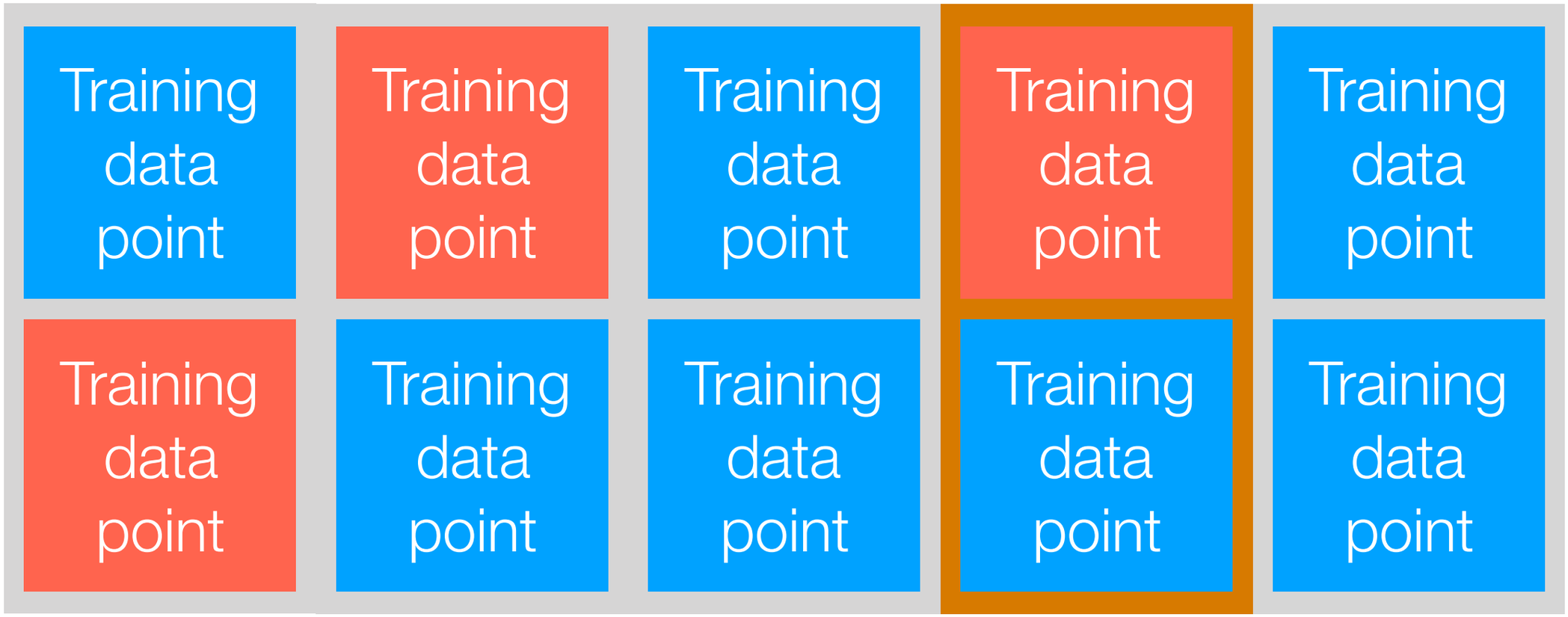
Carnegie Mellon University
Heinz College

