Carnegie Mellon University
Heinz College

95-865 Pittsburgh Lecture 8: Introduction to Predictive Data Analytics

George Chen

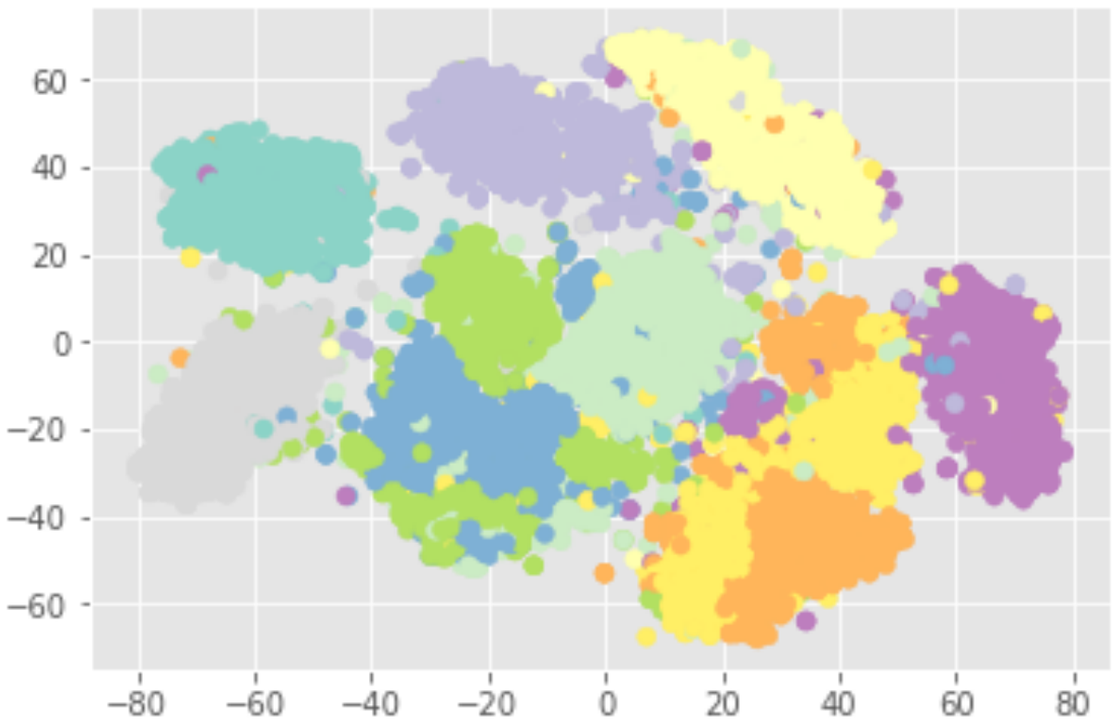
Disclaimer: unfortunately "***k***"
means many things

Previous Lecture: Topic Modeling

- There are actually *many* topic models, not just LDA
  - Correlated topic models, Pachinko allocation, biterm topic models, anchor word topic models, …

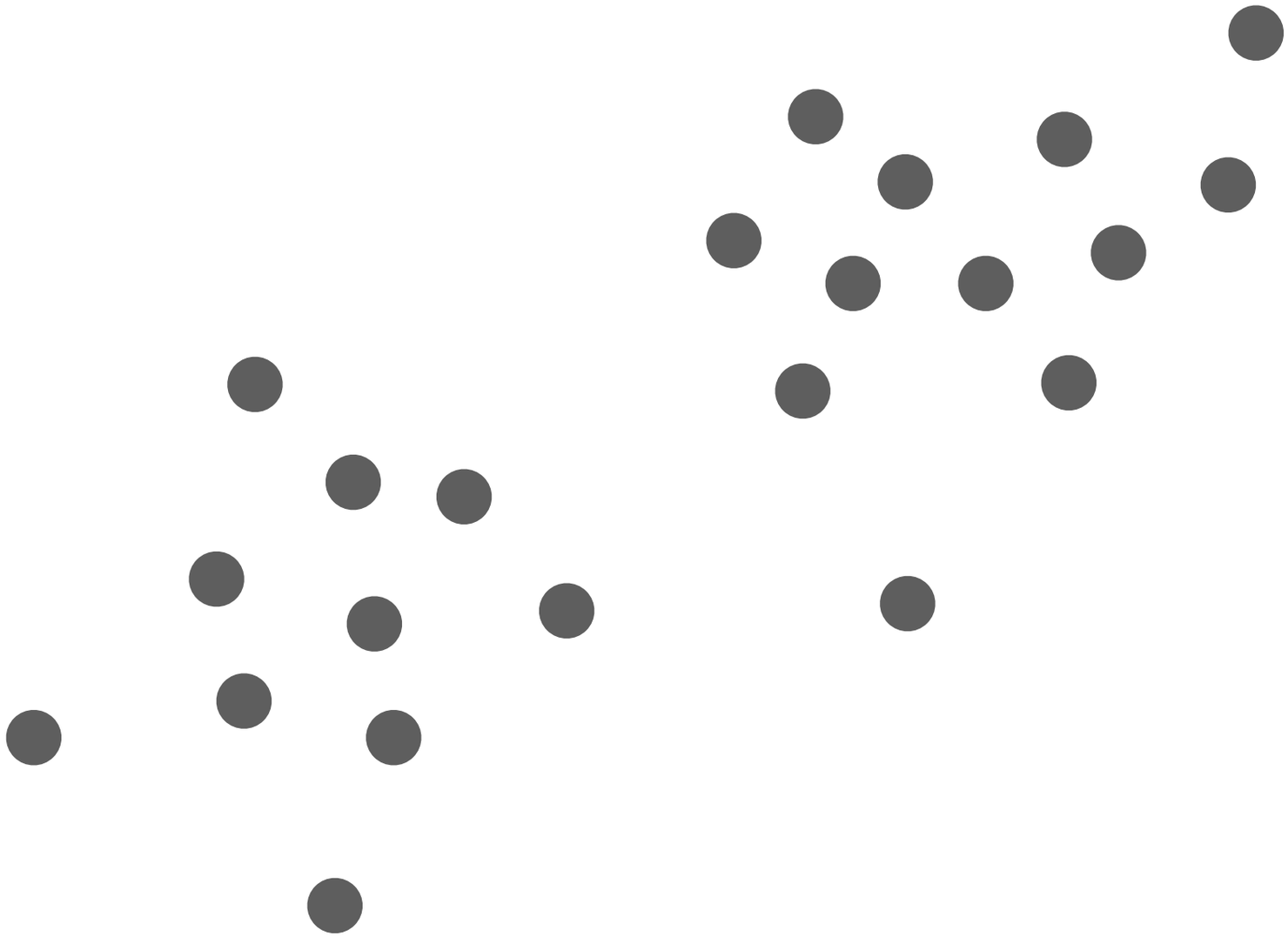
- Dynamic topic models: tracks how topics change *over time*
  - This sort of idea could be used to figure out how user tastes change over time in a recommendation system
    - Could try to see if there are existing patterns for how certain topics become really popular

What if we have labels?



Example: MNIST handwritten digits have known labels

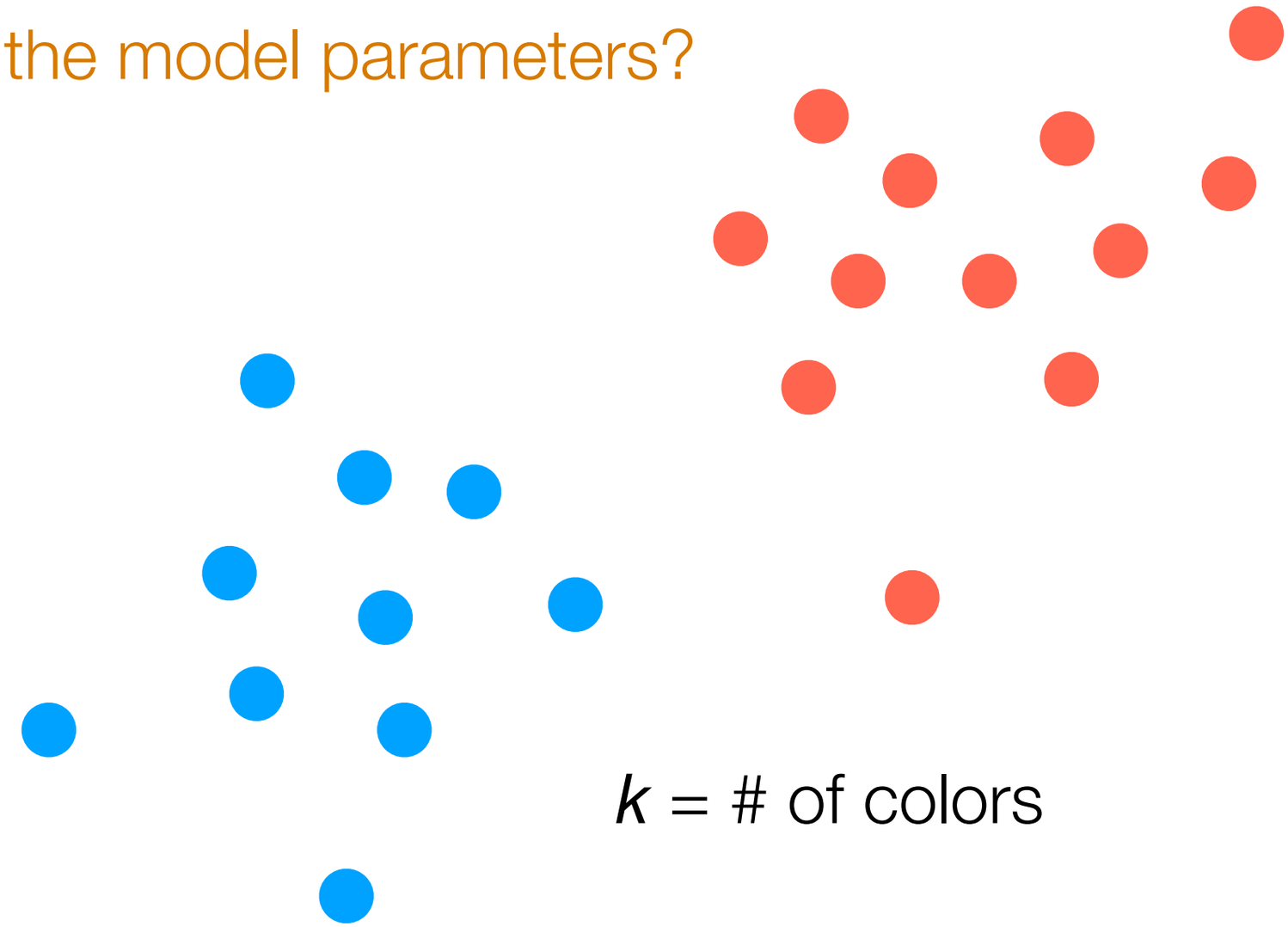
If the labels are known…



If the labels are known…

And we assume data generated by GMM…

What are the model parameters?



k = # of colors

We can directly estimate cluster means, covariances

Flashback: Learning a GMM

Don’t need this top part if we know the labels!

