activity recognition | Is this person walking?                         |

Each uses information available now to predict **an outcome we do not yet know**.

**Our running case:** the bank phone campaign. A **term-deposit account** holds money for a fixed period to earn interest.

# | Each practical question needs a mathematical counterpart

| Practical question             | Bank-case answer                               | Mathematical counterpart |
|--------------------------------|------------------------------------------------|--------------------------|
| What information is available? | New customer information known before the call | ?                        |
| What answer do we want?        | Whether the customer opens the offered account | ?                        |
| What makes a predictor good?   | Few mistakes across unseen customers           | ?                        |

We will fill in one mathematical counterpart at a time.

| Available information becomes **the input**  $x$

**Question:** What information is available?

**Bank-case answer:** New customer information known before the call.

**Formalization:**

$$x \in \mathcal{X}$$

$x$  is one customer's available information;  $\mathcal{X}$  is the set of valid input representations.

| The desired answer becomes **the output**  $y$

**Question:** What answer do we want?

**Bank-case answer:** Whether the customer opens the offered account.

**Formalization:**

$$x \xrightarrow{f} \hat{y} \in \{0, 1\}$$

$y \in \{0, 1\}$  is the actual outcome. The prediction is correct when  $\hat{y} = y$ .

# | Population error formalizes **few mistakes on unseen customers**

A **distribution**  $p$  describes how common different unseen customer–outcome pairs are. A random pair  $(\mathbf{x}, y)$  drawn from  $p$  contains the available information and what actually happens.

The **population error** is the chance that our prediction is wrong:

$$R_p(f) = \Pr_{(\mathbf{x}, y) \sim p} (f(\mathbf{x}) \neq y).$$

**Goal:** choose  $f^* \in \arg \min_{f: \mathcal{X} \rightarrow \{0,1\}} R_p(f)$ .

This completes the goal—but it does **not** tell us how to find  $f$ .

# | The prediction contract now has a mathematical form

| Practical question             | Bank-case answer                               | Mathematical counterpart                               |
|--------------------------------|------------------------------------------------|--------------------------------------------------------|
| What information is available? | New customer information known before the call | Input $x \in \mathcal{X}$                              |
| What answer do we want?        | Whether the customer opens the offered account | Outcome $y \in \{0, 1\}$ ; prediction $\hat{y} = f(x)$ |
| What makes a predictor good?   | Few mistakes across unseen customers           | Small population error $R_p(f)$                        |

**The unresolved question:** How do we find the right function  $f$ ?

# | Machine learning **infers the rule $f$ from examples**

In ordinary computation, the rule is given. For example, a circuit simulator is given equations and parameters, then computes an output from an input.

Ordinary computation

$$f \text{ given, } \quad x \longmapsto f(x)$$

Machine learning

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n \longmapsto \text{infer } f$$

The historical input–outcome pairs  $\mathcal{D}$  are the **training data**. They give examples of the mapping, but not the rule itself.

# | A dataset column is not automatically **an allowed input**

The real prediction context defines what each  $x_i$  may contain.

**Prediction moment:** before calling this customer.

| Historical column                       | Status at prediction time               |
|-----------------------------------------|-----------------------------------------|
| Recorded age and job                    | Available before the call               |
| Duration of this call                   | <b>Unavailable until after the call</b> |
| Whether the customer opens this account | The outcome $y_i$ , not an input        |

The dataset records each call's duration after that call ends.  
**Can call duration be used to predict whether this customer will open the account before the call begins?**

- A. Yes, because it appears in the historical training data.
- B. Yes, if the historical dataset is sufficiently large.
- C. No, because it is unavailable at the prediction moment.

**C. No.** Training inputs must represent what would actually be known when the prediction is made; more records do not repair that mismatch.

**Larger principle:** available columns describe a dataset; allowed inputs are determined by the real decision we are trying to support.