94-775/95-865 Lecture 9: Model Validation, Decision Trees/Forests

George Chen



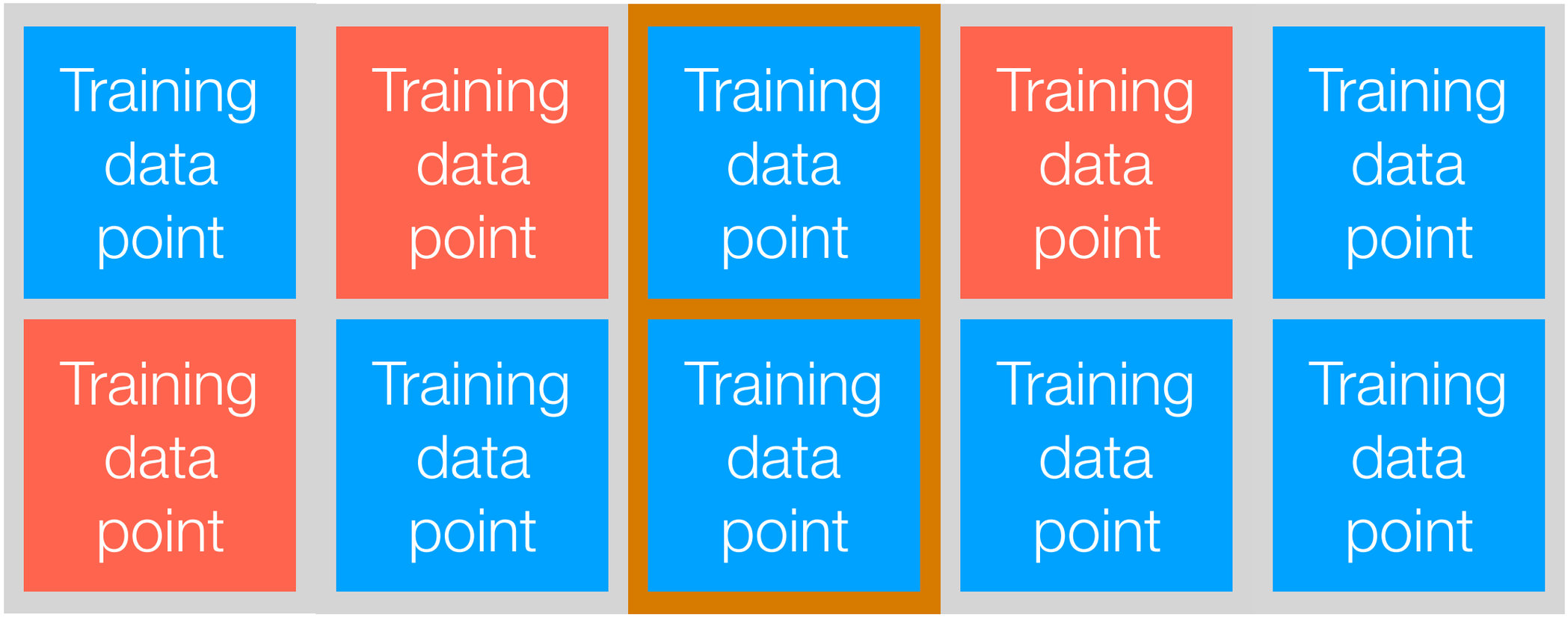
Train method on data in gray

Predict on data in orange

Compute prediction error

0%

50%



Train method on data in gray

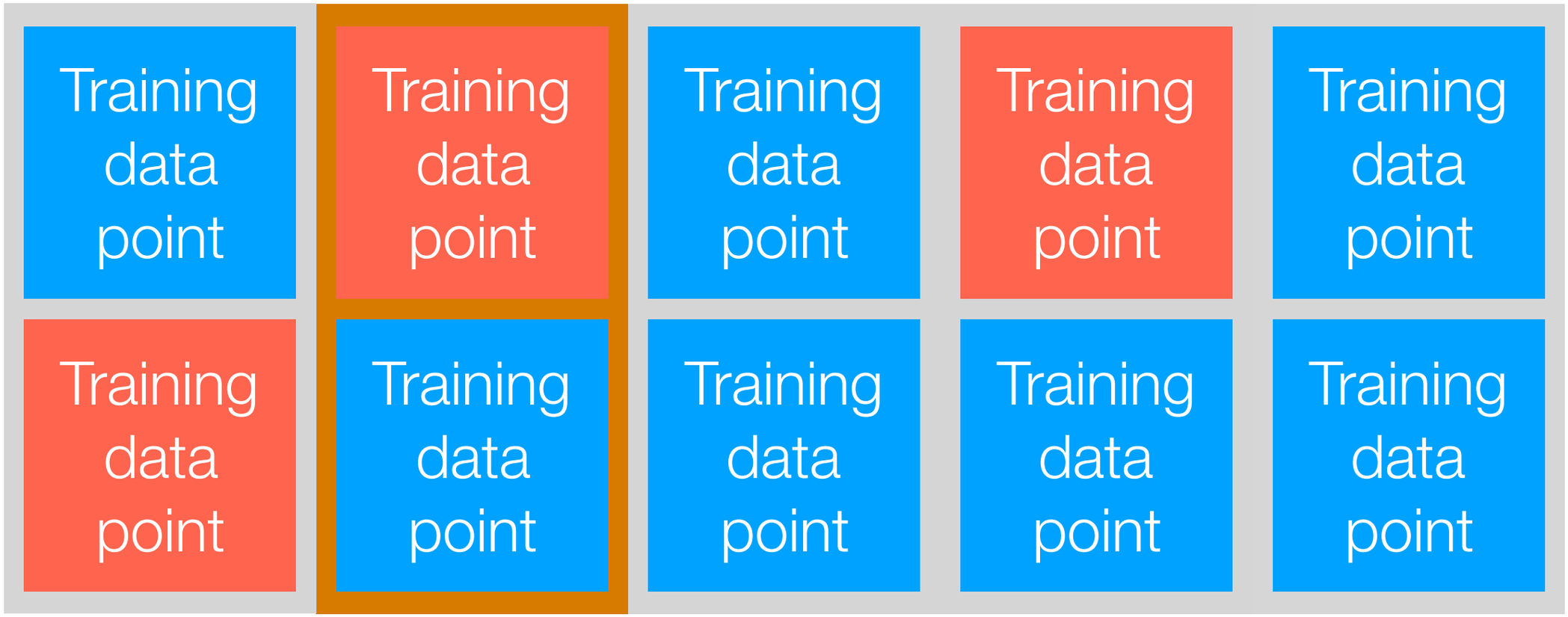
Predict on data in orange

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%

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"

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 errors from simple data splitting and cross-validation are *estimates* of the true error on test data!

Example: earlier, we got a cross validation score of 20% error

This is a guess for the error we will get 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

Cross-Validation Remarks