Step 0: Pick *k*

Step 1: Pick guesses for **cluster means and covariances**

Repeat until convergence:

Step 2: Compute probability of each point belonging to each of the *k* clusters

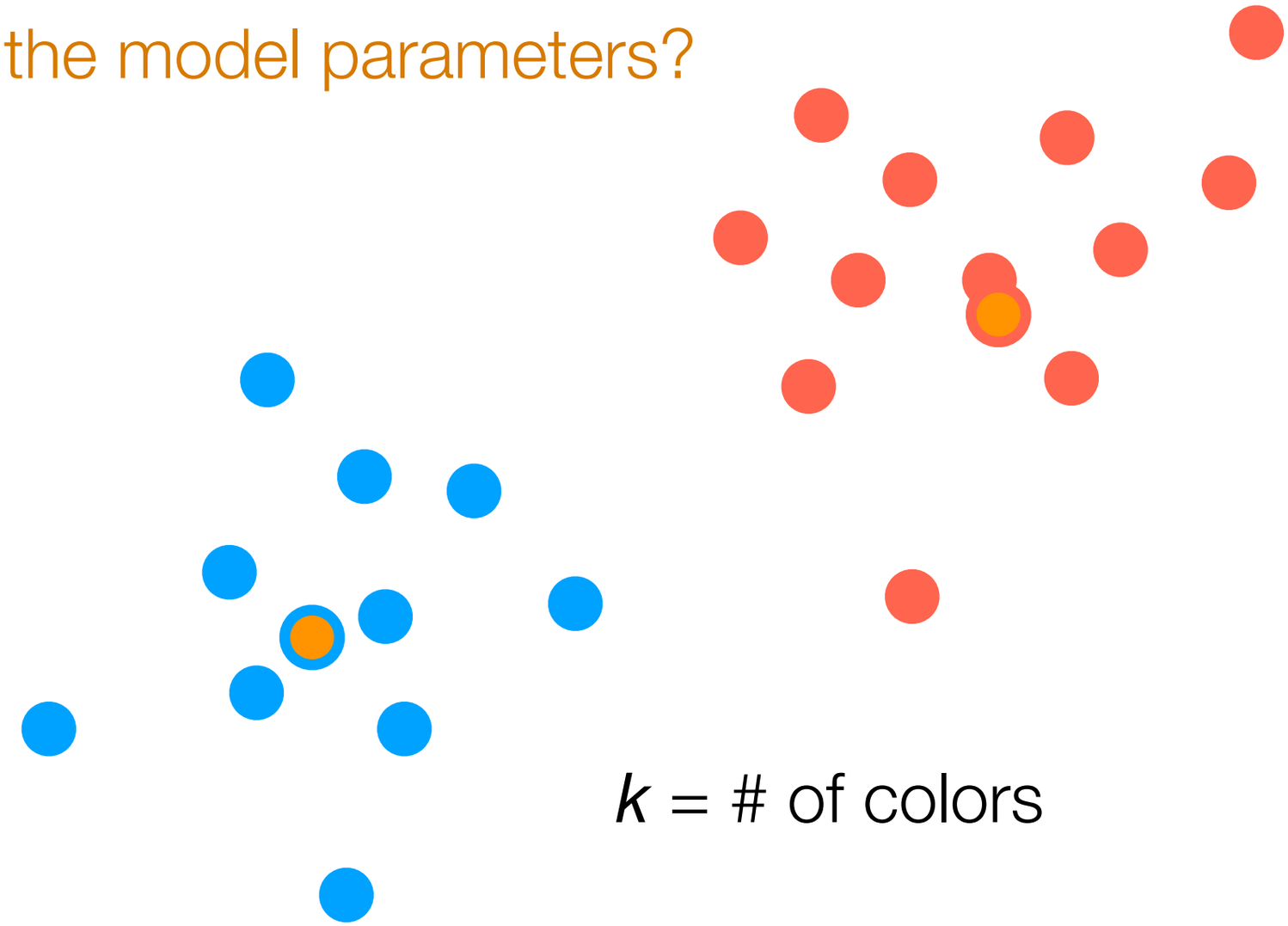
Step 3: Update **cluster means and covariances** carefully accounting for probabilities of each point belonging to each of the clusters

We don’t need to repeat until convergence

If the labels are known…

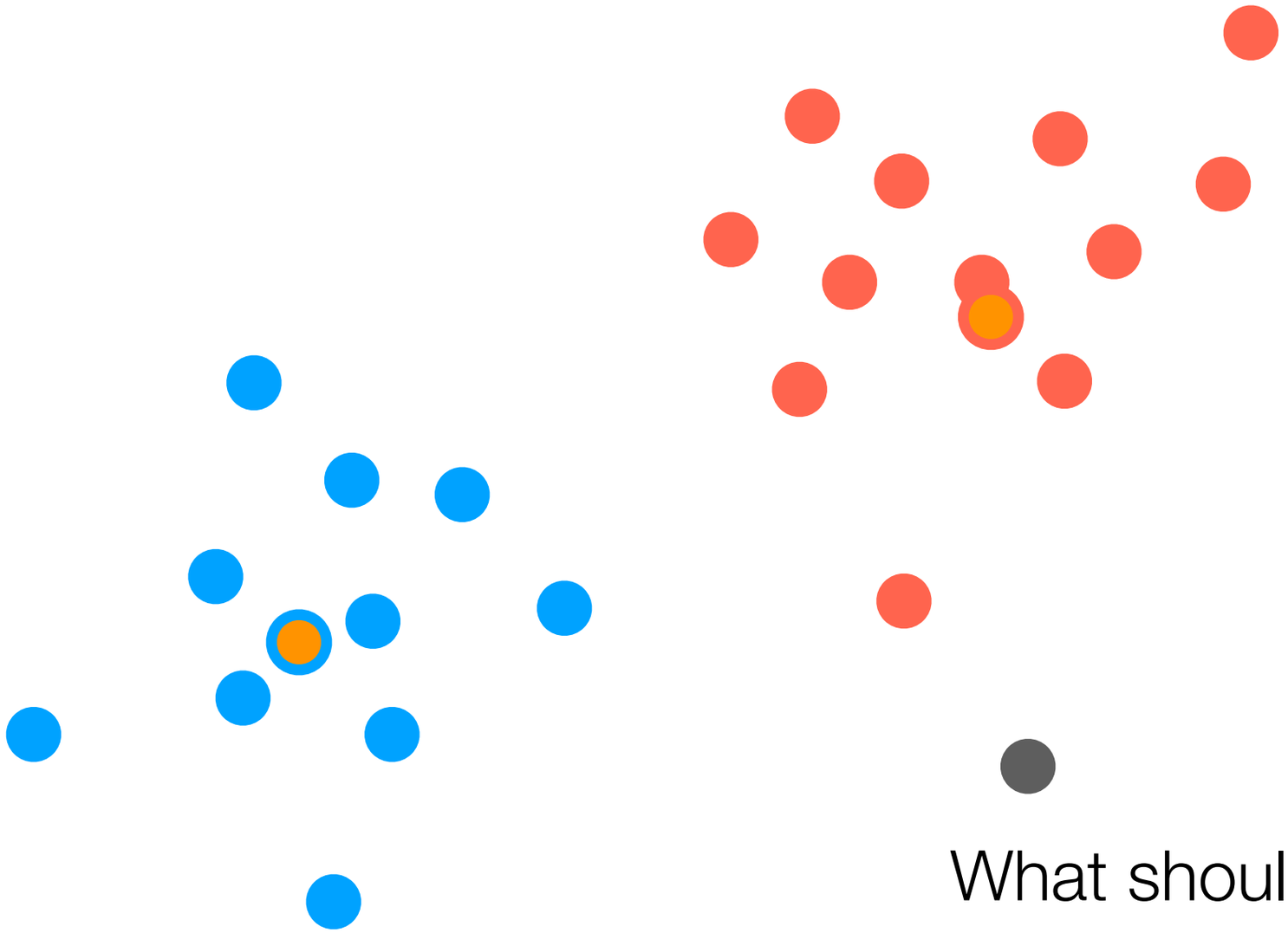
And we assume data generated by GMM…

What are the model parameters?



k = # of colors

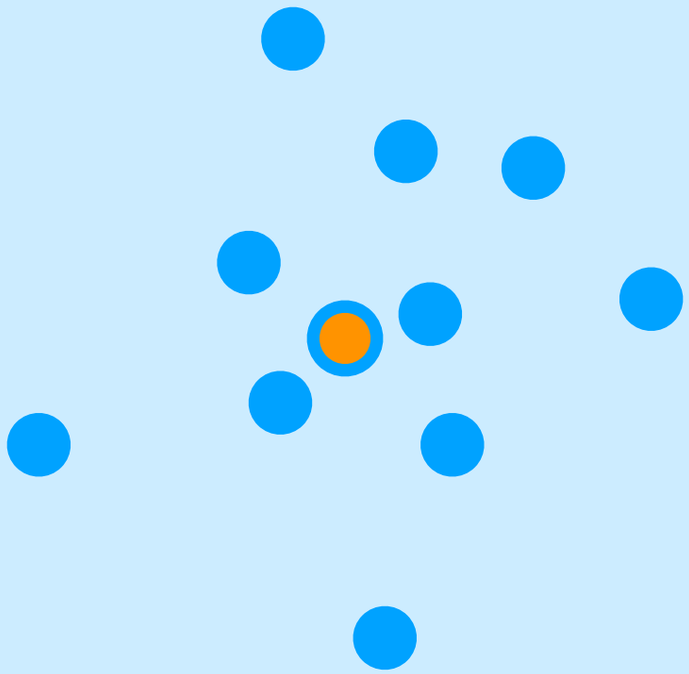
We can directly estimate cluster means, covariances



What should the label of this new point be?

Whichever cluster has higher probability!

