Adaptive Nearest Neighbor Classification and Regression Based on Decision Trees

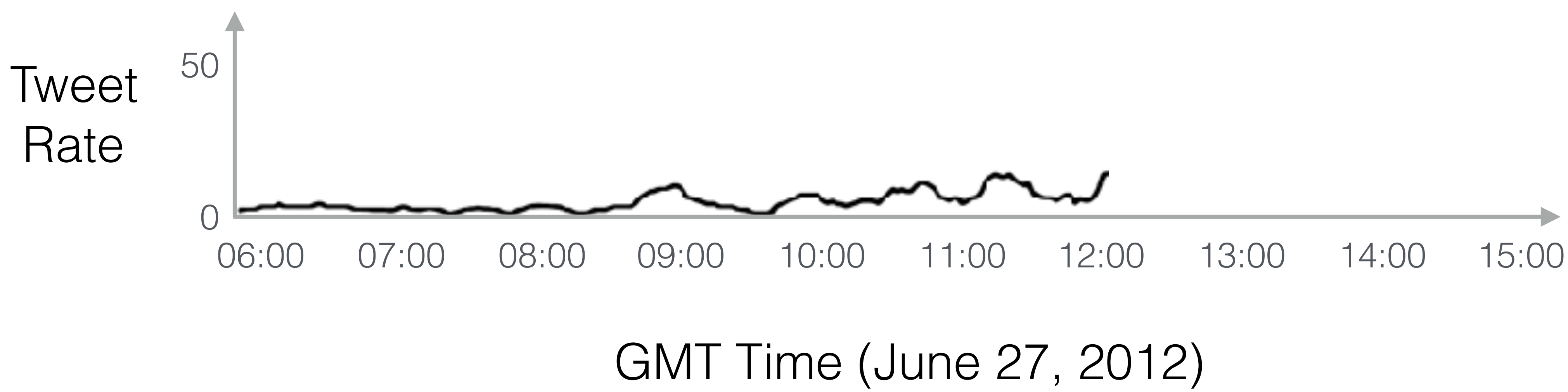
slides by
George Chen
Carnegie Mellon University
Fall 2017

NN and Kernel Classification and Regression

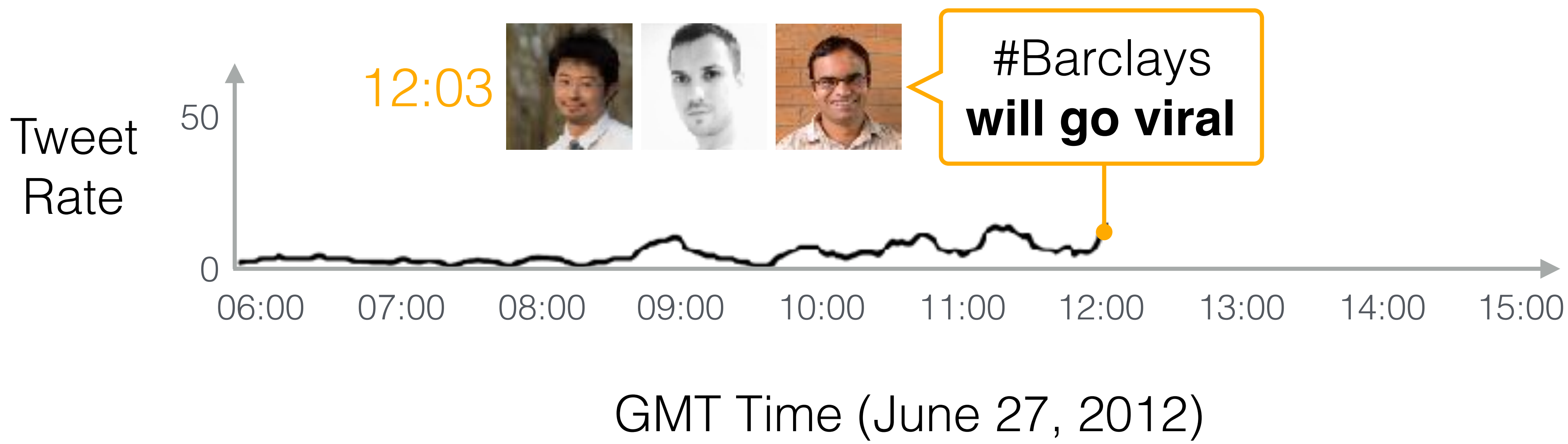

BARCLAYS

News Activity for #Barclays

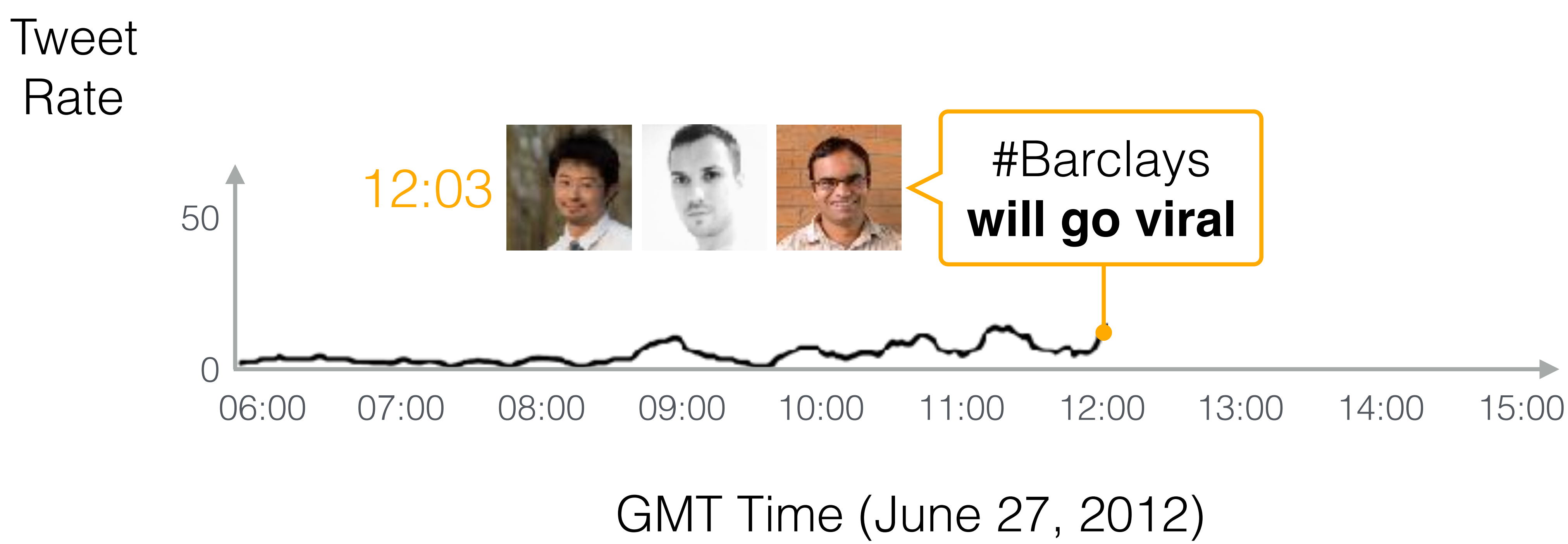
News Activity for #Barclays



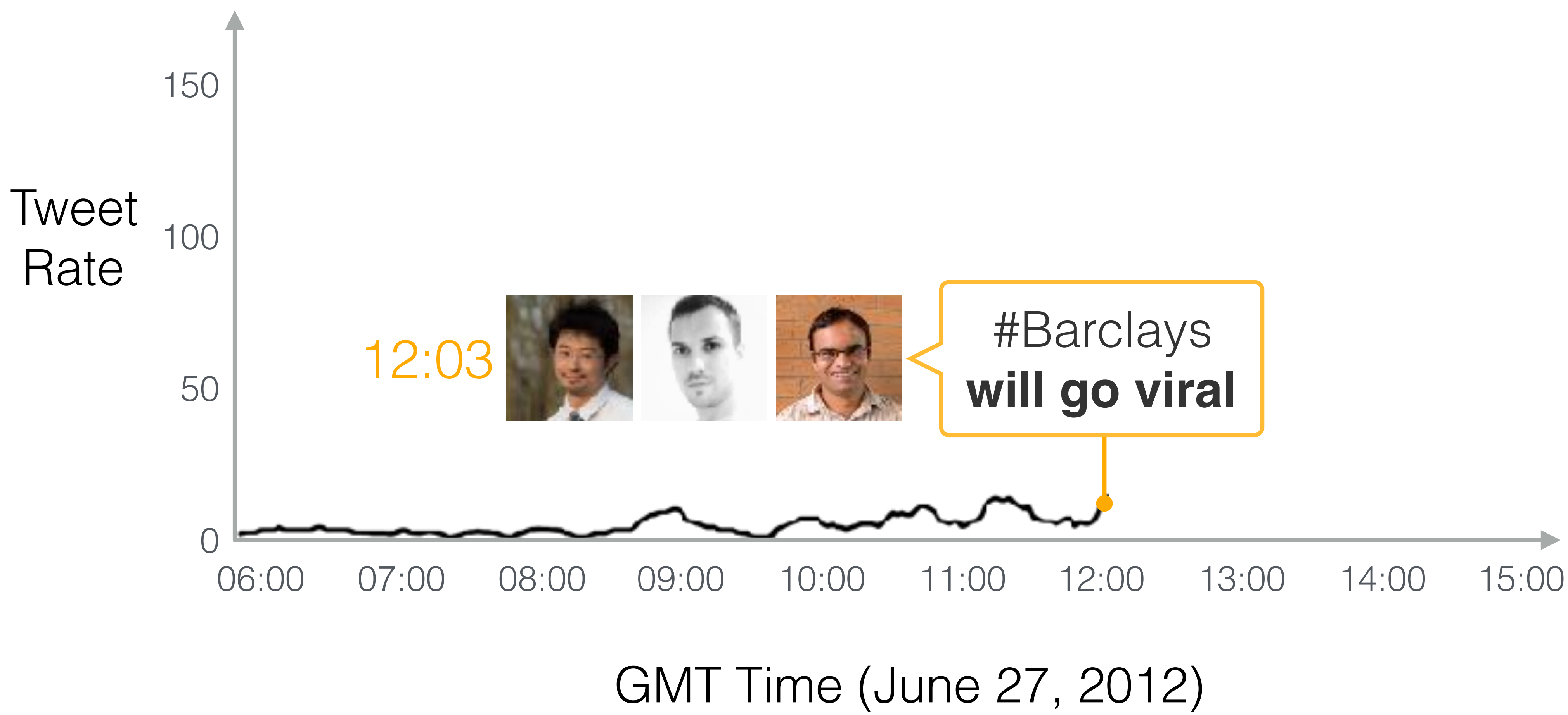
News Activity for #Barclays



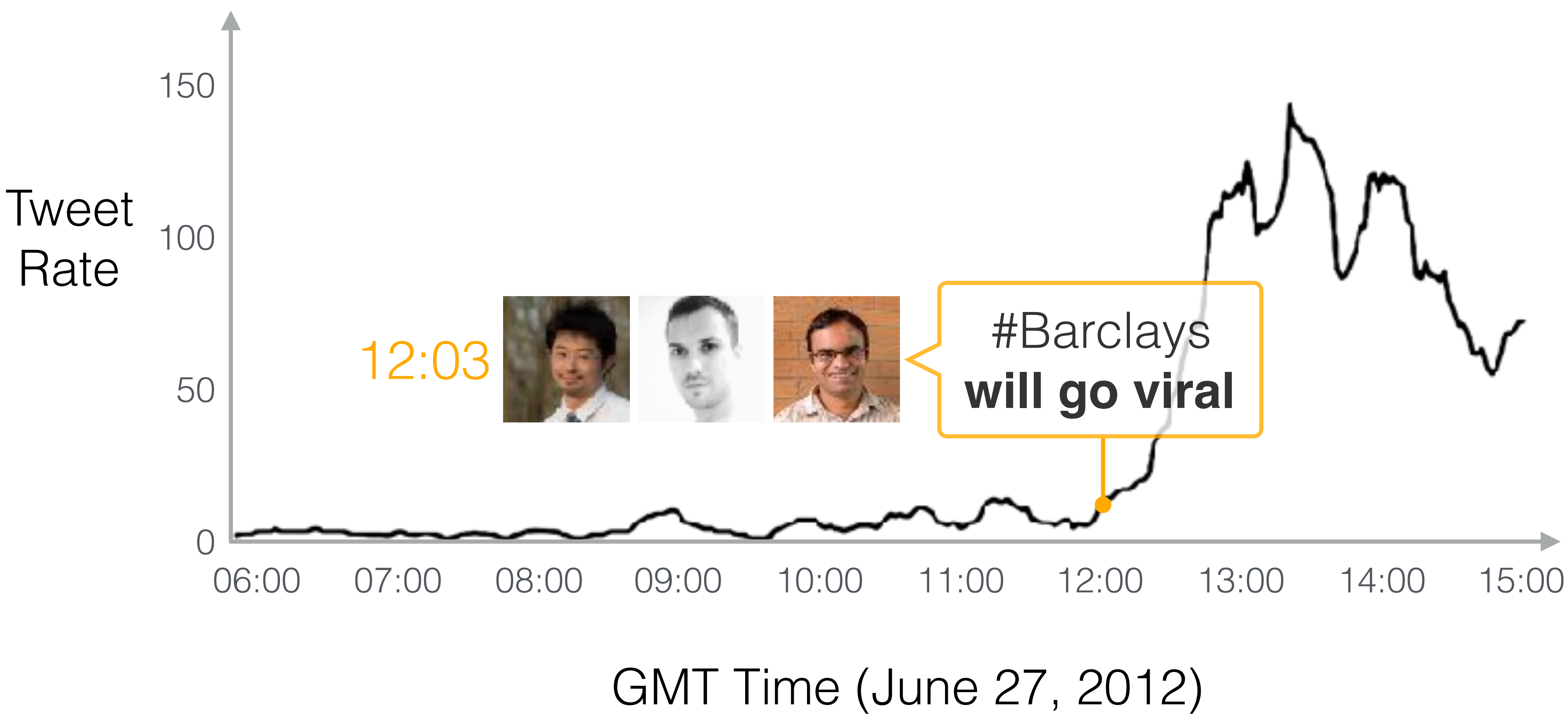
News Activity for #Barclays

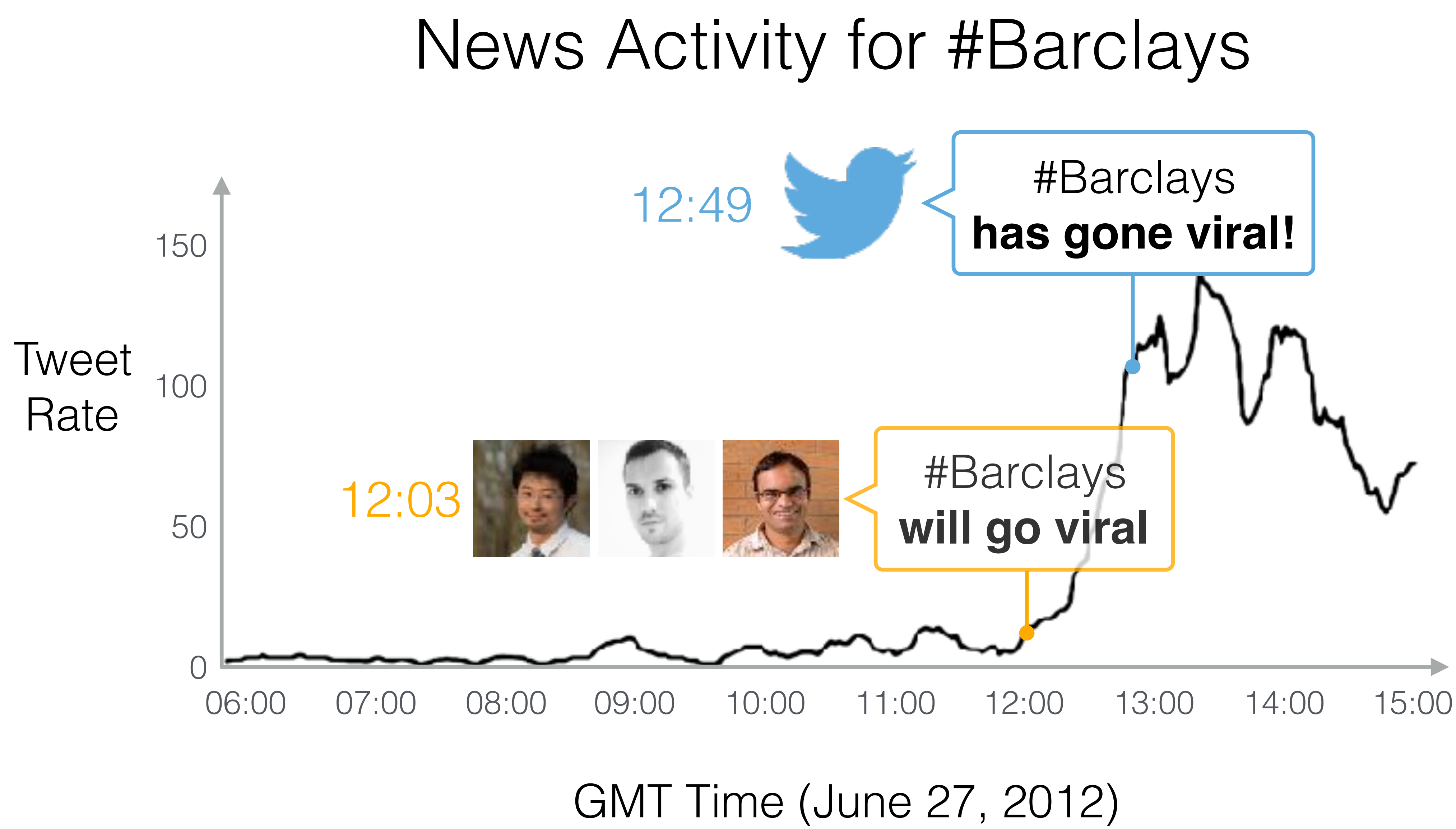


News Activity for #Barclays

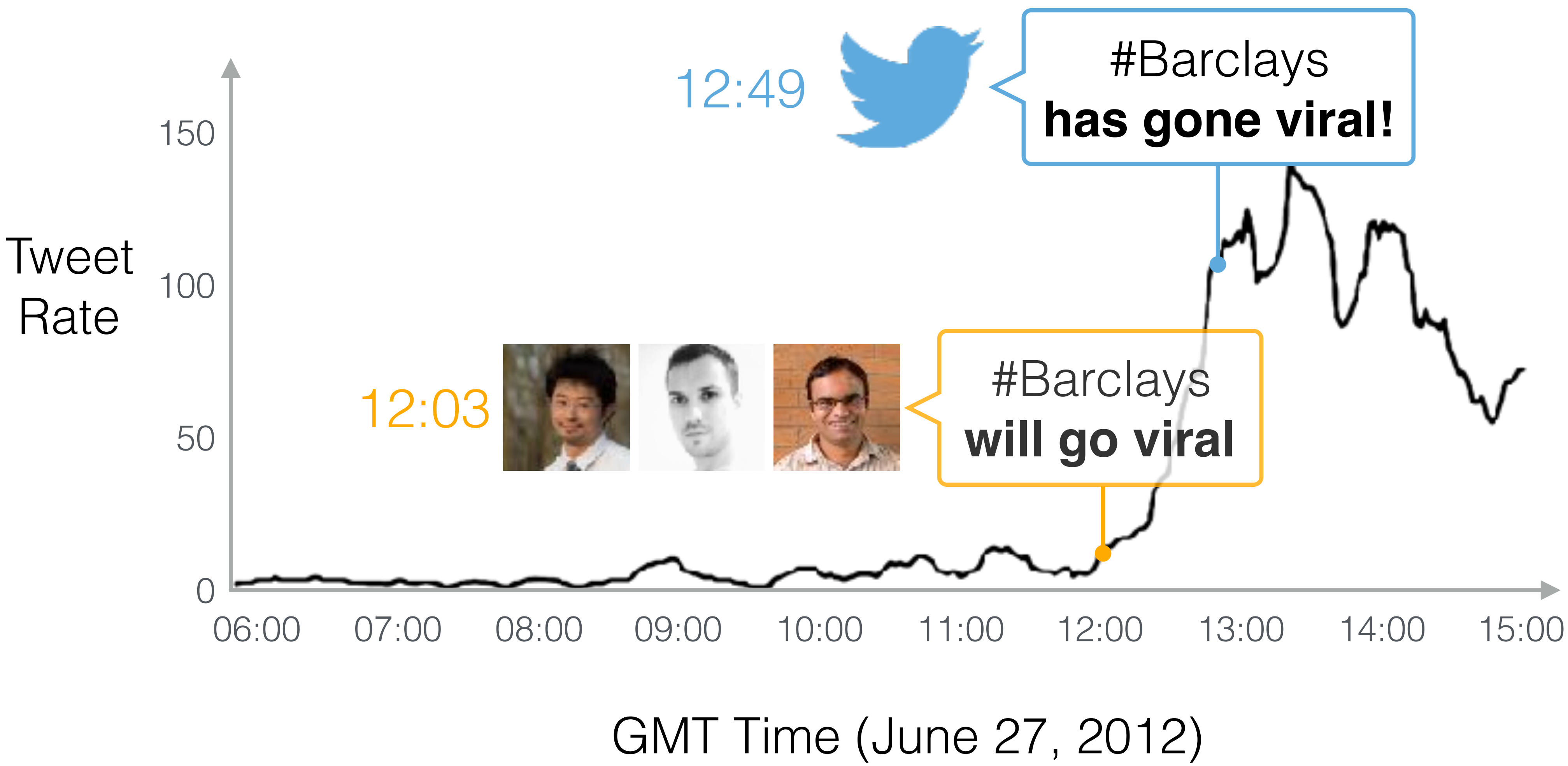


News Activity for #Barclays




News Activity for #Barclays
Tweet Rate
150
100
50
0
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
GMT Time (June 27, 2012)
12:49
#Barclays
has gone viral!
12:03
#Barclays
will go viral

News Activity for #Barclays



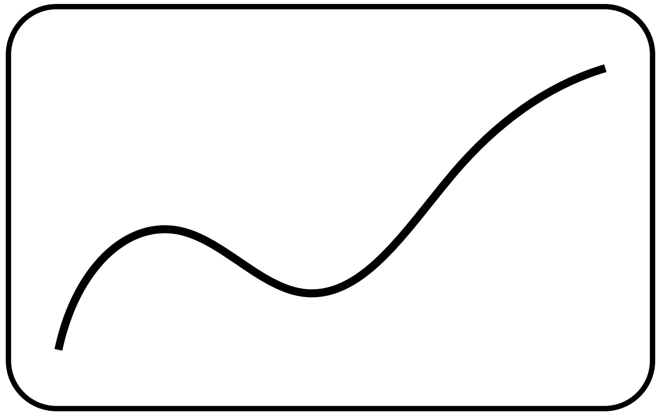
How we did this: **weighted majority voting**

Chen, Nikolov, and Shah. A Latent Source Model for Nonparametric Time Series Classification. NIPS 2013.