- *k*-fold cross-validation is a randomized procedure
  - Re-running CV results in different cross-validation scores!
- Suppose there are *n* data points and *k* folds
  - If we are trying 10 different hyperparameter settings, how many models do we fit?
    - If this number is similar in size to *n*, CV can overfit!
  - How many training data are used to train each model during cross-validation?
    - Smaller # folds typically means faster training
- If $k = n$, would re-running cross-validation result in different cross-validation scores? What about $k = 2$?

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:**

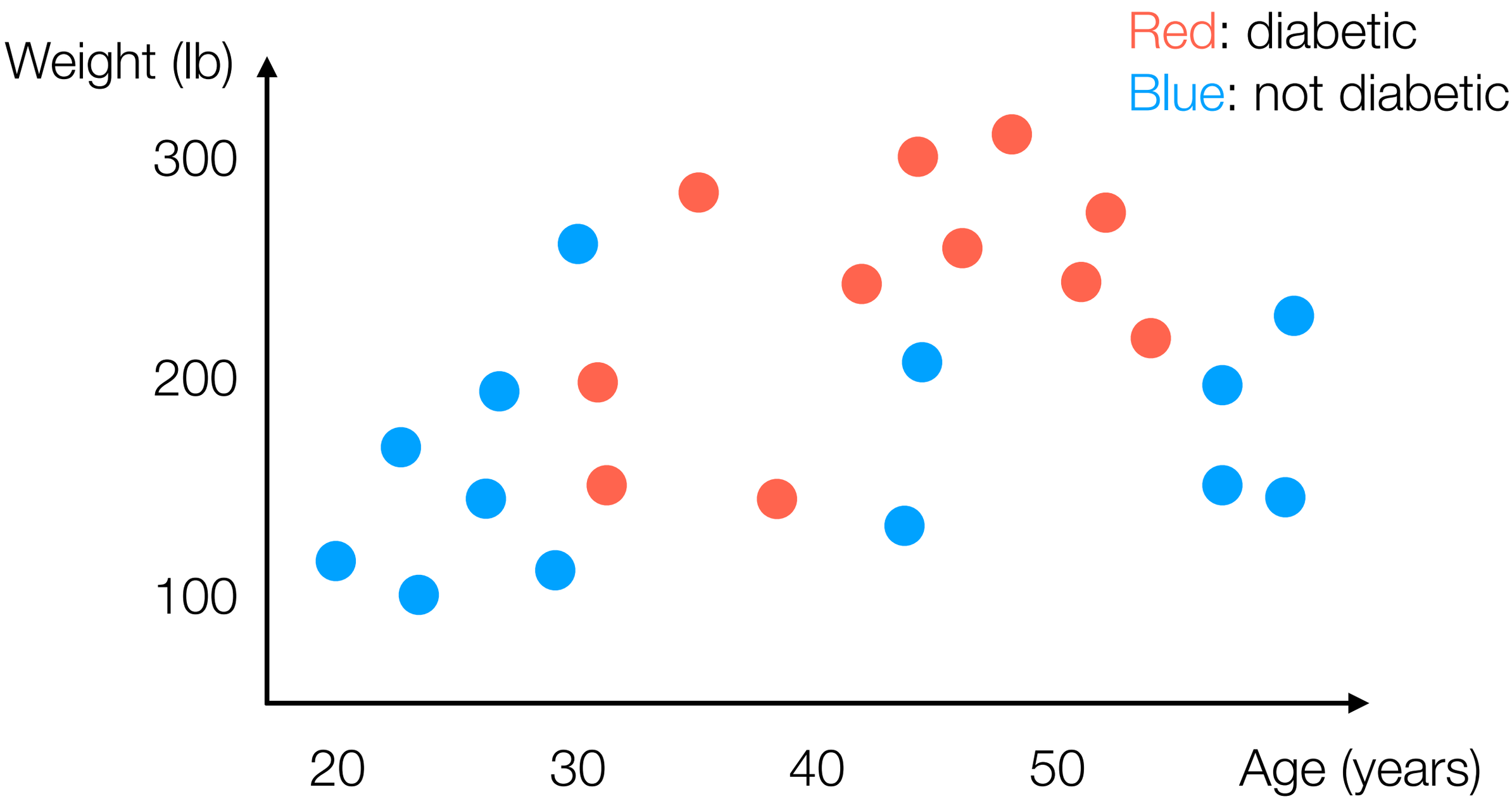
$$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Prediction and Model Validation