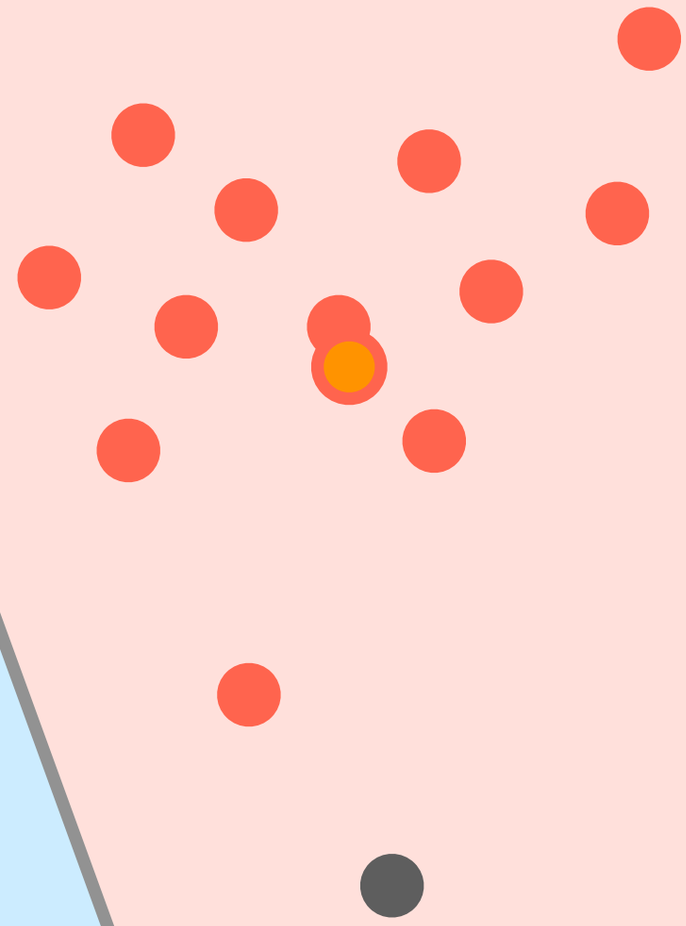
Decision boundary



This classifier we've created assumes a *generative model*

We just created a **classifier**

(a procedure that given a new data point tells us what "class" it belongs to)



What should the label of this new point be?

Whichever cluster has higher probability!

You've seen generative models before for prediction

Linear regression!

Model parameters: slope *m*, intercept *b*

y

Label
(1D in this case)

x

Feature vector
(1D in this case)

Label
(1D in this case)

y

Model parameters: slope *m*, intercept *b*

x

Feature vector
(1D in this case)

For specific value of *x*,
assume *y* drawn from
Gaussian with mean
mx+*b*, standard dev σ

Predictive Data Analysis

Training data

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

Goal: Given new feature vector x, predict label y

- y is discrete (such as colors red and blue)
  ➔ prediction method is called a **classifier**

- y is continuous (such as a real number)
  ➔ prediction method is called a **regressor**

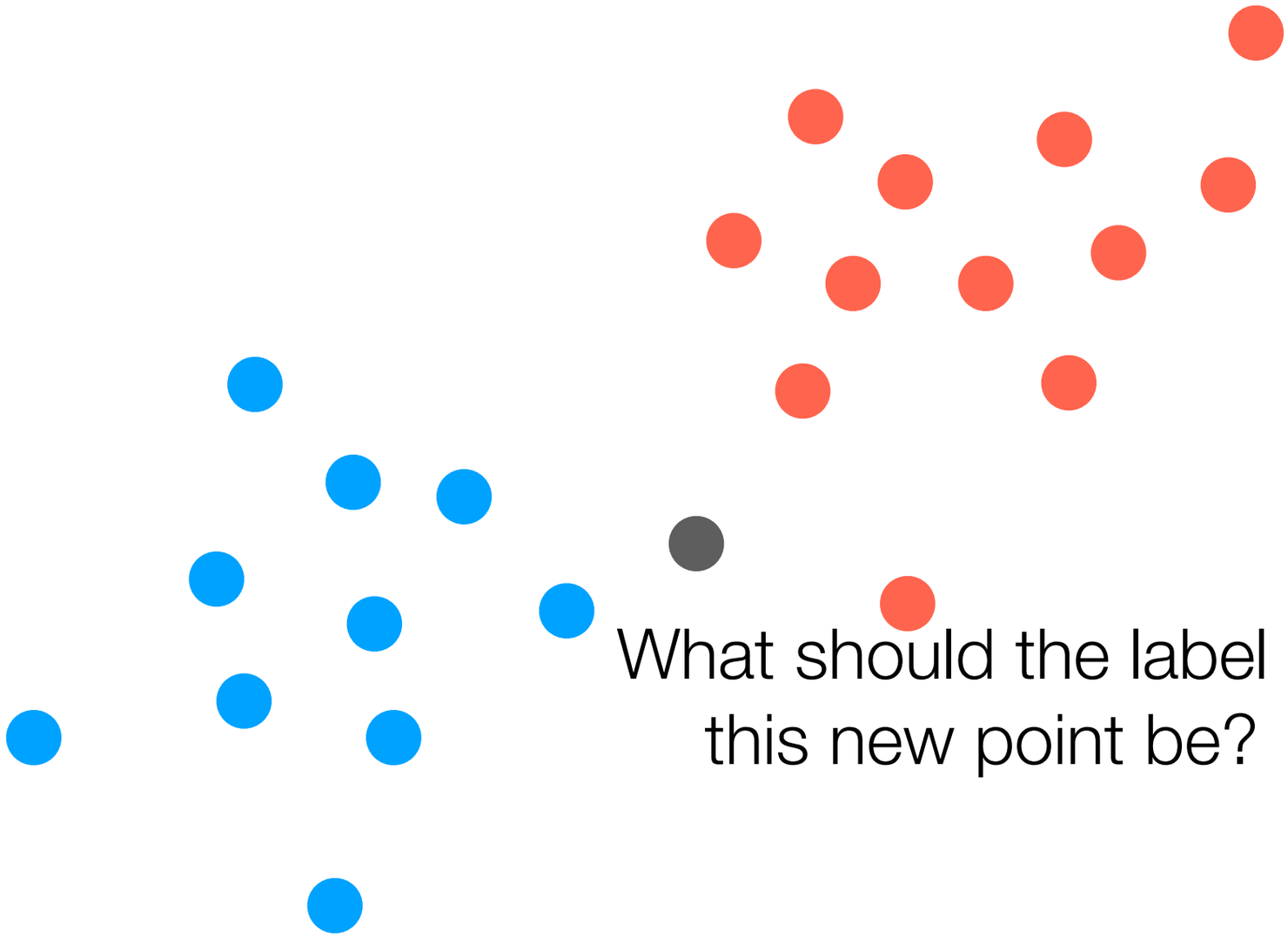
A giant zoo of methods

Generative Models

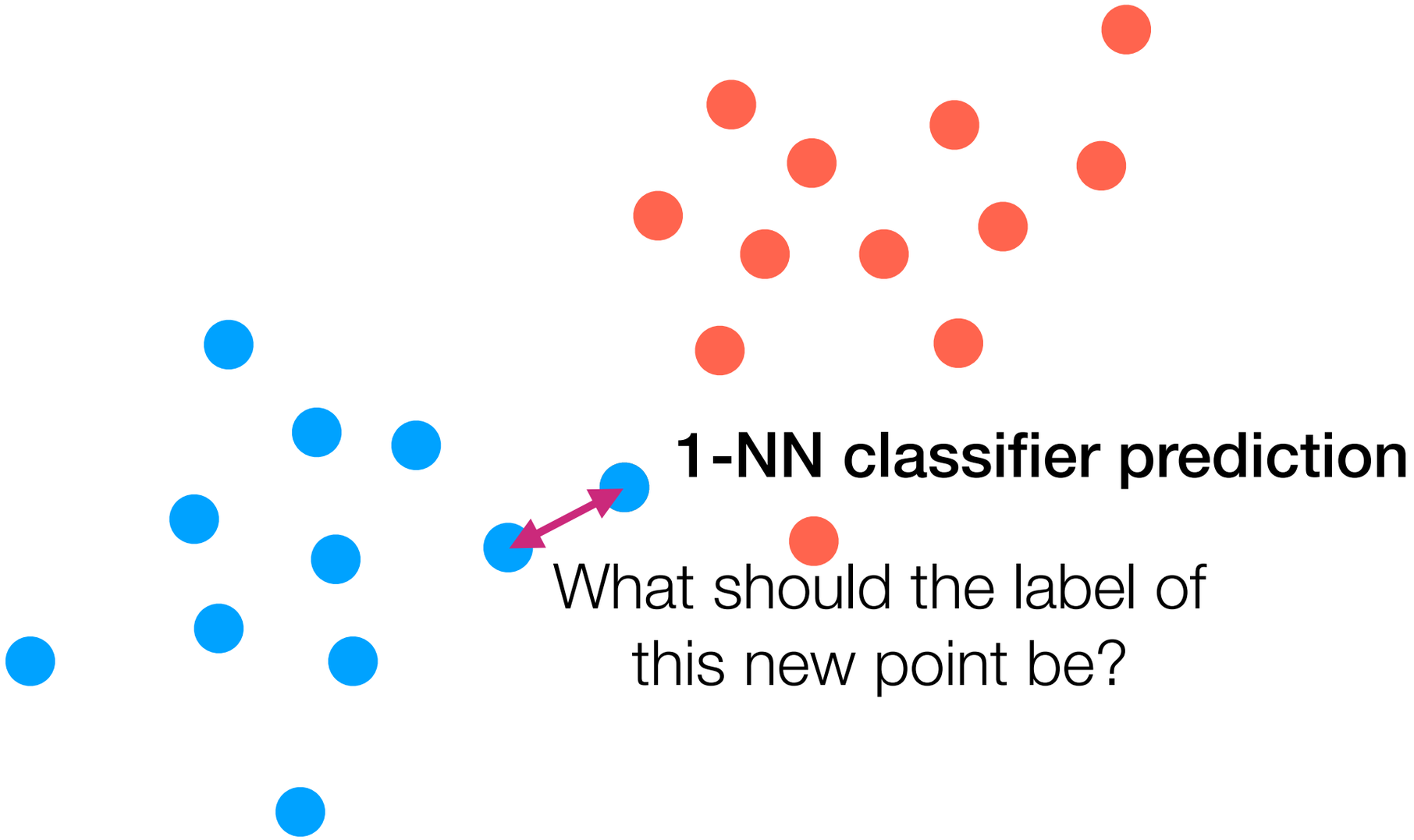
- Hypothesize a specific way in which data are generated
- After learning a generative model:
    - We can generate new synthetic data from the model
        - Usually generative models are probabilistic and we can evaluate probabilities for a new data point
- In contrast to generative models, there are *discriminative* methods that just care about learning a prediction rule