Weighted Majority Voting

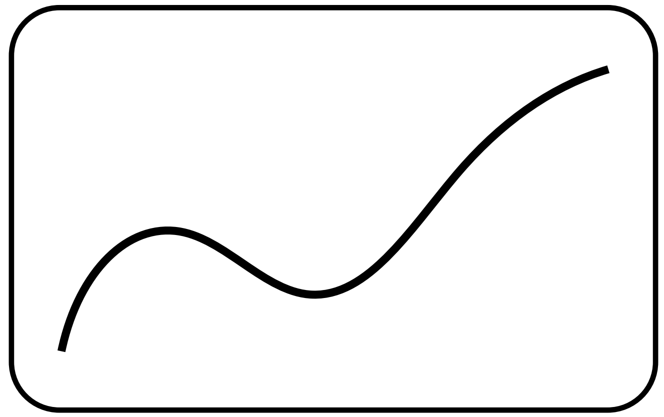
Weighted Majority Voting

Test data



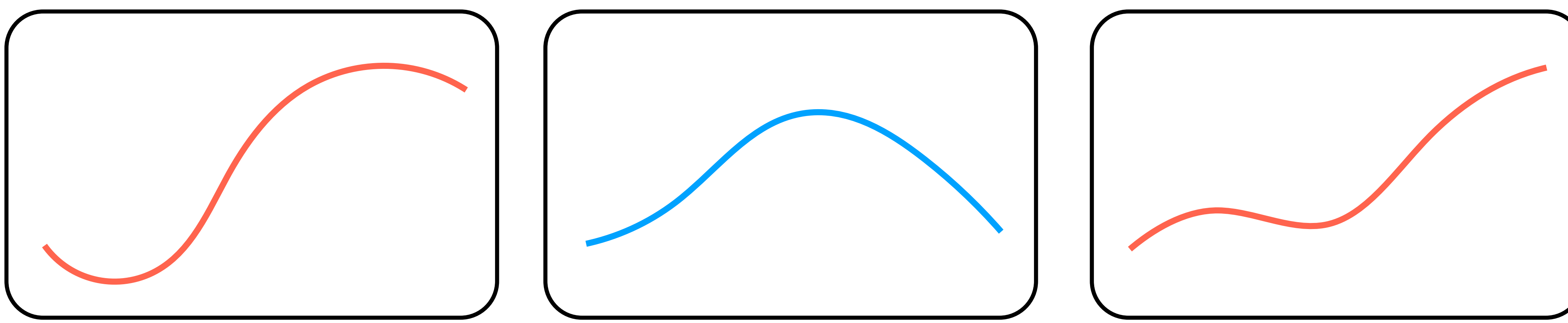
Weighted Majority Voting

Test data



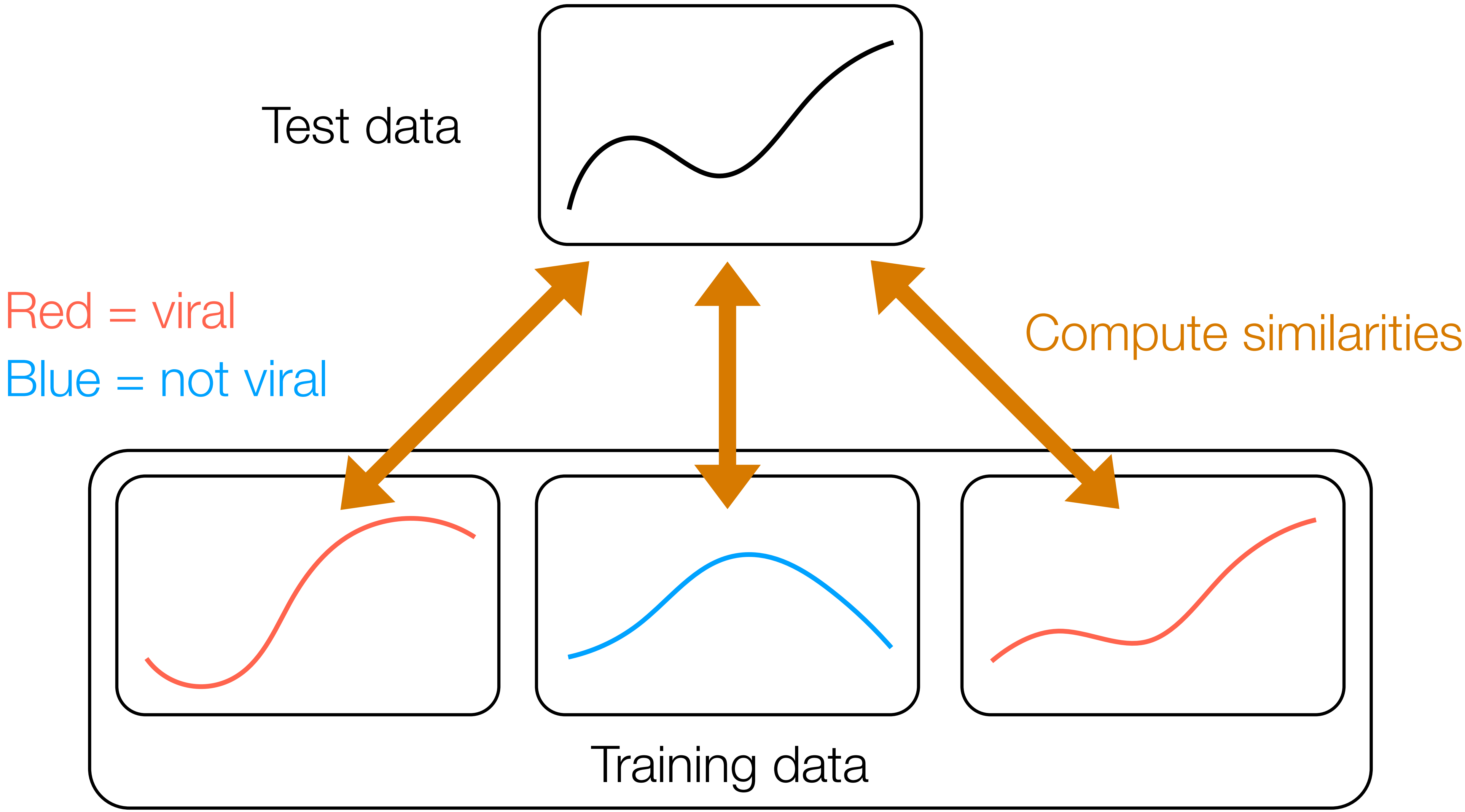
Red = viral

Blue = not viral

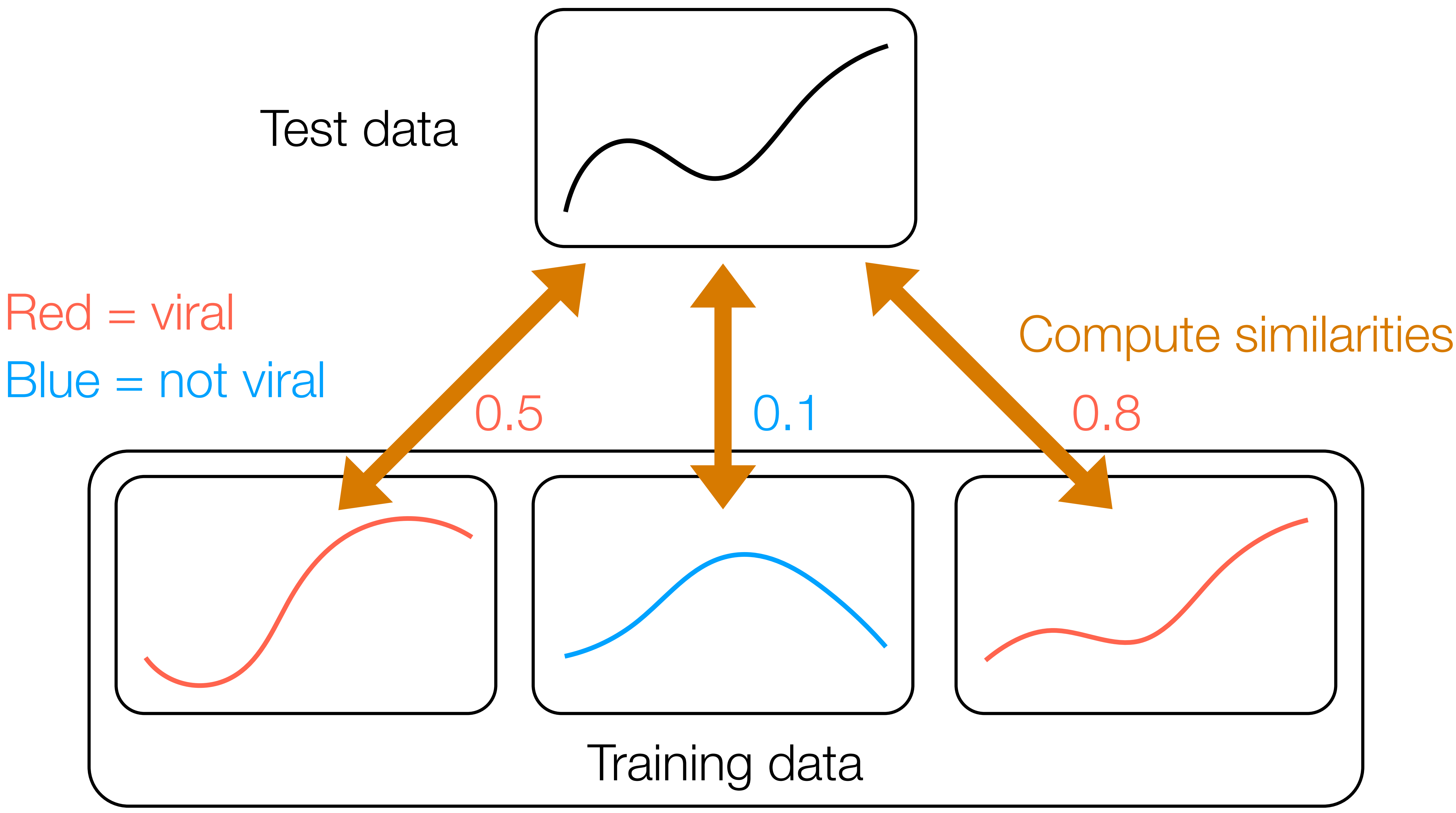


Training data

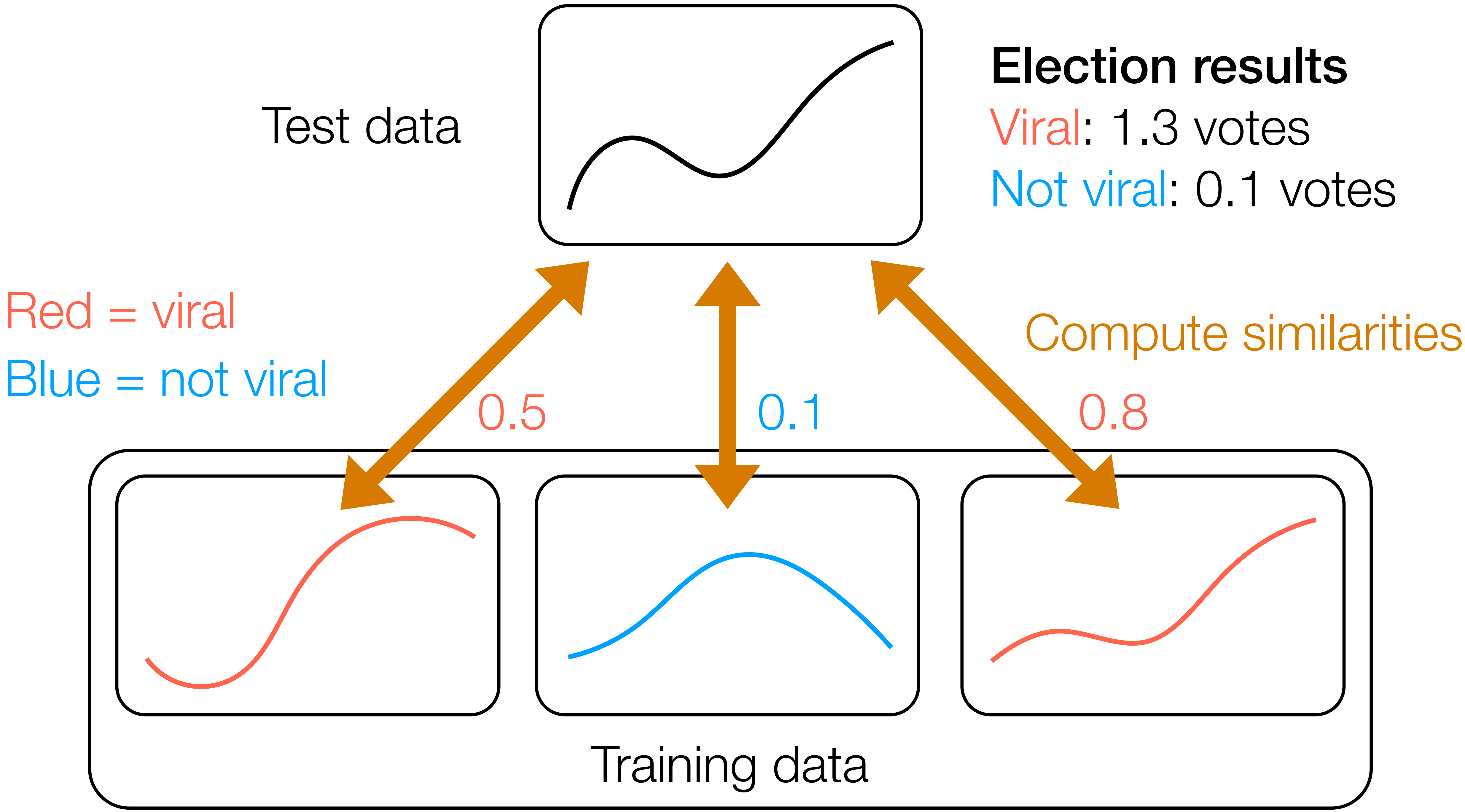
Weighted Majority Voting



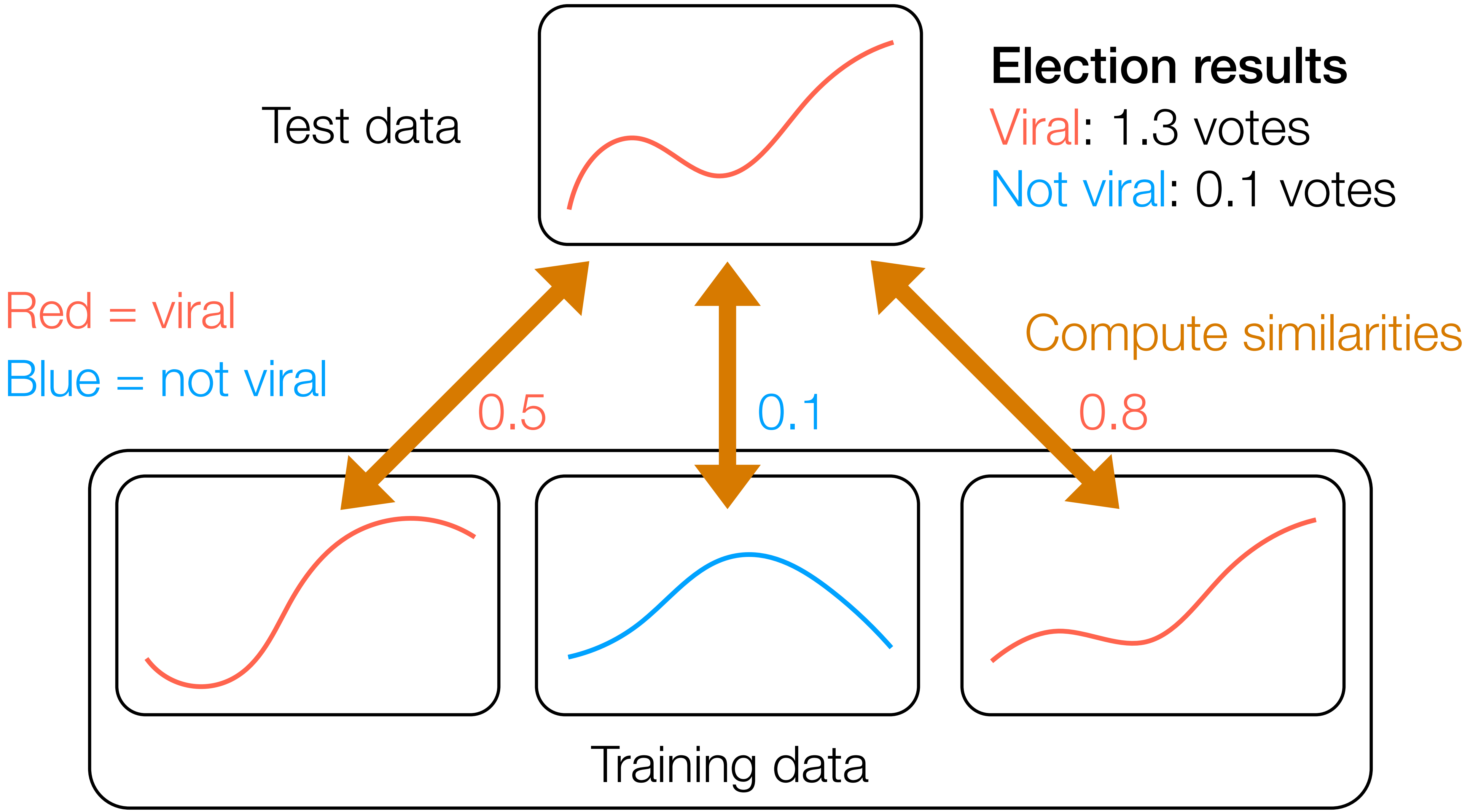
Weighted Majority Voting



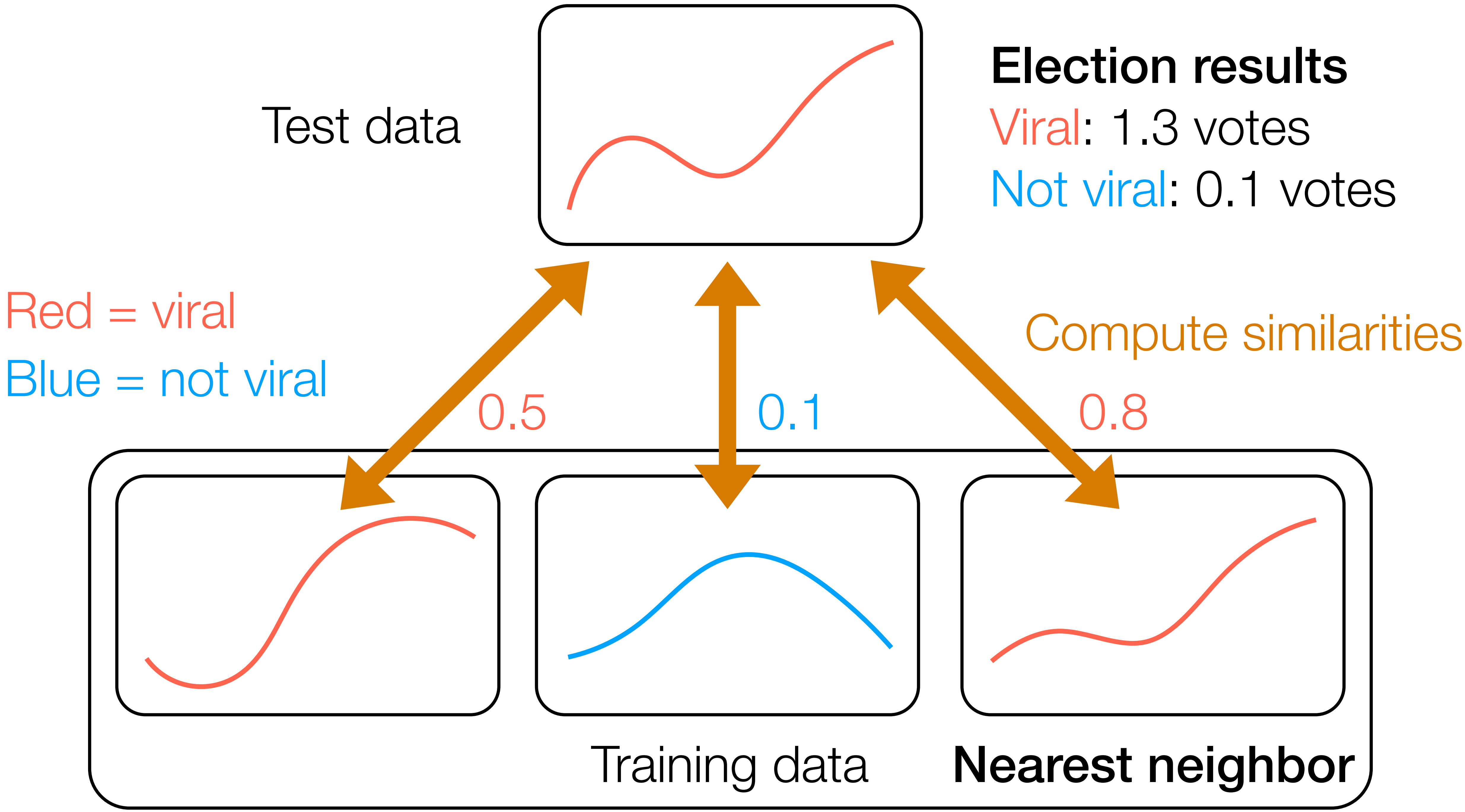
Weighted Majority Voting



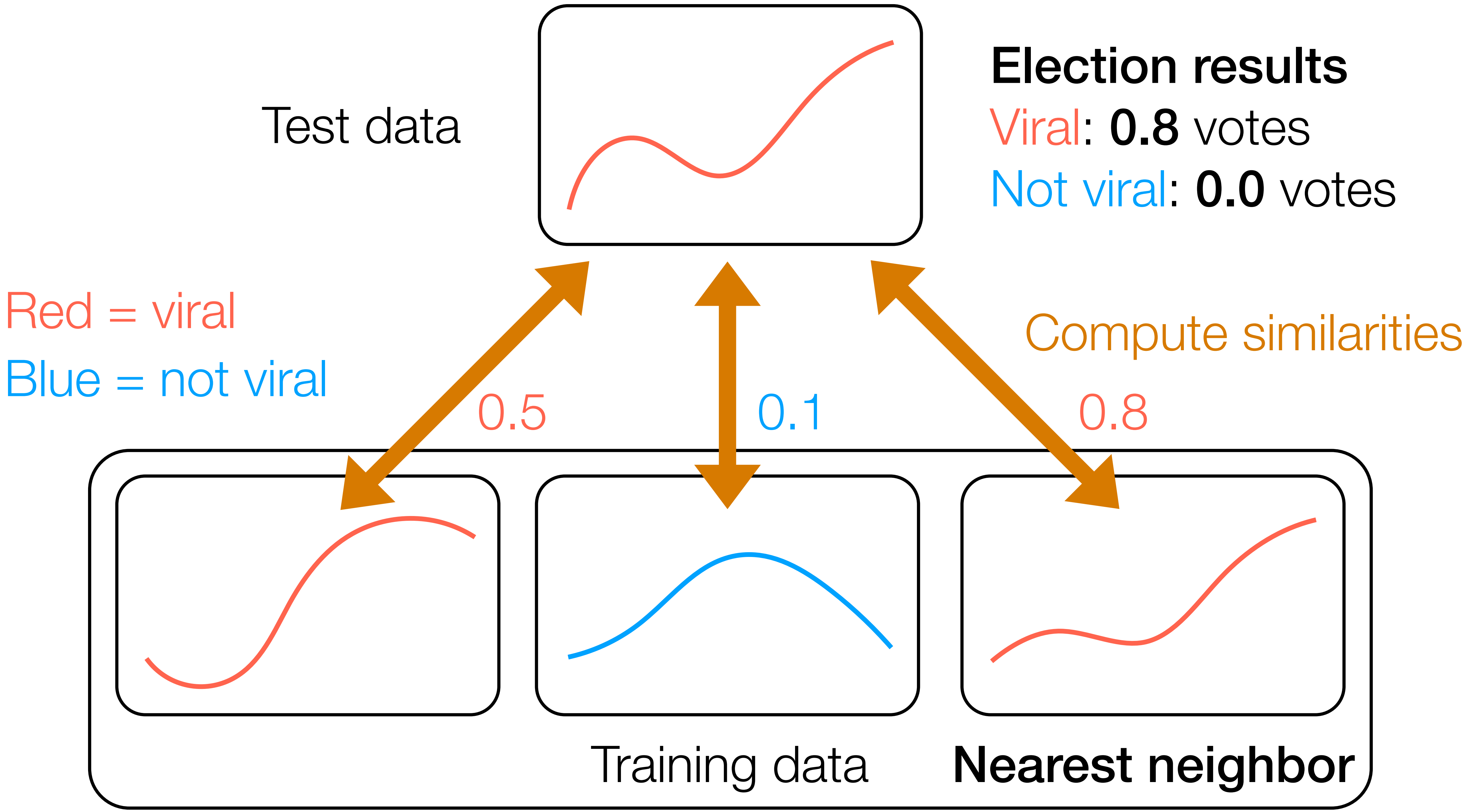
Weighted Majority Voting



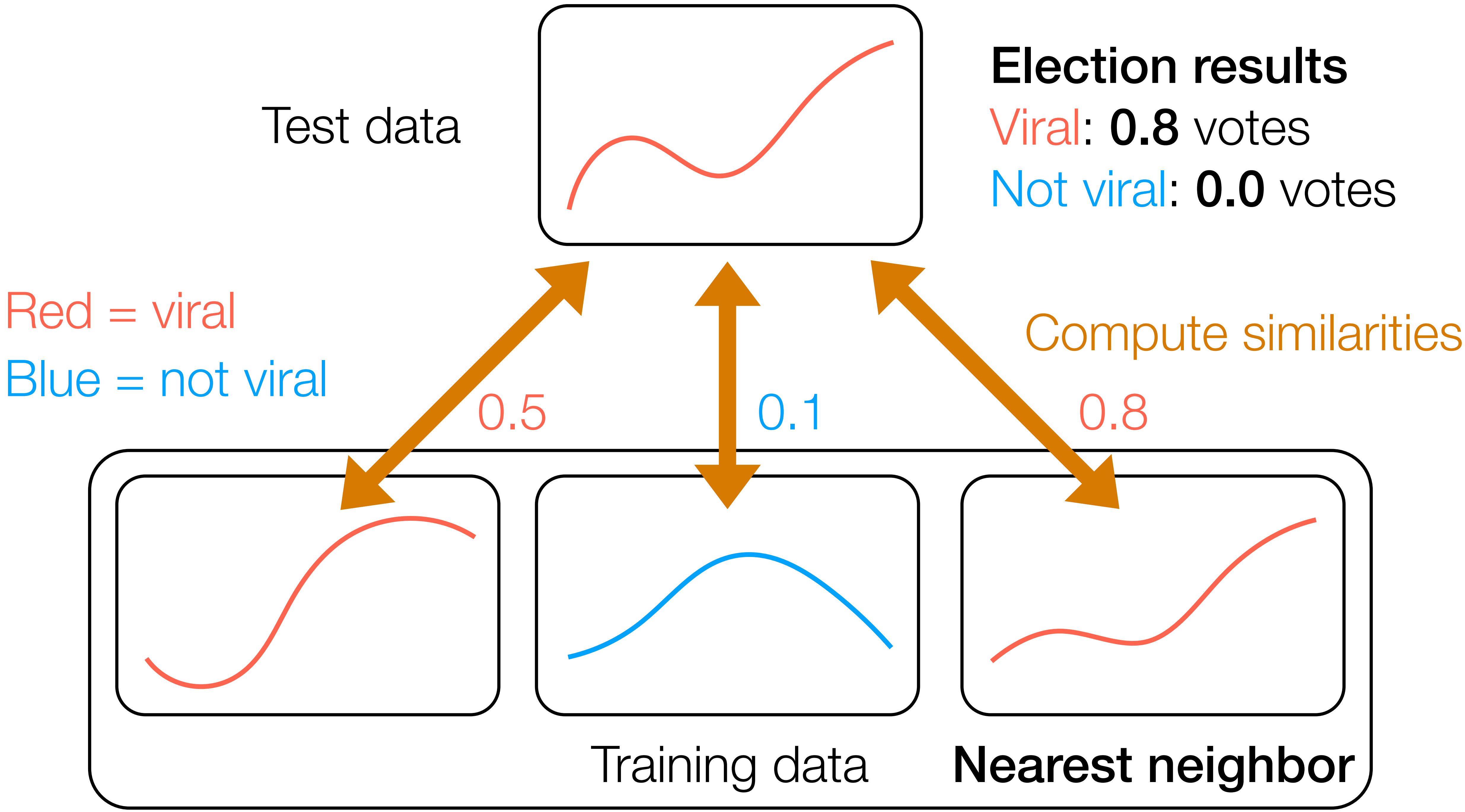
Weighted Majority Voting



Weighted Majority Voting



Nearest Neighbor Classification



NN Classification Variants

NN Classification Variants

- ***k*-NN classification:** consider *k* most similar training data to test data point

NN Classification Variants

- ***k*-NN classification:** consider *k* most similar training data to test data point
  - **Weighted:** when tallying up votes, use the similarities that we computed

NN Classification Variants

- ***k*-NN classification:** consider *k* most similar training data to test data point
  - **Weighted:** when tallying up votes, use the similarities that we computed
  - **Unweighted:** when tallying up votes, have each of the *k* nearest neighbors have an equal vote of 1
    (usually *k*-NN classification refers to unweighted case)

NN Classification Variants

- ***k*-NN classification:** consider *k* most similar training data to test data point
  - **Weighted:** when tallying up votes, use the similarities that we computed
  - **Unweighted:** when tallying up votes, have each of the *k* nearest neighbors have an equal vote of 1
    (usually *k*-NN classification refers to unweighted case)

- **Fixed-radius near neighbor classification:** consider all training data at least some similarity threshold close to test data point (i.e., use all training data distance ≤ *h* away)

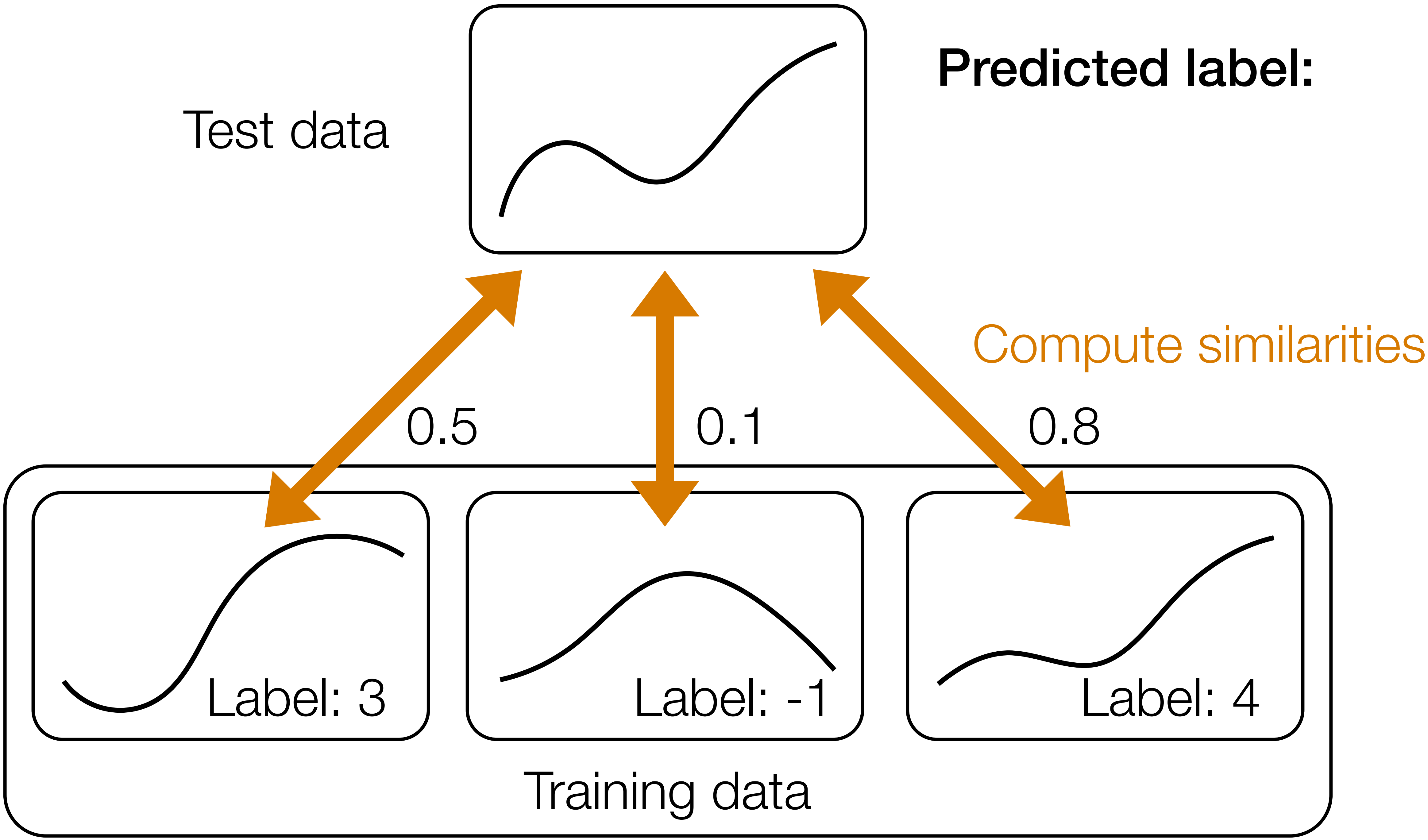
NN Classification Variants

- ***k*-NN classification:** consider *k* most similar training data to test data point
  - **Weighted:** when tallying up votes, use the similarities that we computed
  - **Unweighted:** when tallying up votes, have each of the *k* nearest neighbors have an equal vote of 1
    (usually *k*-NN classification refers to unweighted case)

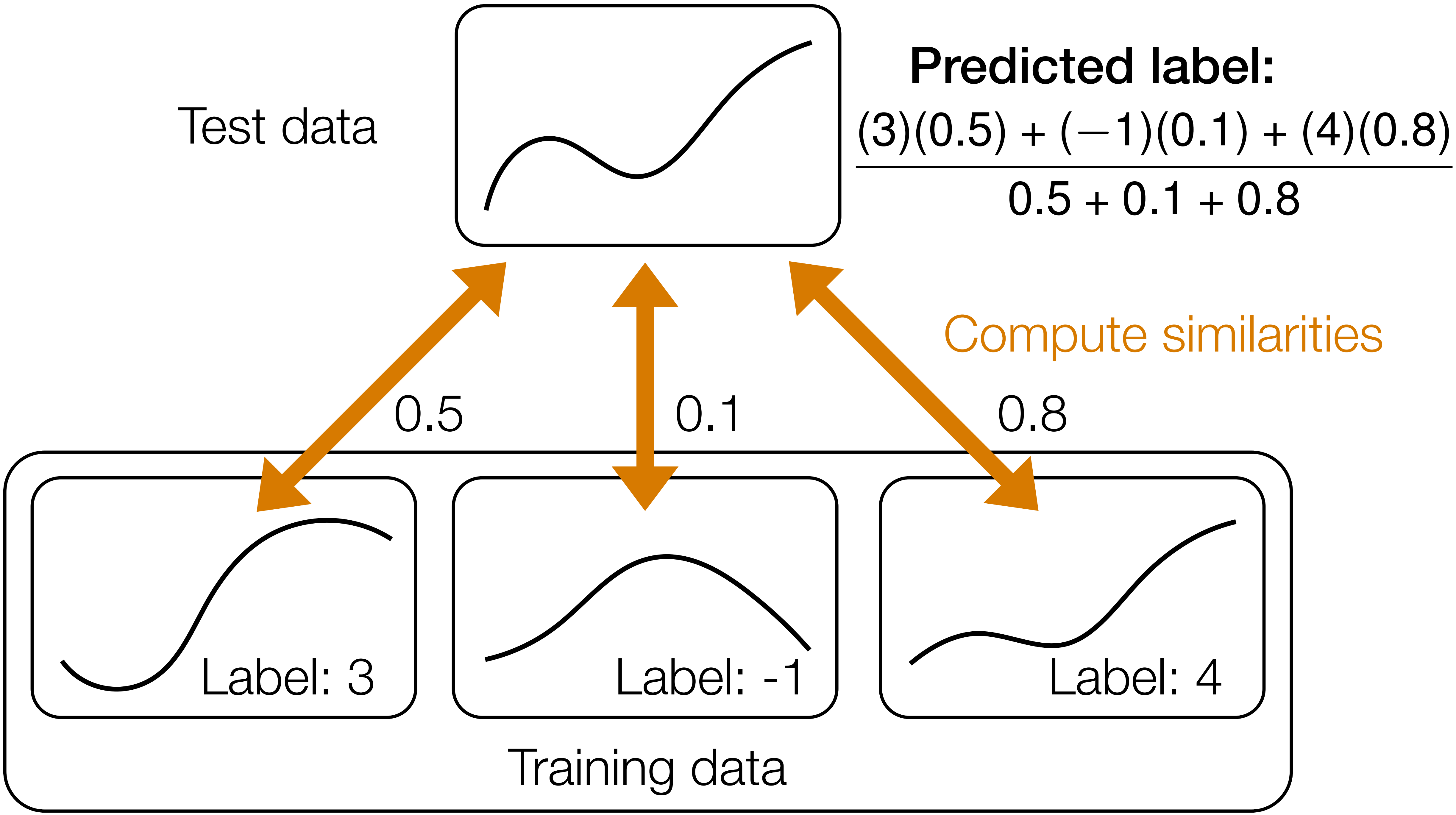
- **Fixed-radius near neighbor classification:** consider all training data at least some similarity threshold close to test data point (i.e., use all training data distance ≤ *h* away)
  - Once again, can use weighted or unweighted votes

Regression: Each label is *continuous* instead of *discrete*

Kernel Regression

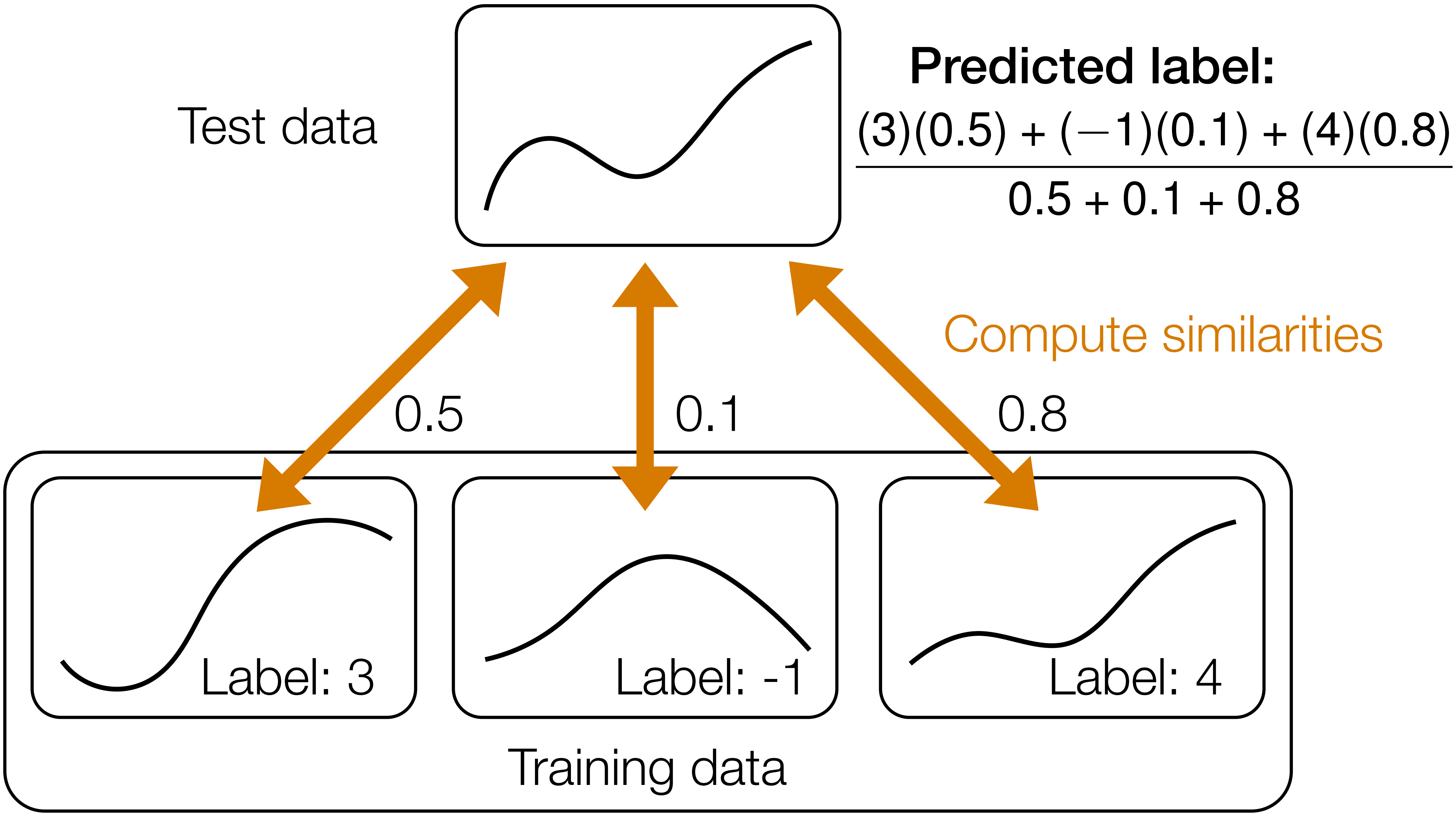


Kernel Regression



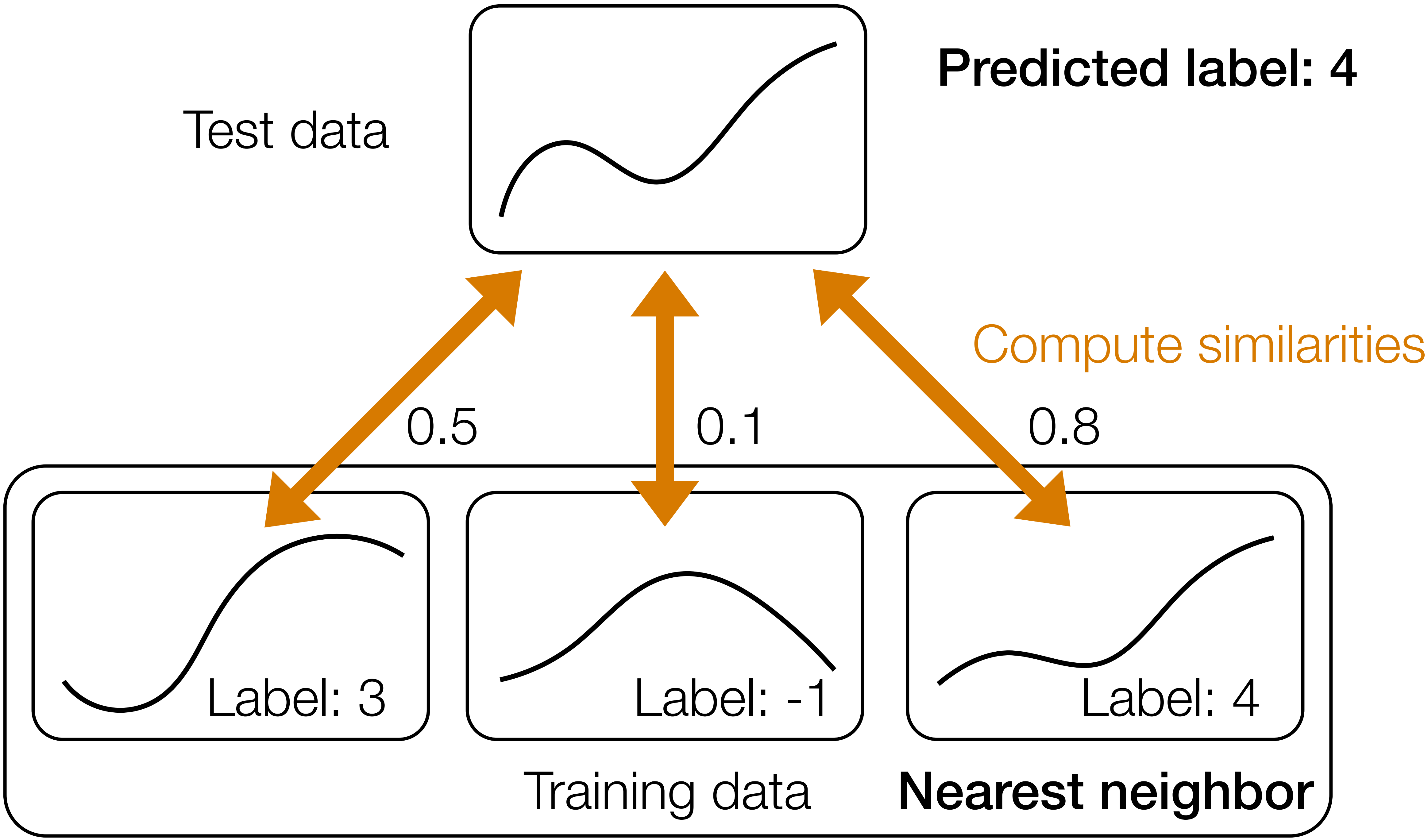
Kernel Regression

Weighted average instead of weighted majority vote

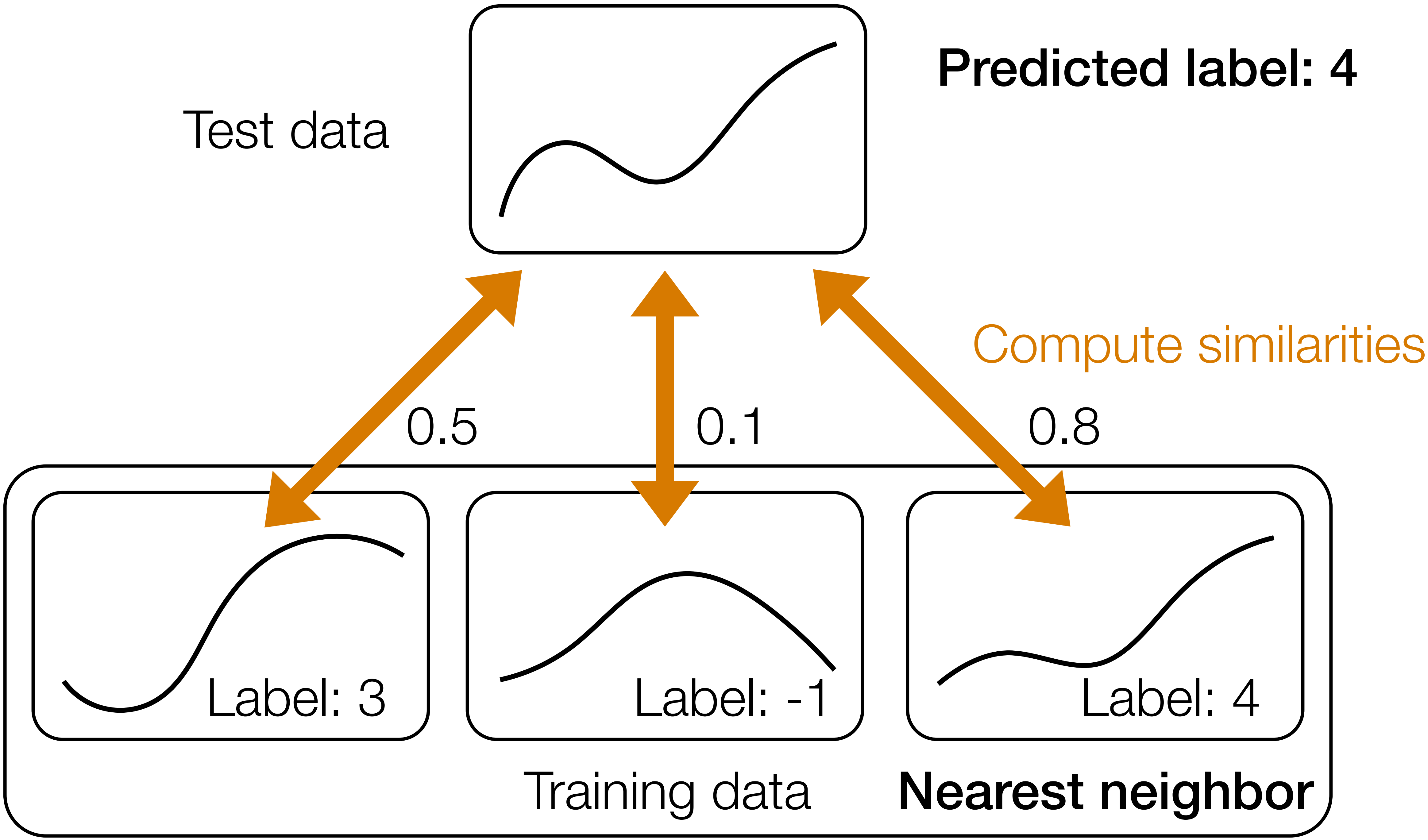


$$\frac{(3)(0.5) + (-1)(0.1) + (4)(0.8)}{0.5 + 0.1 + 0.8}$$

NN Regression



NN Regression

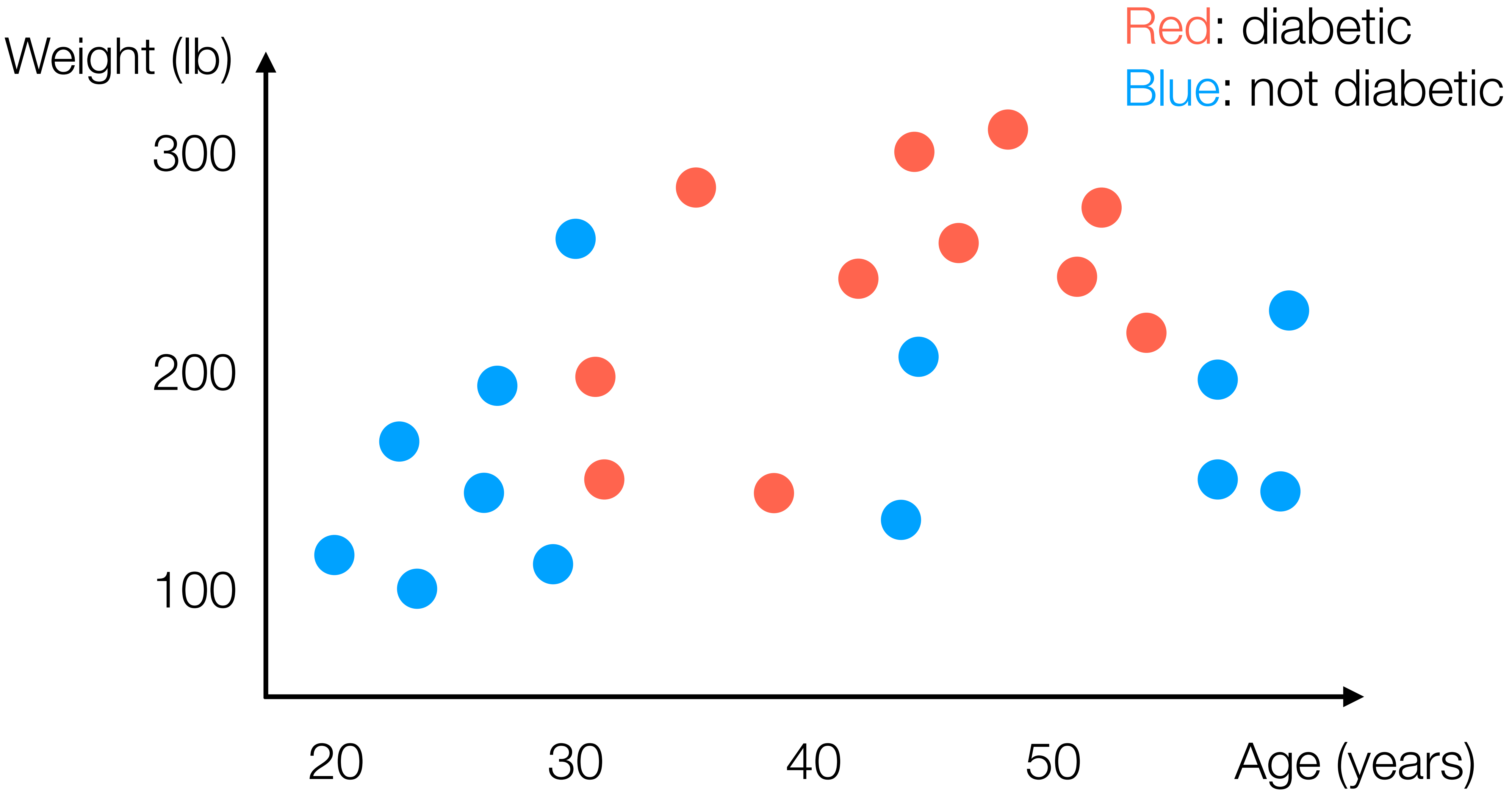


Just like classification: *k*-NN and fixed-radius NN variants, also weighted and unweighted