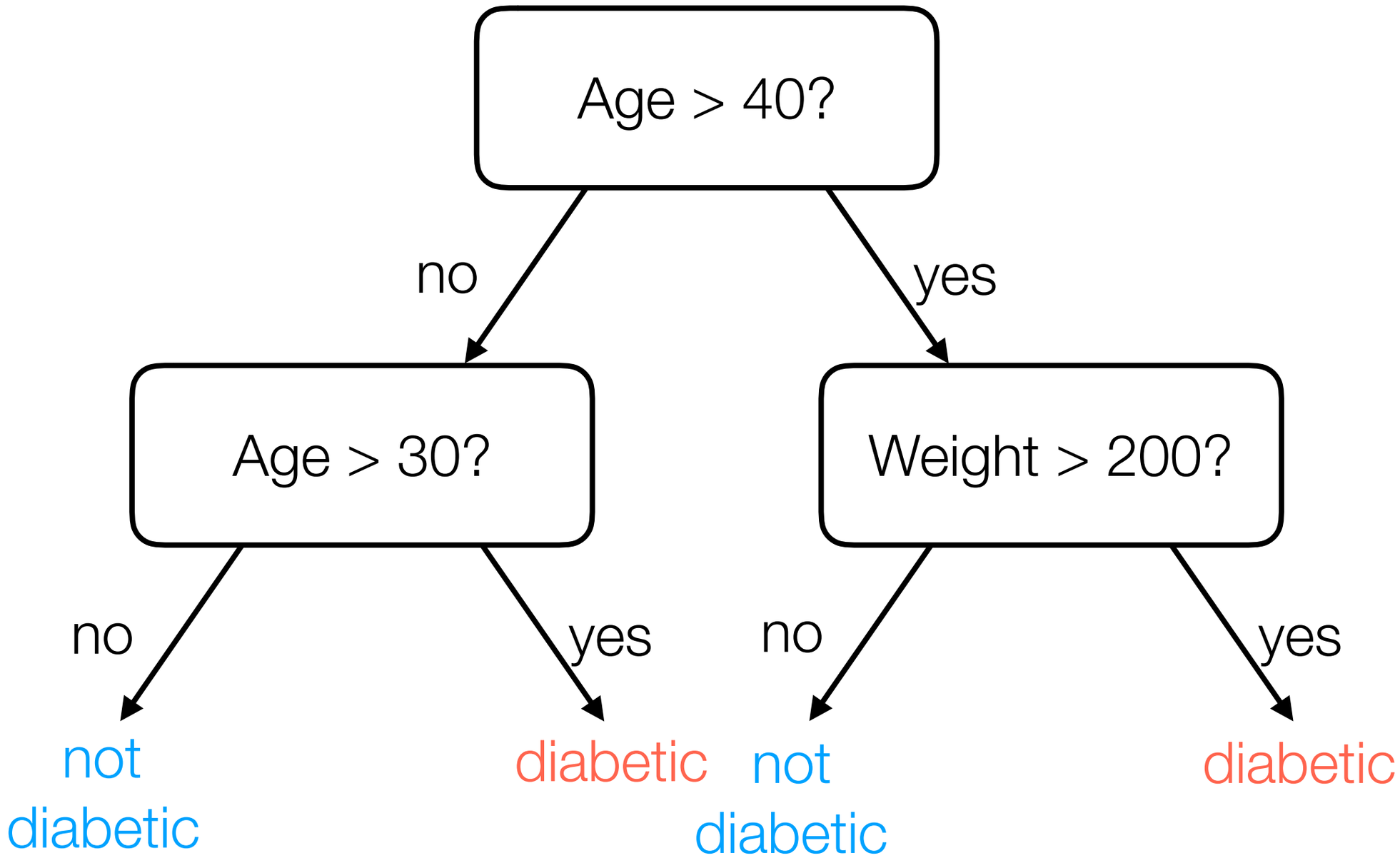
Demo

Decision Trees

Example Made-Up Data



Example Decision Tree