Example of a Discriminative Method: *k*-NN Classification

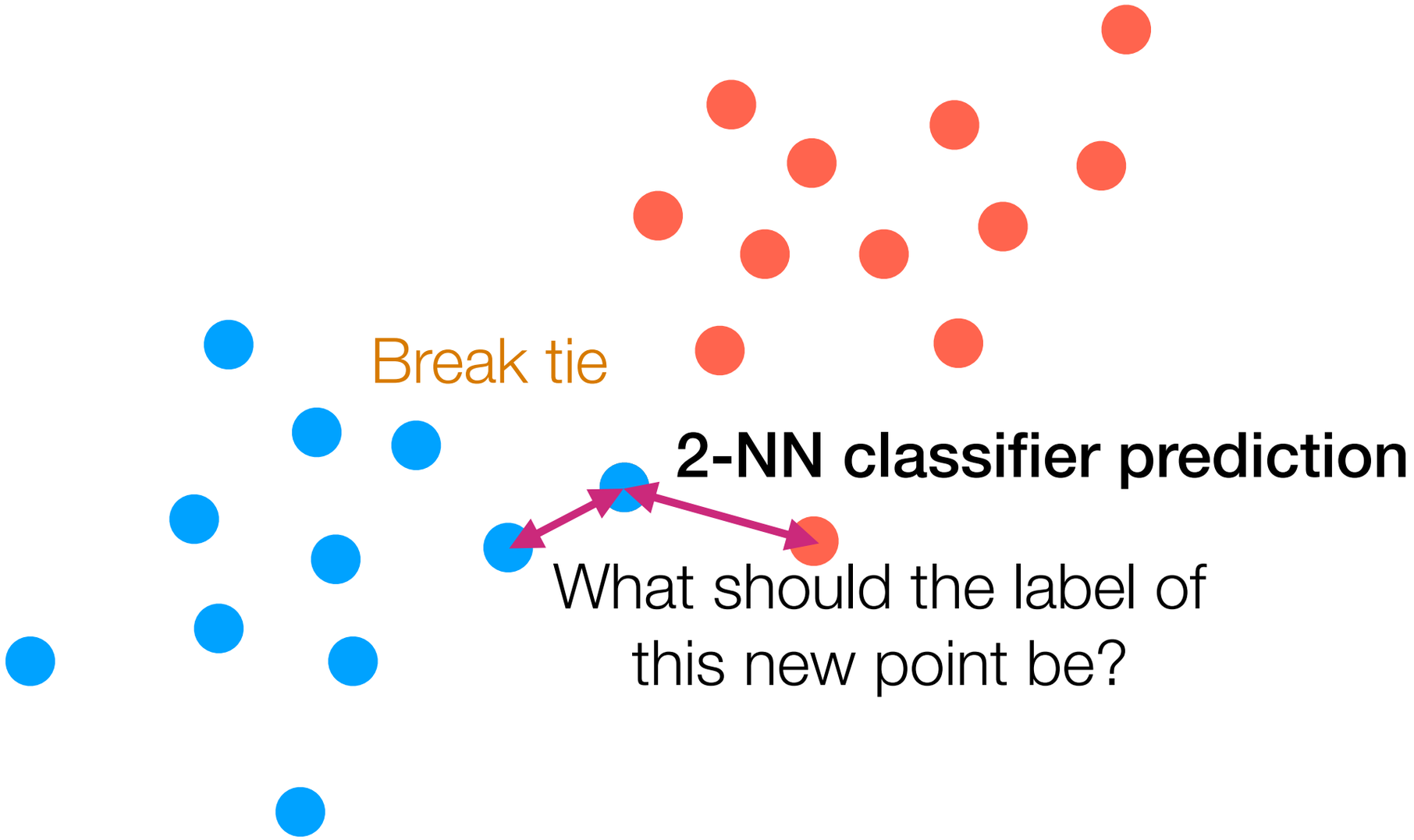
Example: *k*-NN Classification



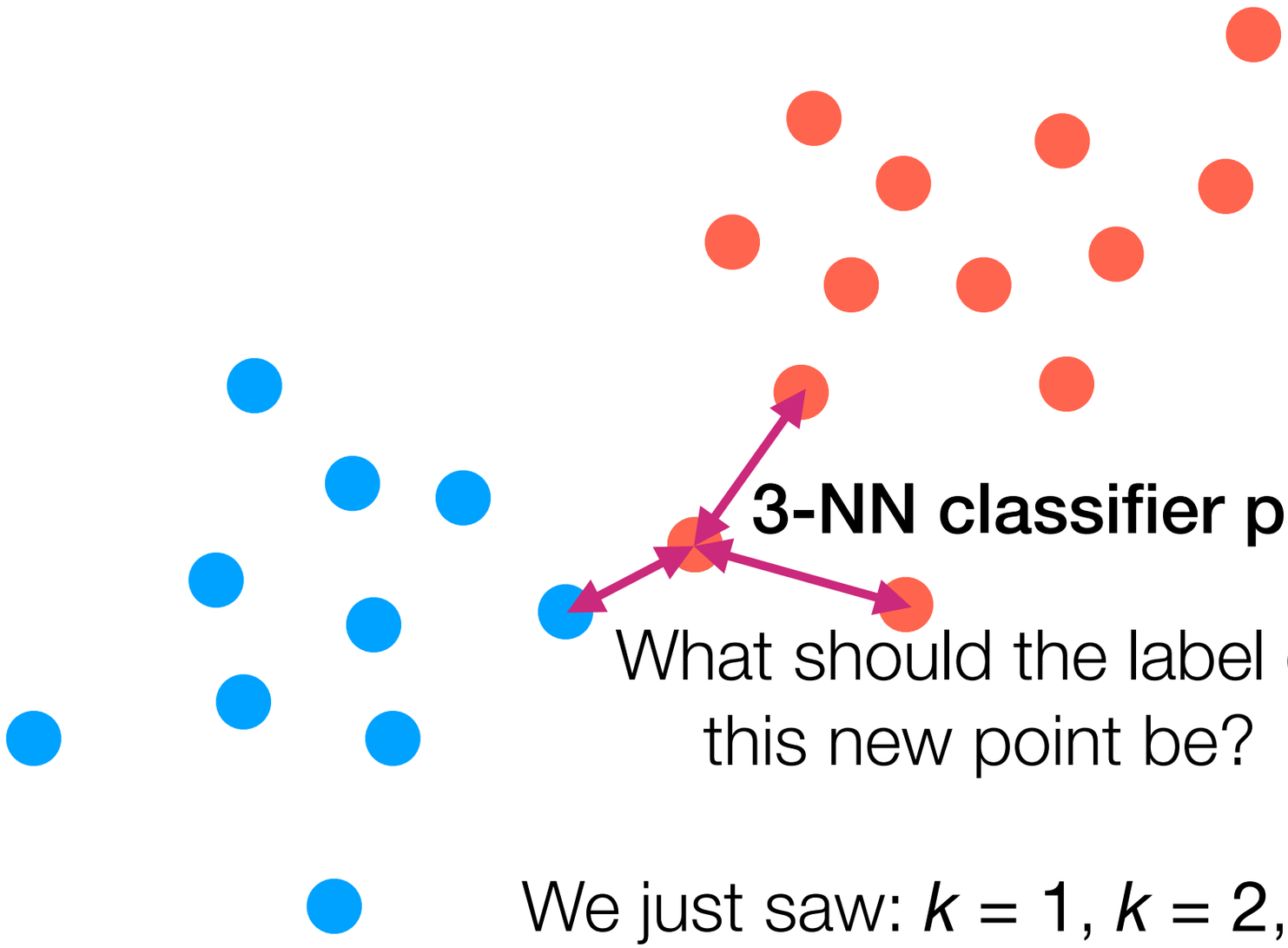
Example: *k*-NN Classification



Example: *k*-NN Classification



Example: *k*-NN Classification



3-NN classifier prediction

What should the label of this new point be?

We just saw: $k = 1$, $k = 2$, $k = 3$

What happens if $k = n$?

How do we choose *k*?

What I'll describe next can be used to select hyperparameter(s) for any prediction method

First: How do we assess how good a prediction method is?

Hyperparameters vs. Parameters

- We fit a model's parameter to training data (terminology: we "learn" the parameters)
- We pick values of hyperparameters and they do *not* get fit to training data

- Example: Gaussian mixture model
  - Hyperparameter: number of clusters *k*
  - Parameters: cluster probabilities, means, covariances

- Example: *k*-NN classification
  - Hyperparameter: number of nearest neighbors *k*
  - Parameters: N/A