# | Assumptions mark where a learned predictor can break

Training pairs are the evidence. Performance on unseen pairs is the claim. An explicit assumption must connect them:



The most common simple starting point is **i.i.d. sampling**: model the training pairs and unseen pairs as independent draws from the same distribution  $p$ .

I.i.d. is deliberately naive. When it reasonably approximates the problem, it provides a useful foundation for analysis. **We will use it for now.**

# | Independence rules out **linked training examples**

The first “I” in **i.i.d.** is **independent**.

Knowing one sampled pair  $(\mathbf{x}_i, y_i)$  should not change the probability of another draw once the population distribution is fixed. The unseen case should also be independent of the sample used to construct  $f$ .

**Possible violation:** if the same customer appears after several campaign contacts, those rows and outcomes may be linked rather than independent examples.

# | Identically distributed connects training pairs to **unseen cases**

The second “I” in **i.i.d.** is **identically distributed**.

Training pairs and the unseen pairs we care about are modeled as draws from the same joint distribution  $p$ —both the inputs and how they relate to the outcome.

**Possible violation:** if the bank changes who it contacts or changes the offered account, deployment cases may no longer follow the training distribution.

Same distribution does not mean the same customers or repeated feature values.

## | The contract and evidence now motivate a first method

The contract asks for low error on unseen customers using only information available before the call. Training pairs become evidence under our working assumption that they and the unseen cases are connected.

**First idea:** find past customers similar to the new customer and consult their outcomes.

**Next question:** What should “similar” mean, and why should their outcomes help?

---

## 2. How can examples produce a predictor?

KNN makes learning from local evidence visible without making the algorithm the point.

# | KNN predicts from **nearby observed outcomes**

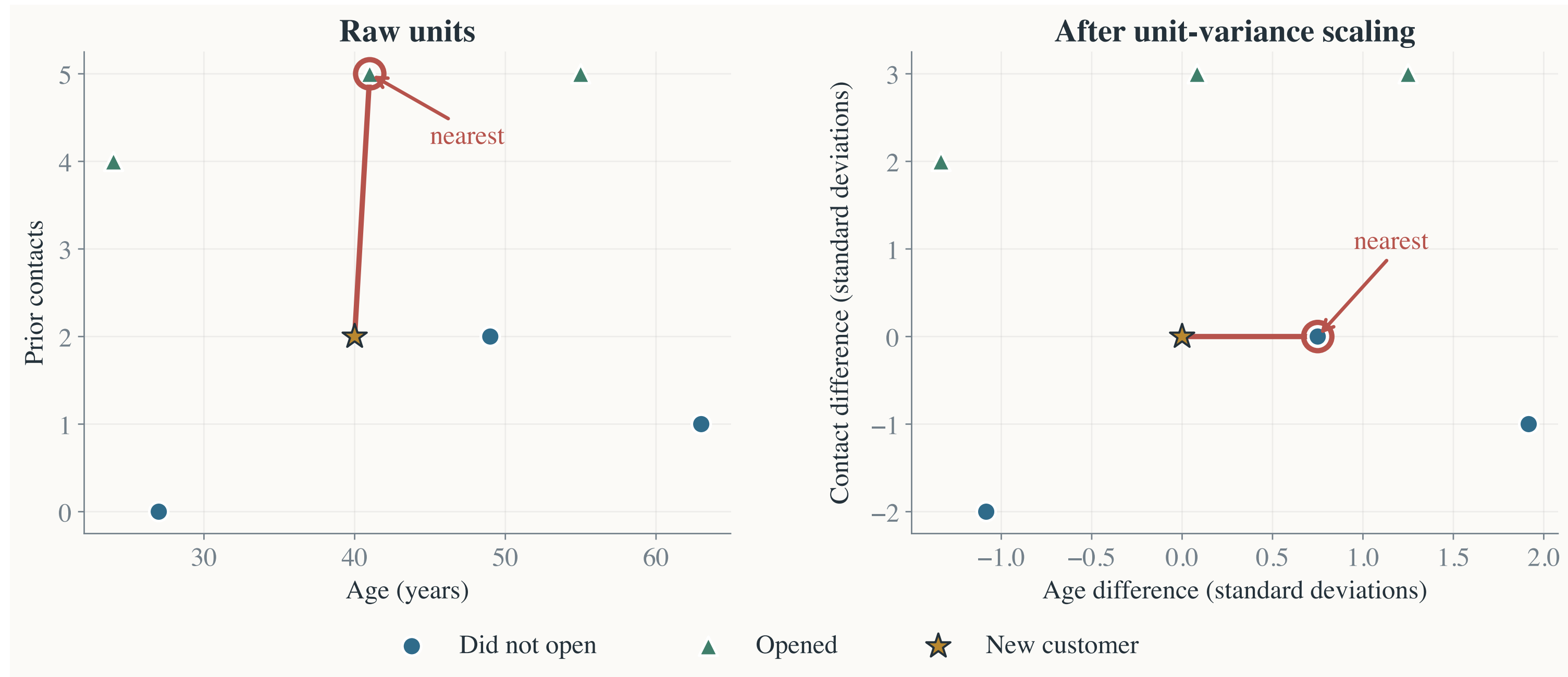
For a new customer  $\boldsymbol{x}$ :