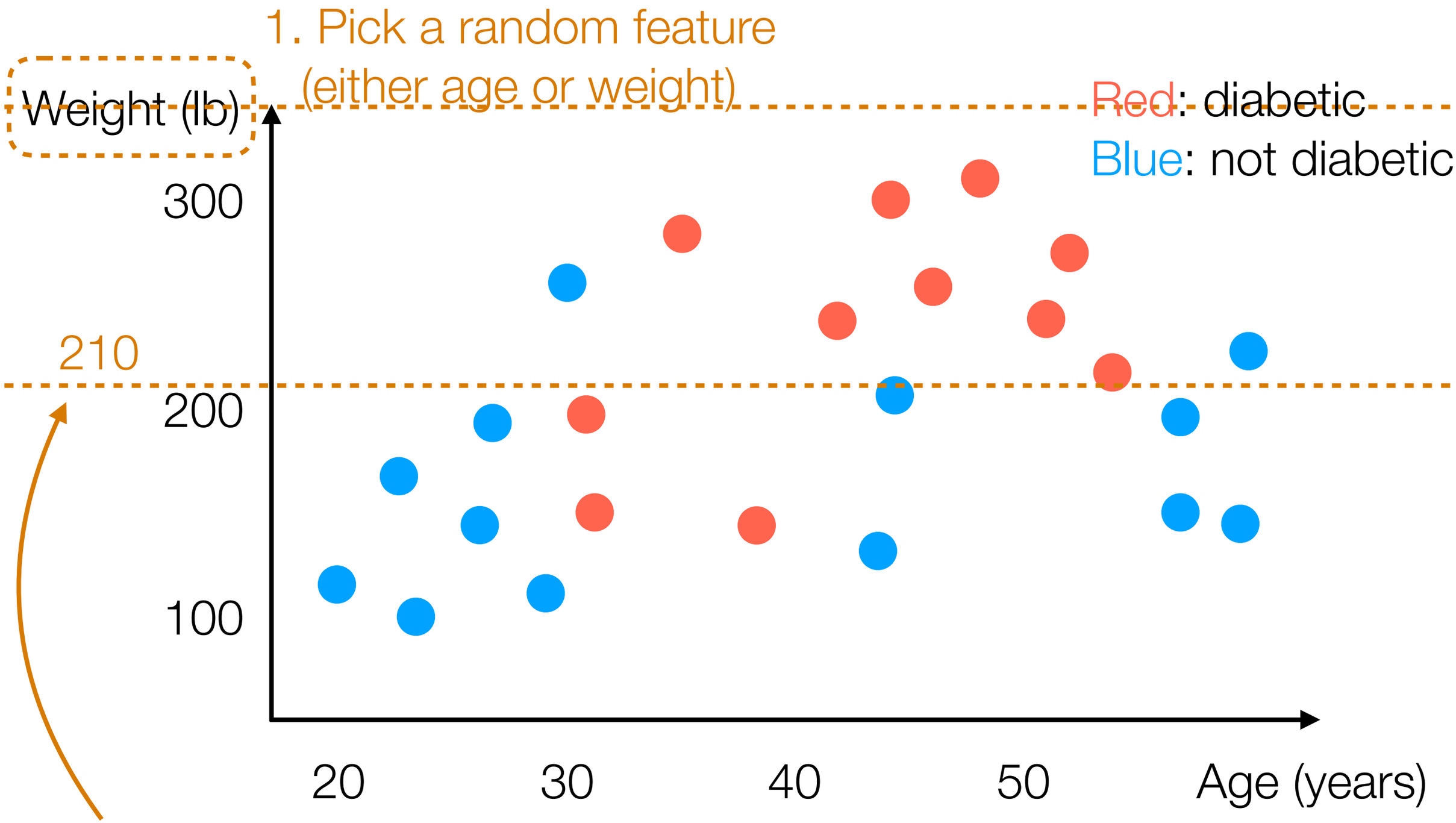


Learning a Decision Tree

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