Training data

Training data point

Training data point

Training data point

Training data point

Training data point

Training data point

Training data point

Training data point

Training data point

Training data point

Example: Each data point is an email and we know whether it is spam/ham

Want to classify these points correctly

Test data point

Test data point

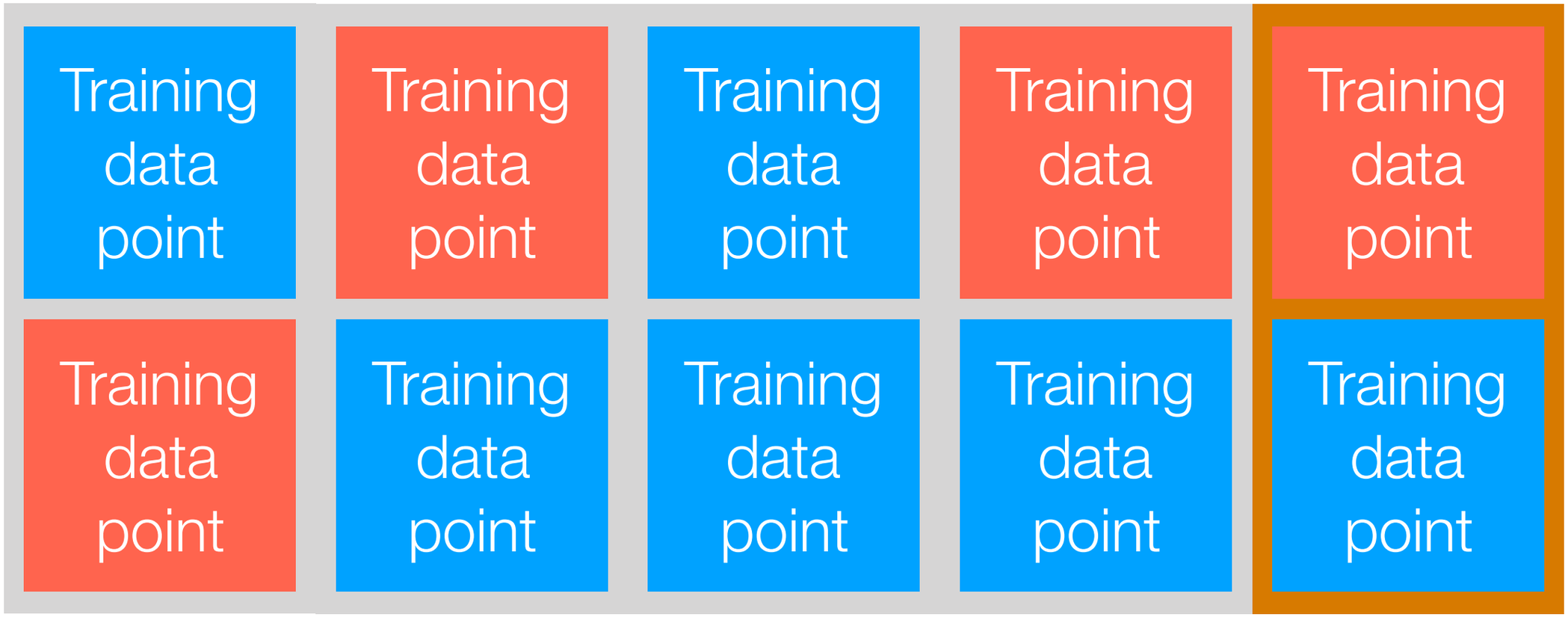
Test data point

Test data point

Test data point

Example: future emails to classify as spam/ham

Predicted labels

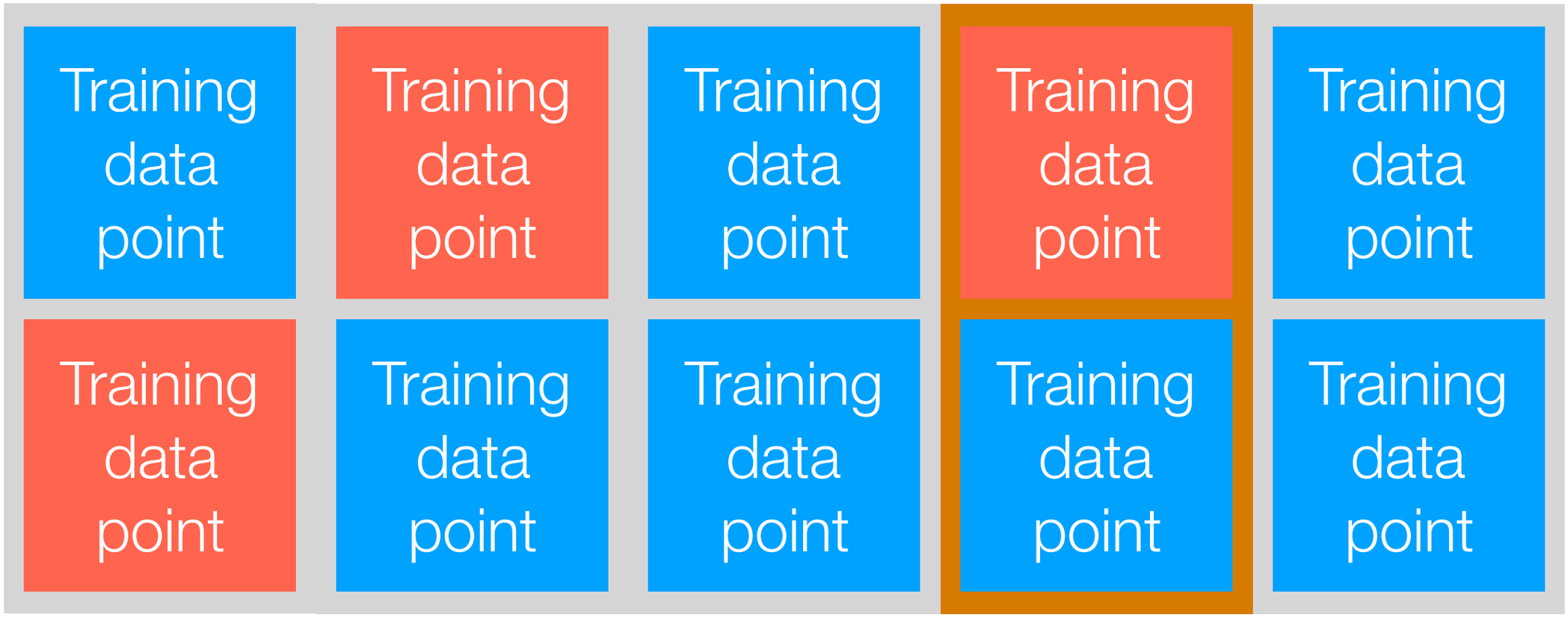


Train method on data in gray

Predict on data in orange

Compute prediction error

50%



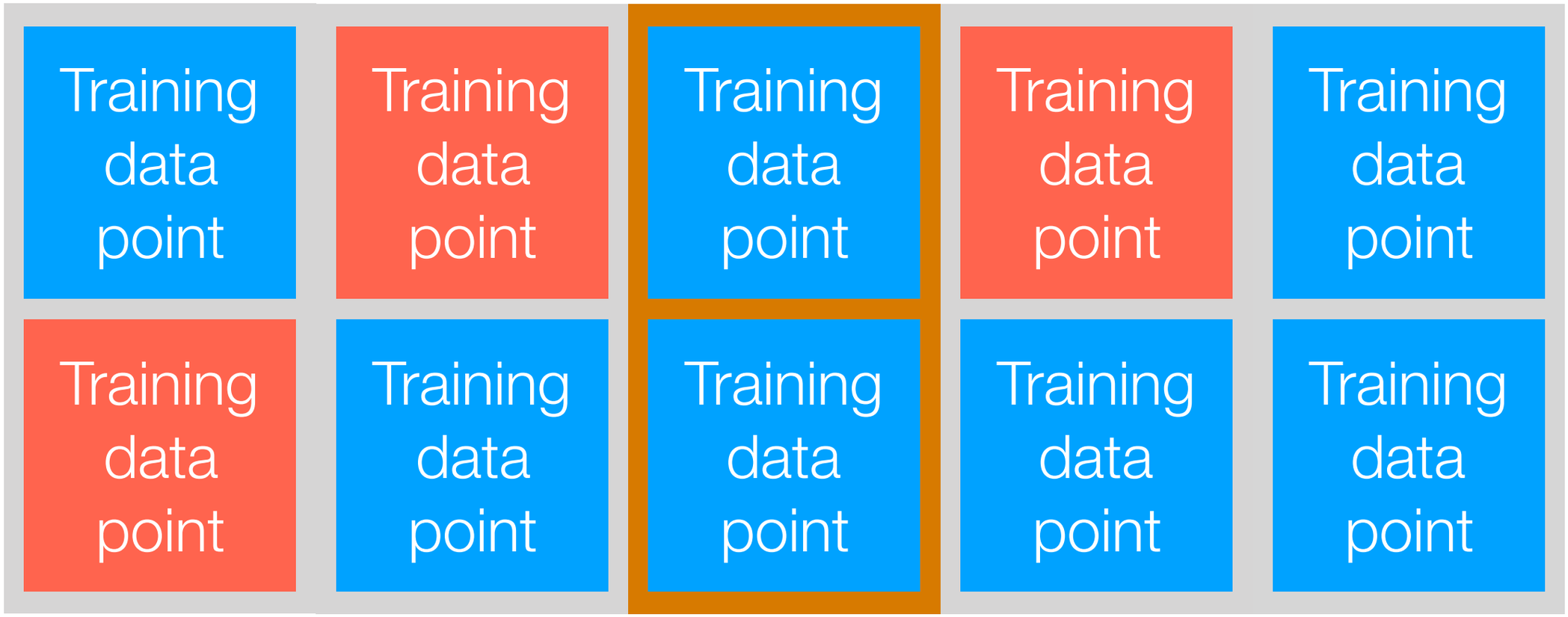
Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%



Train method on data in gray

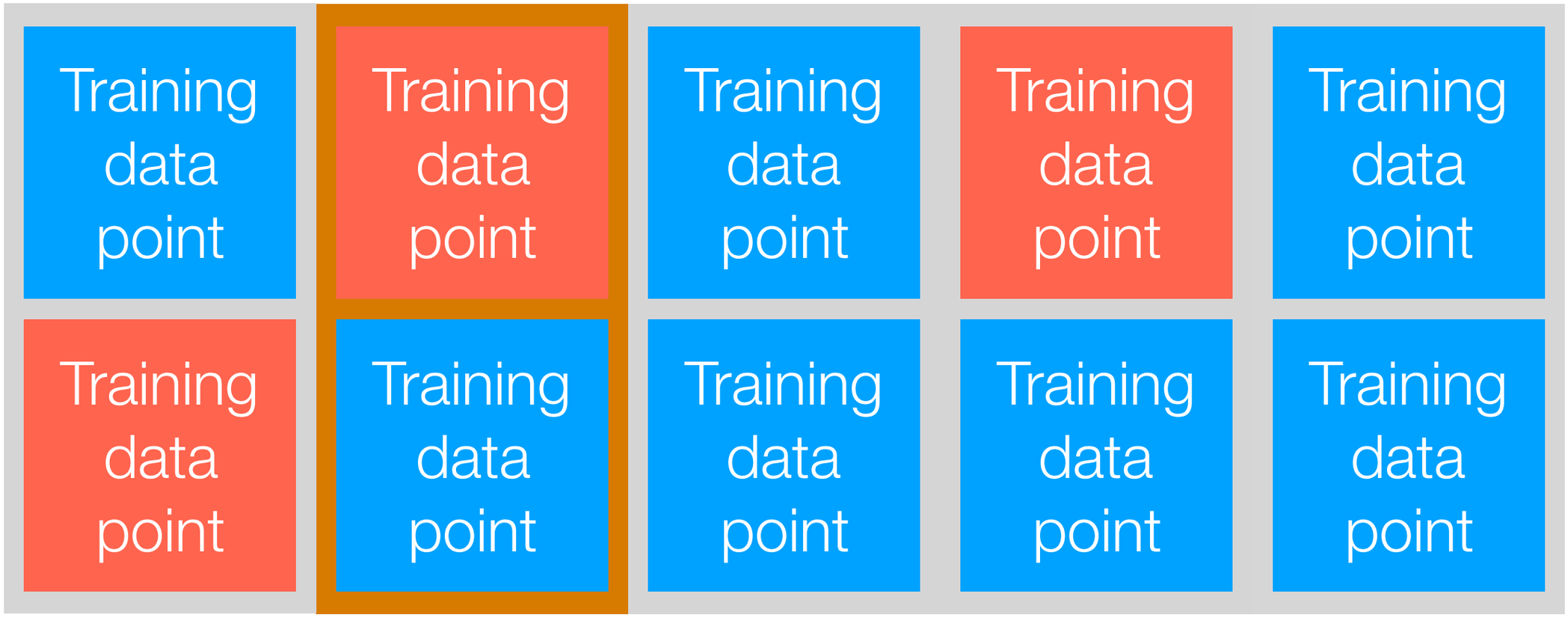
Predict on data in orange

Compute prediction error

50%

0%

50%



Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%

0%

50%

Training data point	Training data point	Training data point	Training data point	Training data point
Training data point	Training data point	Training data point	Training data point	Training data point