1. compute the distance to every training input  $\boldsymbol{x}_i$ ;
2. keep the indices  $\mathcal{N}_k(\boldsymbol{x})$  of the  $k$  nearest;
3. average their observed outcomes and take a majority vote.

$$\hat{\eta}_k(\boldsymbol{x}) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\boldsymbol{x})} y_i, \quad \hat{y} = \mathbb{1} \left[ \hat{\eta}_k(\boldsymbol{x}) \geq \frac{1}{2} \right].$$

The method constructs a prediction directly from the stored examples.

# | Changing the axes can change **who is nearest**



Distance measures differences in the coordinates we supply—not an intrinsic notion of customer similarity.

## | Unit-variance scaling is a **simple default metric**

For each feature  $j$ , estimate its training mean and standard deviation, then use

$$x'_{ij} = \frac{x_{ij} - \hat{\mu}_j}{\hat{s}_j}.$$

- Apply the same rule to numeric columns and binary indicator columns.
- Encode categorical attributes first—for example, with one indicator per category.
- Estimate  $\hat{\mu}_j$  and  $\hat{s}_j$  from training data only; remove constant columns.

**Interpretation:** every retained coordinate has empirical variance one before distance is computed.

# | The neighbor fraction estimates a **local outcome probability**

The population quantity is

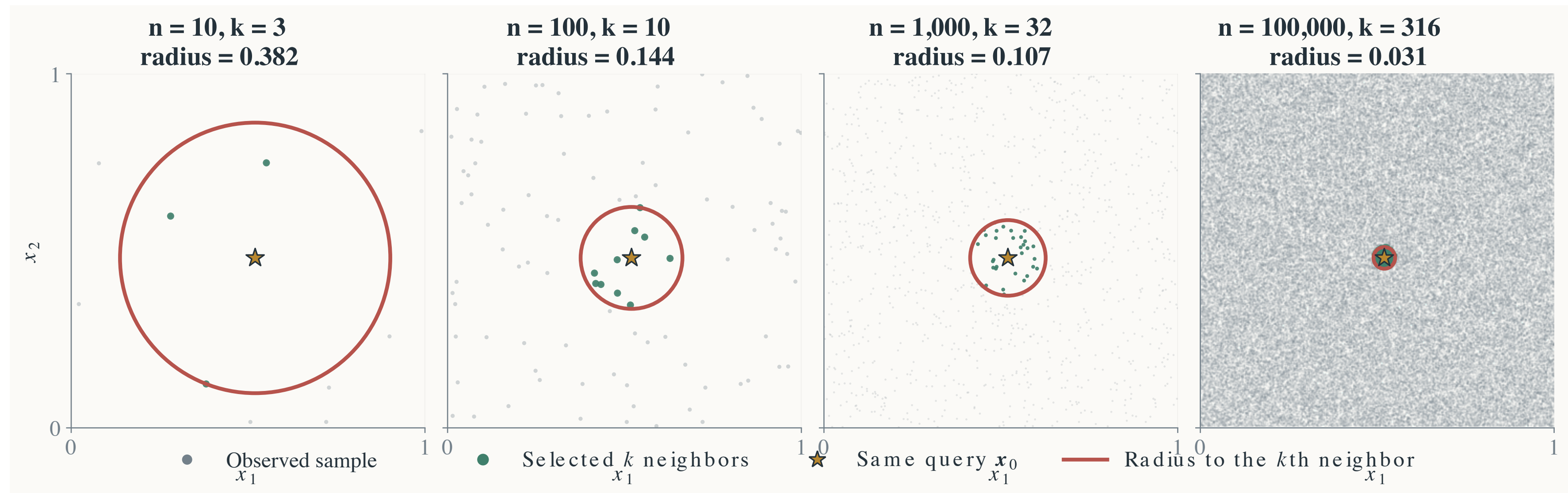
$$\eta(\boldsymbol{x}) = \Pr(y = 1 \mid \mathbf{x} = \boldsymbol{x}).$$