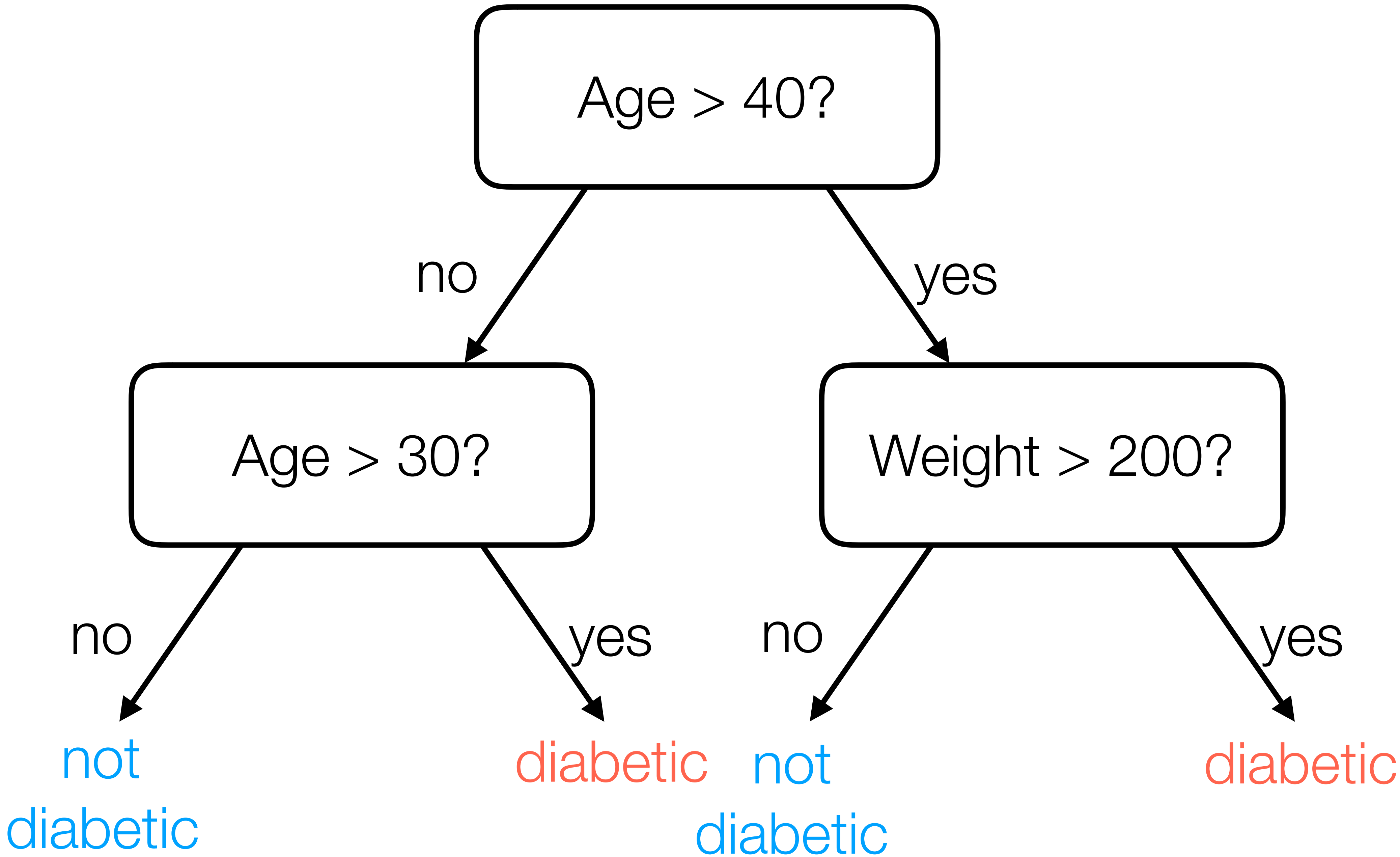
“Adaptive” nearest neighbors: learn the similarity function

Decision Trees

Example Made-Up Data



Example Decision Tree



Learning a Decision Tree

Learning a Decision Tree

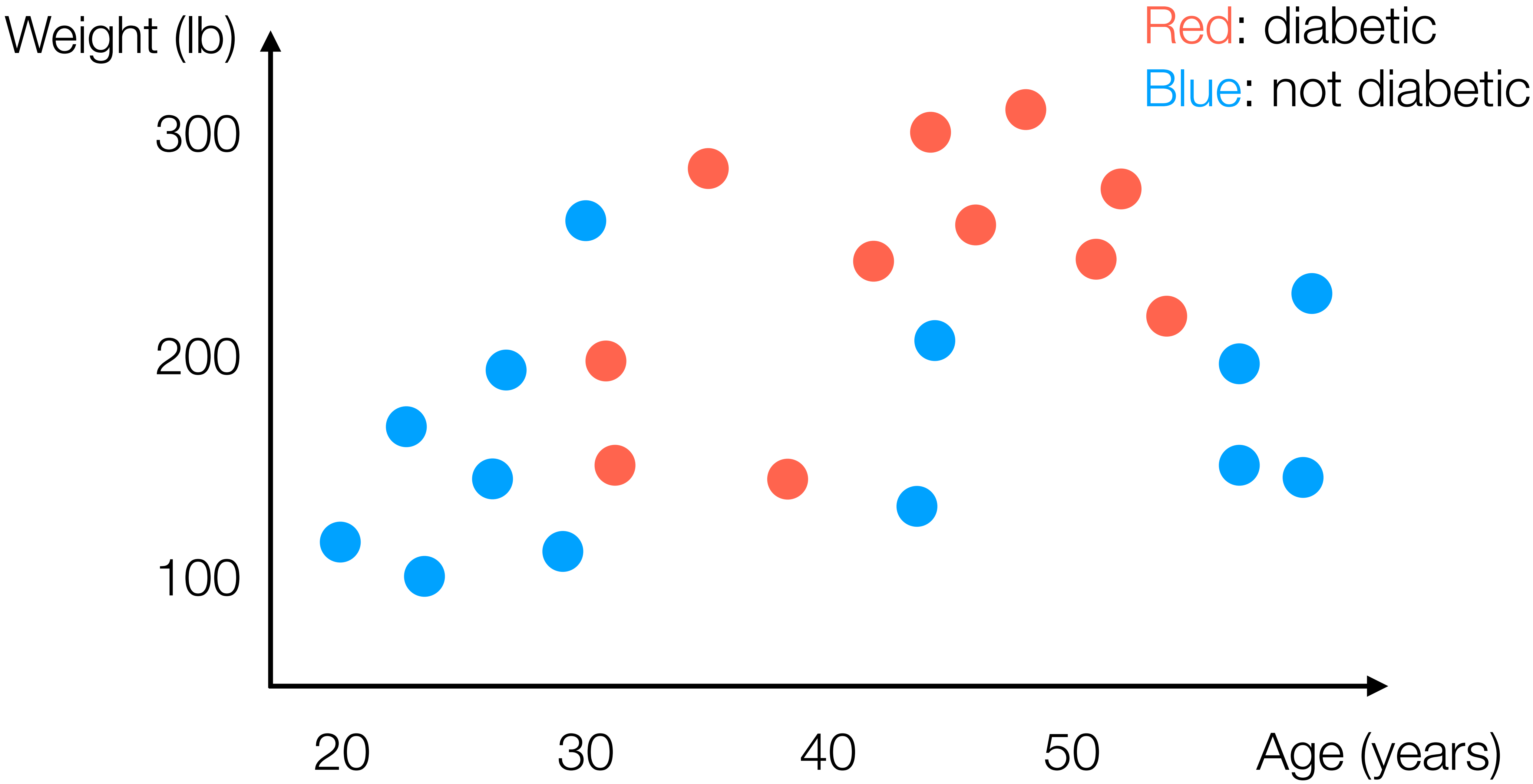
- Many ways: general approach actually looks a lot like divisive clustering *but accounts for label information*

Learning a Decision Tree

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

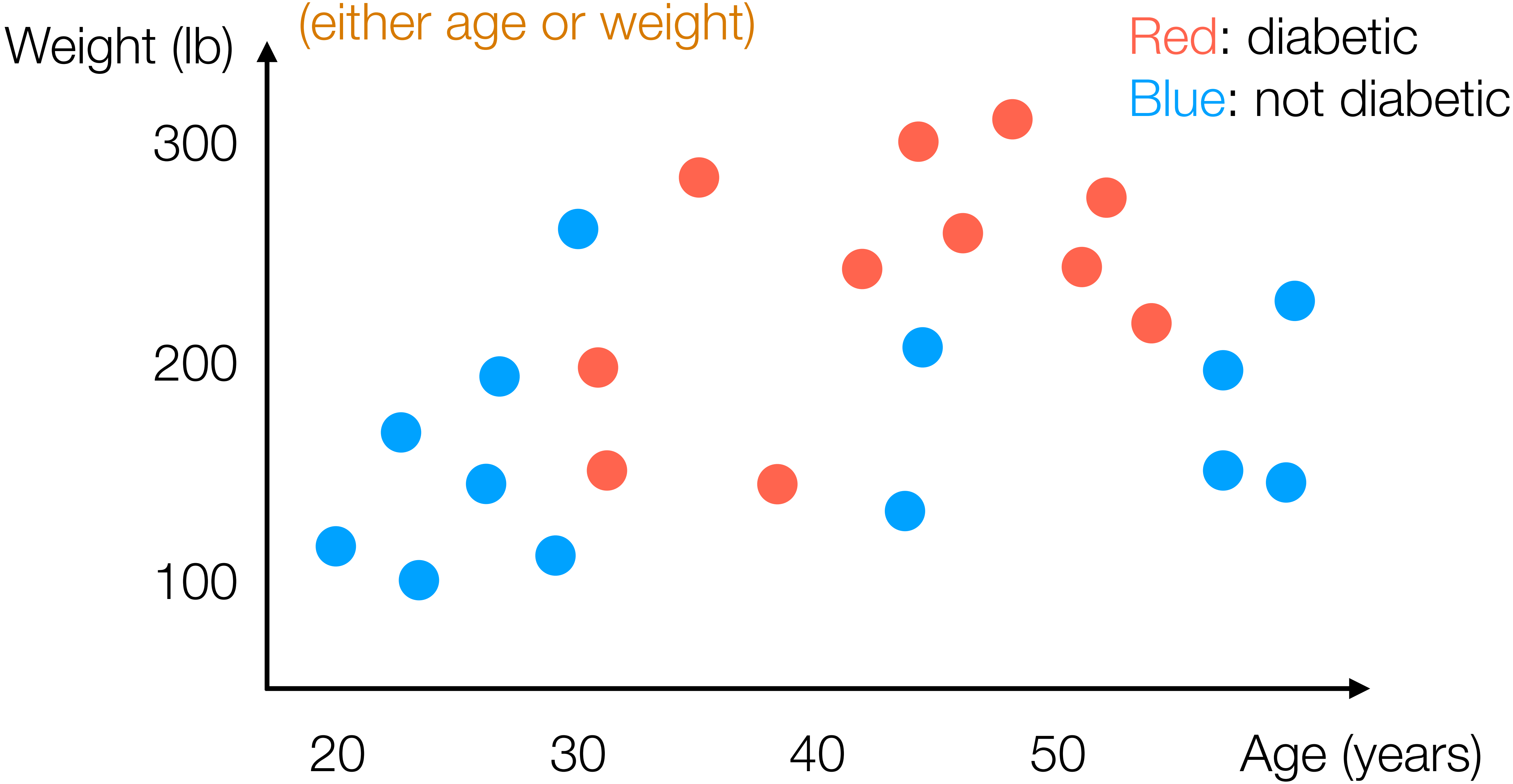
- I'll show one way (that nobody actually uses in practice) but it's easy to explain

Learning a Decision Tree



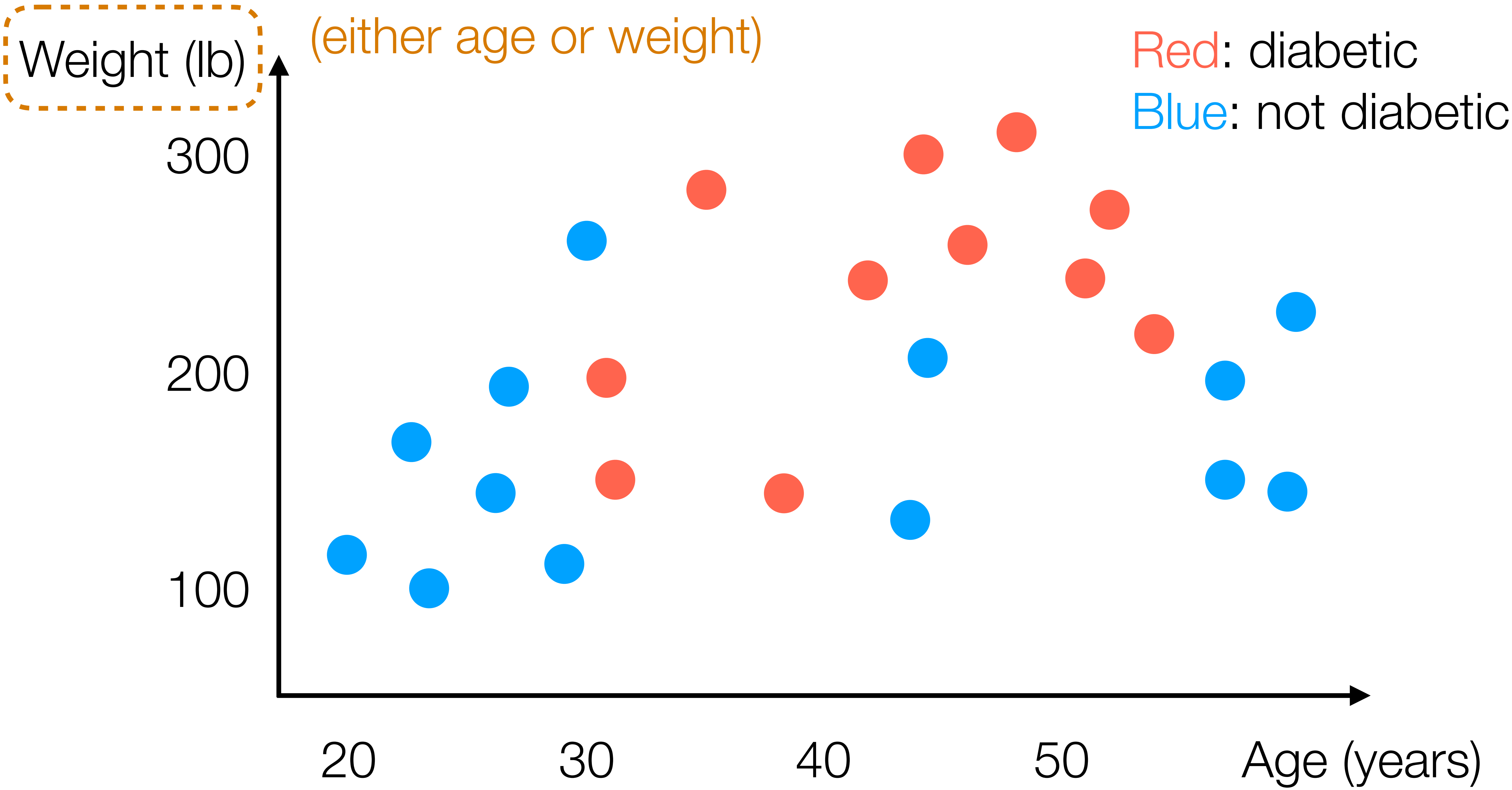
Learning a Decision Tree

1. Pick a random feature (either age or weight)



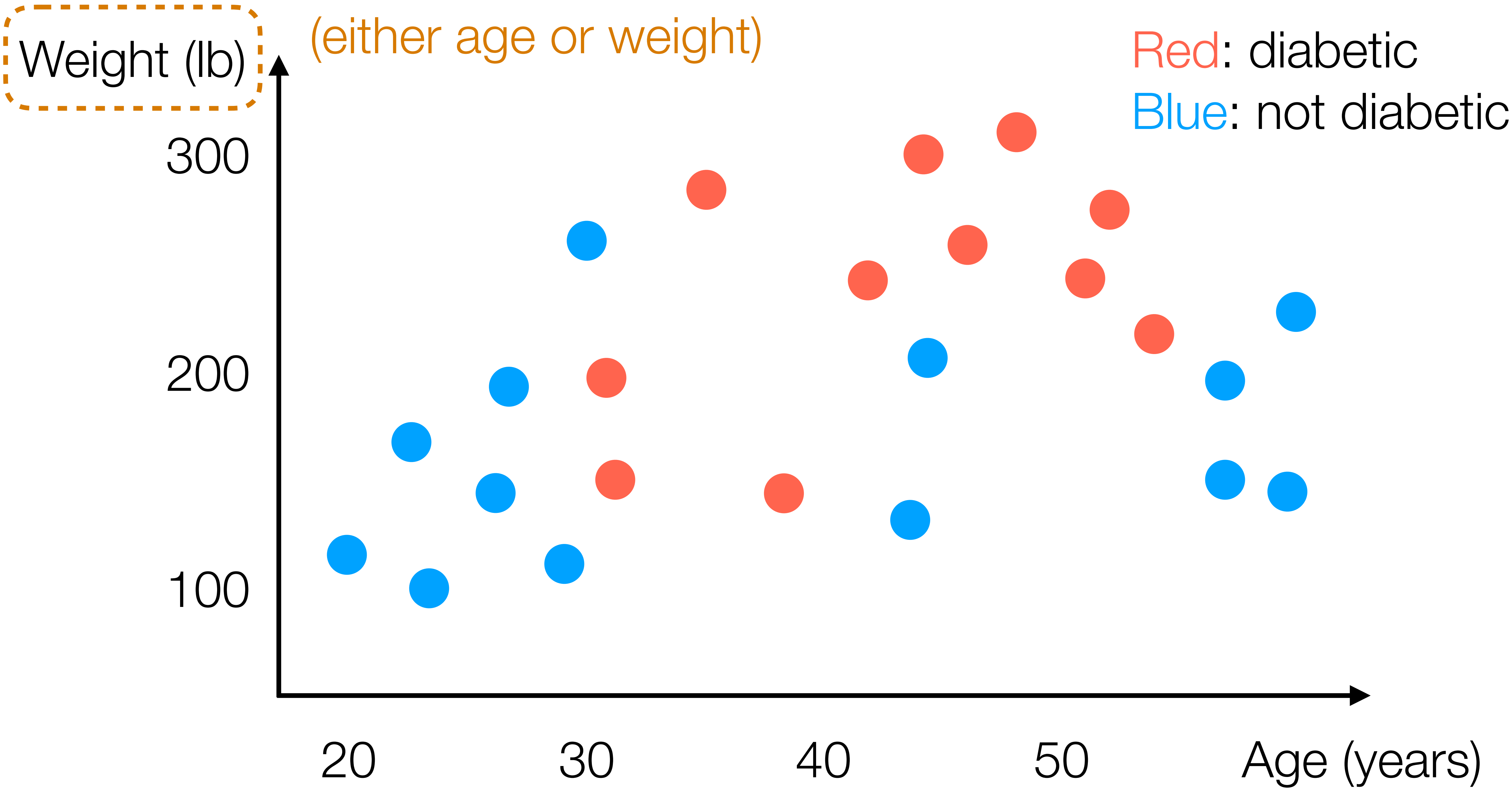
Learning a Decision Tree

1. Pick a random feature
(either age or weight)



Learning a Decision Tree

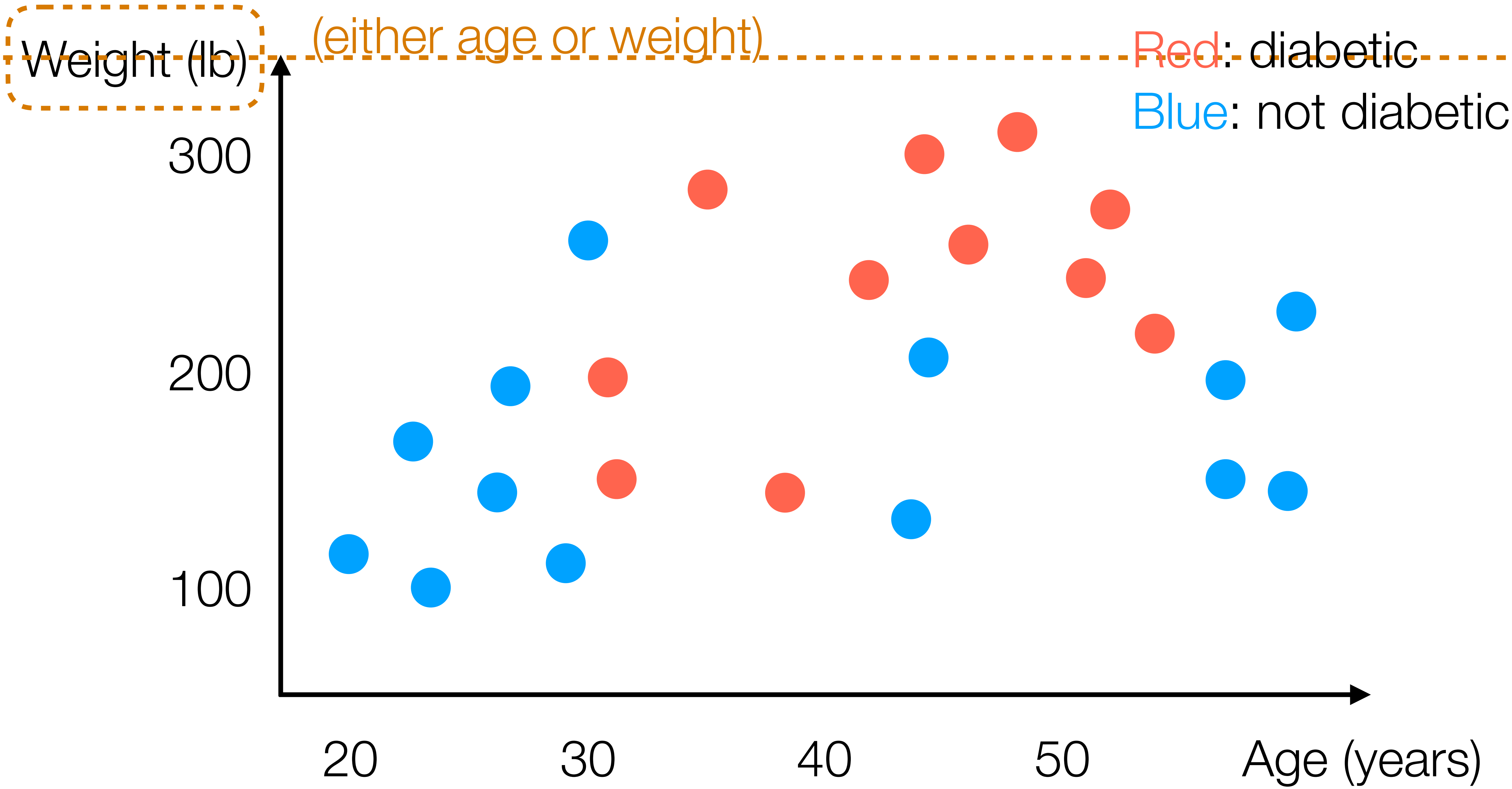
1. Pick a random feature (either age or weight)



2. Find threshold for which red and blue are as “separate as possible” (on one side, mostly red; on other side, mostly blue)

Learning a Decision Tree

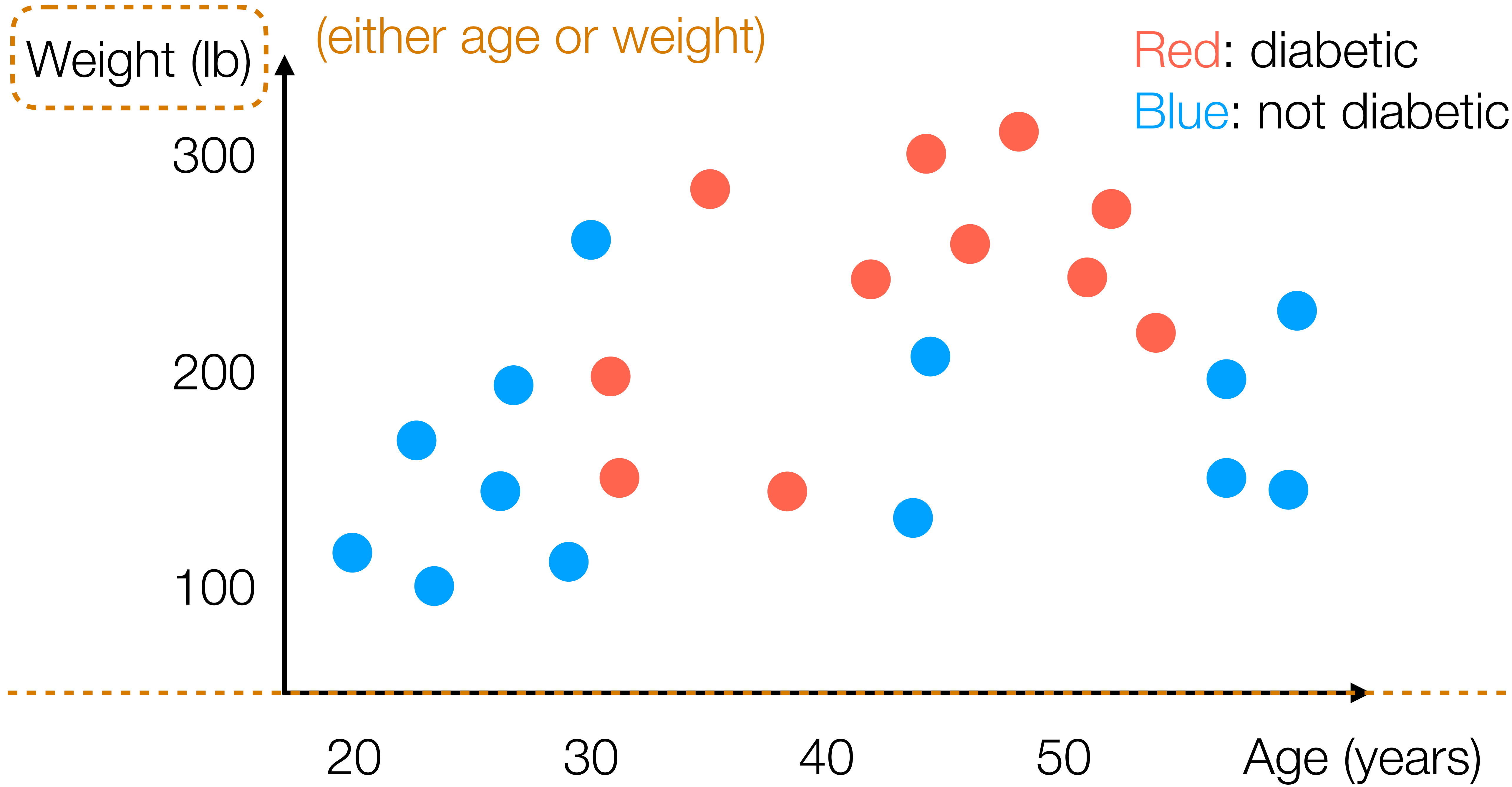
1. Pick a random feature (either age or weight)



2. Find threshold for which red and blue are as “separate as possible” (on one side, mostly red; on other side, mostly blue)

Learning a Decision Tree

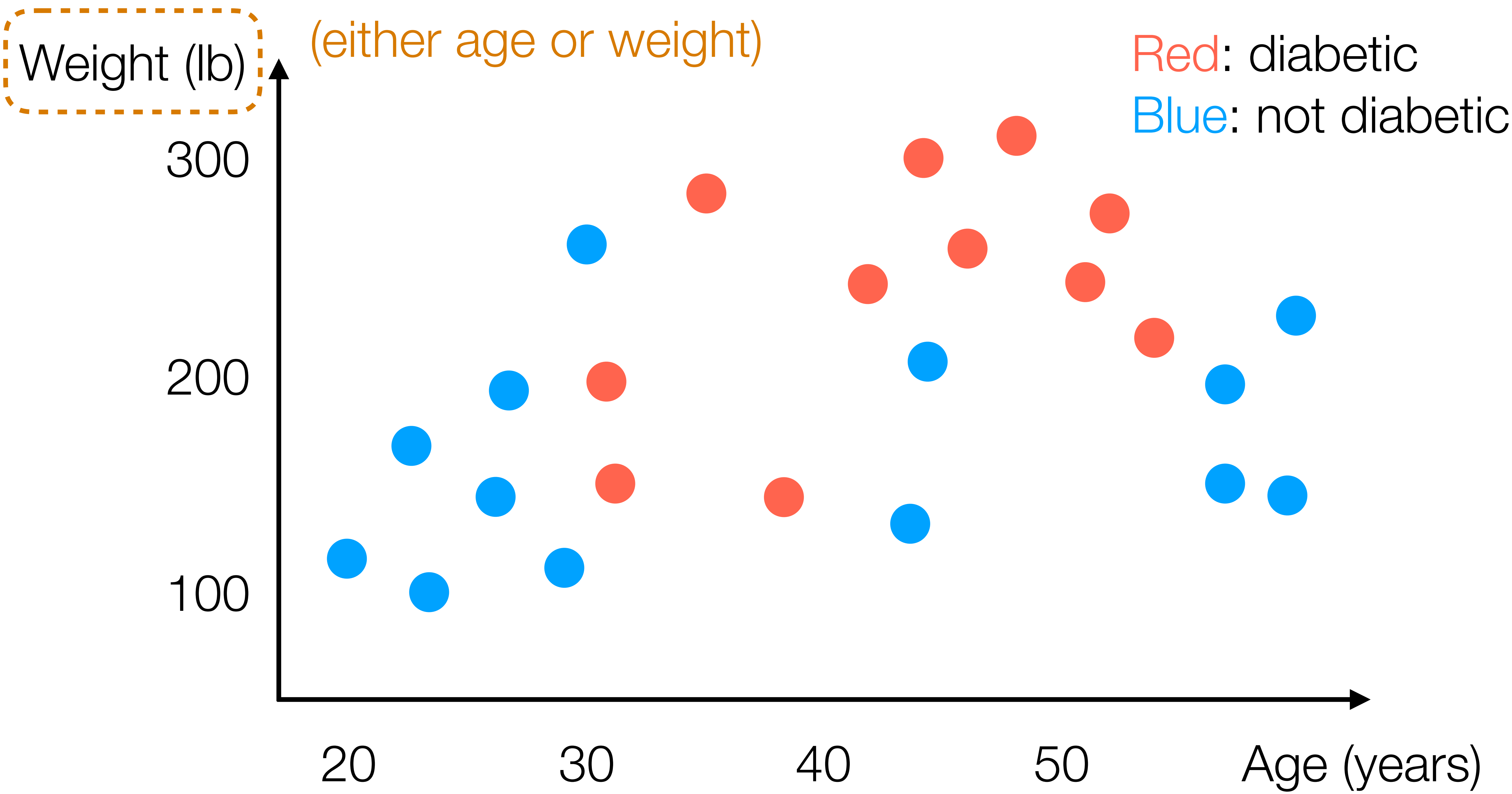
1. Pick a random feature (either age or weight)



2. Find threshold for which red and blue are as “separate as possible” (on one side, mostly red; on other side, mostly blue)

Learning a Decision Tree

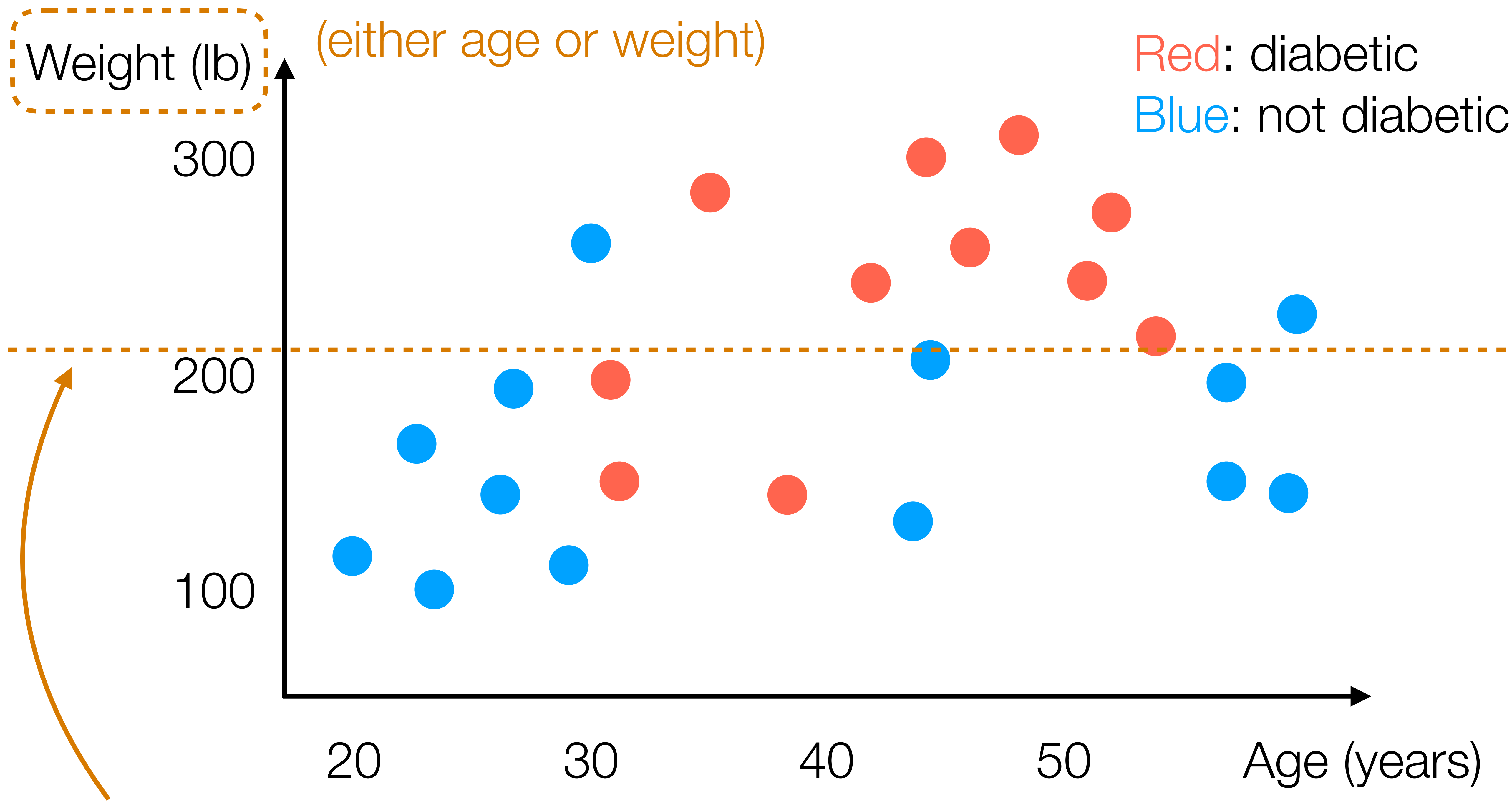
1. Pick a random feature (either age or weight)



2. Find threshold for which red and blue are as "separate as possible" (on one side, mostly red; on other side, mostly blue)