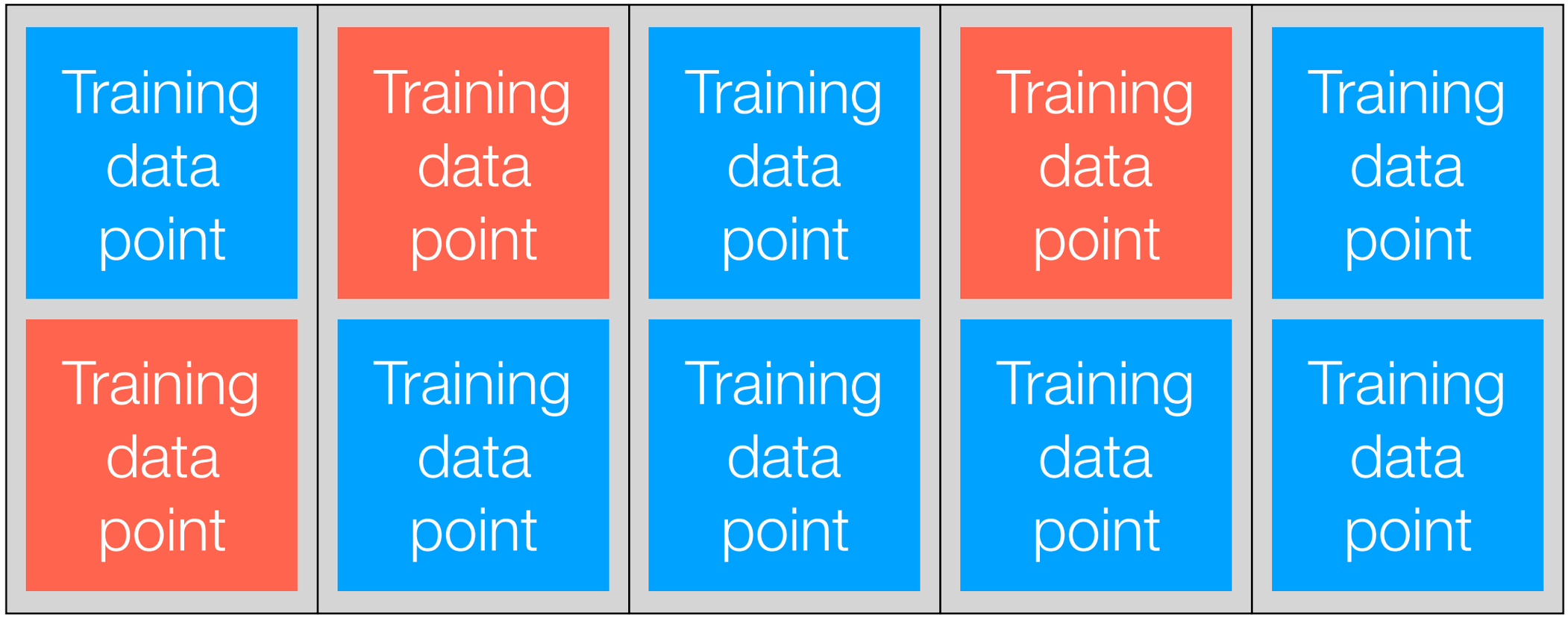
Train method on data in gray

Predict on data in orange

Compute prediction error

0% 0% 50% 0% 50%

Average error: (0+0+50+0+50)/5 = 20%



1. Shuffle data and put them into “folds” (5 folds in this example)
2. For each fold (which consists of its own train/validation sets):
   (a) Train on fold’s training data, test on fold’s validation data
   (b) Compute prediction error
3. Compute average prediction error across the folds

not the same *k* as in *k*-means or *k*-NN classification

k-fold Cross Validation

Training data point	Training data point	Training data point	Training data point	Training data point
Training data point	Training data point	Training data point	Training data point	Training data point

1. Shuffle data and put them into “folds” (*k*=5 folds in this example)
2. For each fold (which consists of its own train/validation sets):
   (a) Train on fold’s training data, test on fold’s validation data
   (b) Compute prediction error
3. Compute average prediction error across the folds

not the same *k* as in *k*-means or *k*-NN classification

k-fold Cross Validation

Training data point	Training data point	Training data point	Training data point	Training data point
Training data point	Training data point	Training data point	Training data point	Training data point

1. Shuffle data and put them into "folds" (*k*=5 folds in this example)
2. For each fold (which consists of its own train/validation sets):
   (a) Train on fold's training data, test on fold's validation data
   (b) Compute **some sort of prediction score**
3. Compute **average prediction score** across the folds

"cross validation score"

Choosing *k* in *k*-NN Classification

Note: *k*-NN classifier has a single hyperparameter *k*

For each *k* = 1, 2, 3, …, the maximum *k* you are willing to try:

Compute 5-fold cross validation score using *k*-NN classifier as prediction method

Use whichever *k* has the best cross validation score

Automatic Hyperparameter Selection

Suppose the prediction algorithm you're using has hyperparameters θ

For each hyperparameter setting θ you are willing to try:

Compute 5-fold cross validation score using your algorithm with hyperparameters θ

Use whichever θ has the best cross validation score

Why 5?

People have found using 10 folds or 5 folds to work well in practice but it's just empirical — there's no deep reason

Important: the cross validation score is trying to predict what the prediction quality will be on the unseen test data

Our earlier example had a cross validation score of 20% error

This is a guess at how well the prediction method should perform on test data

This guess is <u>not</u> always accurate

Example: Each data point is an email and we know whether it is spam/ham

Want to classify these points correctly

Test data point

Test data point

Test data point

Test data point

Test data point

Example: future emails to classify as spam/ham

Different Ways to Measure Accuracy

Simplest way:

- **Raw error rate:** fraction of predicted labels that are wrong (this was in our cross validation example earlier)

In “binary” classification (there are 2 labels such as spam/ham) when 1 label is considered “positive” and the other “negative”:

- **Precision:** among data points predicted to be “positive”, what fraction of these predictions is correct?
- **Recall:** among data points that are actually “positive”, what fraction of these points is predicted correctly as “positive”? (also called **true positive rate**)
- **F1 score:**

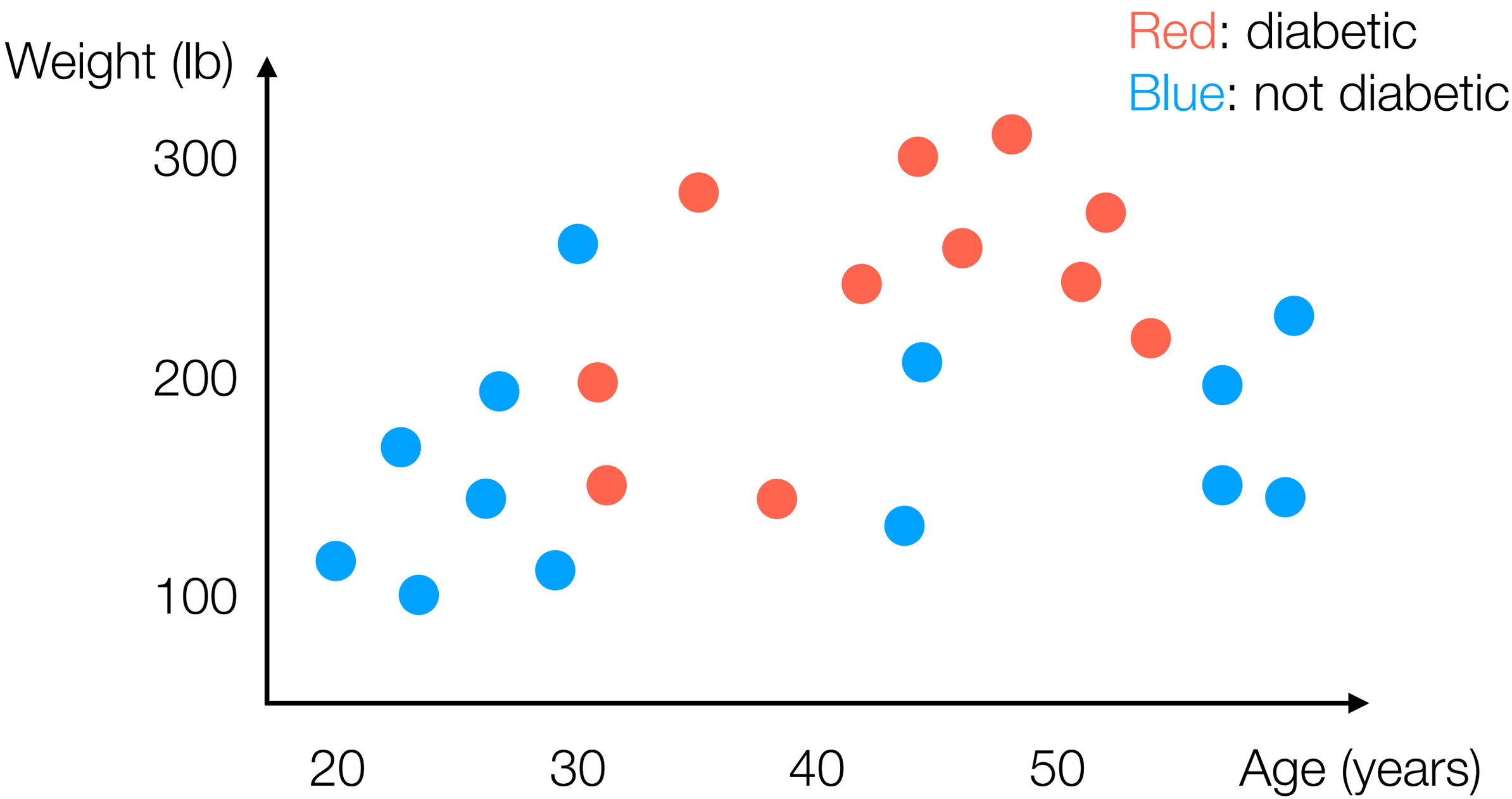
$$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Prediction and Validation