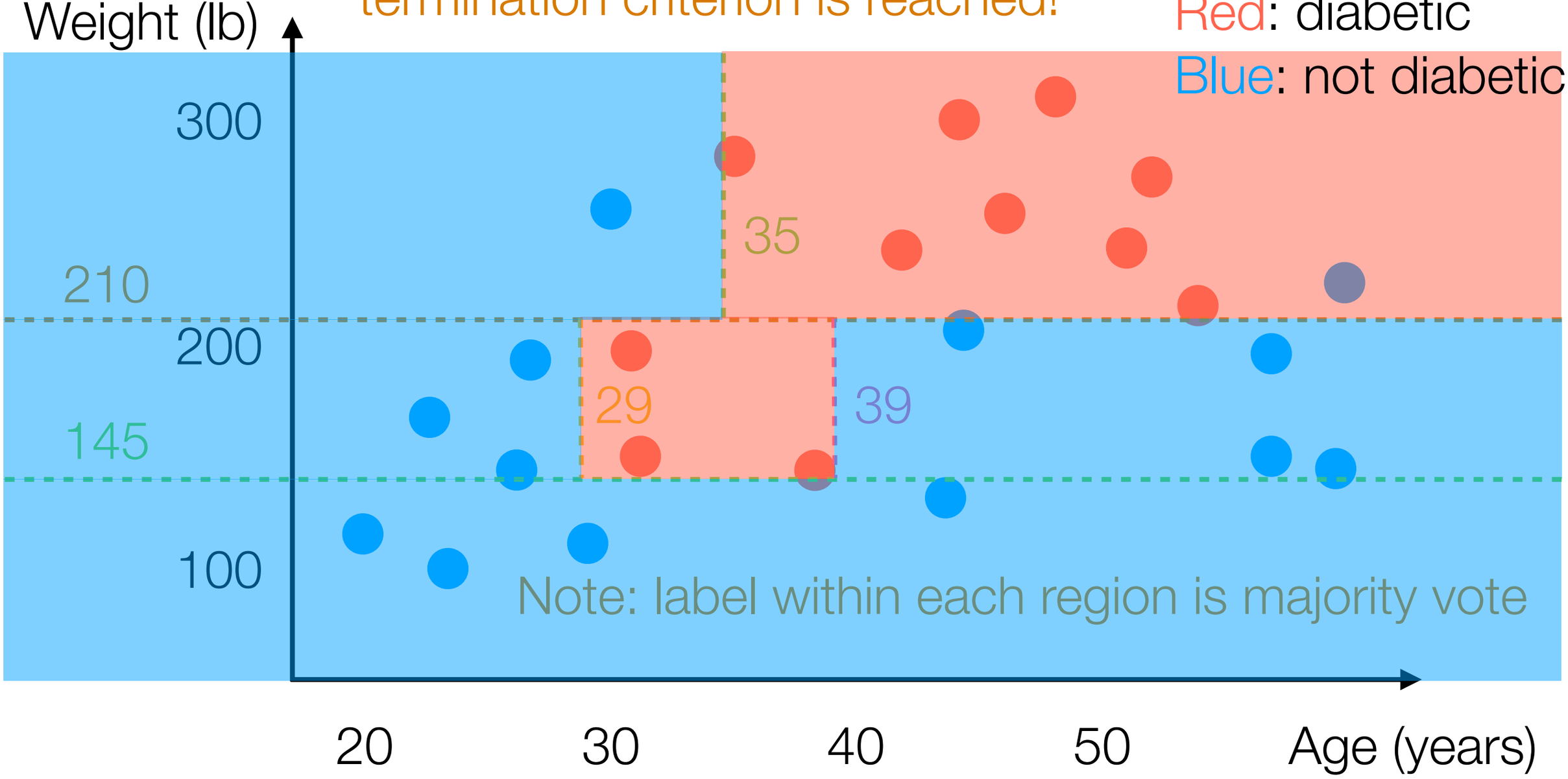
- I'll show one way (that nobody actually uses in practice) but it's easy to explain