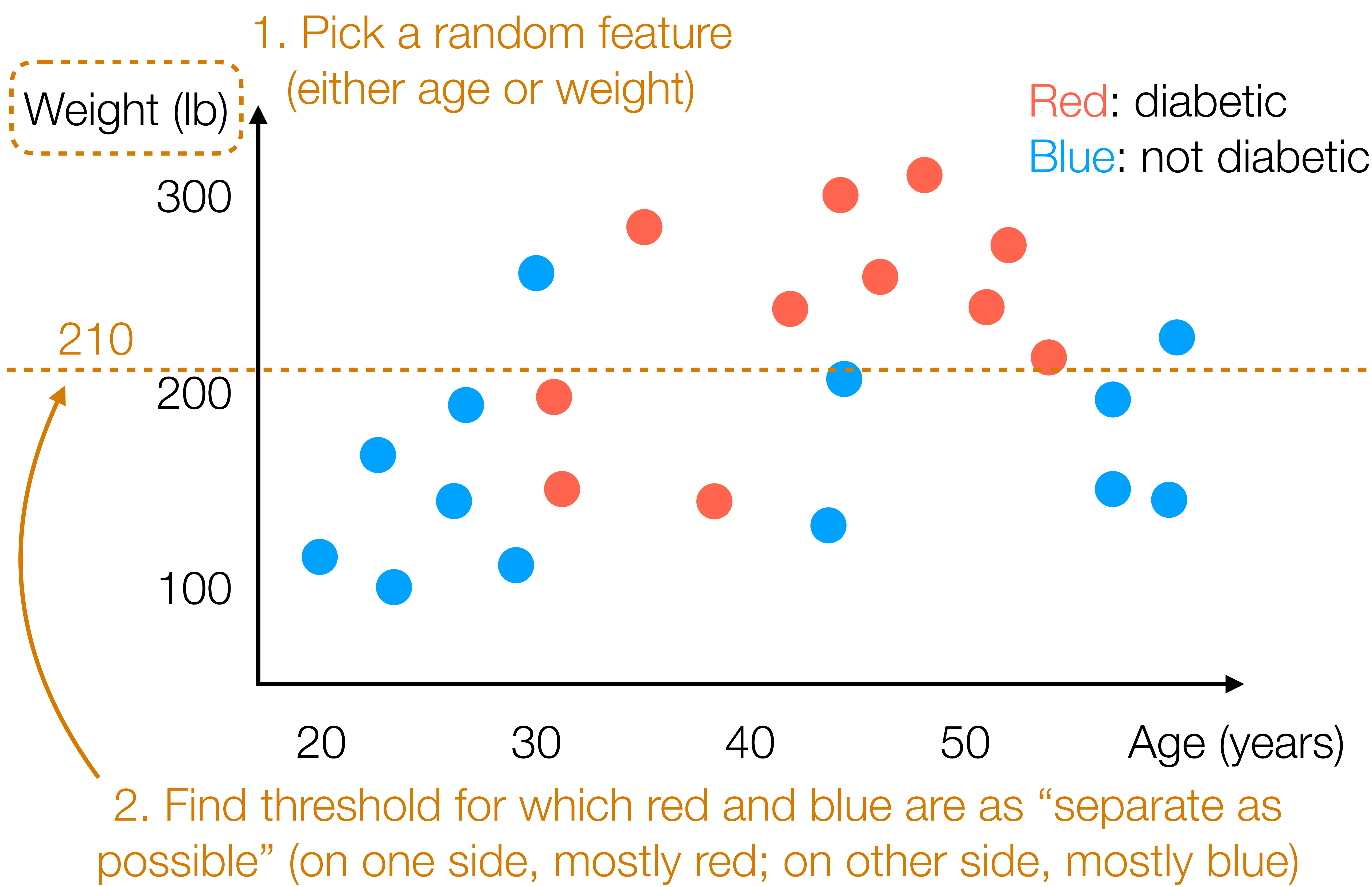
Learning a Decision Tree

1. Pick a random feature (either age or weight)

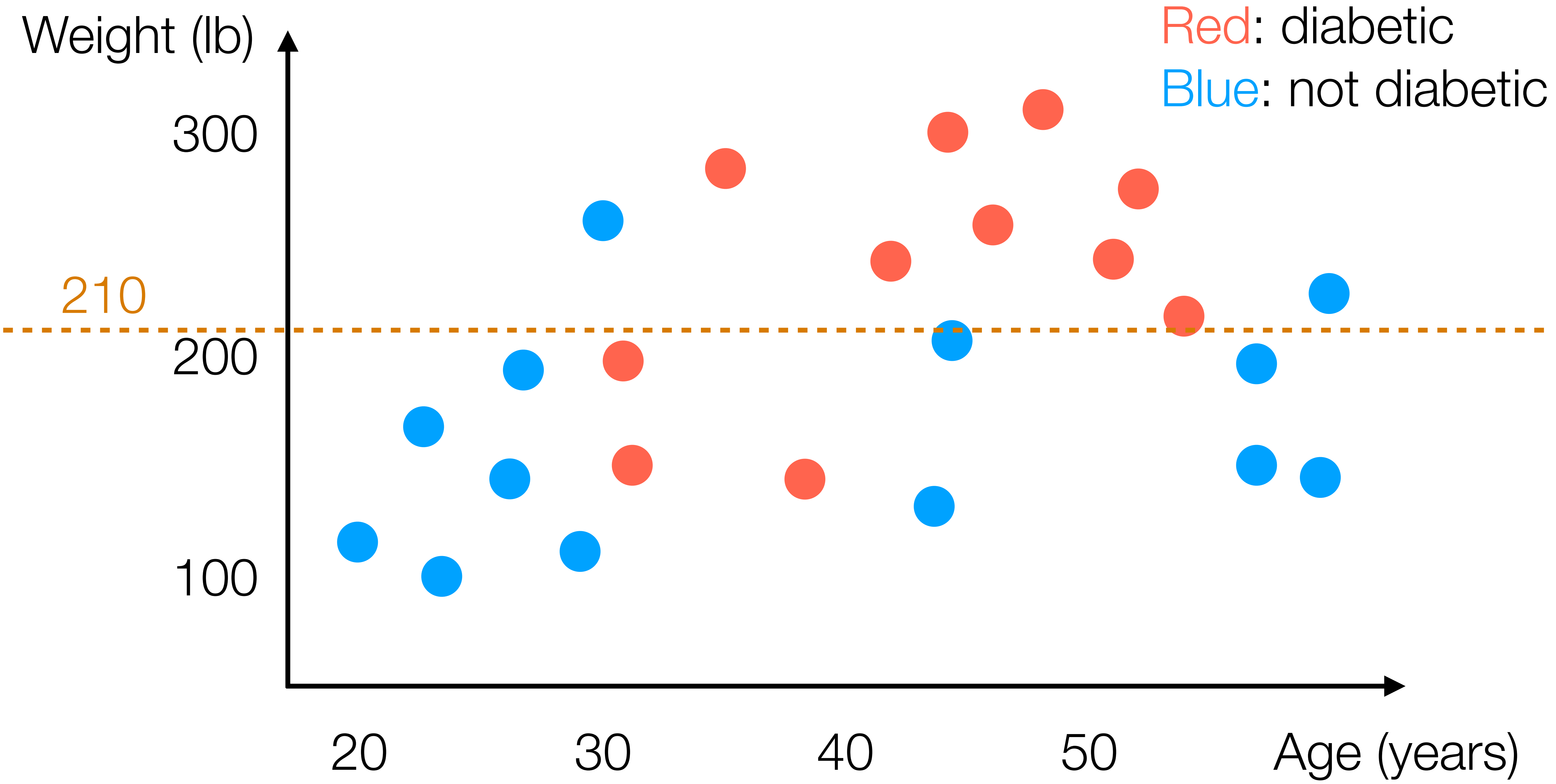


2. Find threshold for which red and blue are as “separate as possible” (on one side, mostly red; on other side, mostly blue)

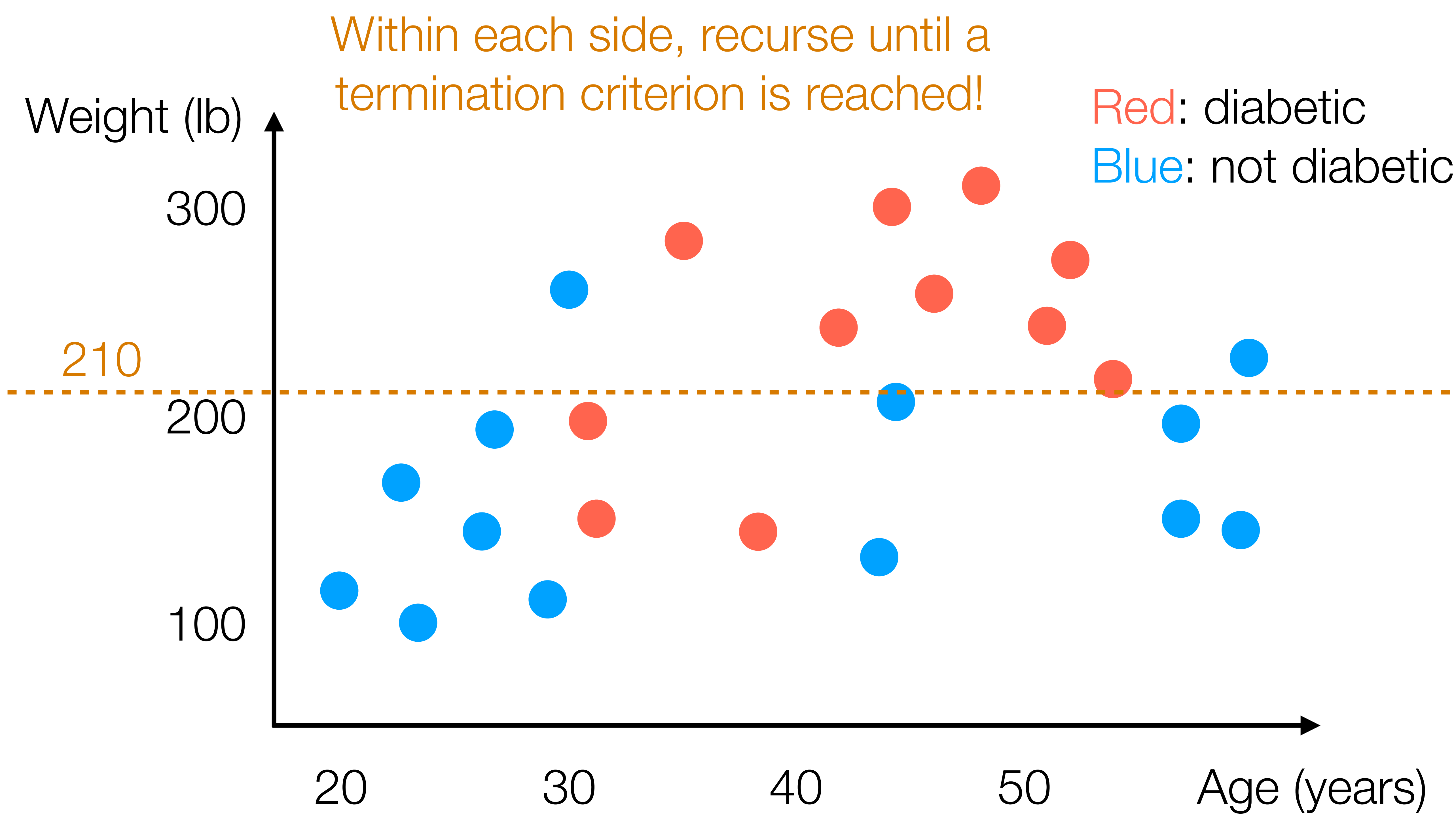
Learning a Decision Tree



Learning a Decision Tree

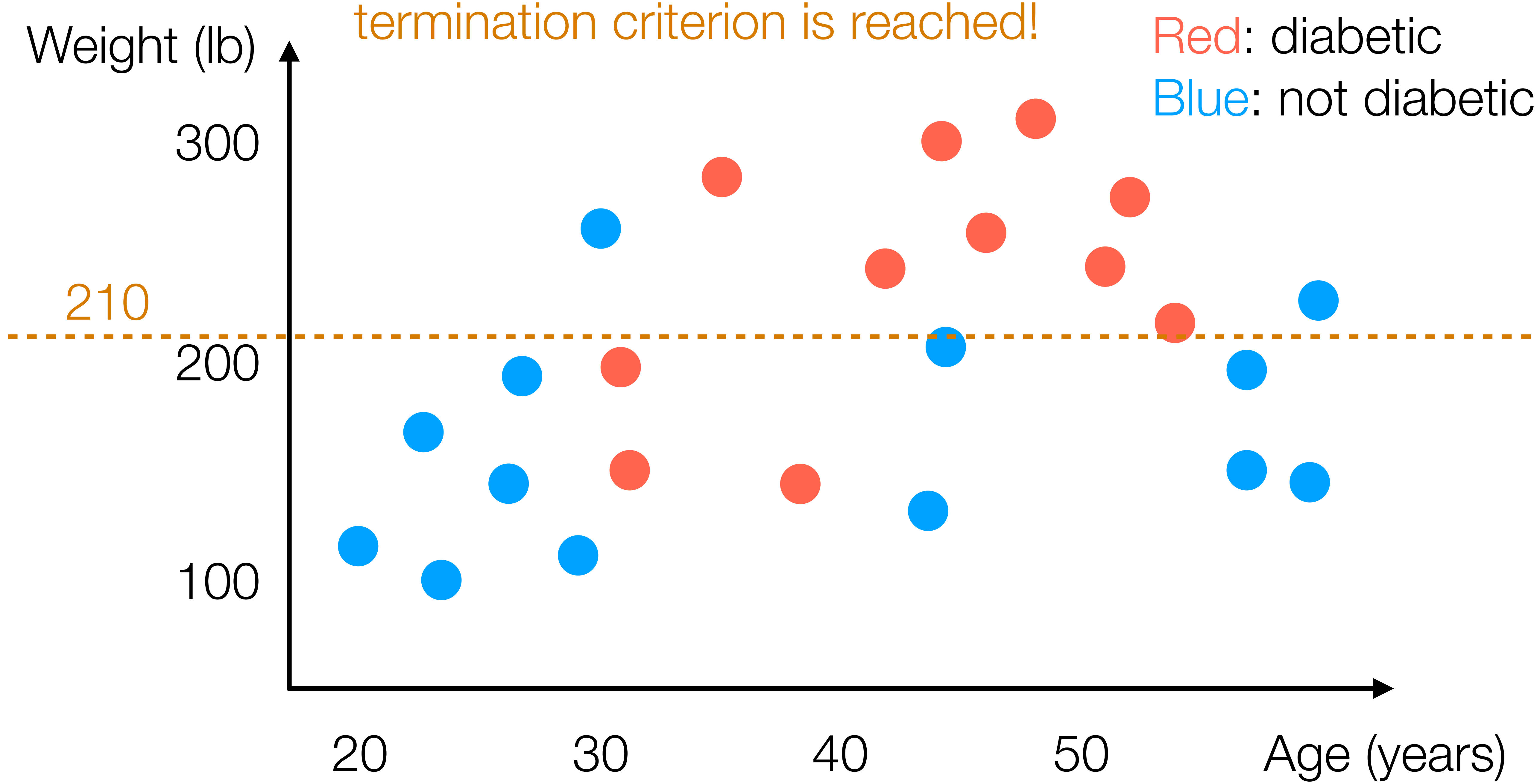


Learning a Decision Tree



Learning a Decision Tree

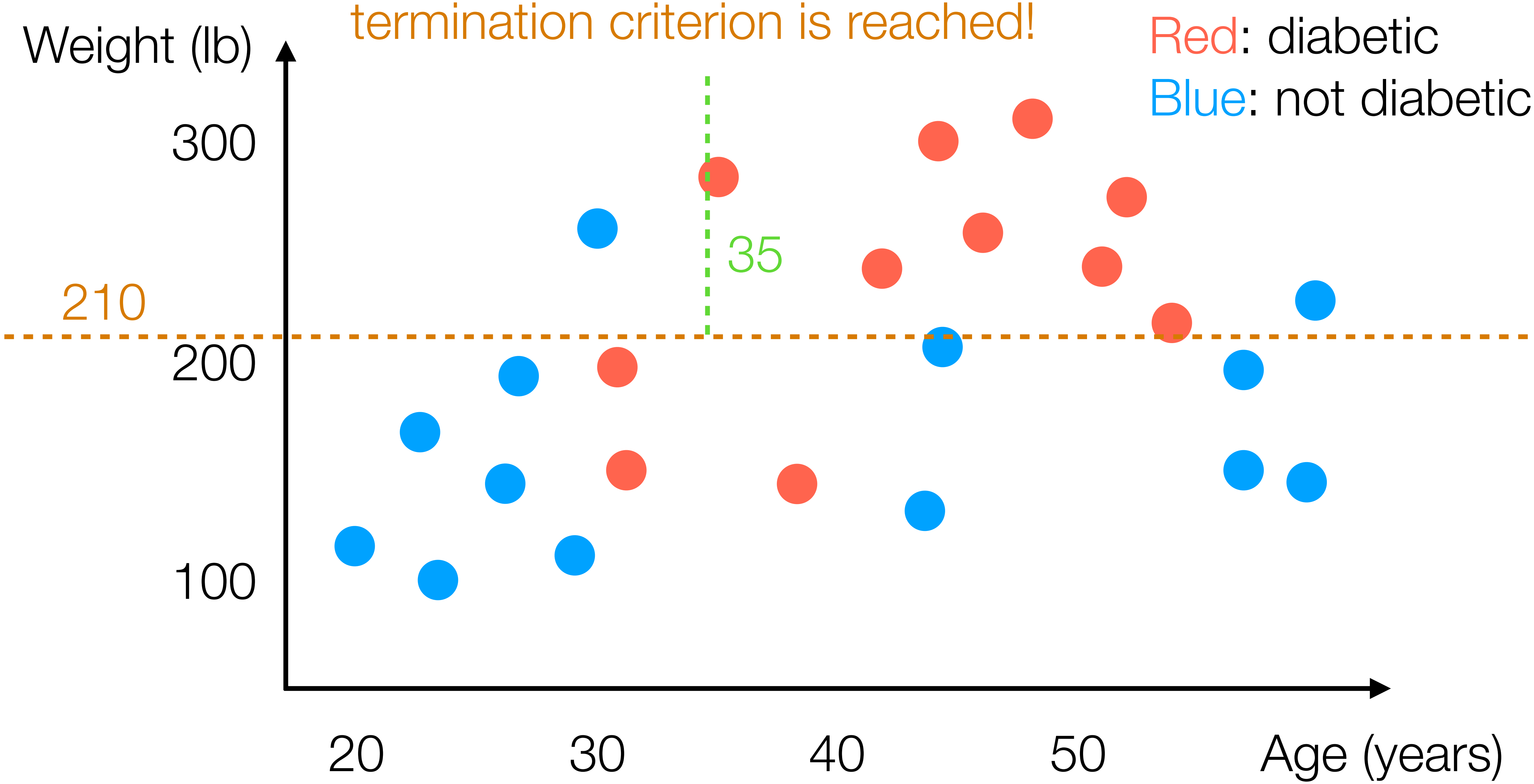
Within each side, recurse until a termination criterion is reached!



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

Learning a Decision Tree

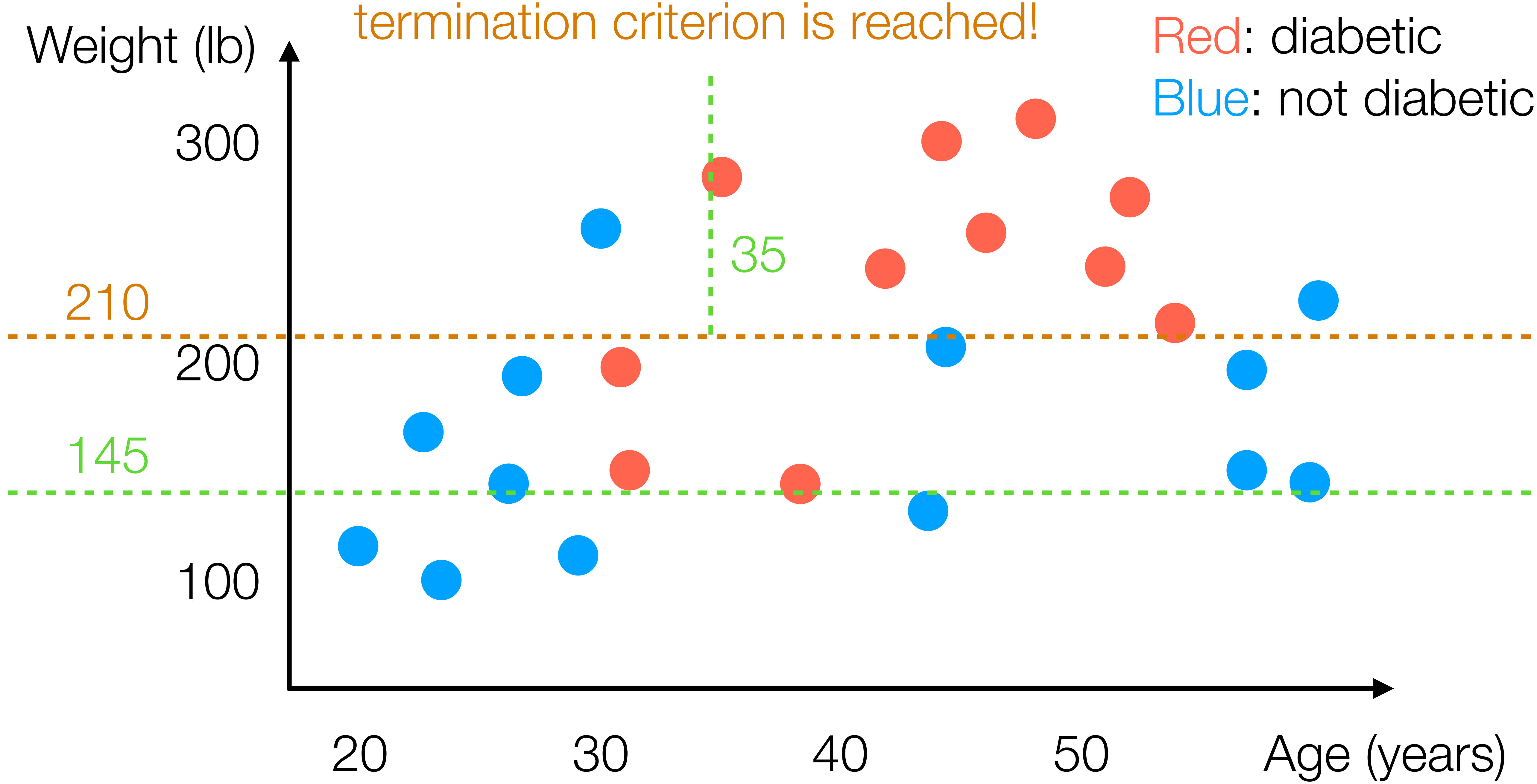
Within each side, recurse until a termination criterion is reached!



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

Learning a Decision Tree

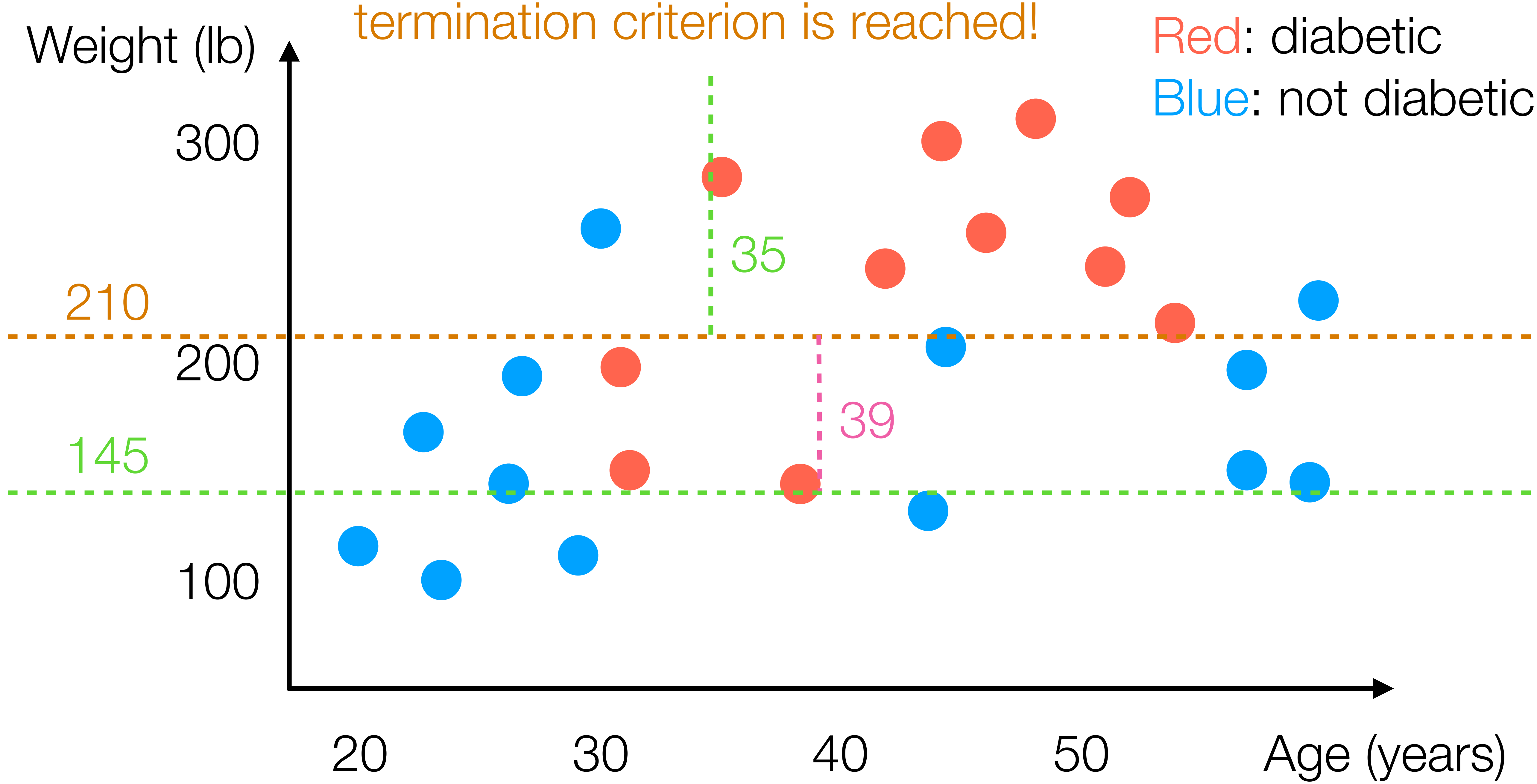
Within each side, recurse until a termination criterion is reached!



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

Learning a Decision Tree

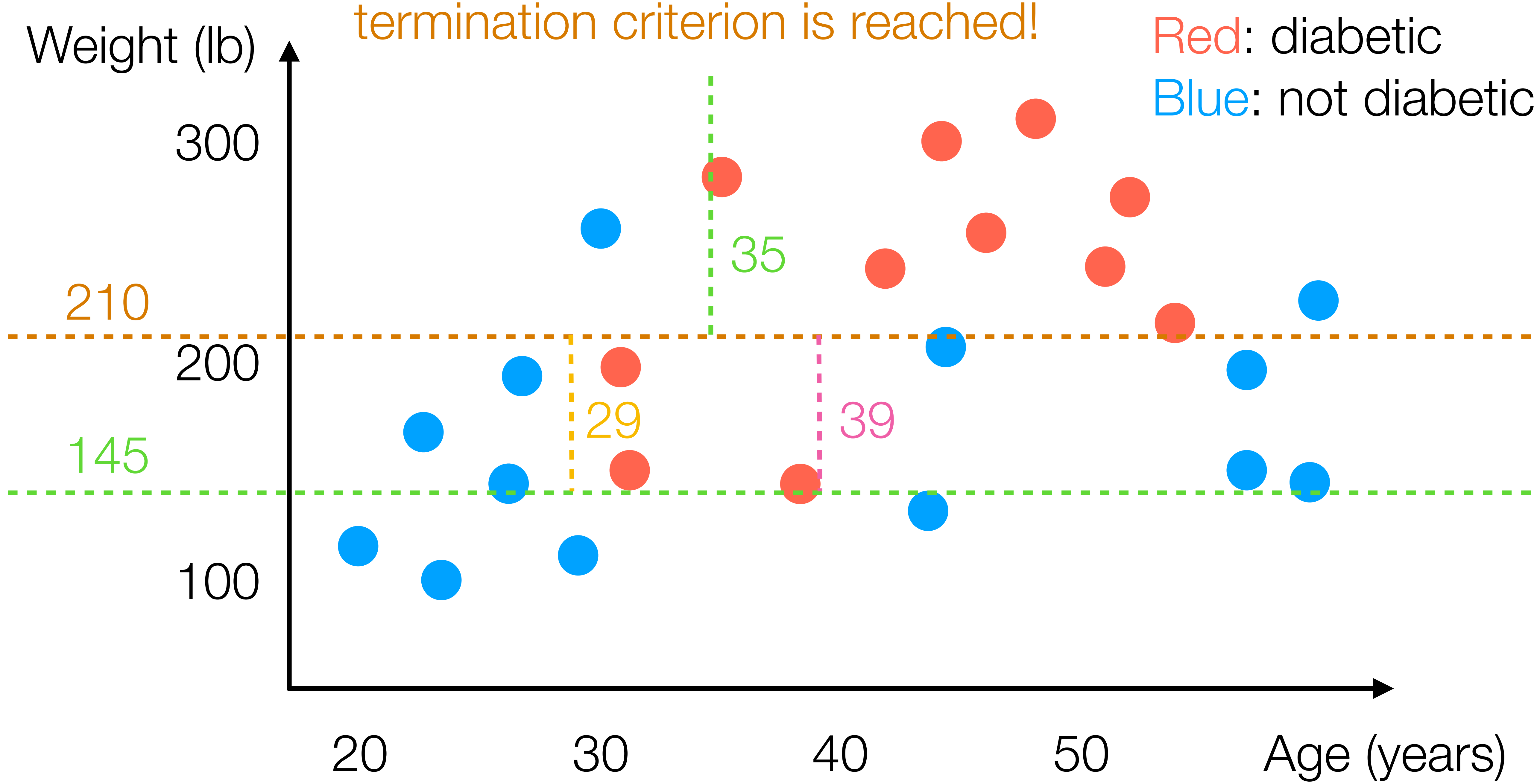
Within each side, recurse until a termination criterion is reached!



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

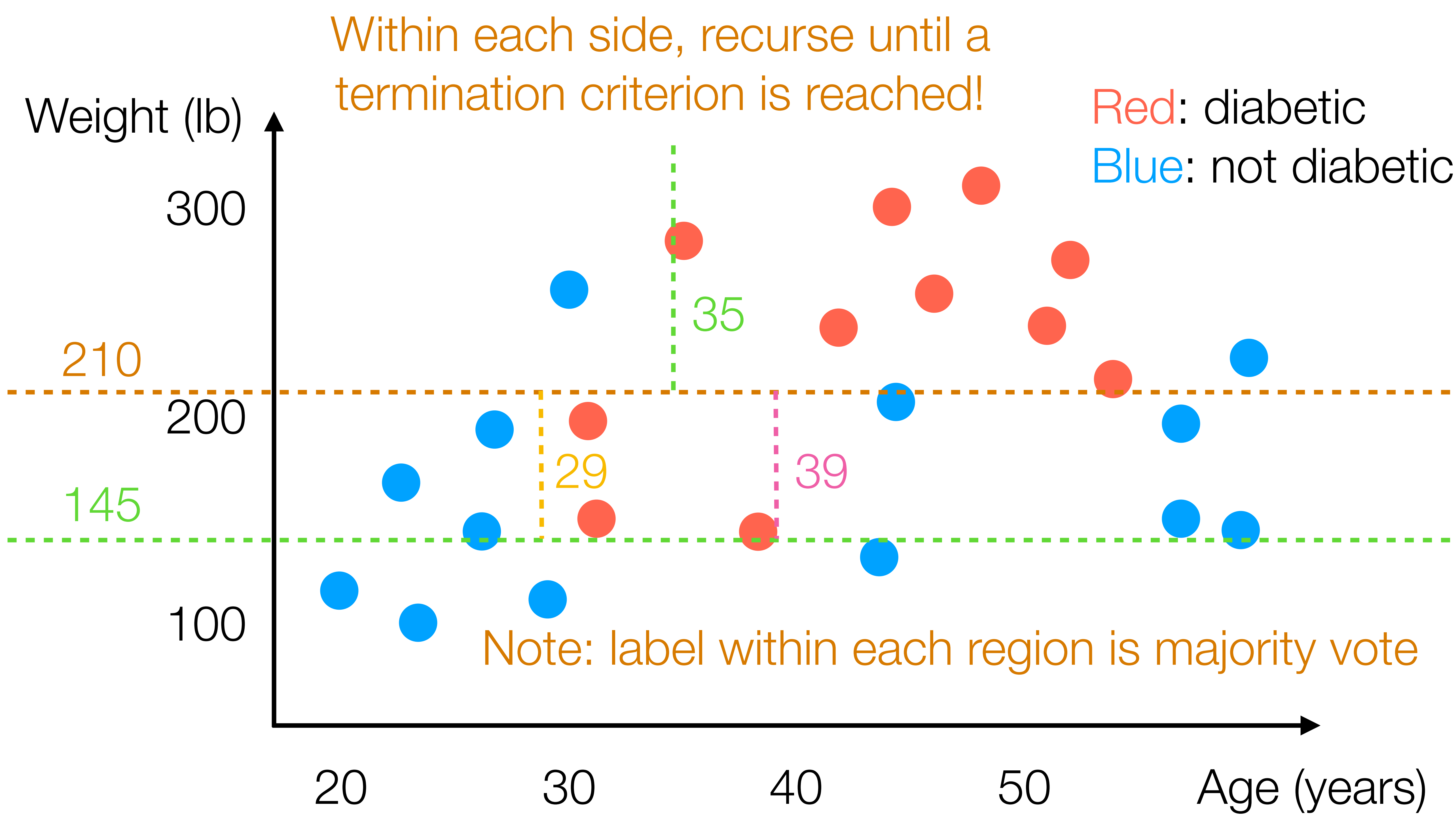
Learning a Decision Tree

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

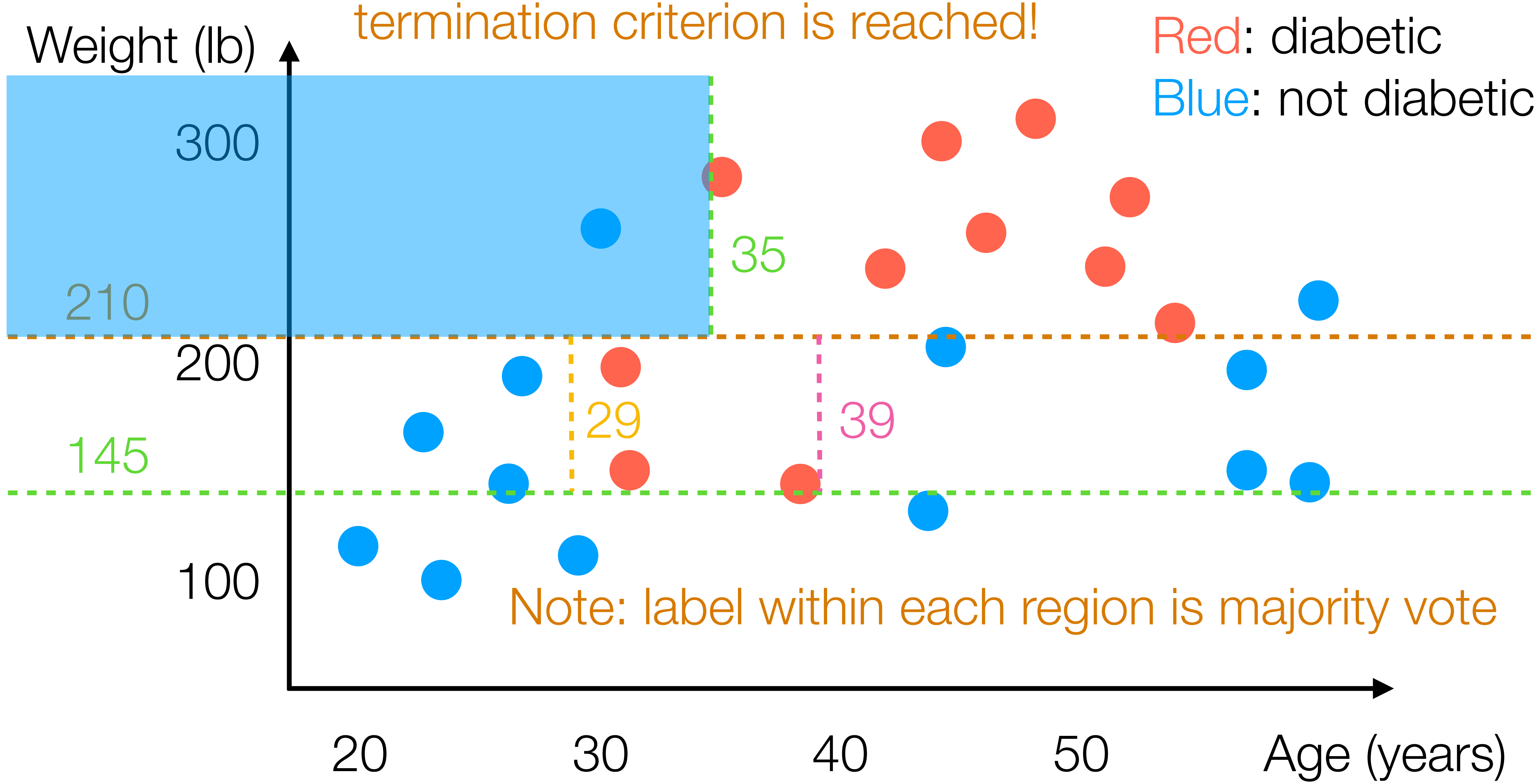
Learning a Decision Tree



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

Learning a Decision Tree

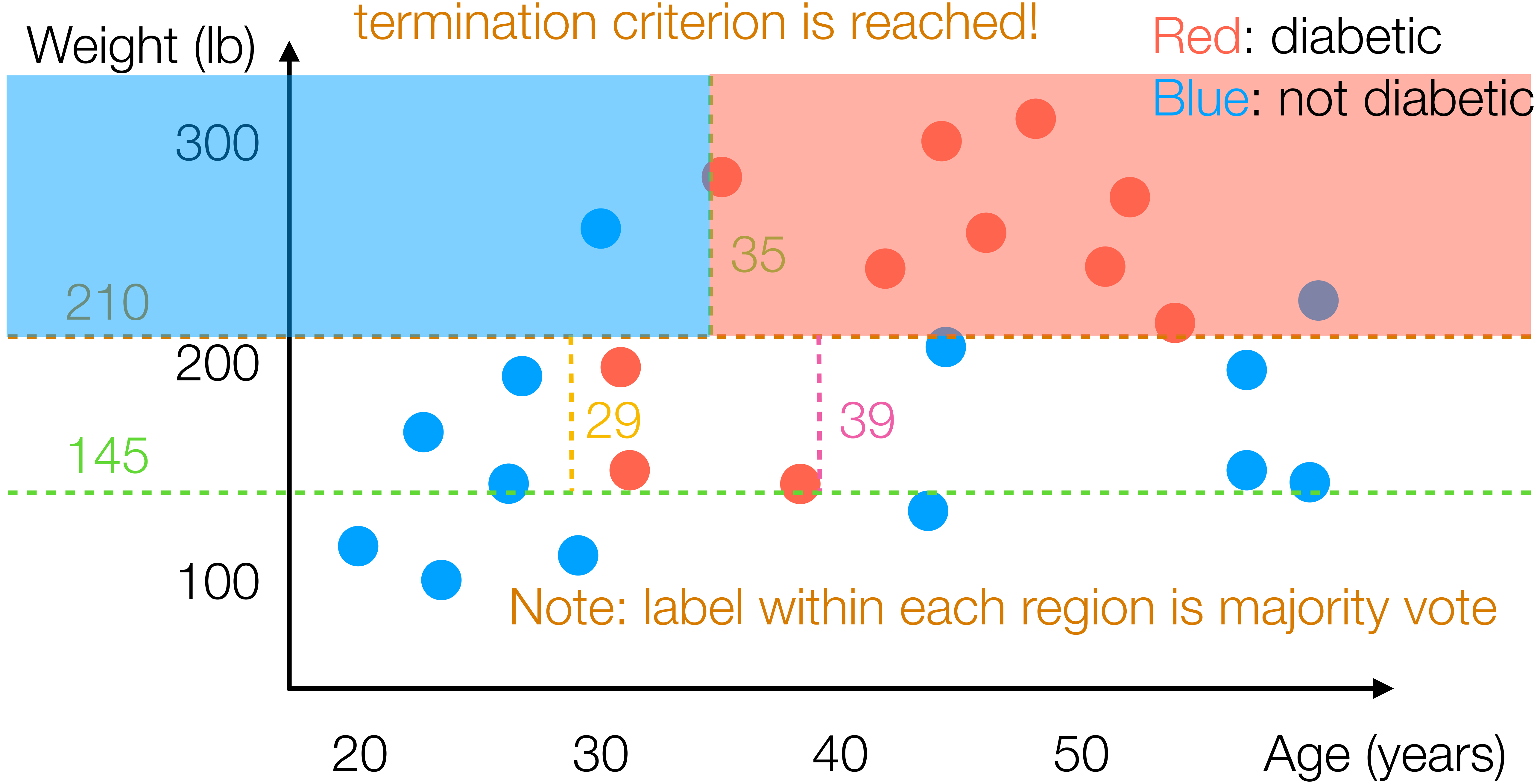
Within each side, recurse until a termination criterion is reached!