KNN substitutes nearby observed outcomes for repeated outcomes at exactly the same input:

$$\hat{\eta}_k(\boldsymbol{x}) = \frac{\text{nearby customers who opened the account}}{k}.$$

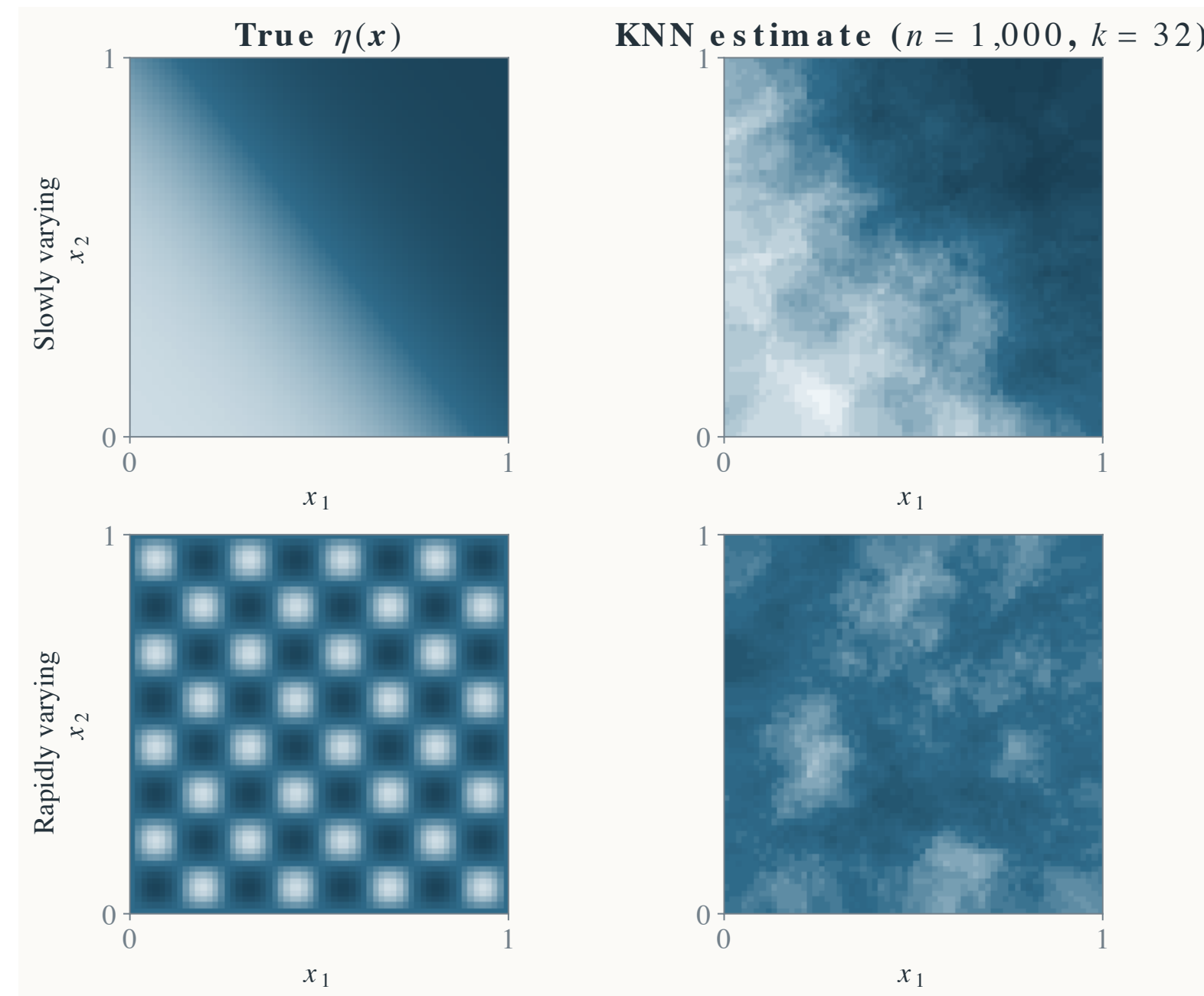
Similar customers need not have identical outcomes; their **probabilities** must be locally related.