Learning a Decision Tree



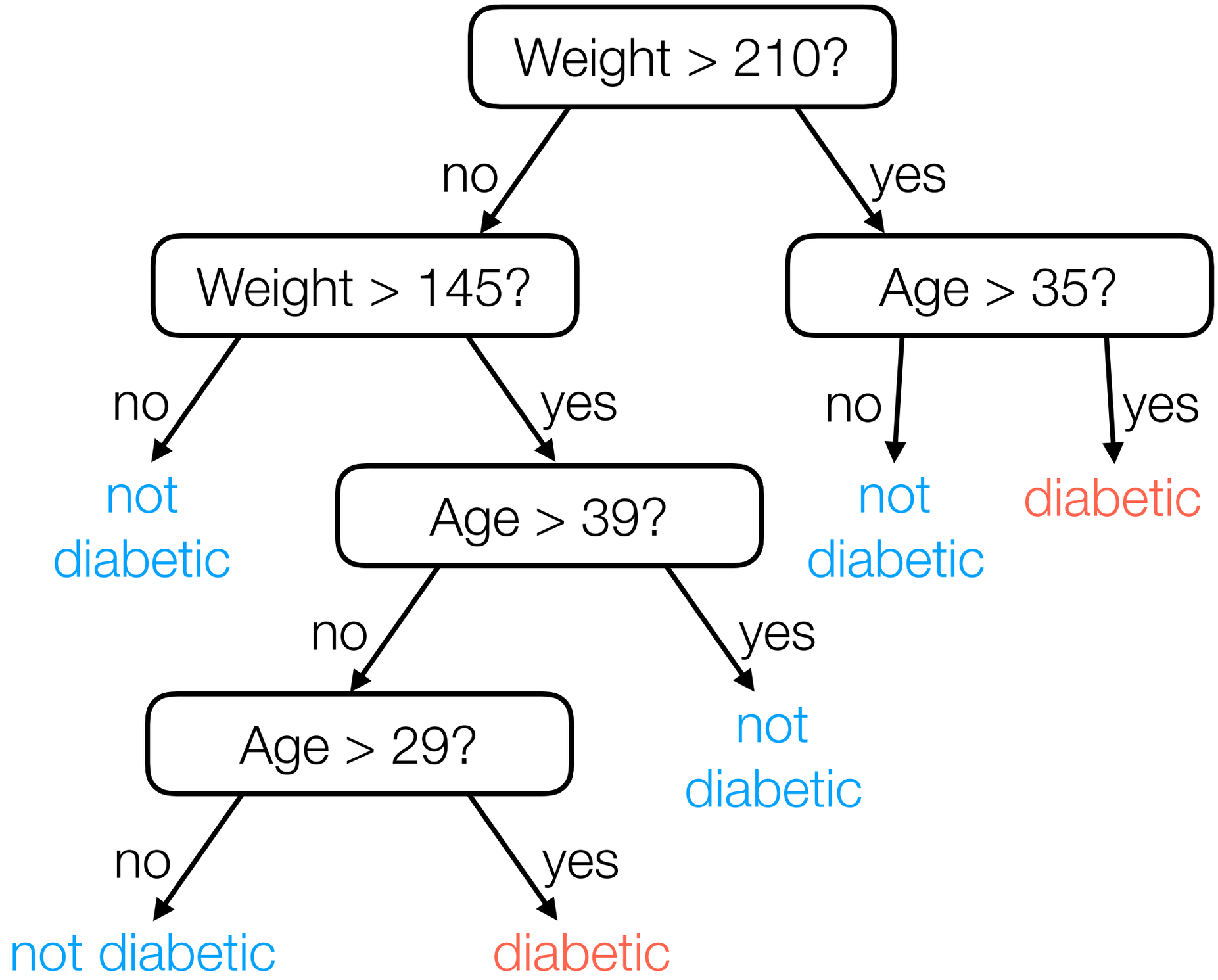
Learning a Decision Tree

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



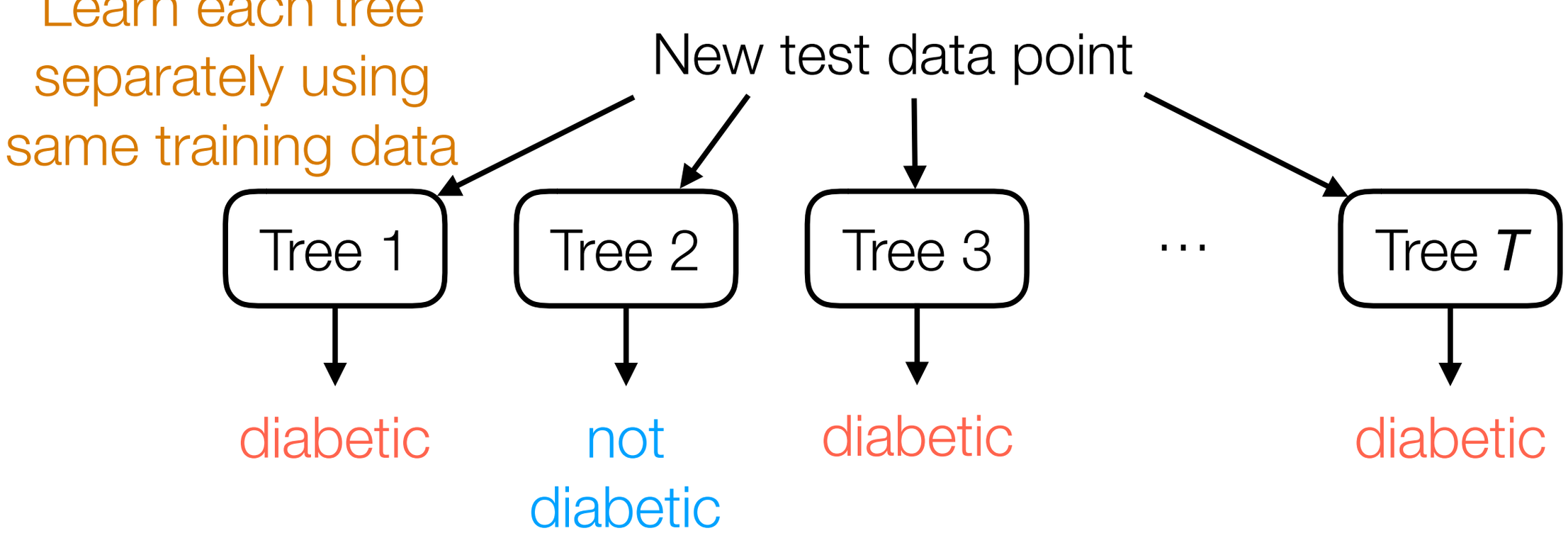
For a new person with feature vector (age, weight), easy to predict!