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

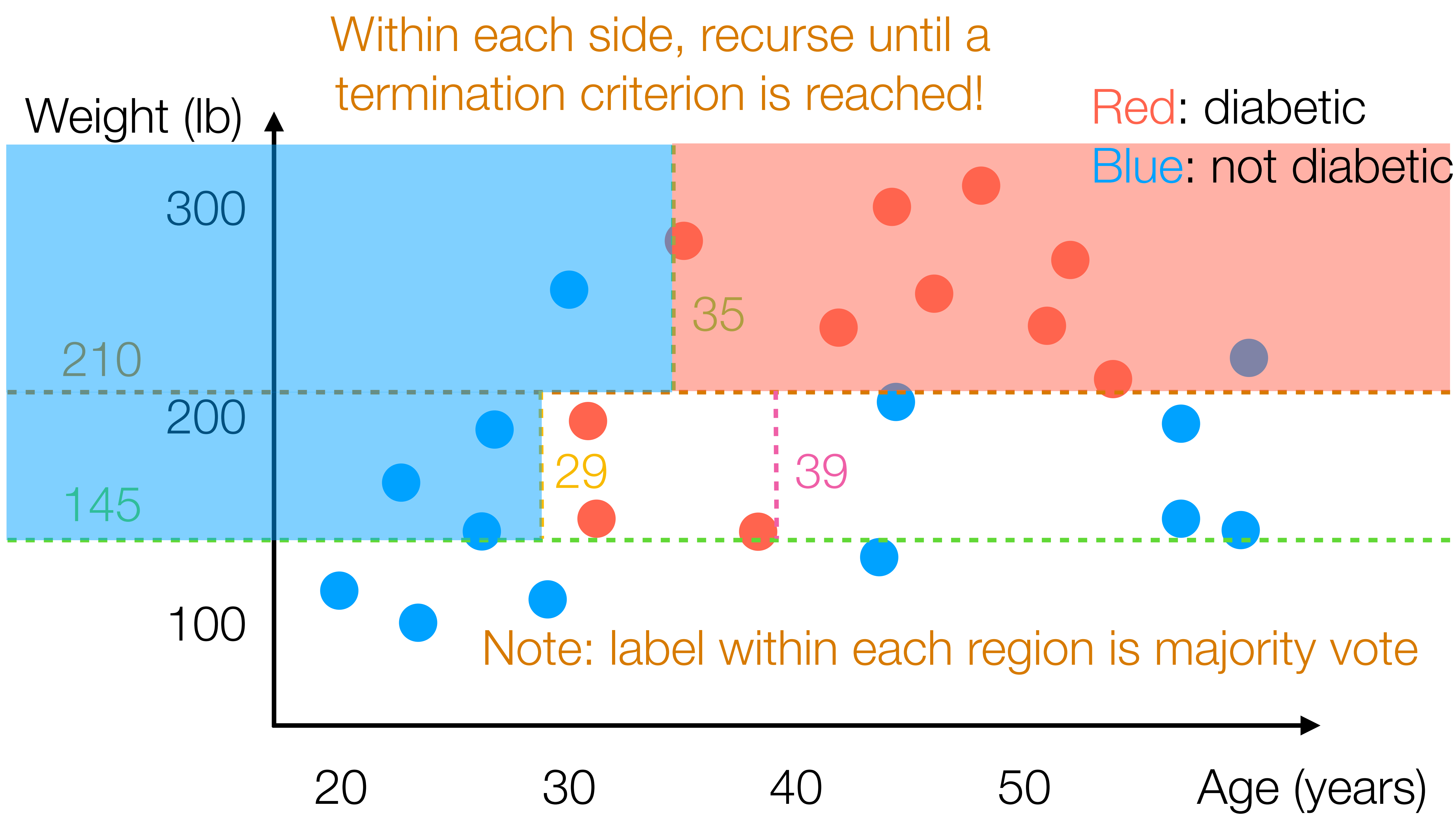
Learning a Decision Tree

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

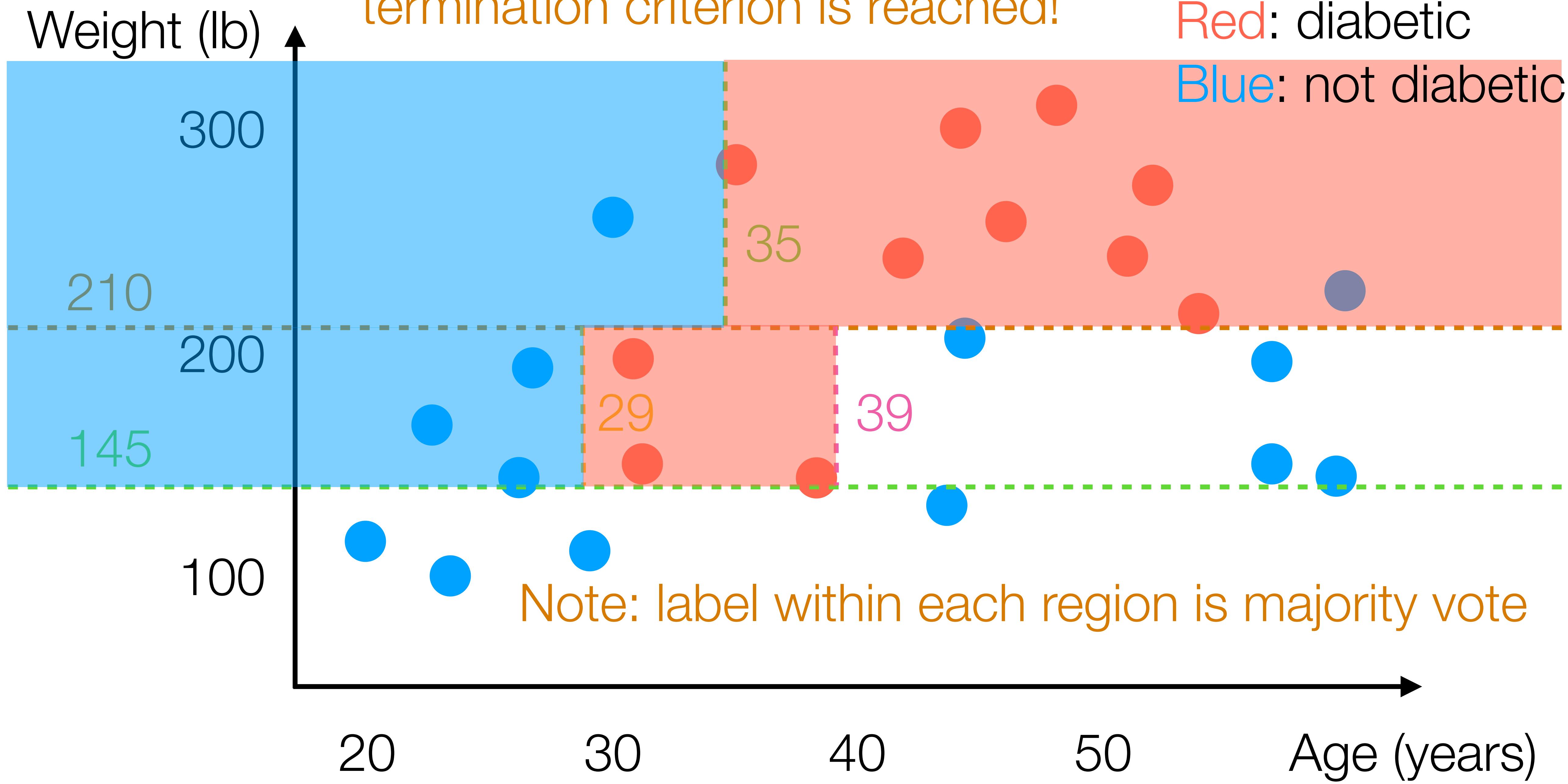
Learning a Decision Tree



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

Learning a Decision Tree

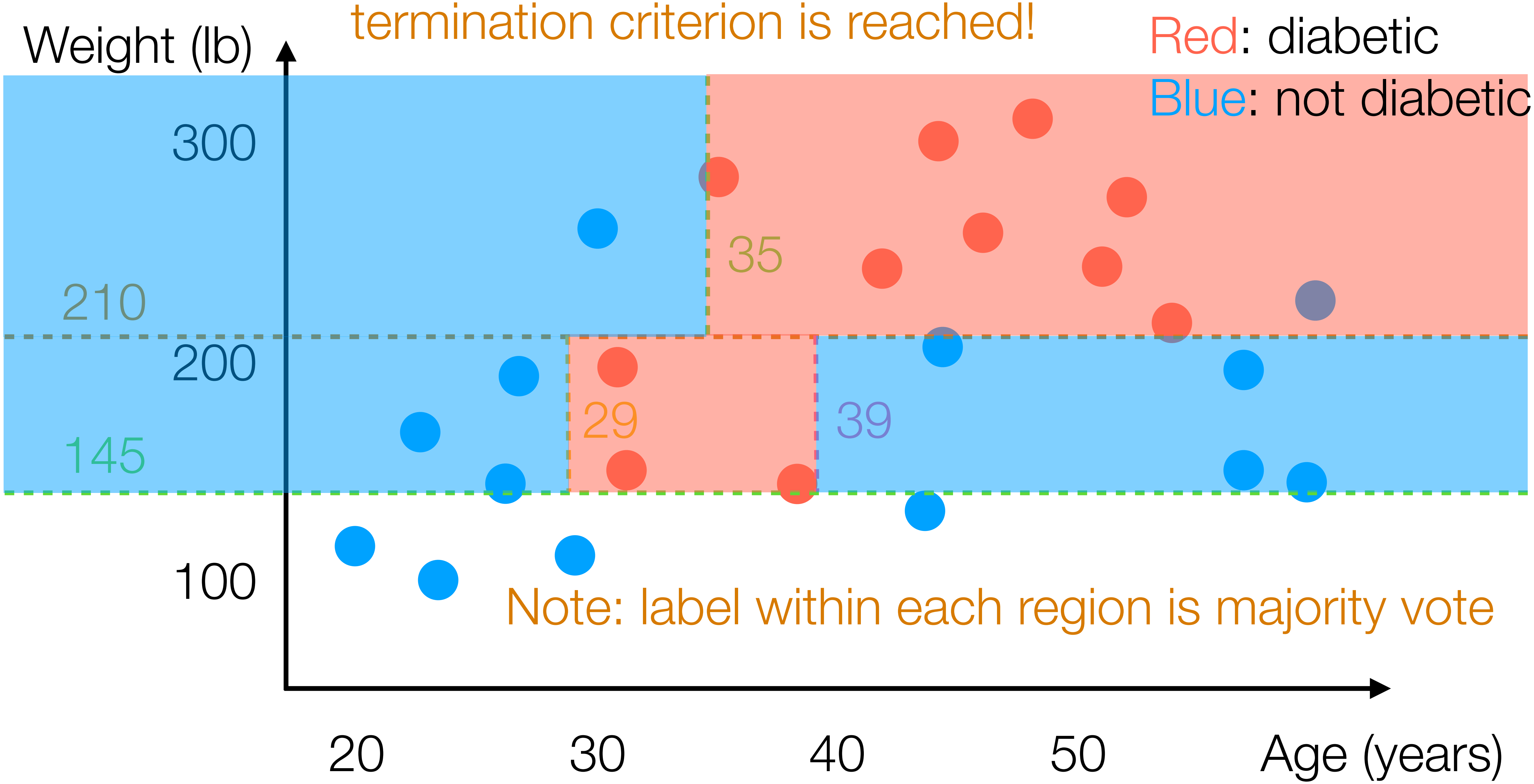
Within each side, recurse until a termination criterion is reached!



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

Learning a Decision Tree

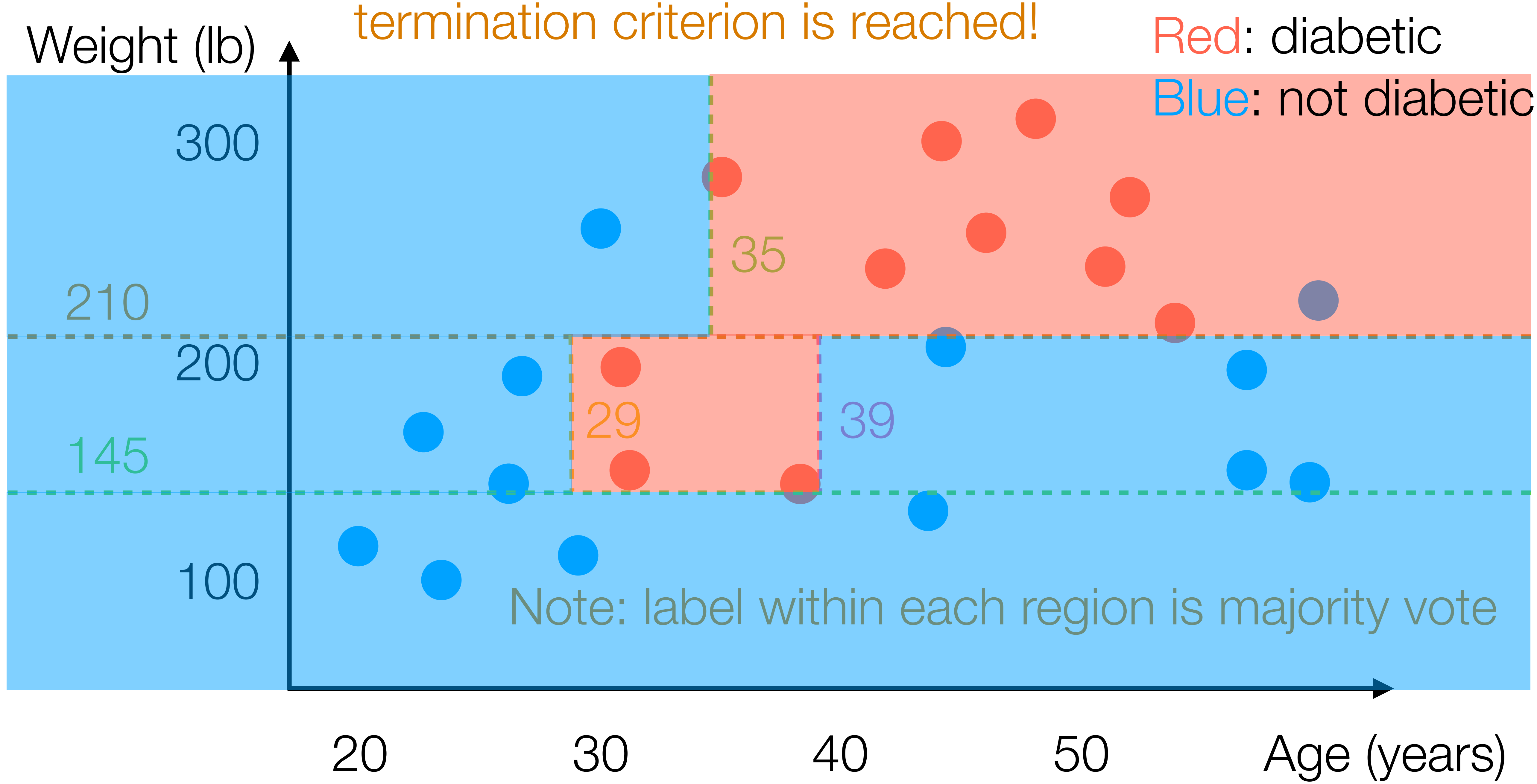
Within each side, recurse until a termination criterion is reached!



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

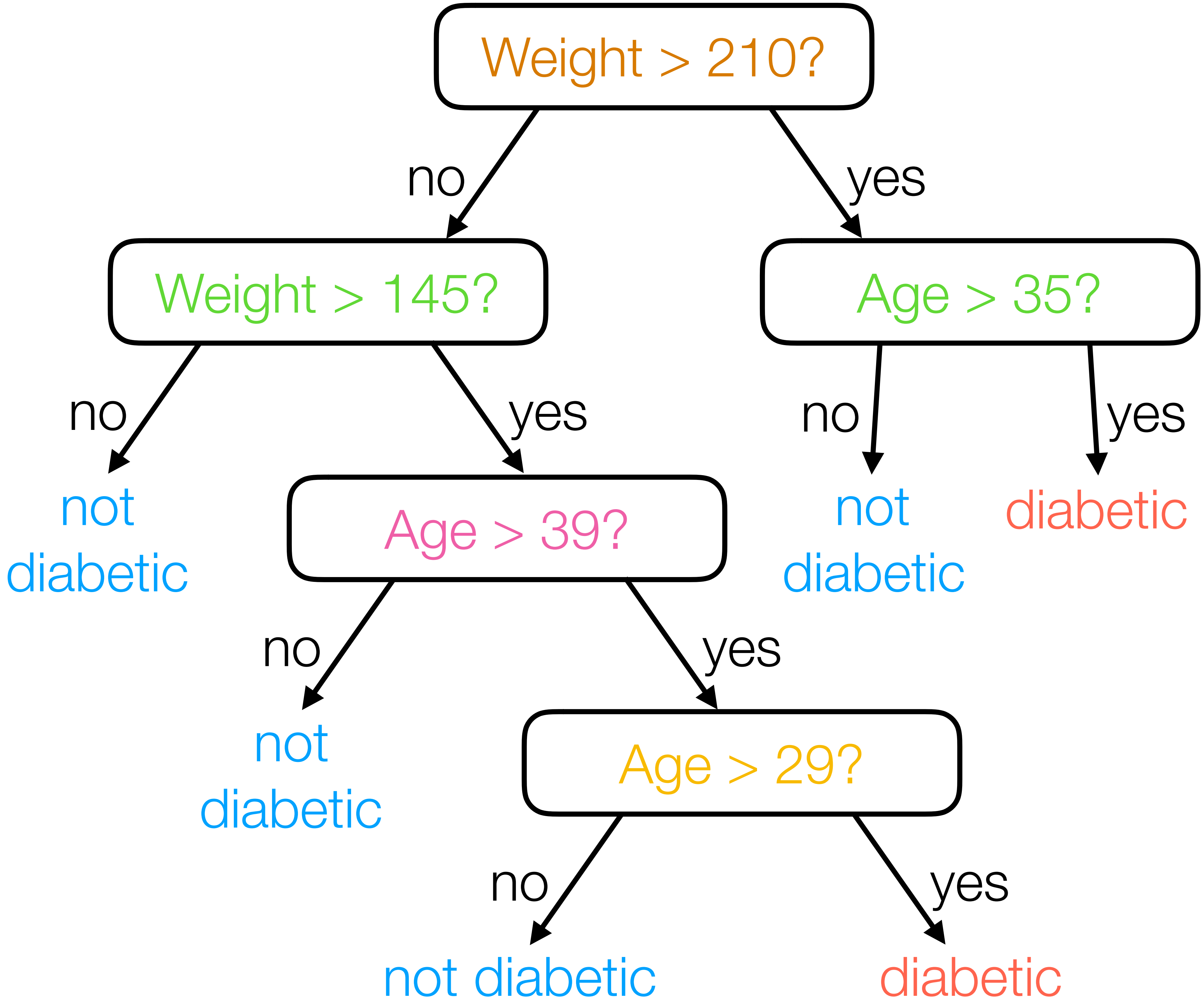
Learning a Decision Tree

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

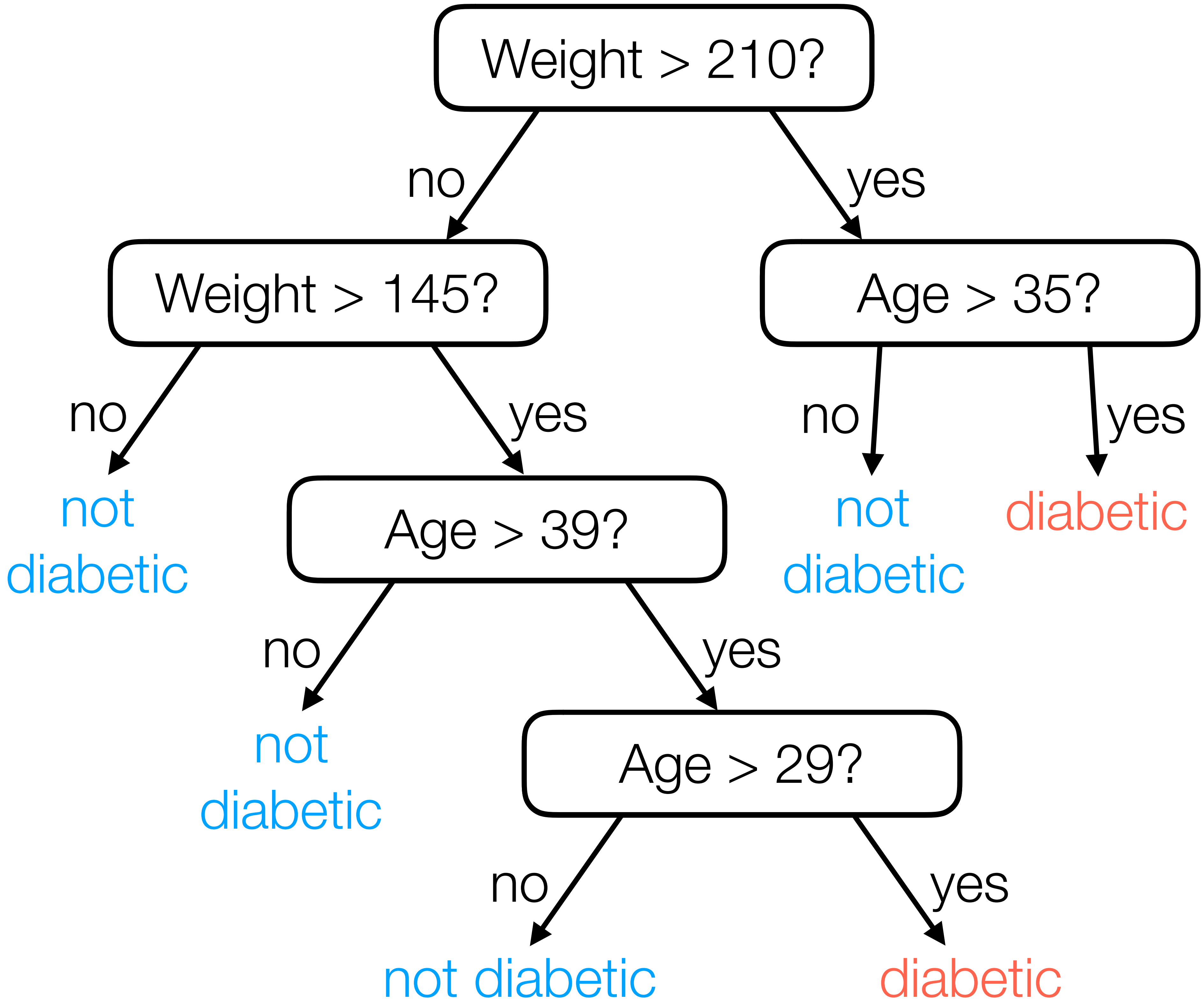


Example termination criteria: ≥90% points within region has same label, number of points within region is <5

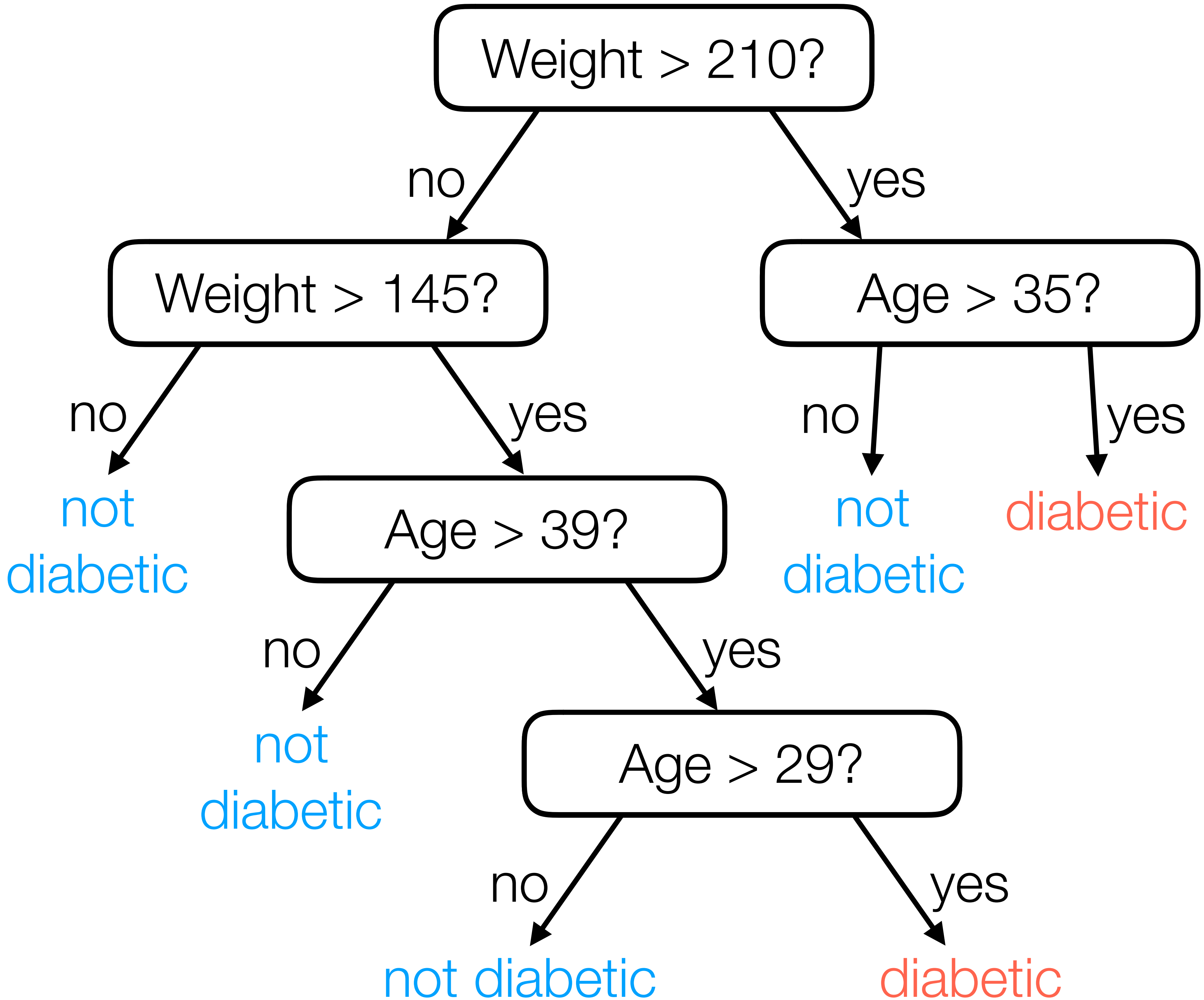
Decision Tree Learned



Decision Tree Learned

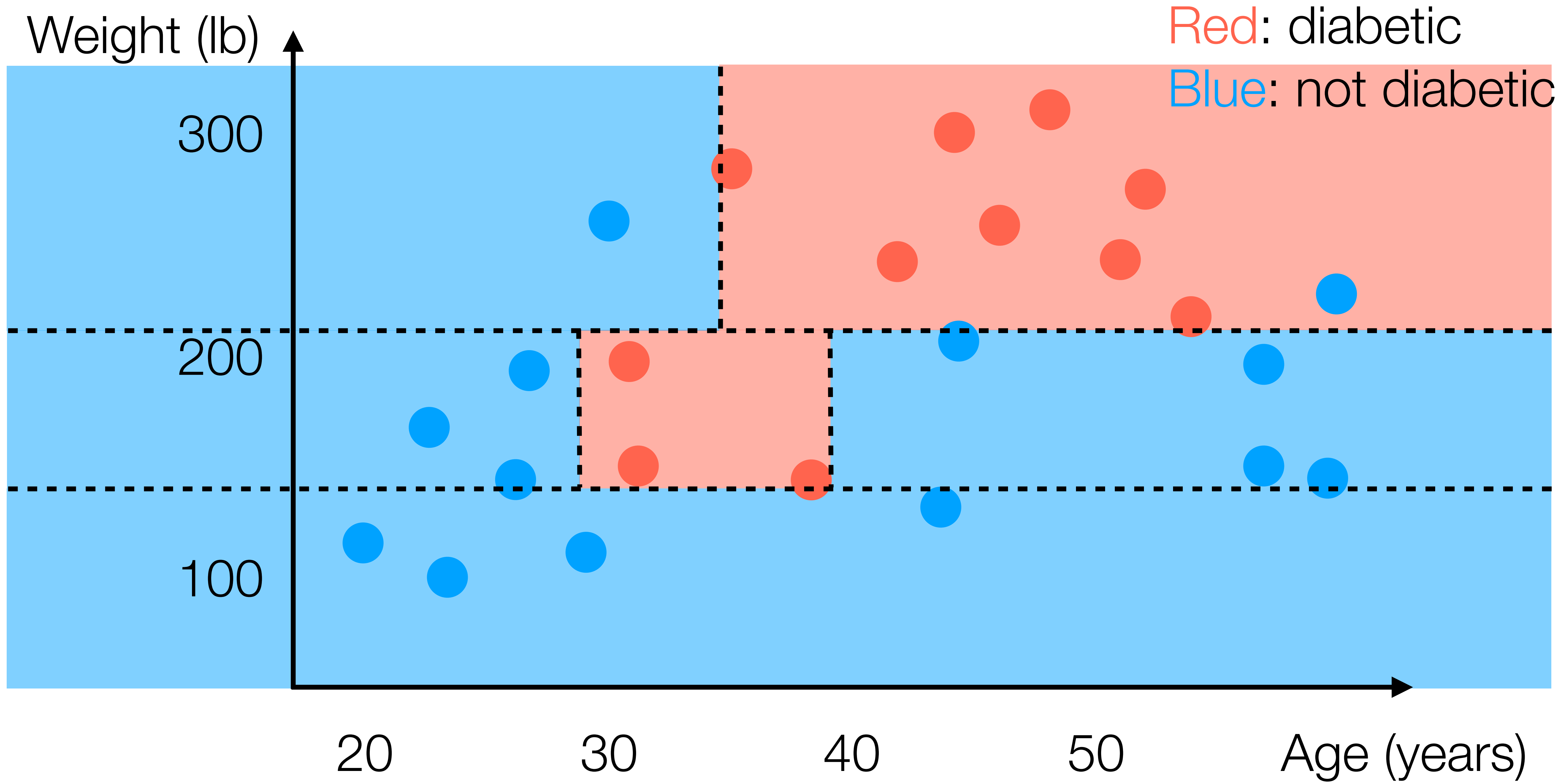


Decision Tree Learned



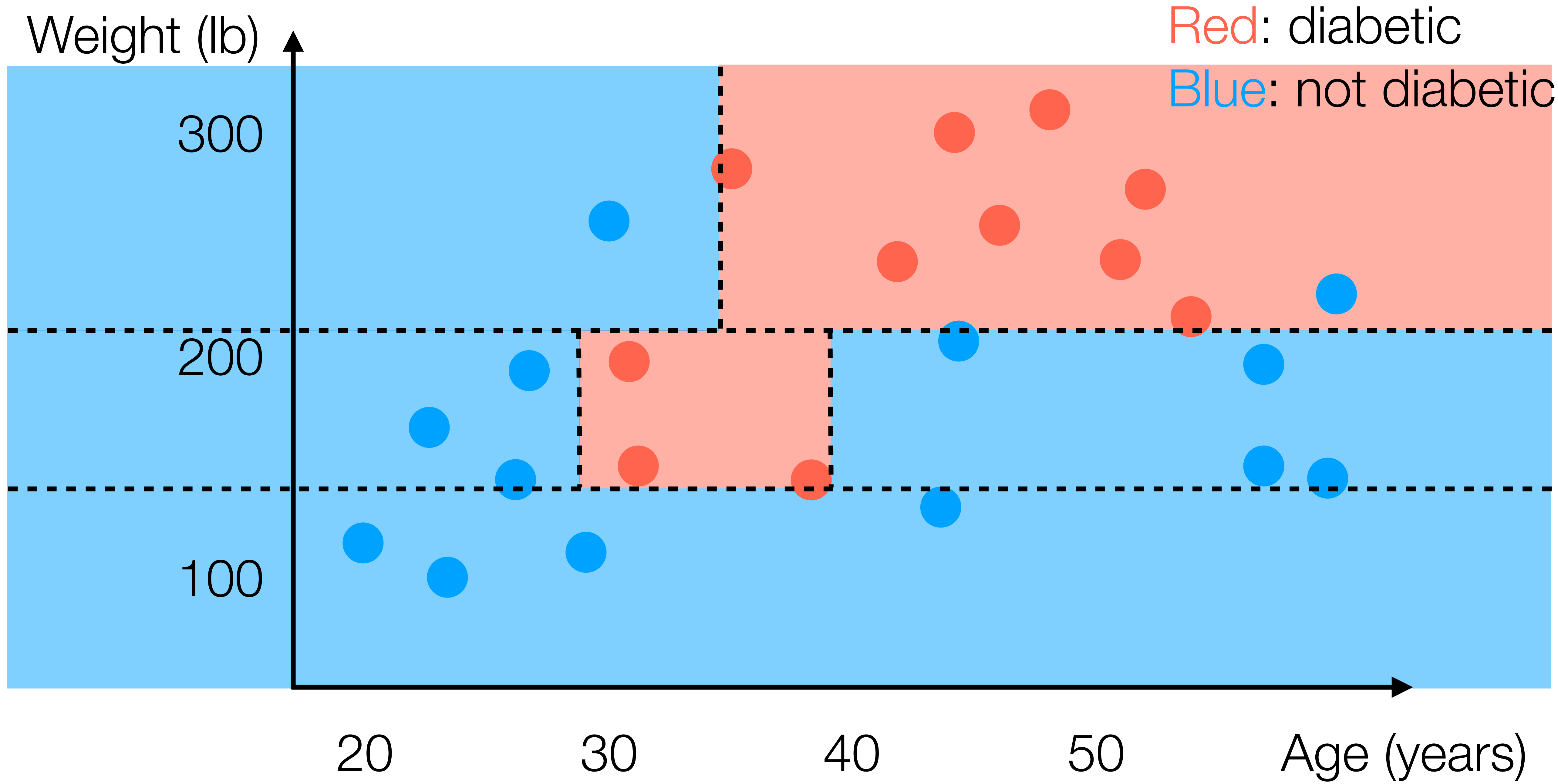
For a new person with feature vector (age, weight), easy to predict!