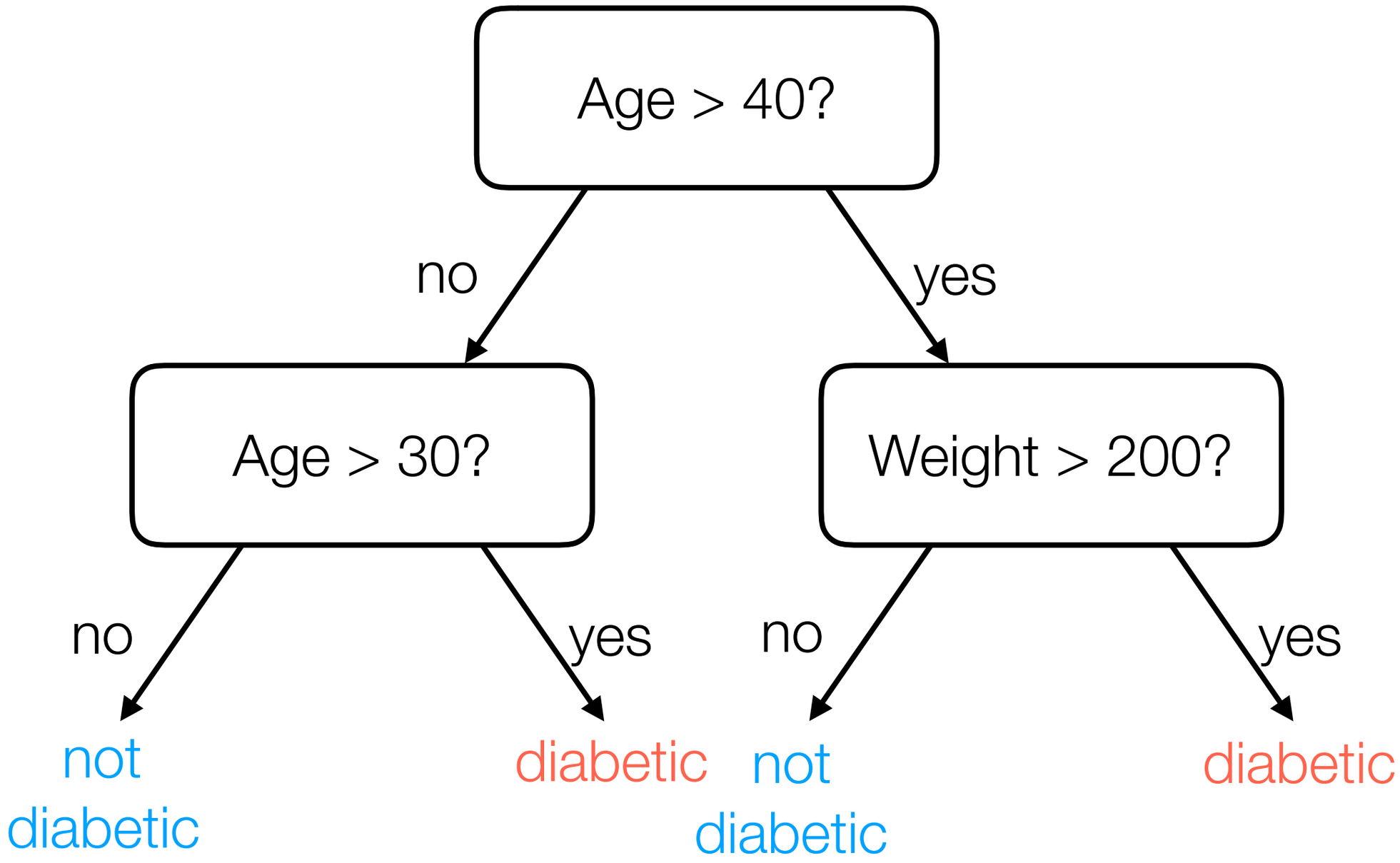
Demo

Decision Trees

Example Made-Up Data



Example Decision Tree

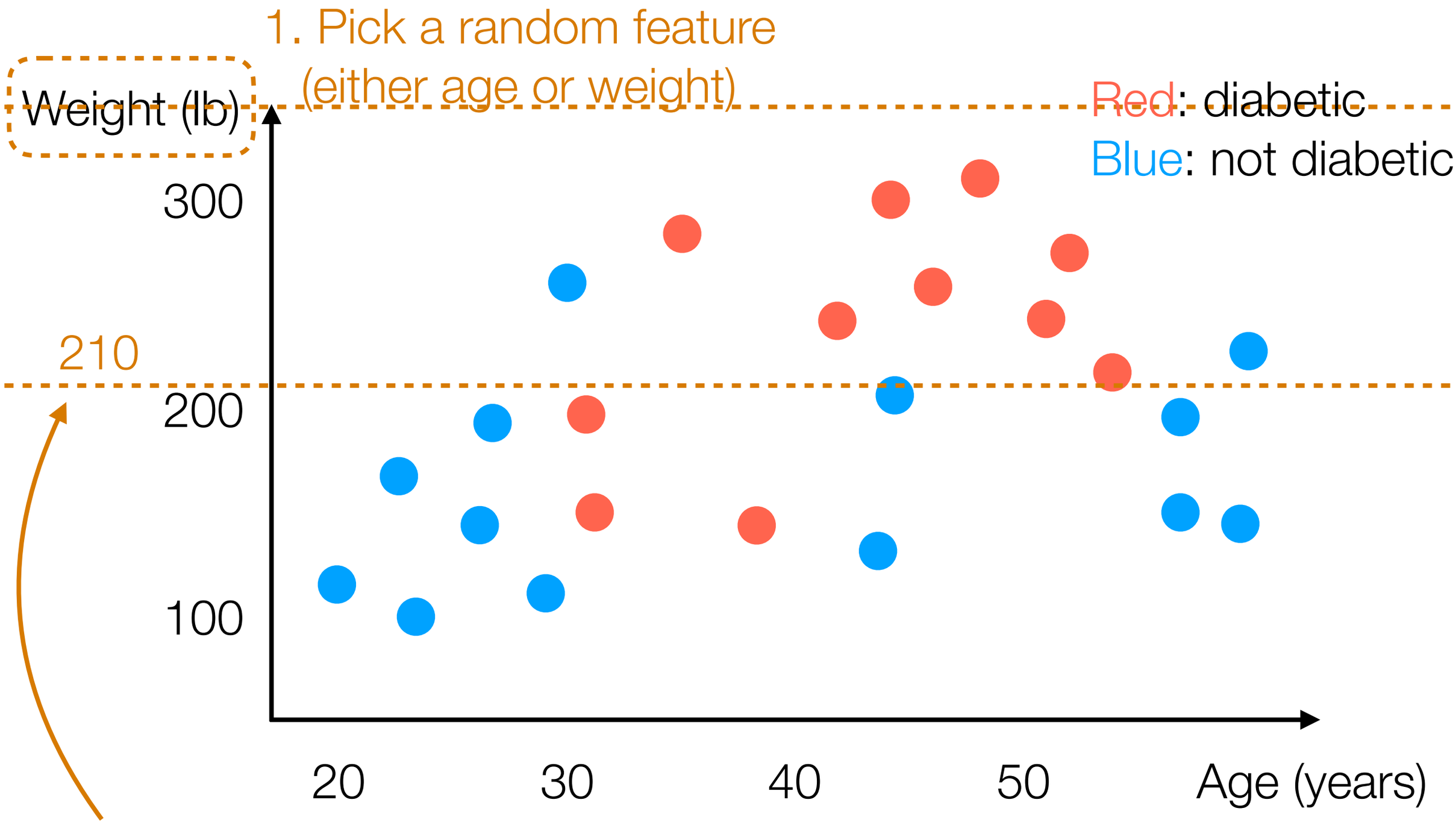


Learning a Decision Tree

- Many ways: general approach actually looks a lot like divisive clustering *but accounts for label information*

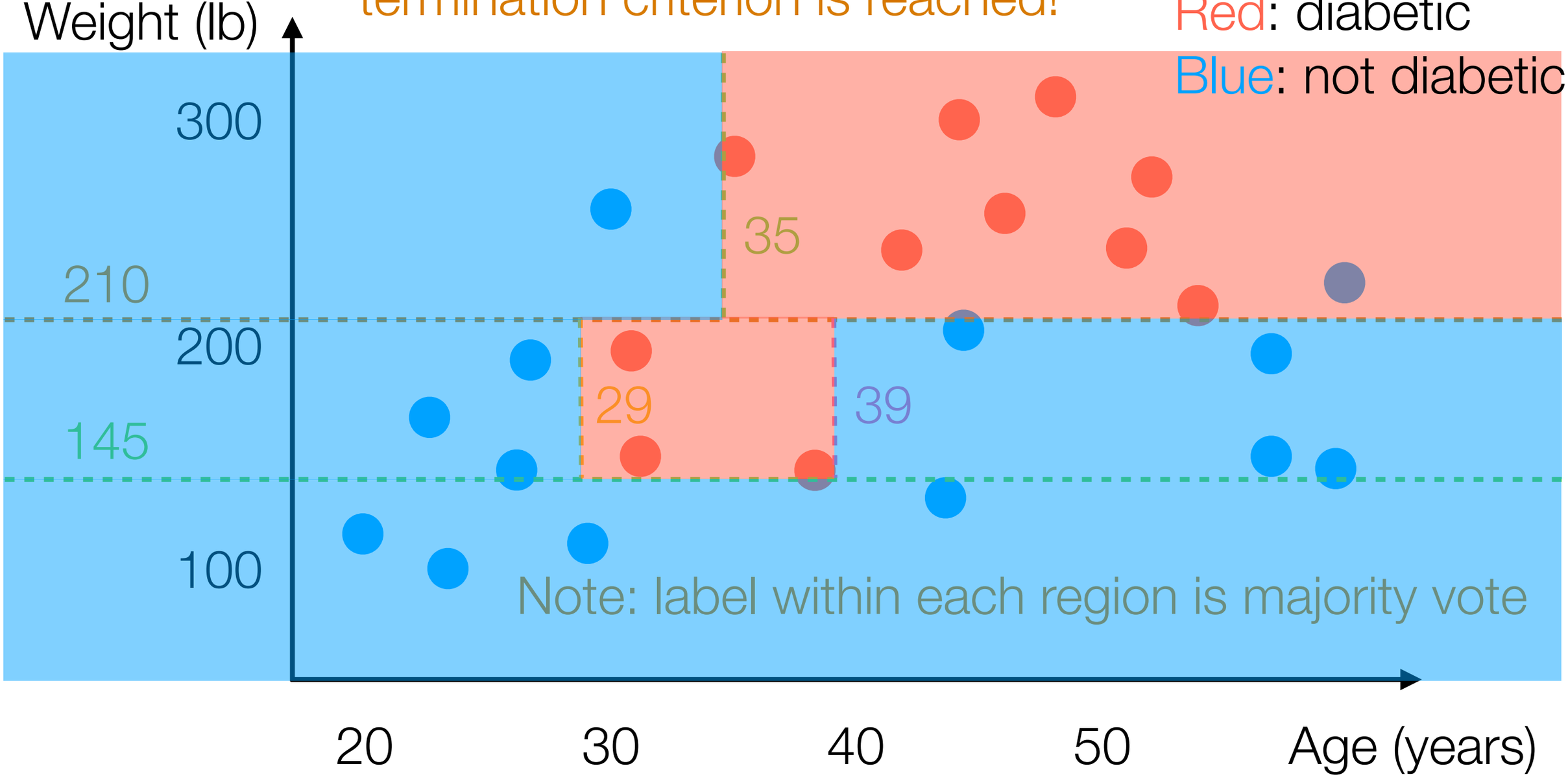
- I'll show one way (that nobody actually uses in practice) but it's easy to explain