# | With more data, $k$ can grow while neighborhoods stay **local**



Here  $k = \lfloor \sqrt{n} \rfloor$ , so  $k \rightarrow \infty$  while  $k/n \rightarrow 0$ : the same query averages more labels inside a smaller neighborhood.

# | Smooth local structure is easier to estimate **with finite data**



Local averaging recovers slowly varying probabilities sooner. Fine-scale or choppy structure can remain blurred until the sample is extraordinarily dense.

# | KNN defers nearly all computation **until prediction**

```
1 def fit(X, y):
2     return np.asarray(X, float), np.asarray(y)
3
4 def predict_proba(model, X_new, k):
5     X, y = model
6     d2 = ((X_new[:, None, :] - X[None, :, :])**2).sum(2)
7     nbr = np.argpartition(d2, k - 1, axis=1)[:k]
8     return y[nbr].mean(axis=1)
```

## Fit

- no parameter optimization
- store  $n \times d$  inputs and  $n$  labels
- memory and copy:  $O(nd)$

## Predict one input

- compute all  $n$  distances
- partial selection avoids a full sort
- time:  $O(nd)$

Small reference sets can be cheap; very large ones motivate models that compress training evidence into learned parameters.

---

# 3. Why can fitting the examples mislead us?

Observed success may reflect memorization, sample luck, or the wrong flexibility.

## | Which $k$ looks best when we score the data used to fit it?

Assume every recorded input is distinct. We fit each KNN model, then compute its error on those same examples.

**Which value of  $k$  should we select using only this error estimate?**

A.  $k = 1$

B.  $k = 15$

C.  $k = 151$

**A.**  $k = 1$ . Every training example is its own nearest neighbor, so this procedure reports zero error.

That makes  $k = 1$  look perfect—but our goal concerns **unseen** customers.

# | Training error answers **an easier question**

| Quantity                  | Examples being scored | Question answered                                    |
|---------------------------|-----------------------|------------------------------------------------------|
| Training error            | Used to construct $f$ | How well does $f$ reproduce evidence it already saw? |
| Population error $R_p(f)$ | Unseen draws from $p$ | How often will $f$ fail on the cases we care about?  |

The same observations cannot simultaneously serve as unrestricted construction material and untouched evidence of generalization.

# | A holdout lets observed data **play the role of unseen data**

We do not have outcomes for the genuinely unseen customers we ultimately care about. How can we estimate their error?



Reserve some observed pairs **before fitting**. Because their outcomes did not help construct  $f$ , predictions on them mimic the information boundary of an unseen case—under the same sampling assumptions.