Nearest Neighbor Interpretation



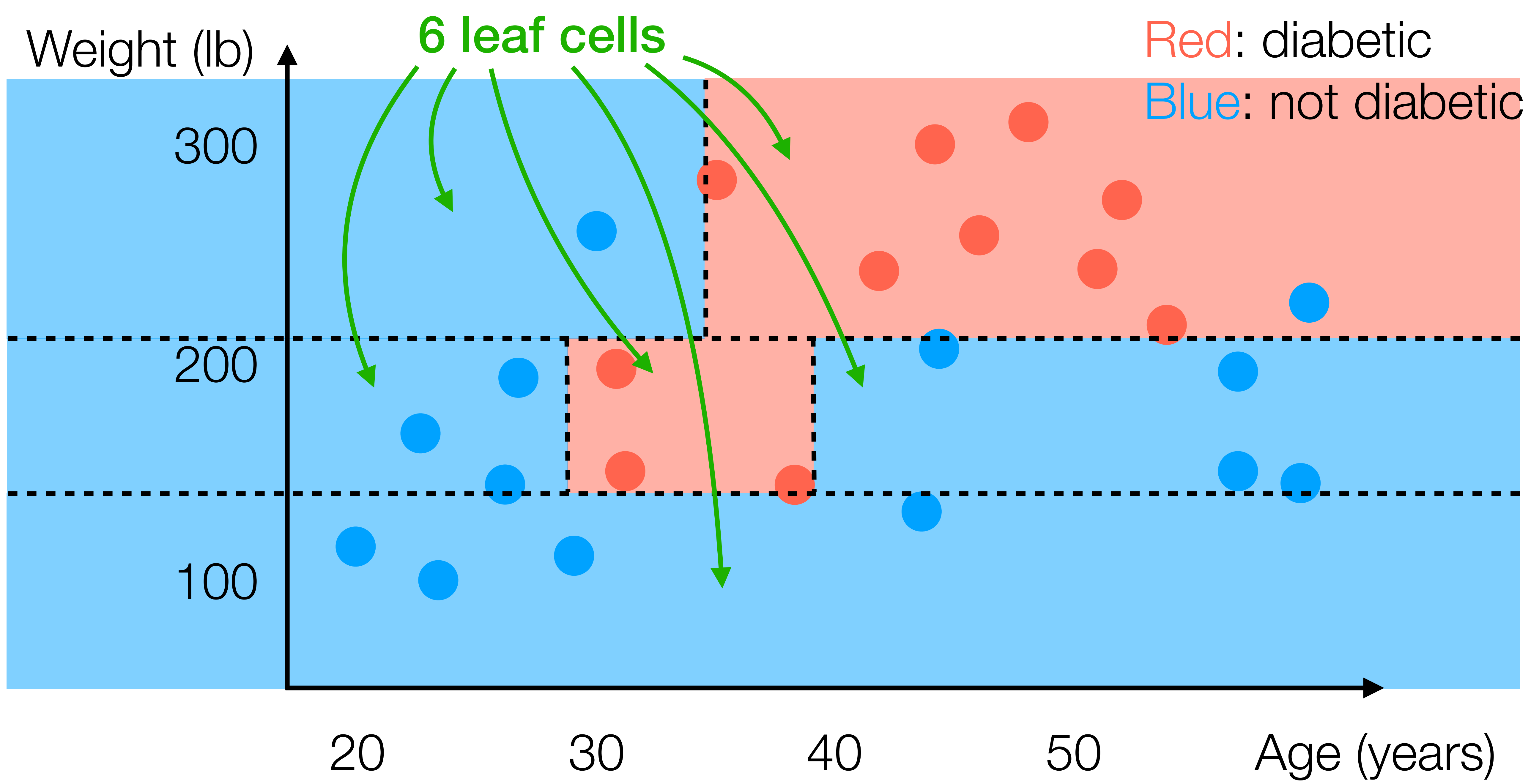
Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

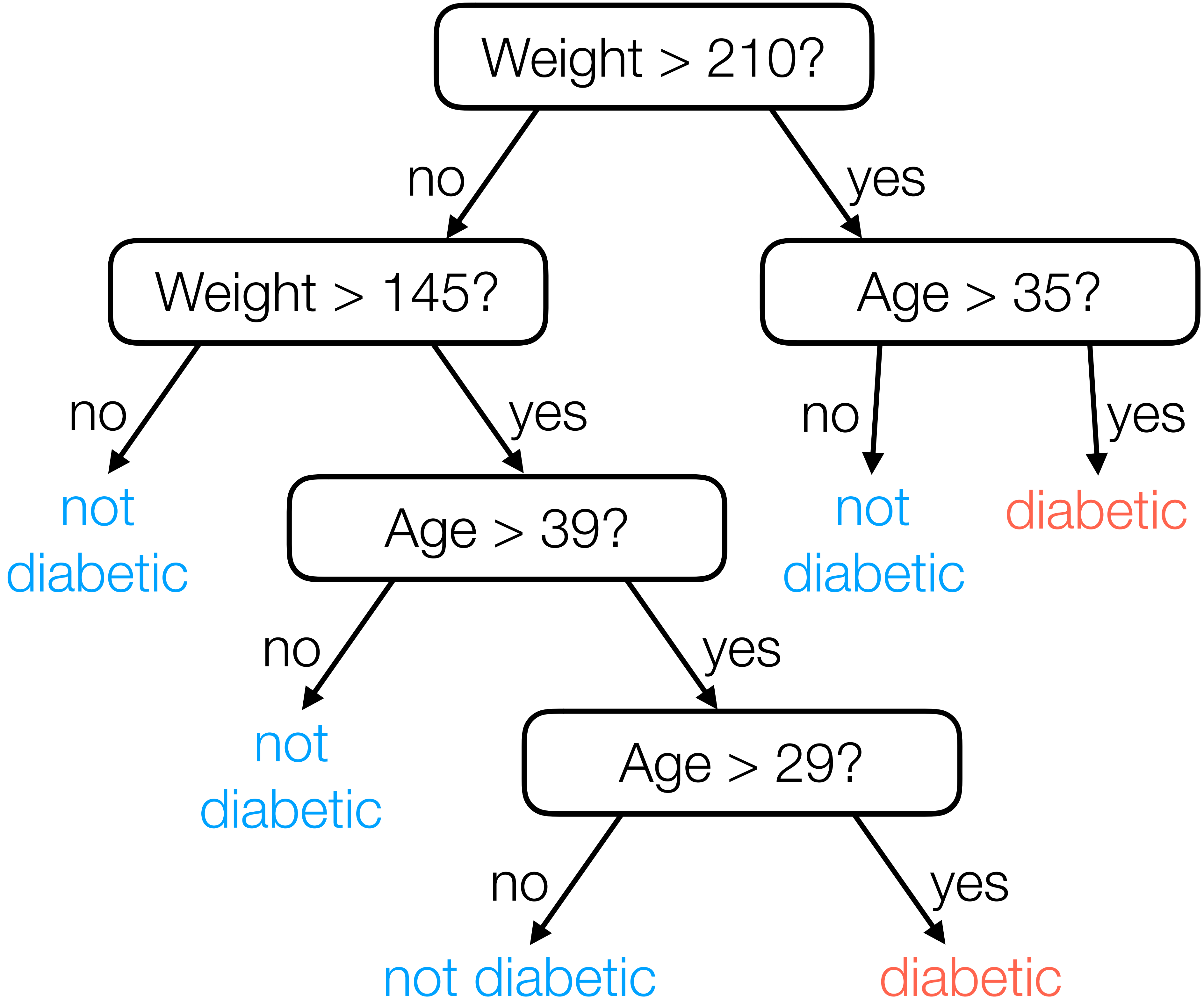


Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"

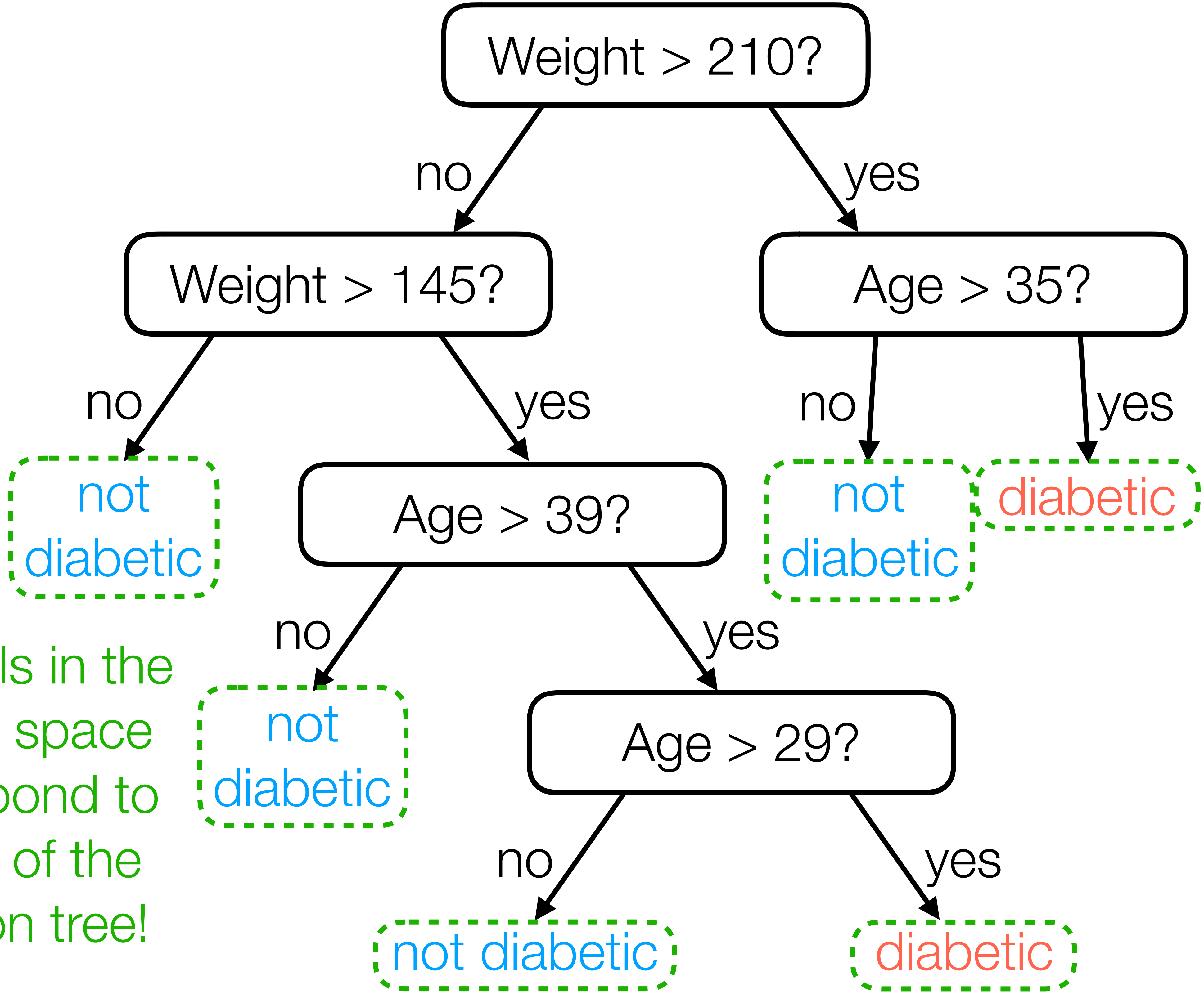


Decision Tree Learned



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

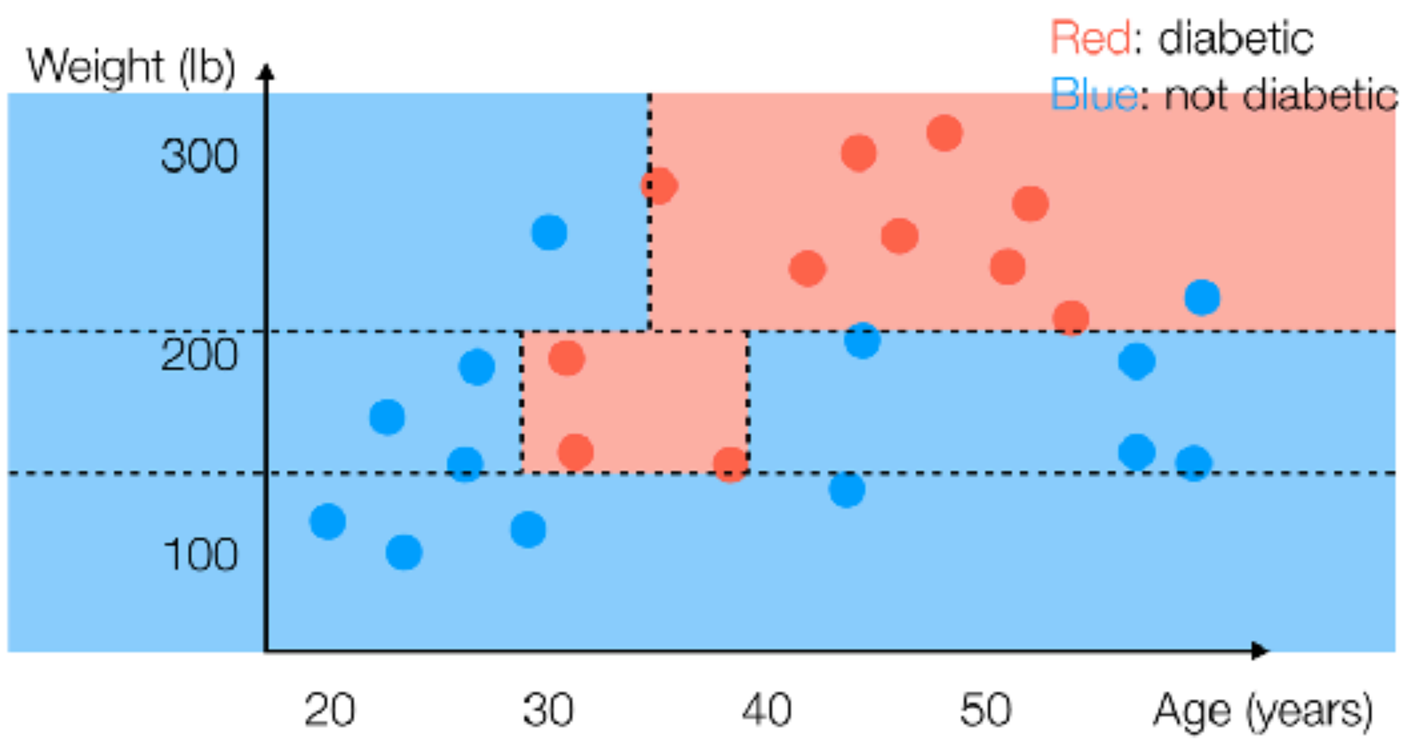
Decision Tree Learned



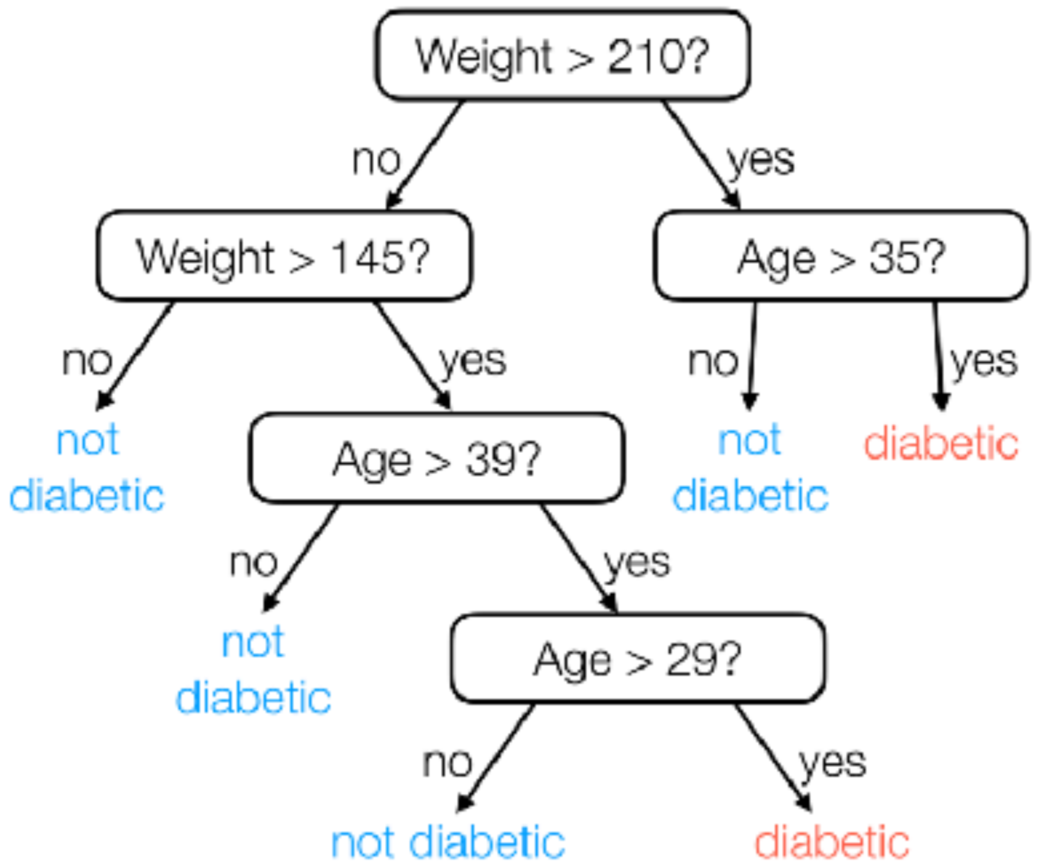
Leaf cells in the feature space correspond to leaves of the decision tree!