Decision Forest for Classification

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

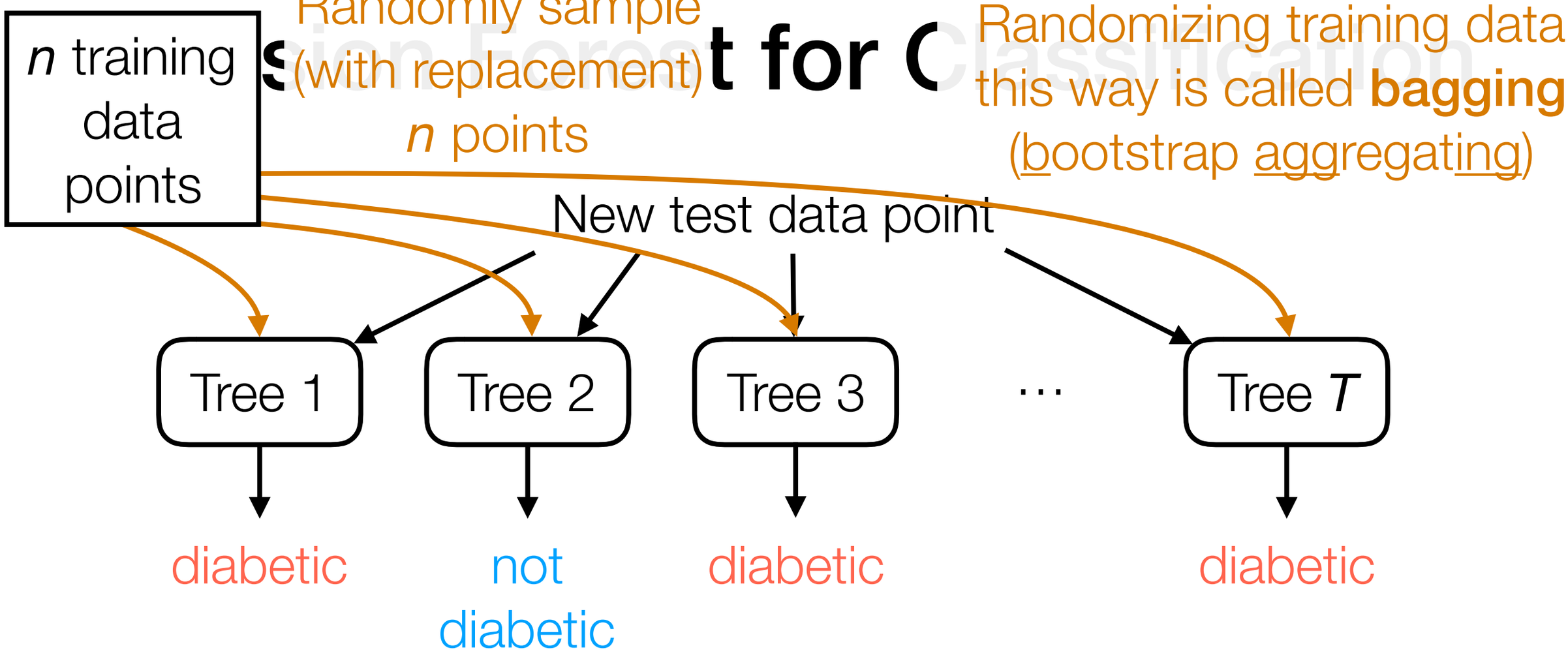
  ➔ by re-running the same learning procedure, we can get different decision trees that make different predictions!

- 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:** randomize training data used for each tree, randomly choose a few features to try to split on (and among these features, choose the best one to split on)

Back to the demo