Learning a Decision Tree



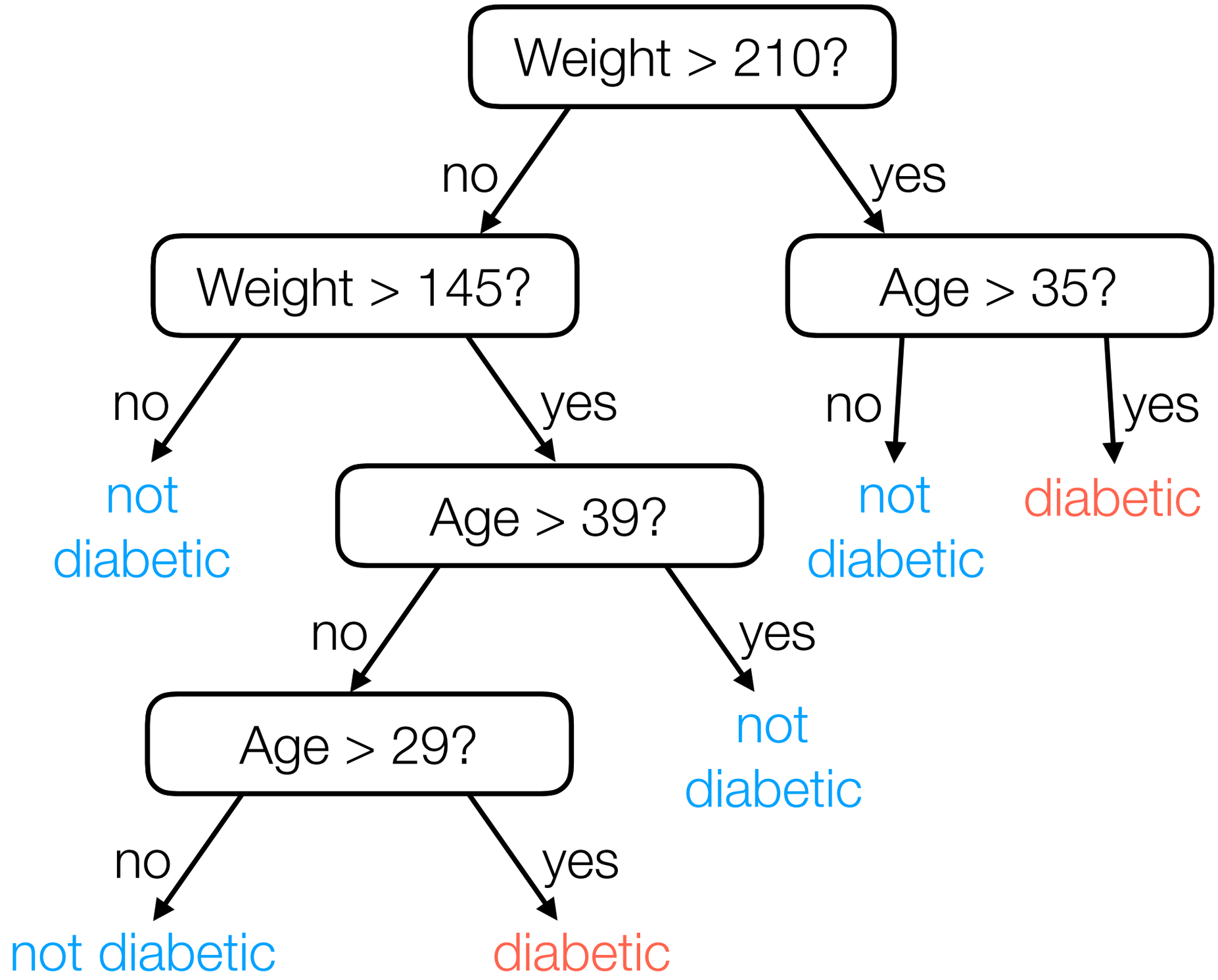
Learning a Decision Tree

Within each side, recurse until a termination criterion is reached!



Example termination criteria: ≥90% points within region has same label, number of points within region is <5

Decision Tree Learned



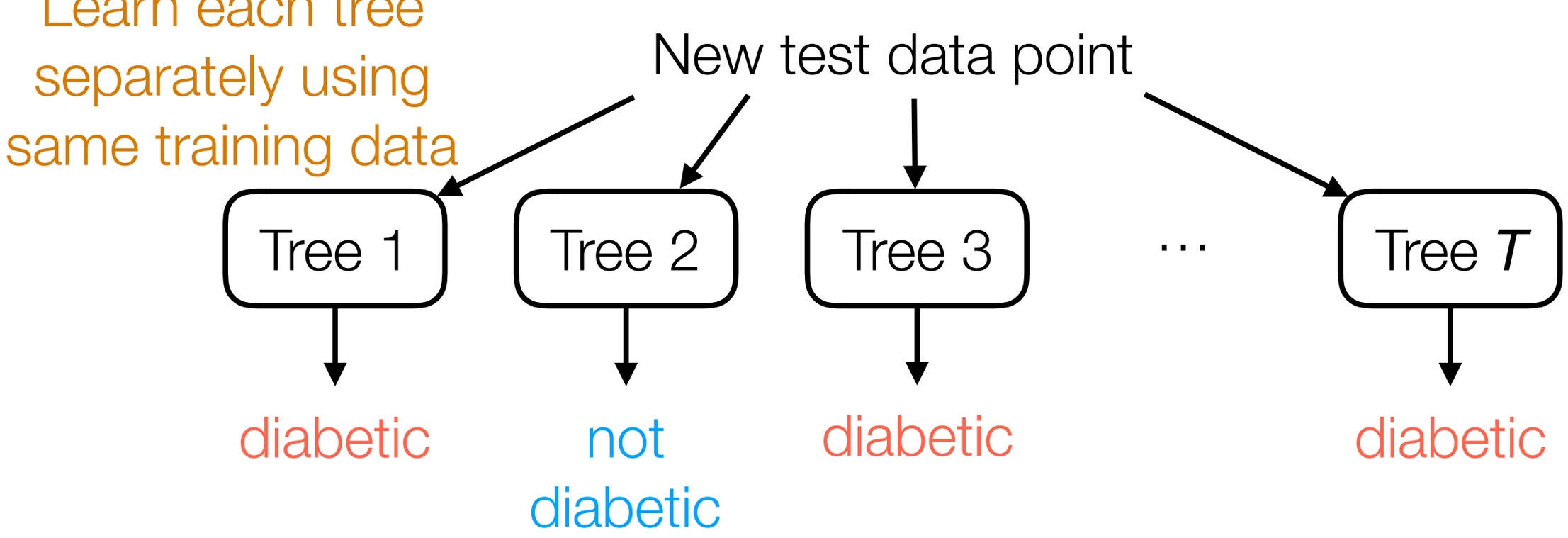
For a new person with feature vector (age, weight), easy to predict!

Decision Forest for Classification

- Typically, a decision tree is learned with randomness (e.g., we randomly chose which feature to threshold)

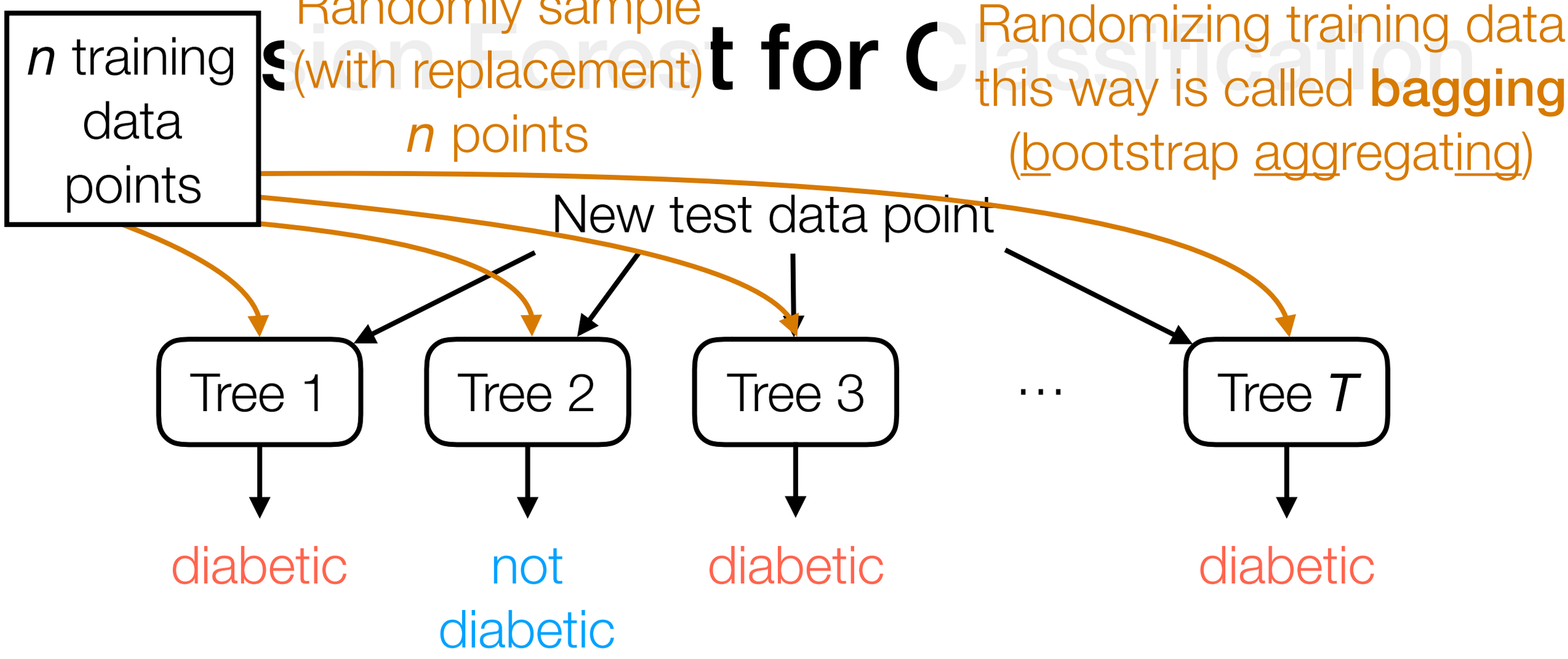
  ➔ by re-running the same learning procedure, we can get different decision trees that make different predictions!

- For a more stable prediction, use many decision trees



Final prediction: majority vote of the different trees' predictions

Random Forest for Classification



Question: What happens if all the trees are the same?

Adding randomness can make trees more different!

- **Random Forest:** in addition to randomly choosing features to threshold, also randomize training data used for each tree
- **Extremely randomized trees:** further randomize thresholds rather than trying to pick clever thresholds