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

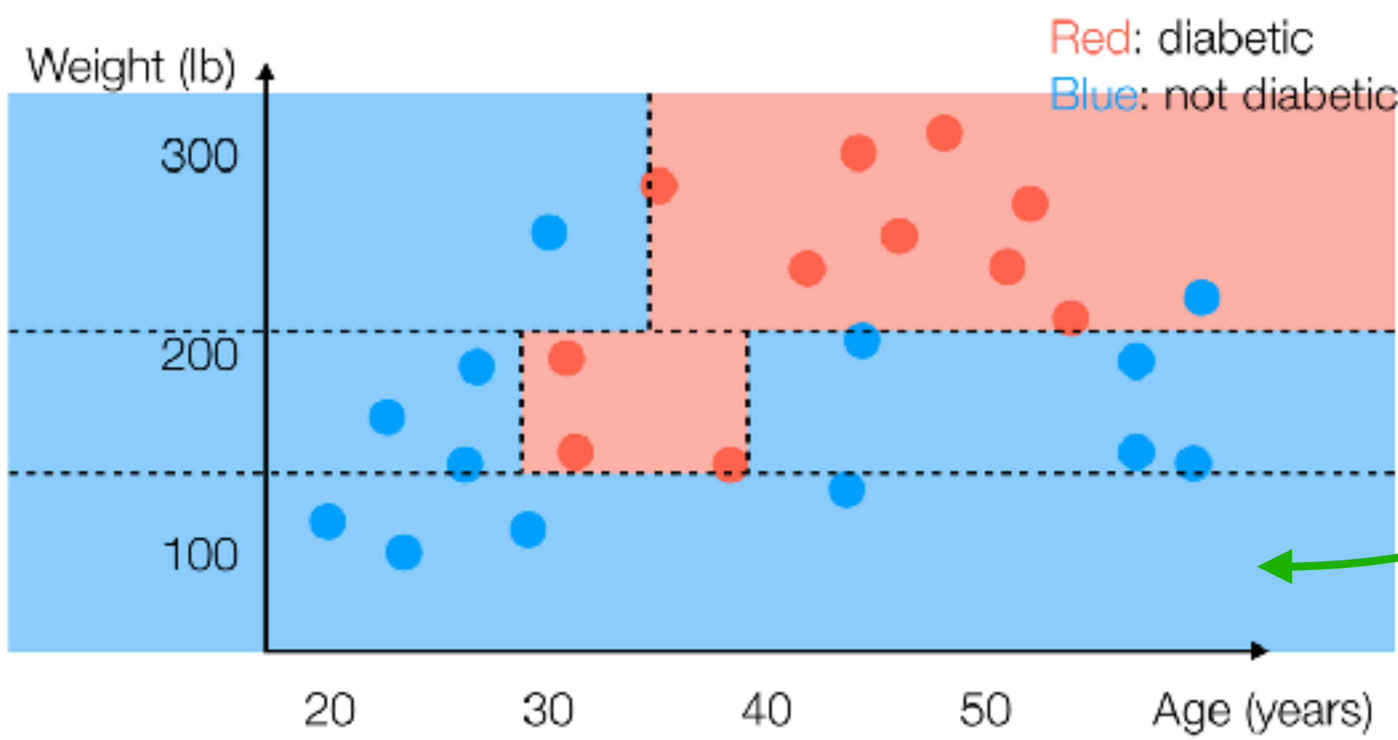
Feature space sliced up into leaf cells



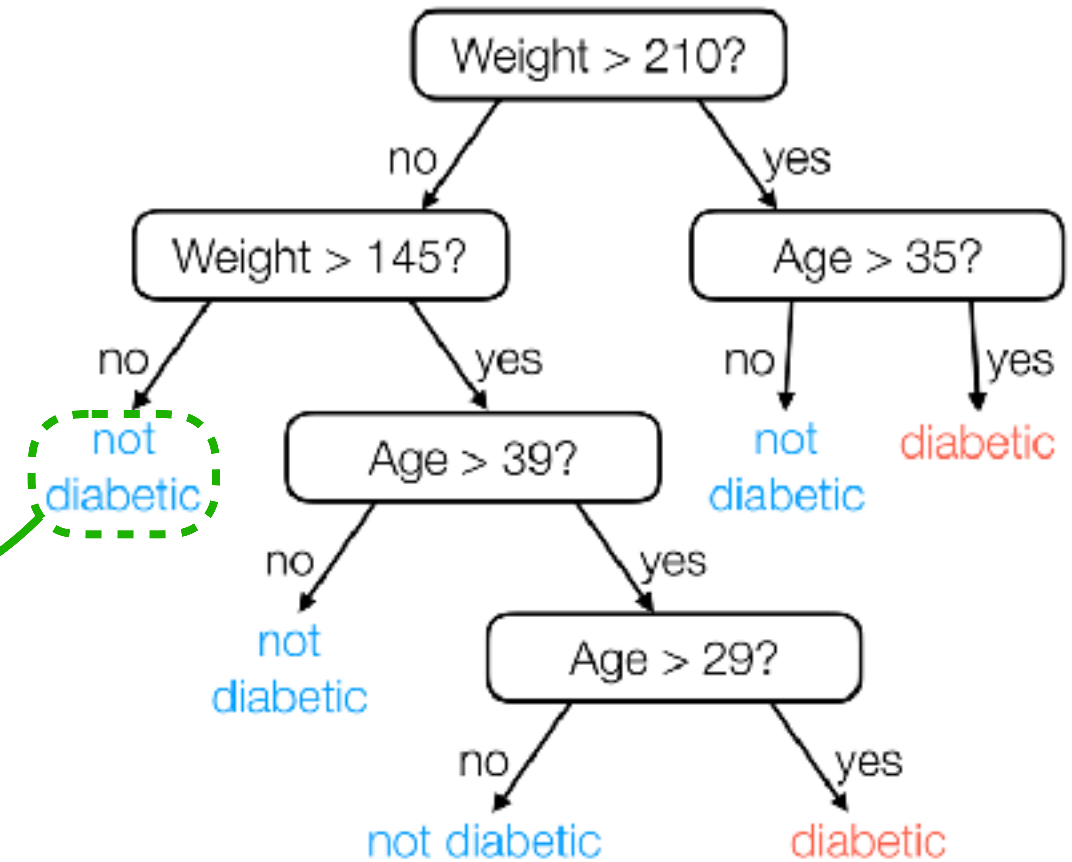
Decision Tree



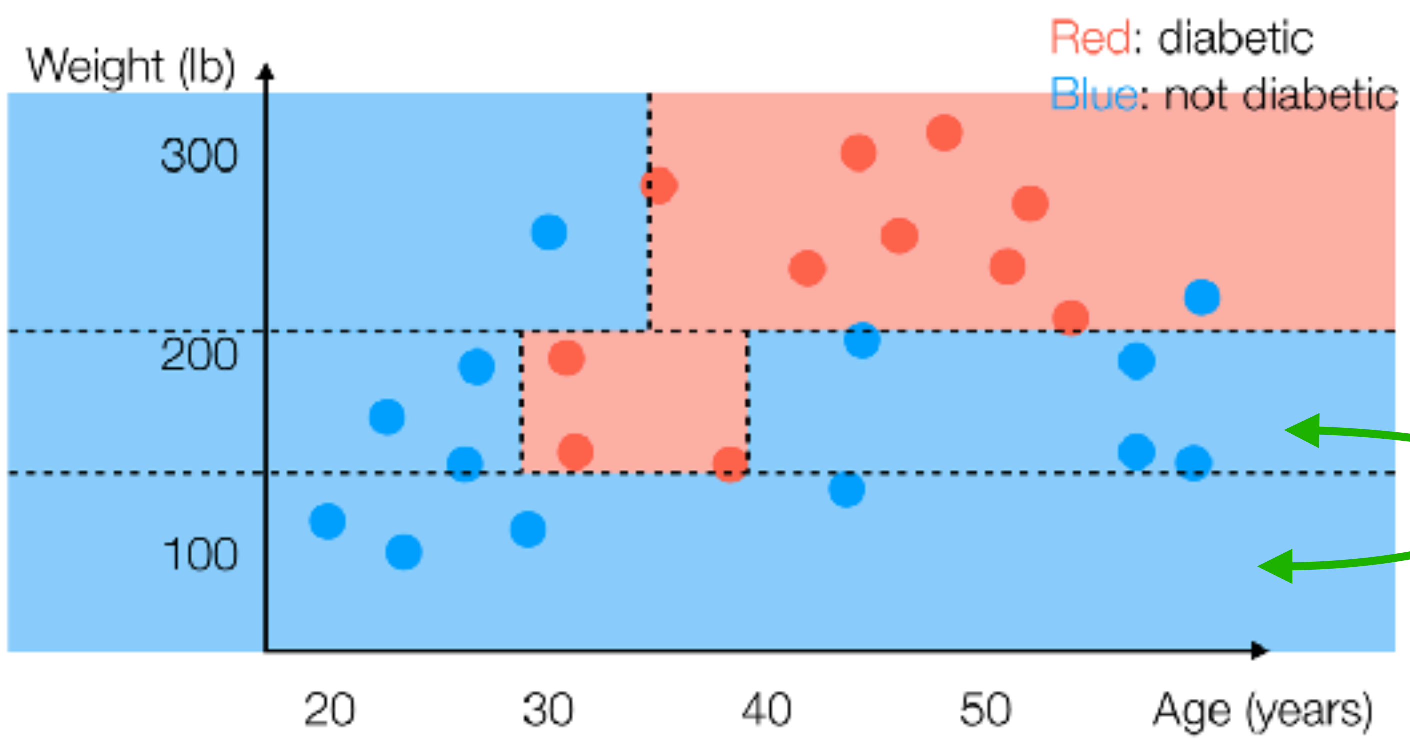
Feature space sliced up into leaf cells



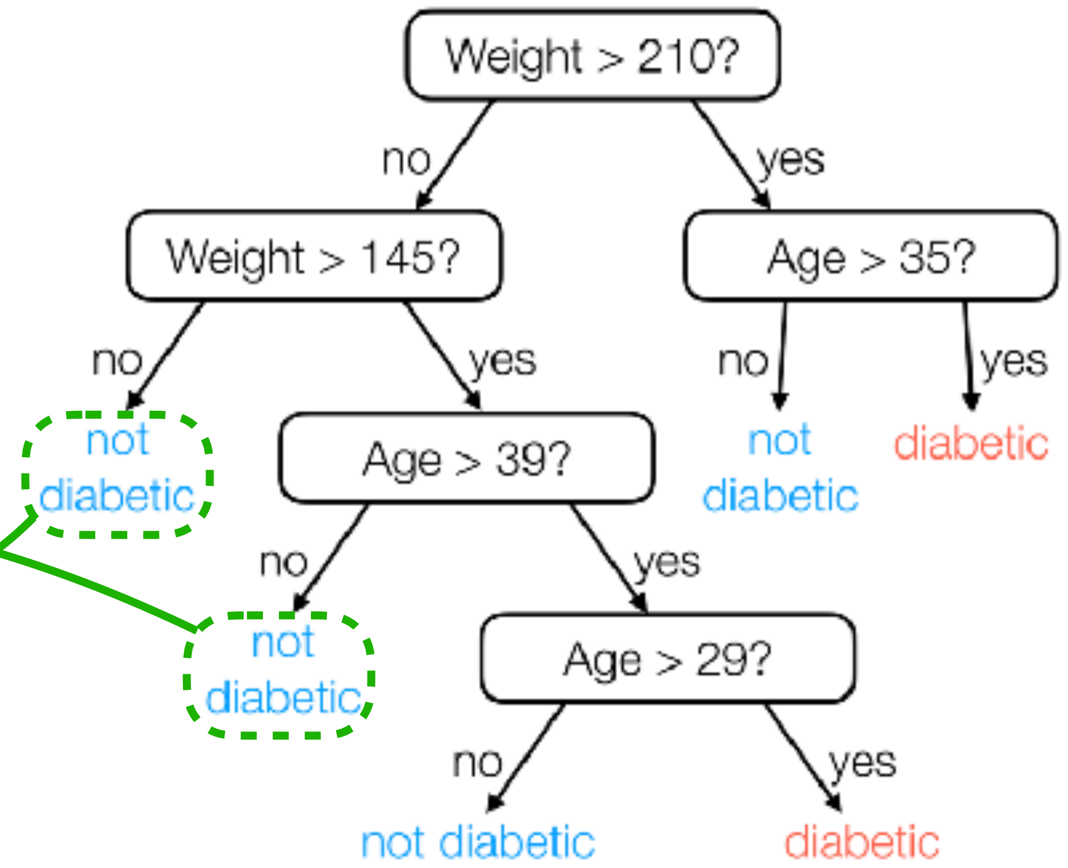
Decision Tree



Feature space sliced up into leaf cells

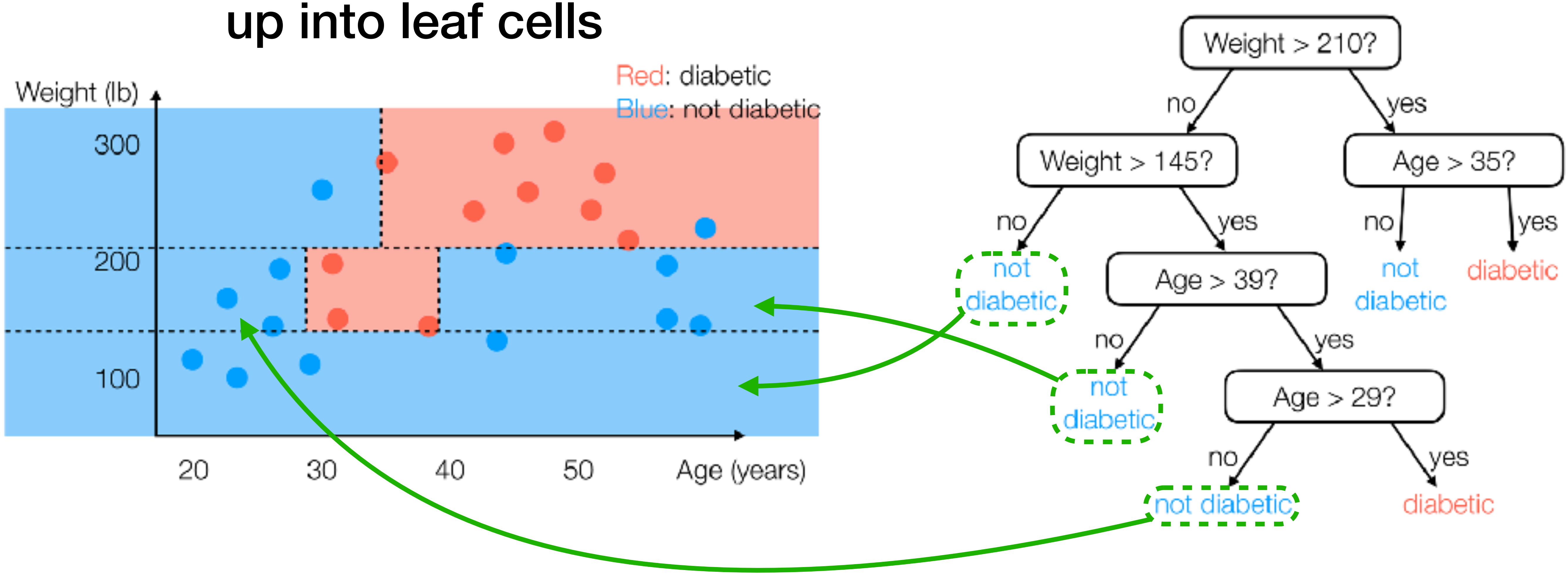


Decision Tree



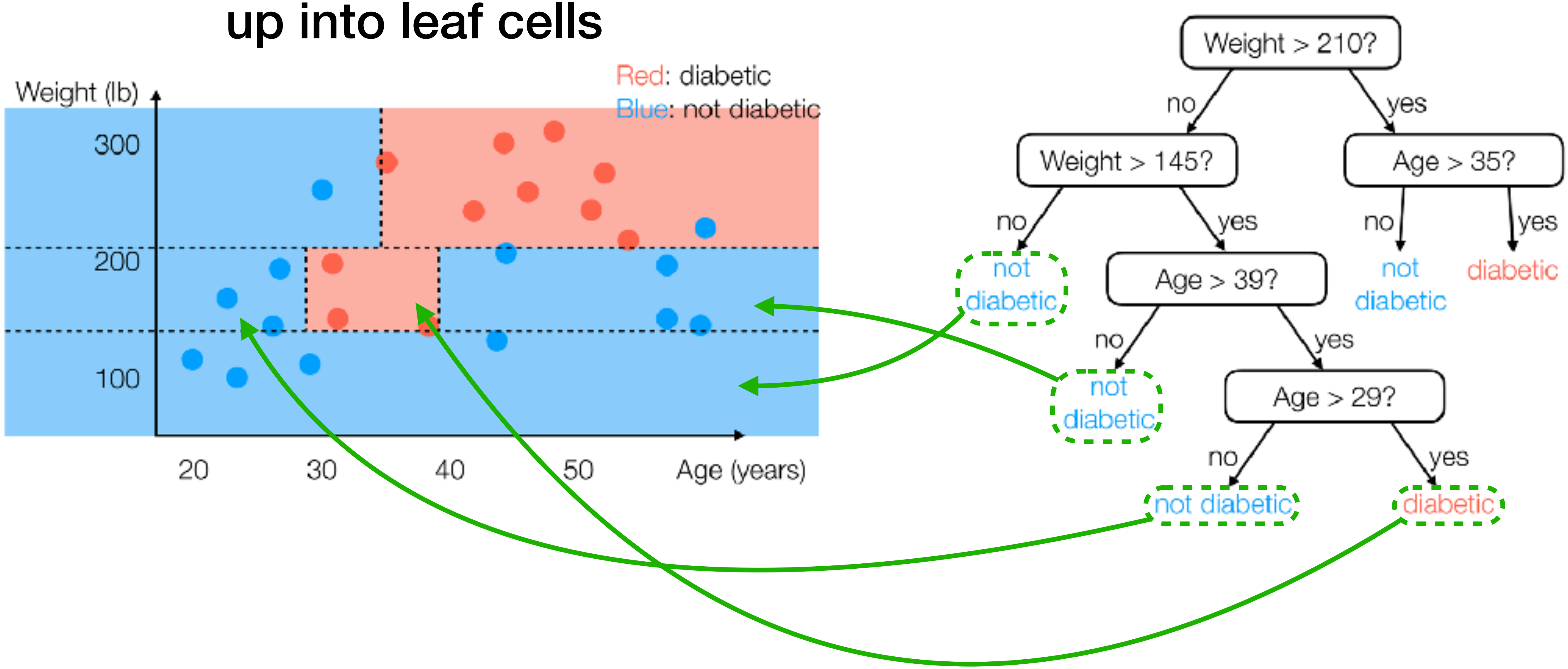
Feature space sliced up into leaf cells

Decision Tree



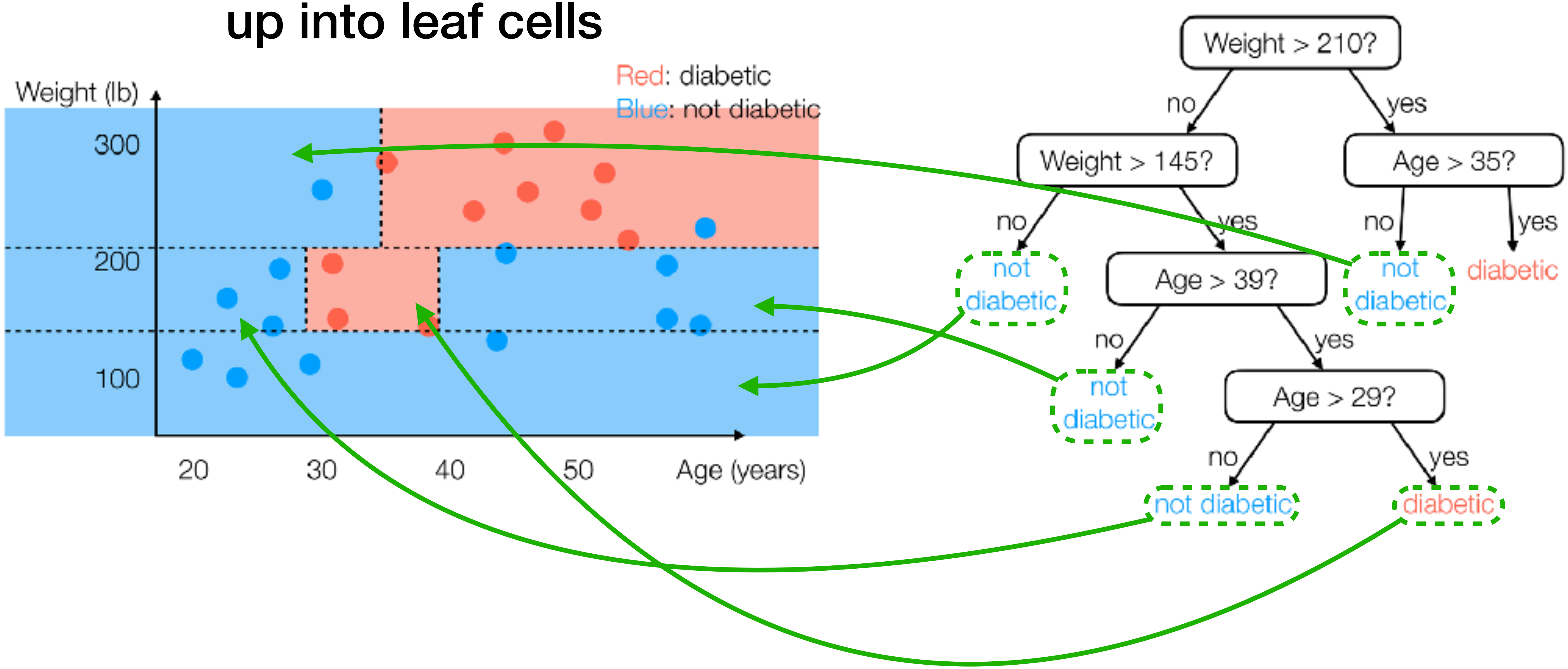
Feature space sliced up into leaf cells

Decision Tree



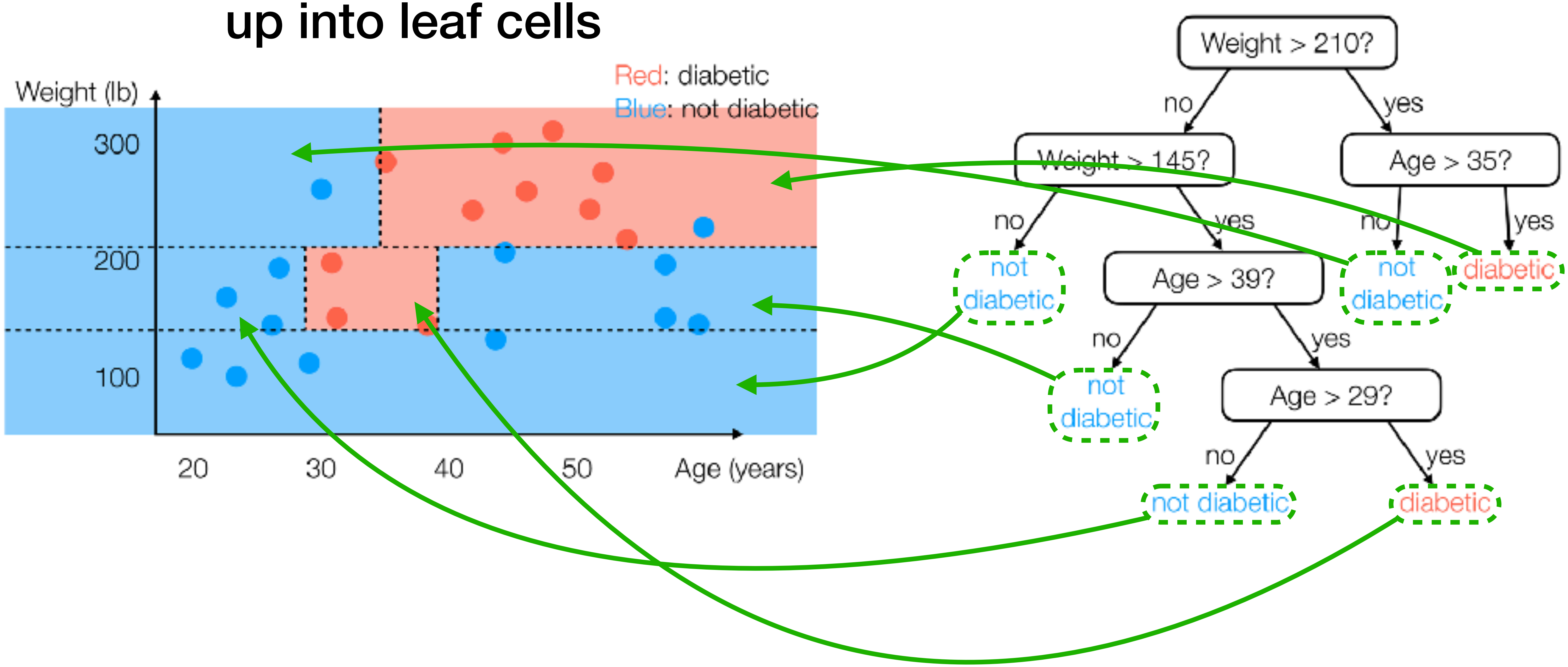
Feature space sliced up into leaf cells

Decision Tree



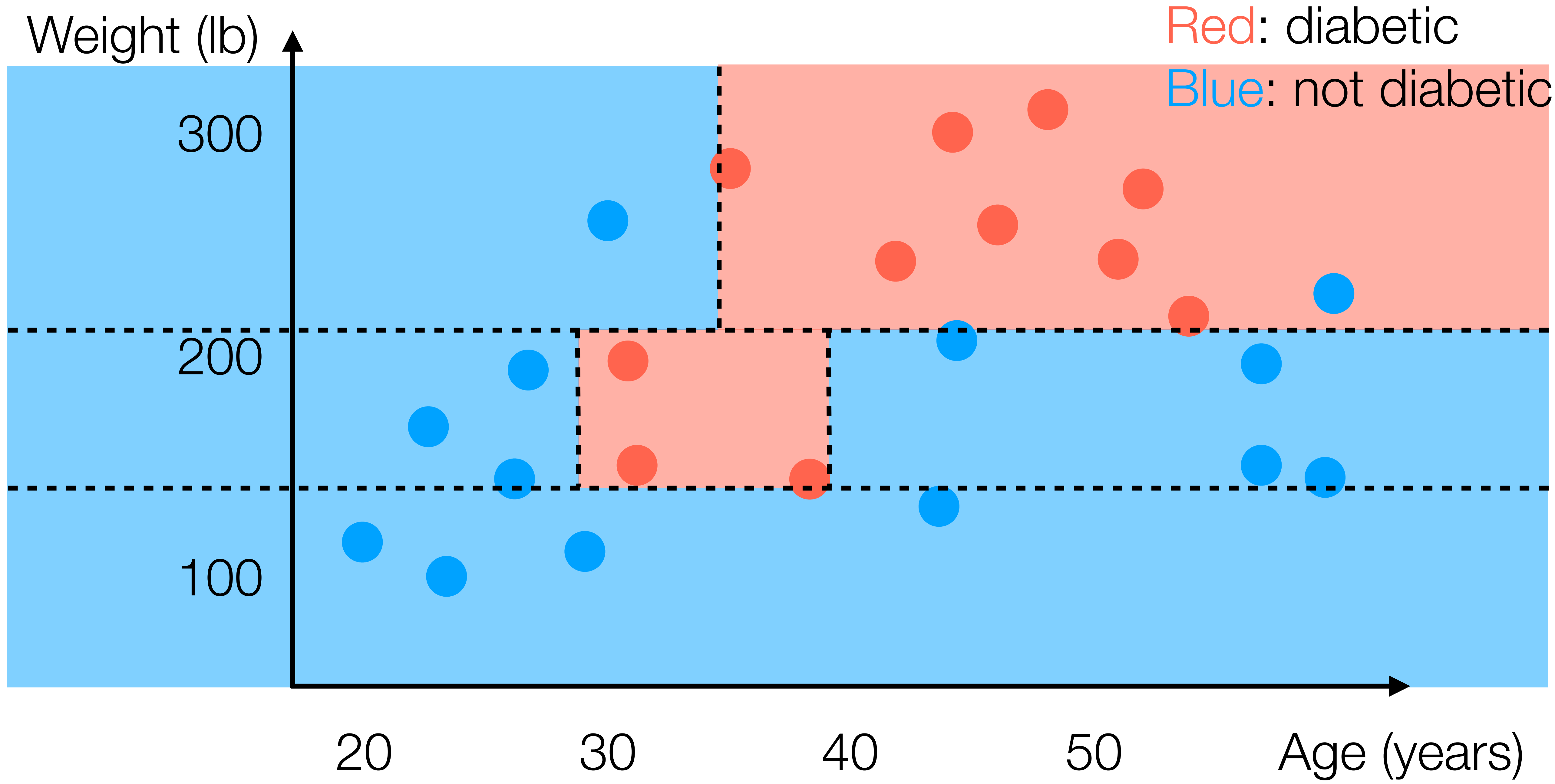
Feature space sliced up into leaf cells

Decision Tree



Nearest Neighbor Interpretation

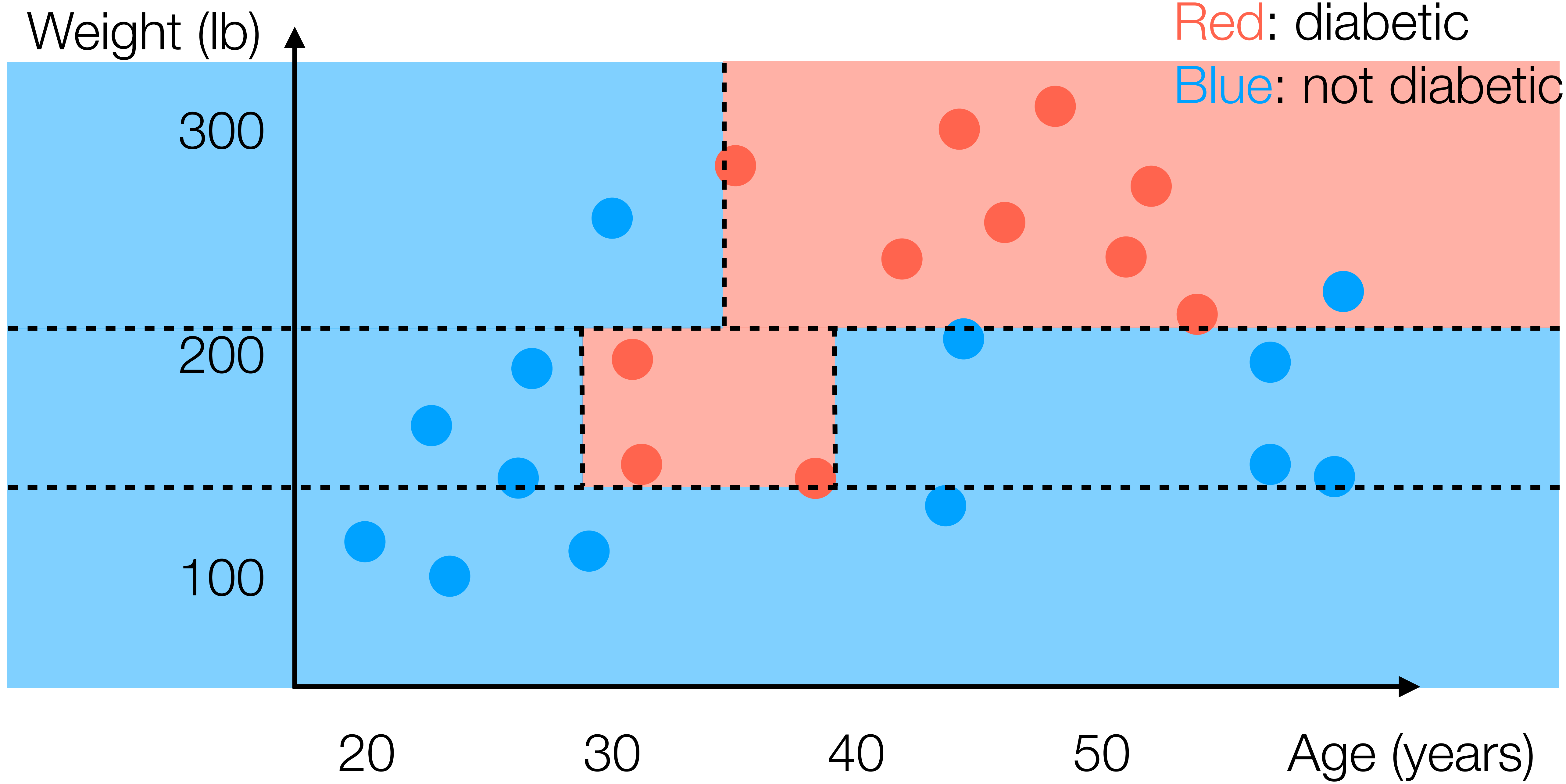
Note: Each training data point lands in one "leaf cell"



Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

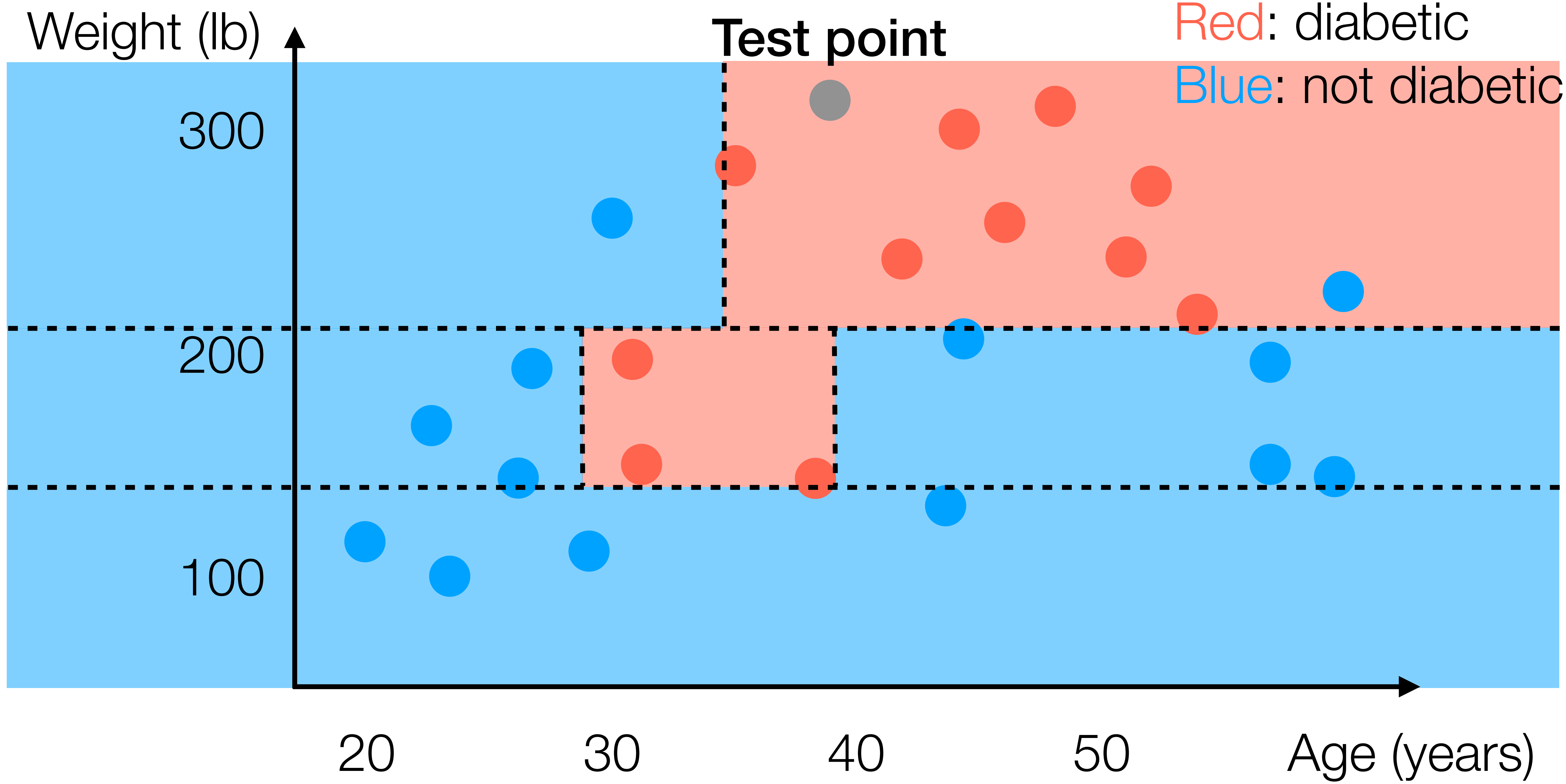
Also: Any test data point lands in one leaf cell



Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

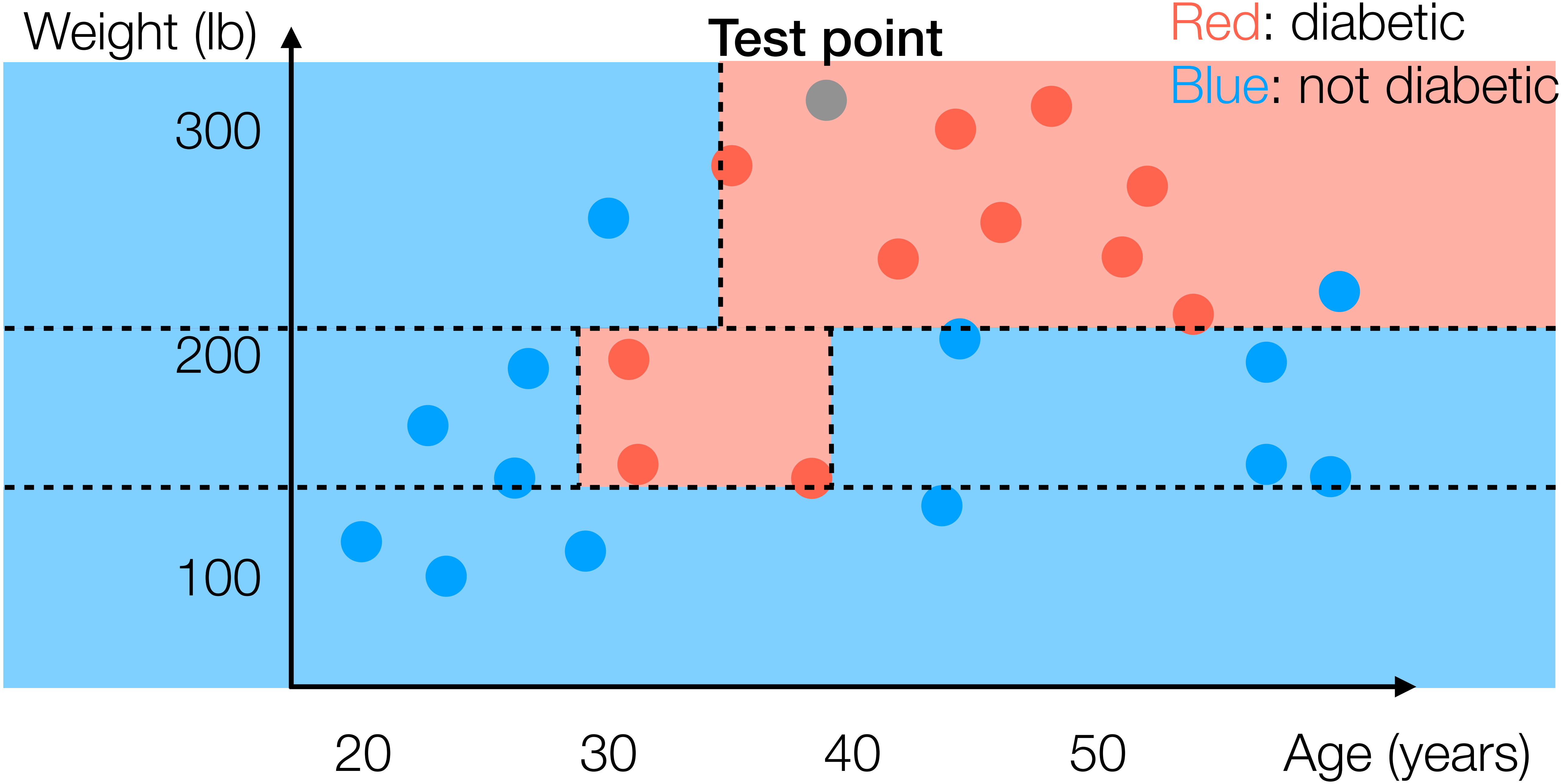
Also: Any test data point lands in one leaf cell



Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

Also: Any test data point lands in one leaf cell



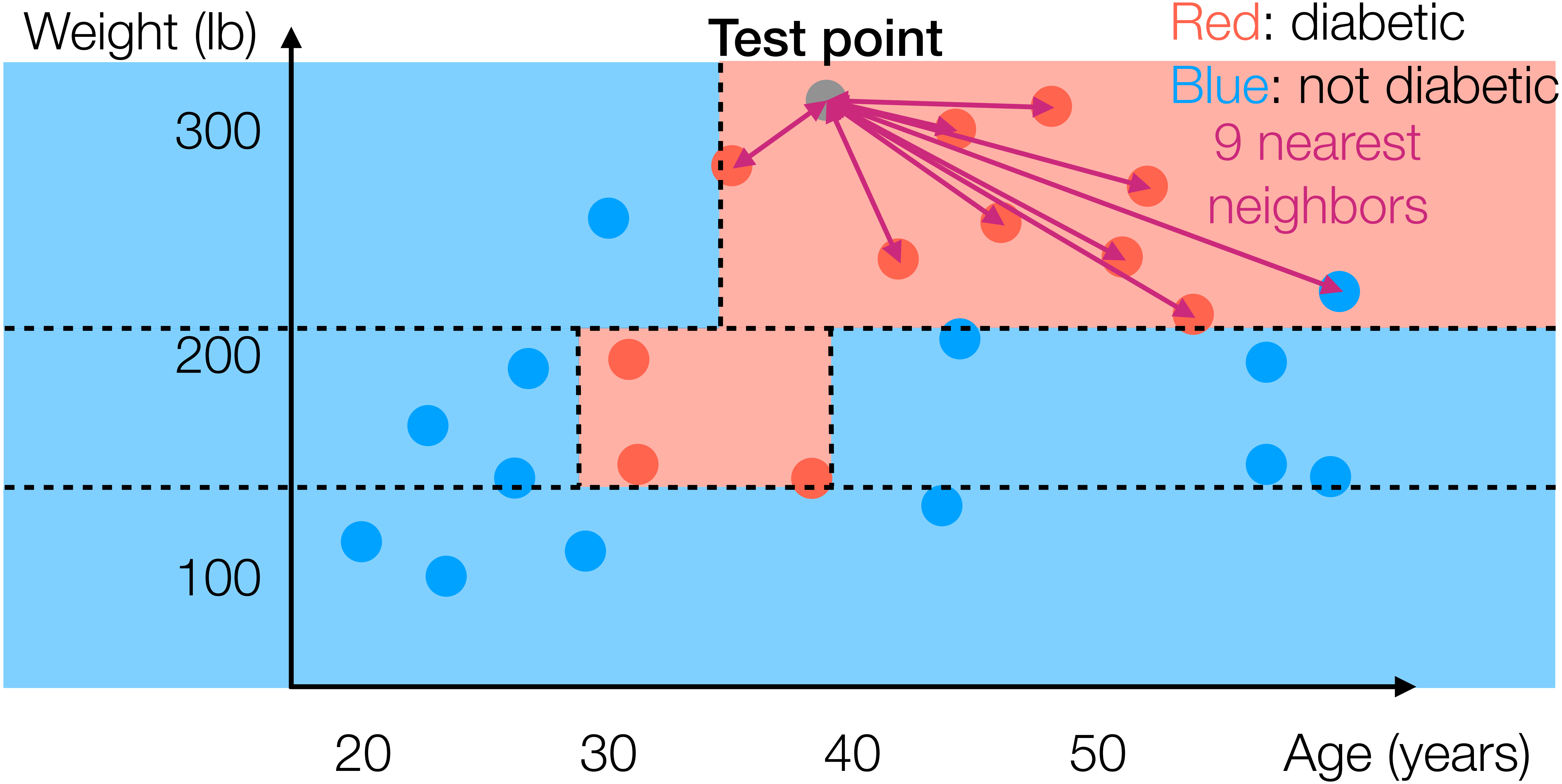
Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"

Also: Any test data point lands in one leaf cell



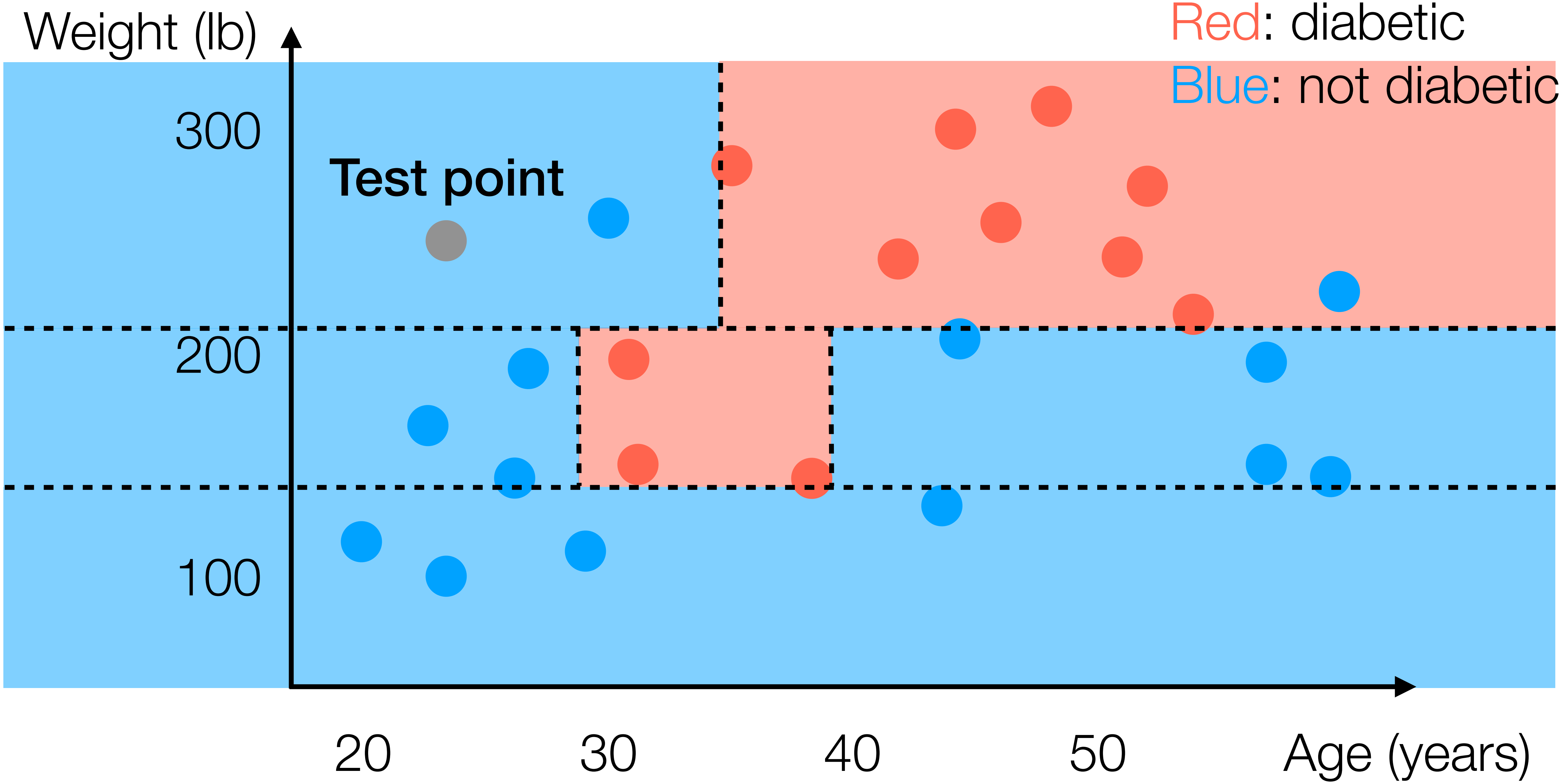
Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"

Also: Any test data point lands in one leaf cell



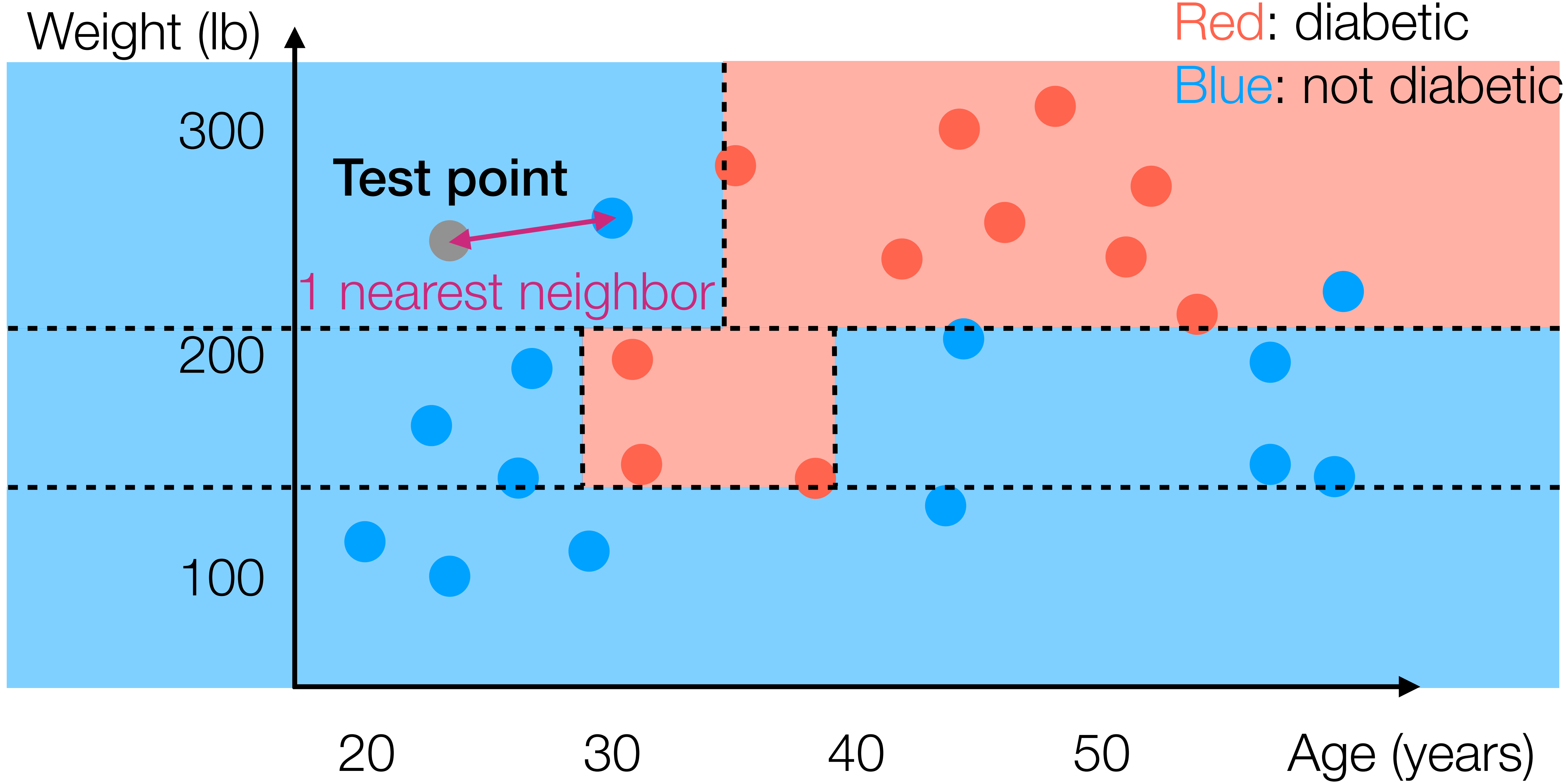
Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

Also: Any test data point lands in one leaf cell



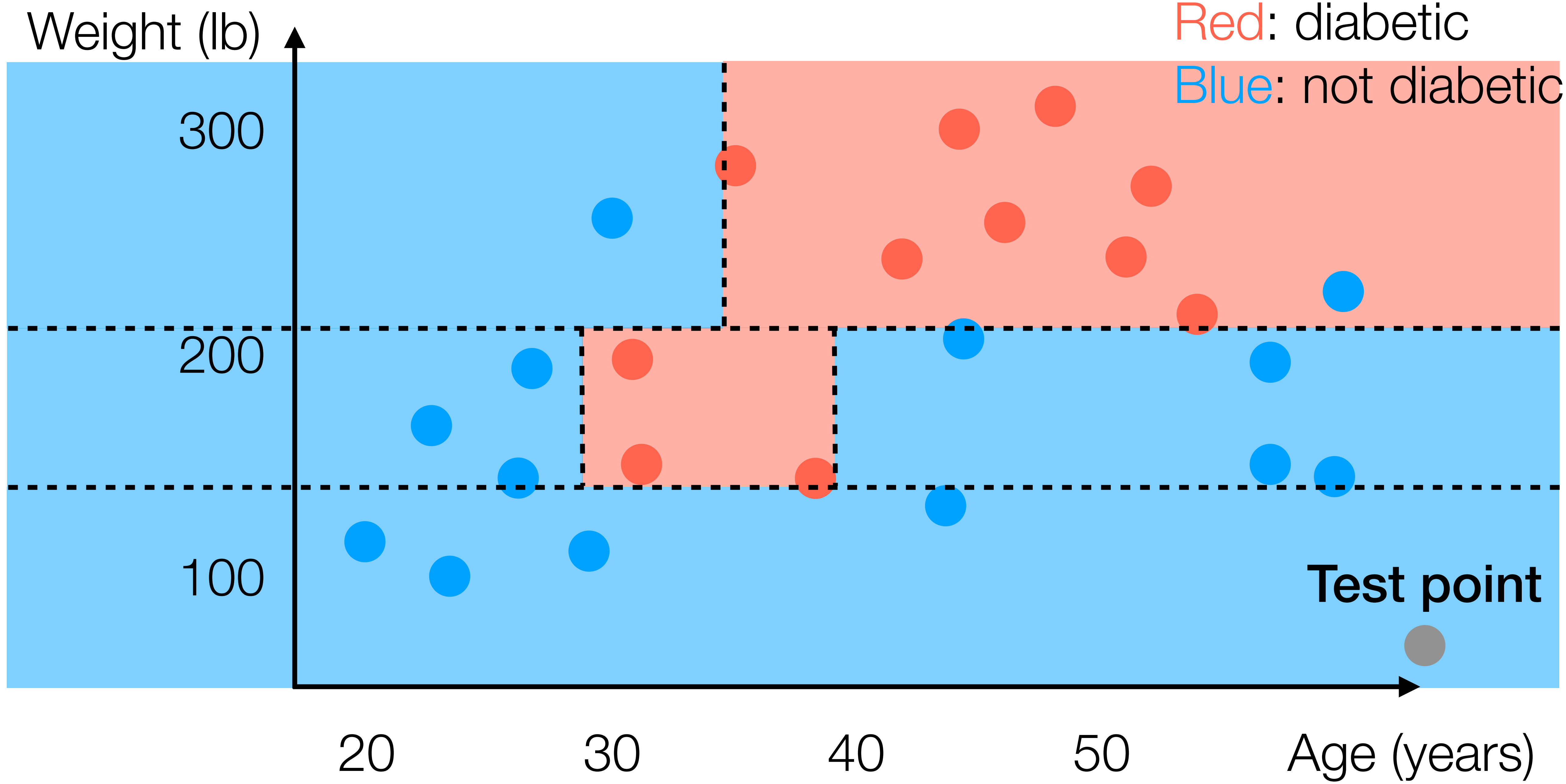
Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"

Also: Any test data point lands in one leaf cell



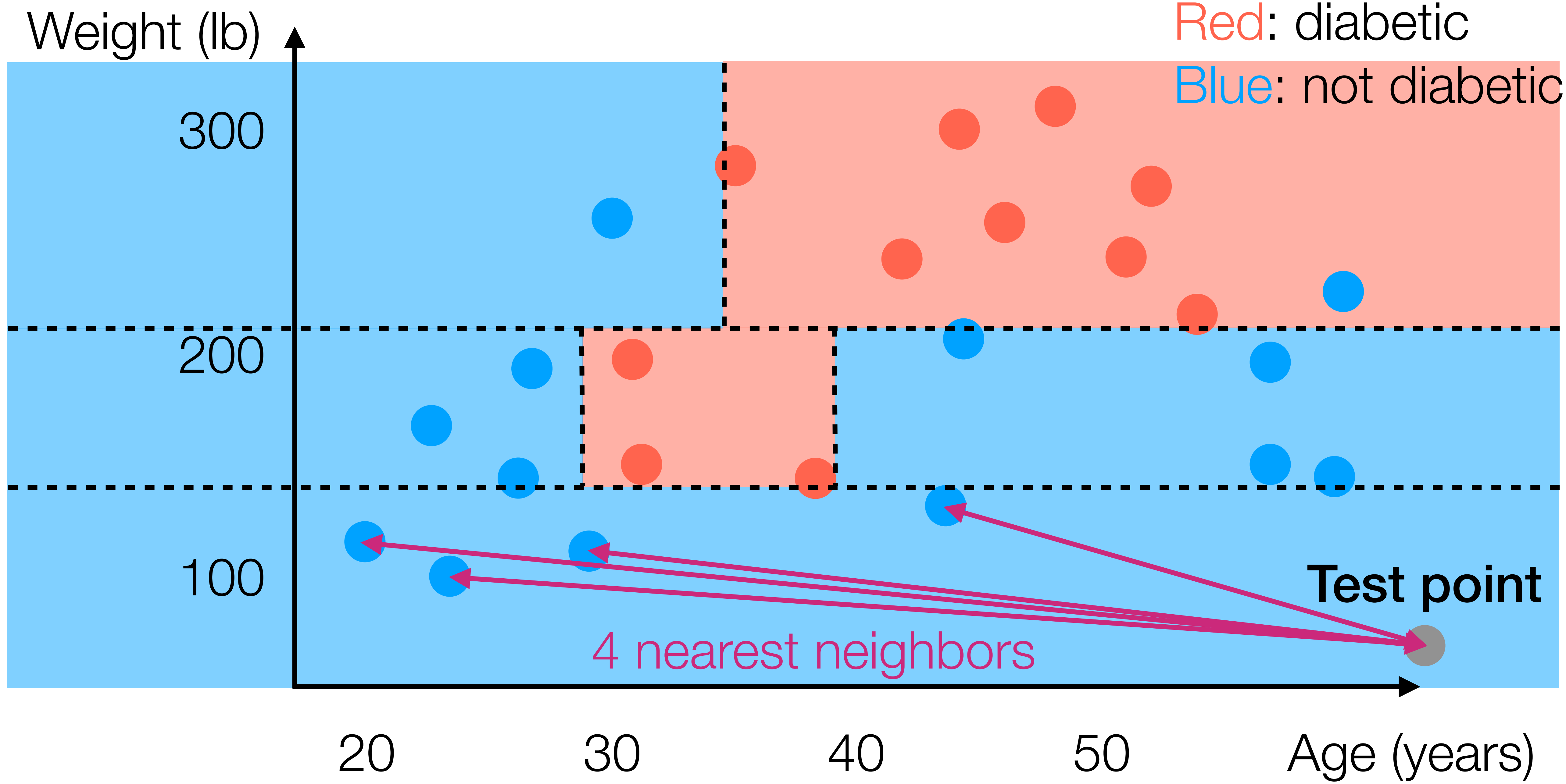
Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

Also: Any test data point lands in one leaf cell

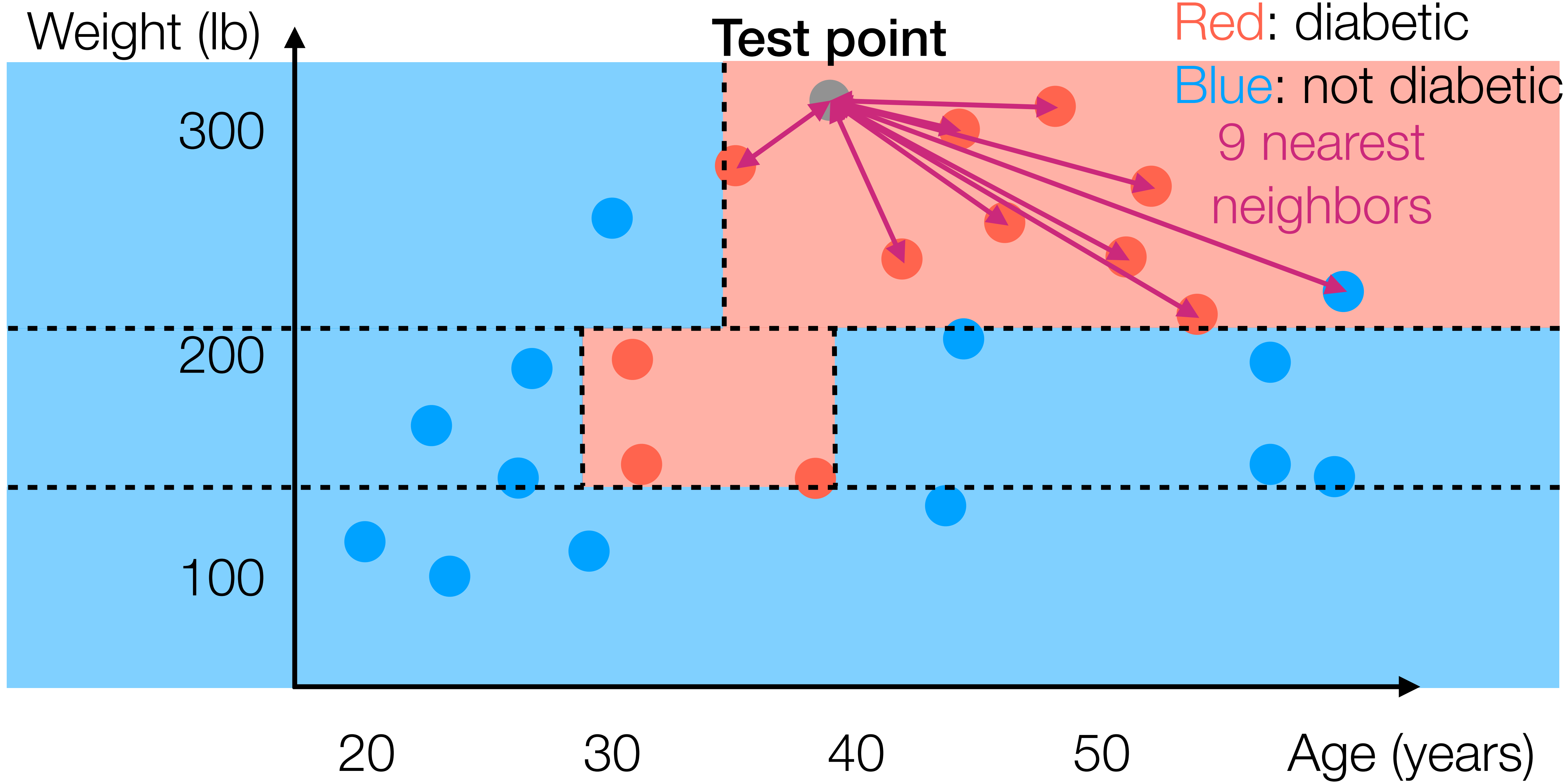


Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"
Also: Any test data point lands in one leaf cell

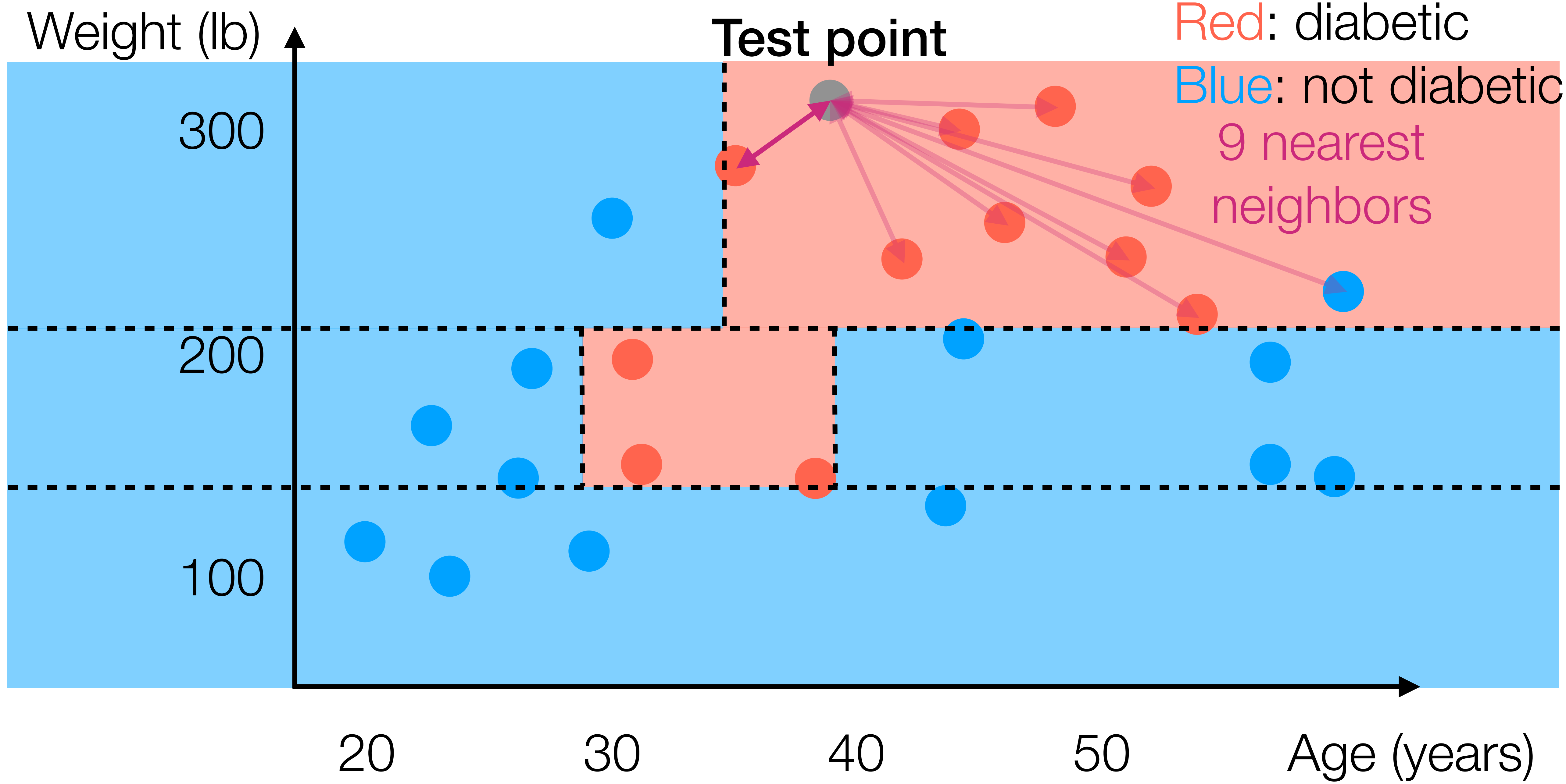


Prediction for test point: majority vote of training points in same leaf cell
(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"

Also: Any test data point lands in one leaf cell

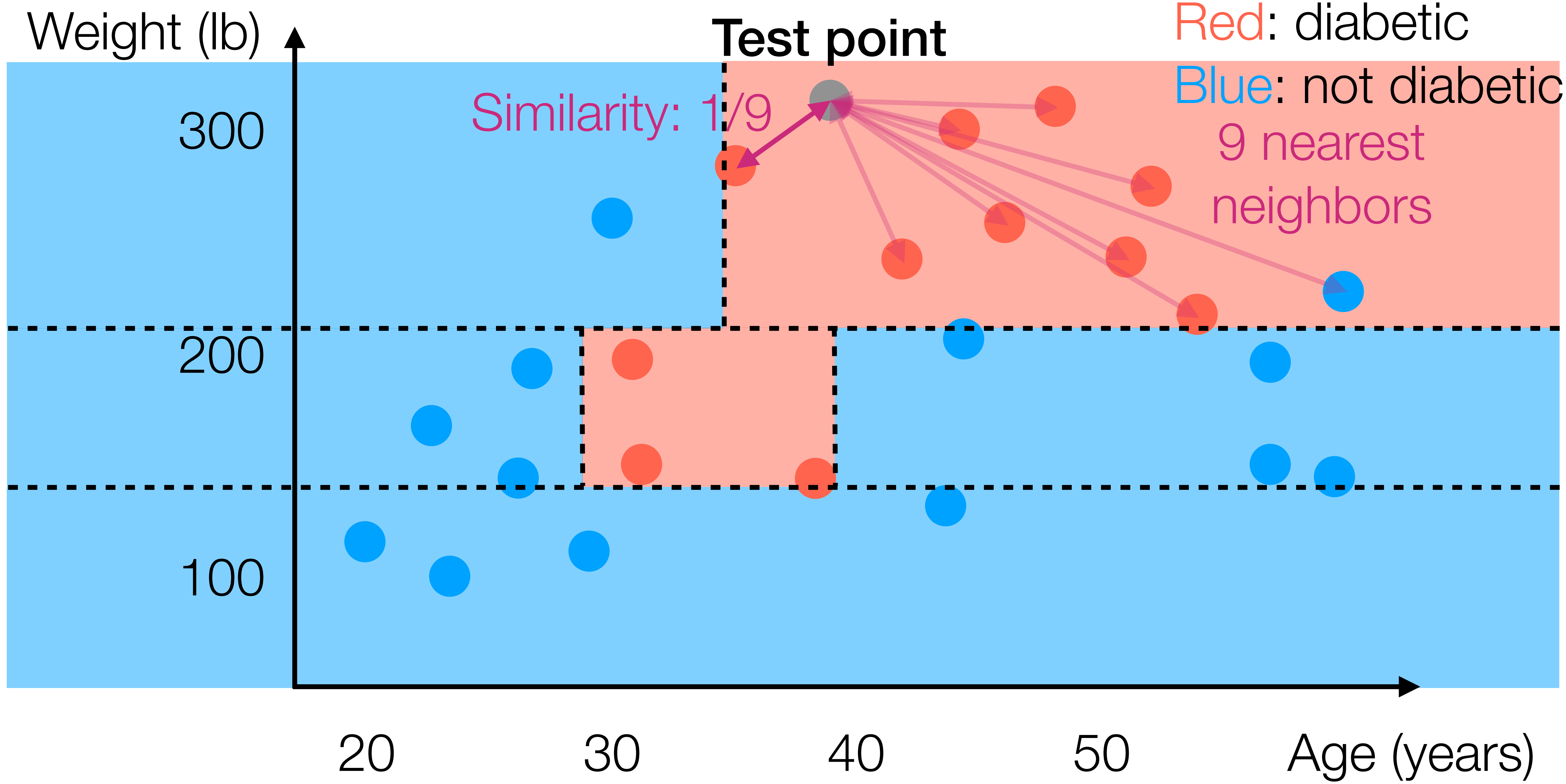


Prediction for test point: majority vote of training points in same leaf cell
(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

Also: Any test data point lands in one leaf cell



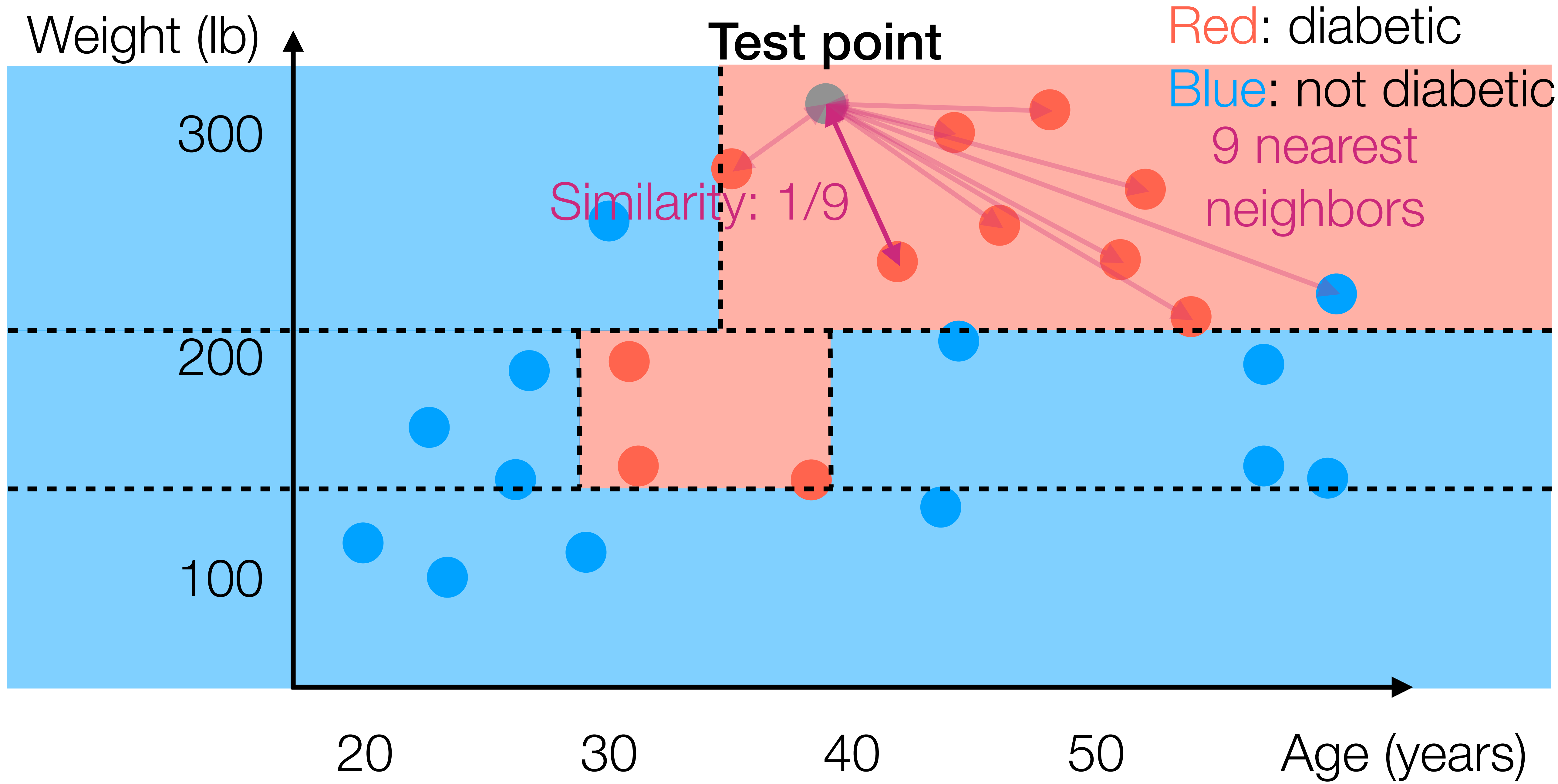
Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

Also: Any test data point lands in one leaf cell

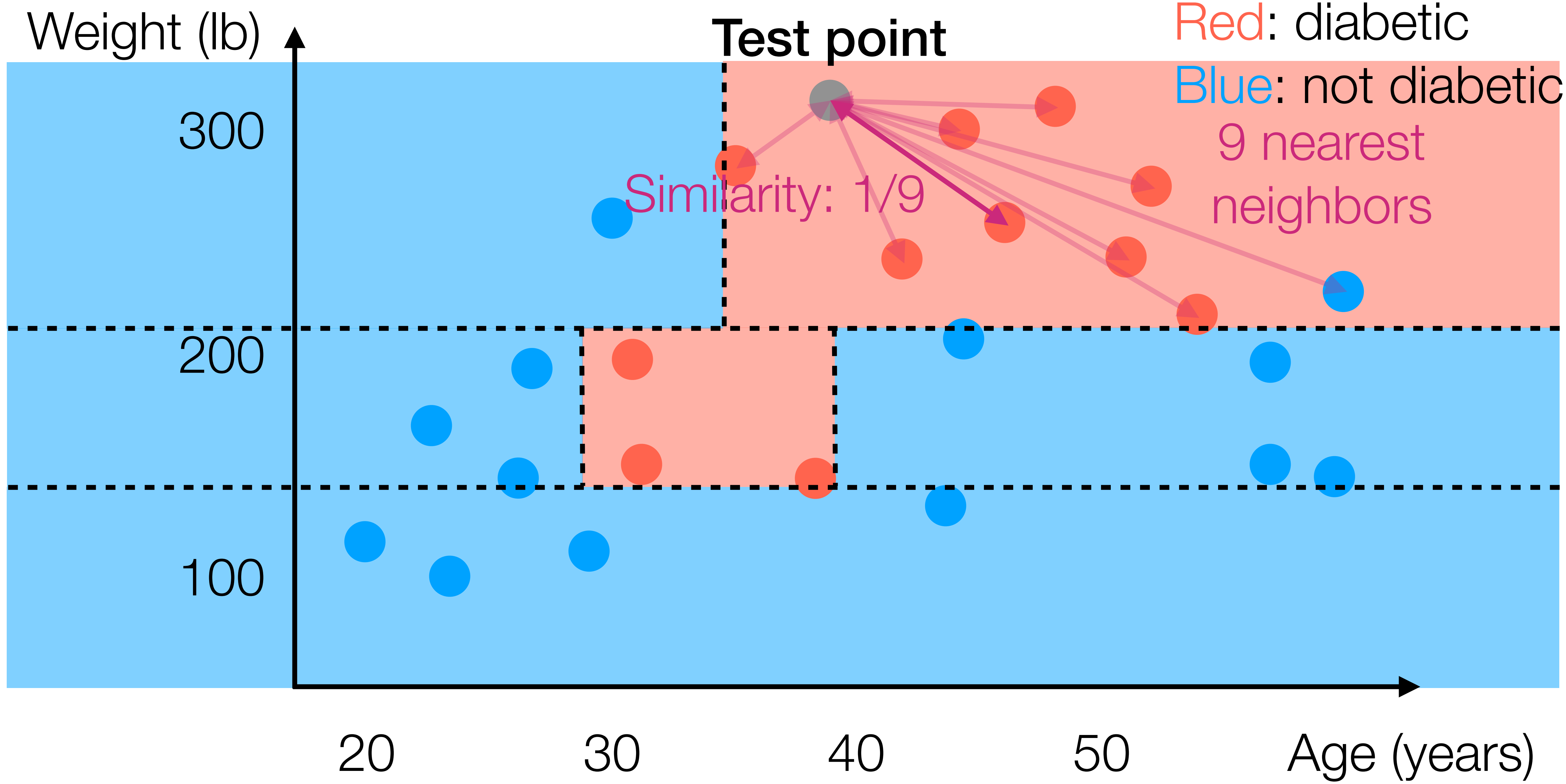


Prediction for test point: majority vote of training points in same leaf cell
(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"

Also: Any test data point lands in one leaf cell

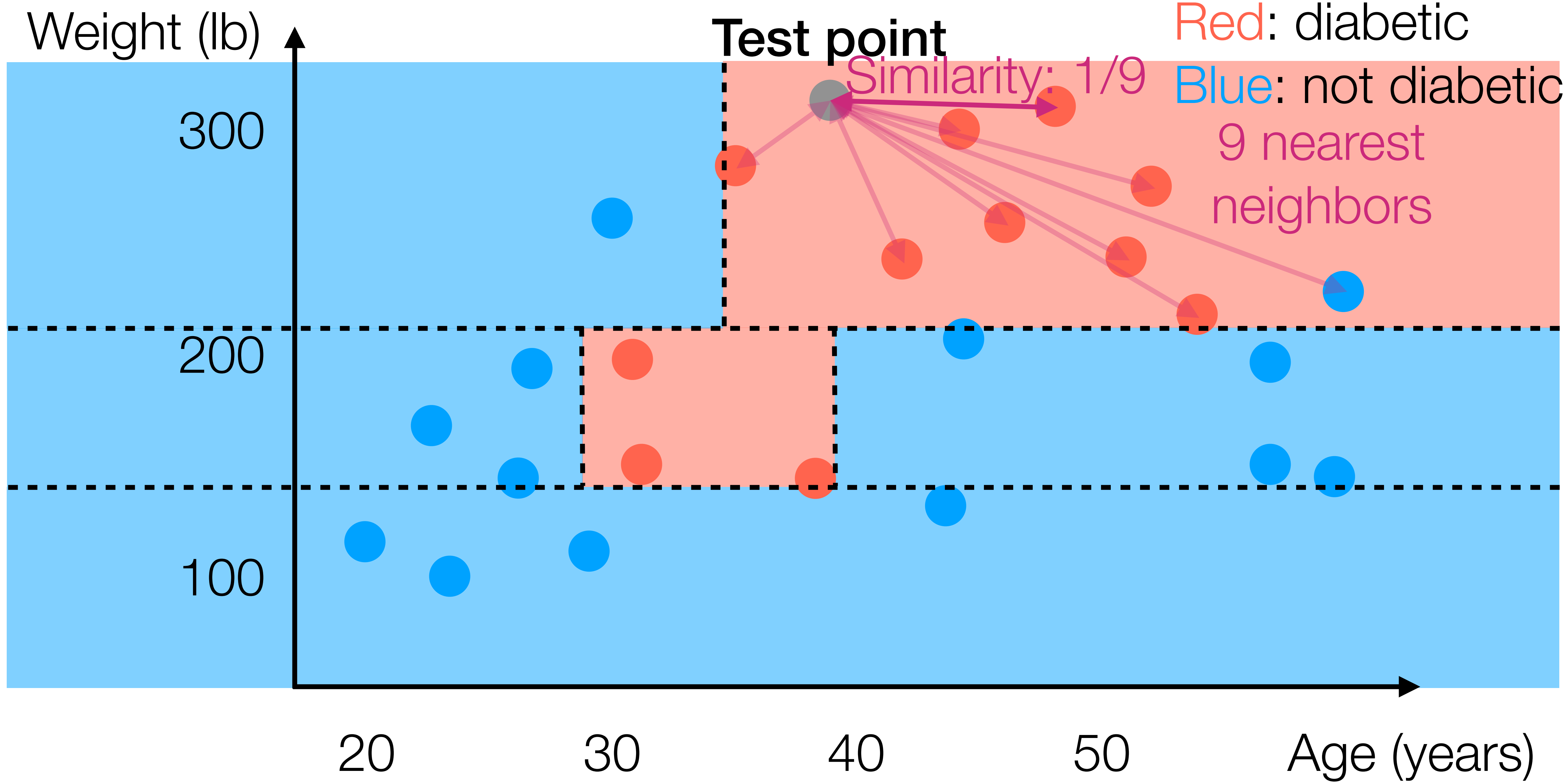


Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"
Also: Any test data point lands in one leaf cell

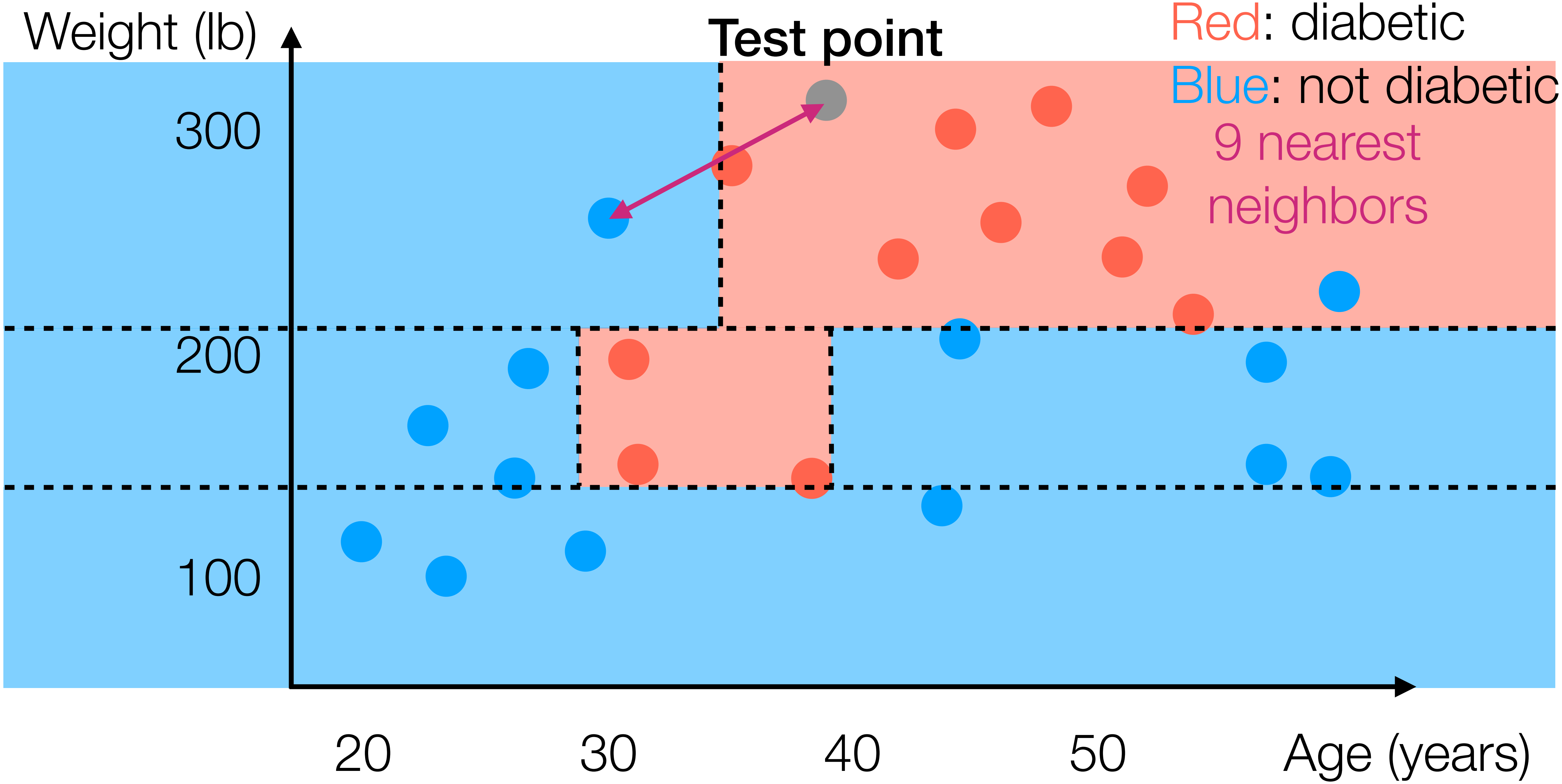


Prediction for test point: majority vote of training points in same leaf cell
(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

Also: Any test data point lands in one leaf cell

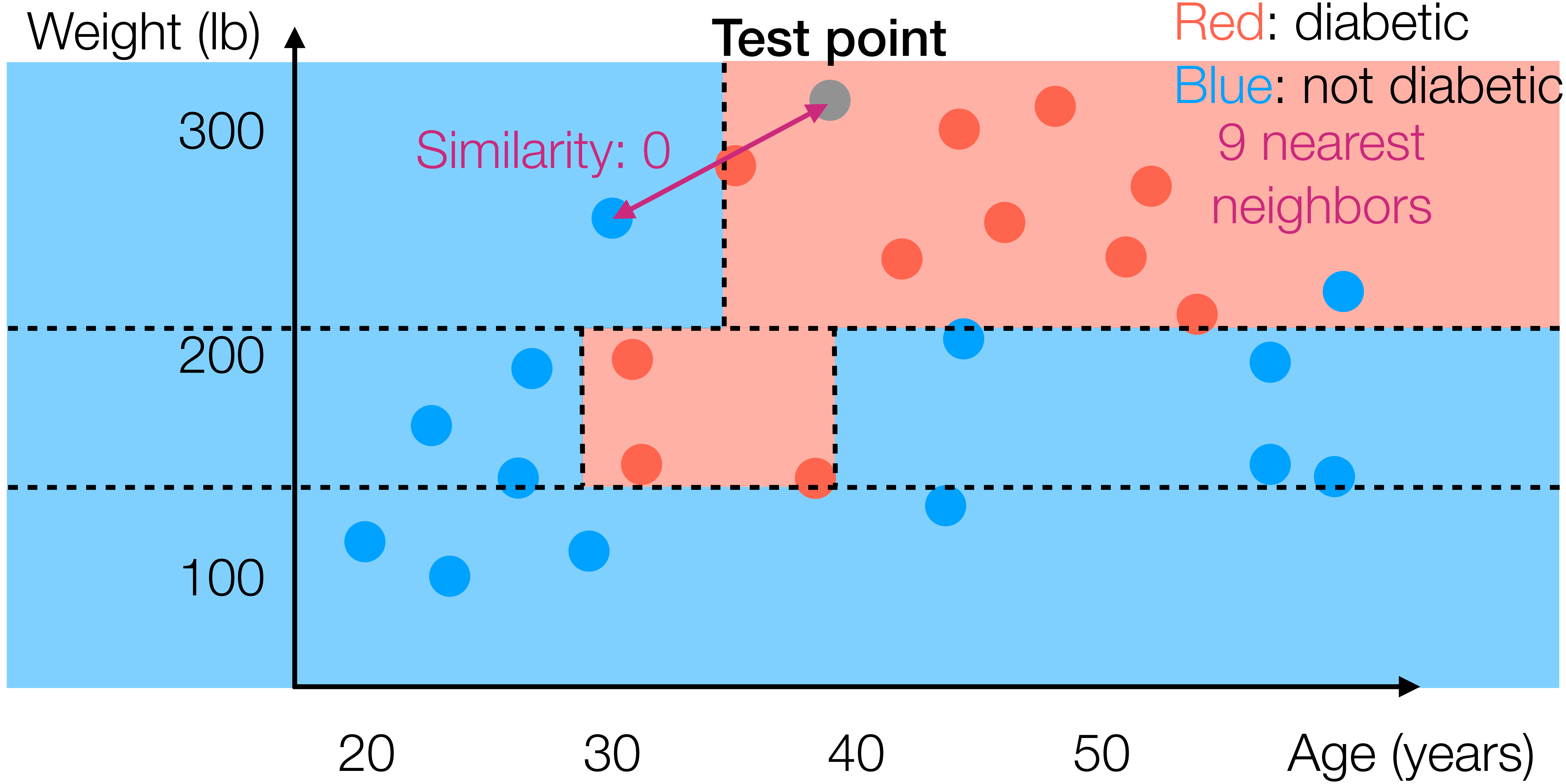


Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

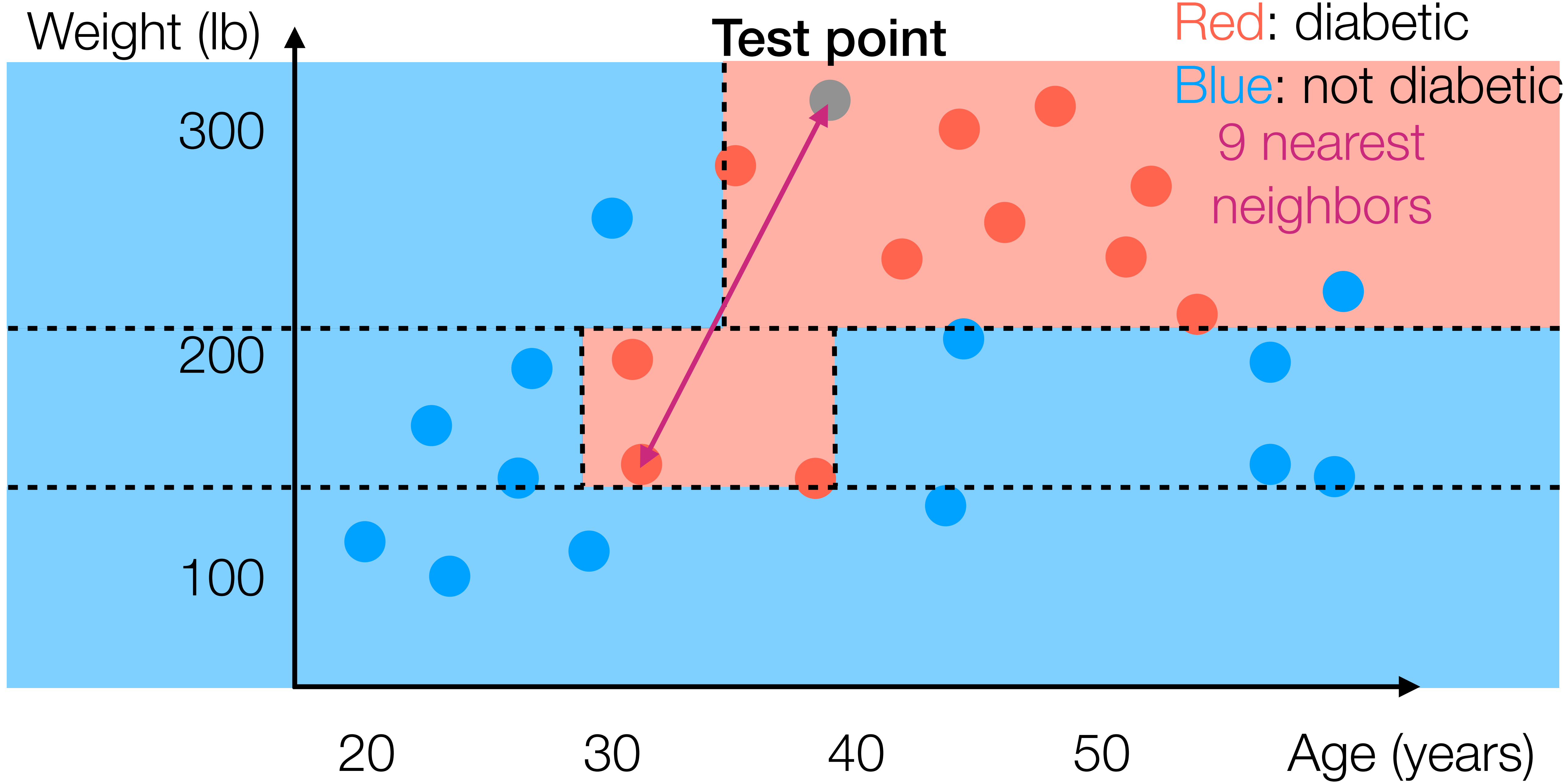
Note: Each training data point lands in one "leaf cell"
Also: Any test data point lands in one leaf cell



Prediction for test point: majority vote of training points in same leaf cell
(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”
Also: Any test data point lands in one leaf cell

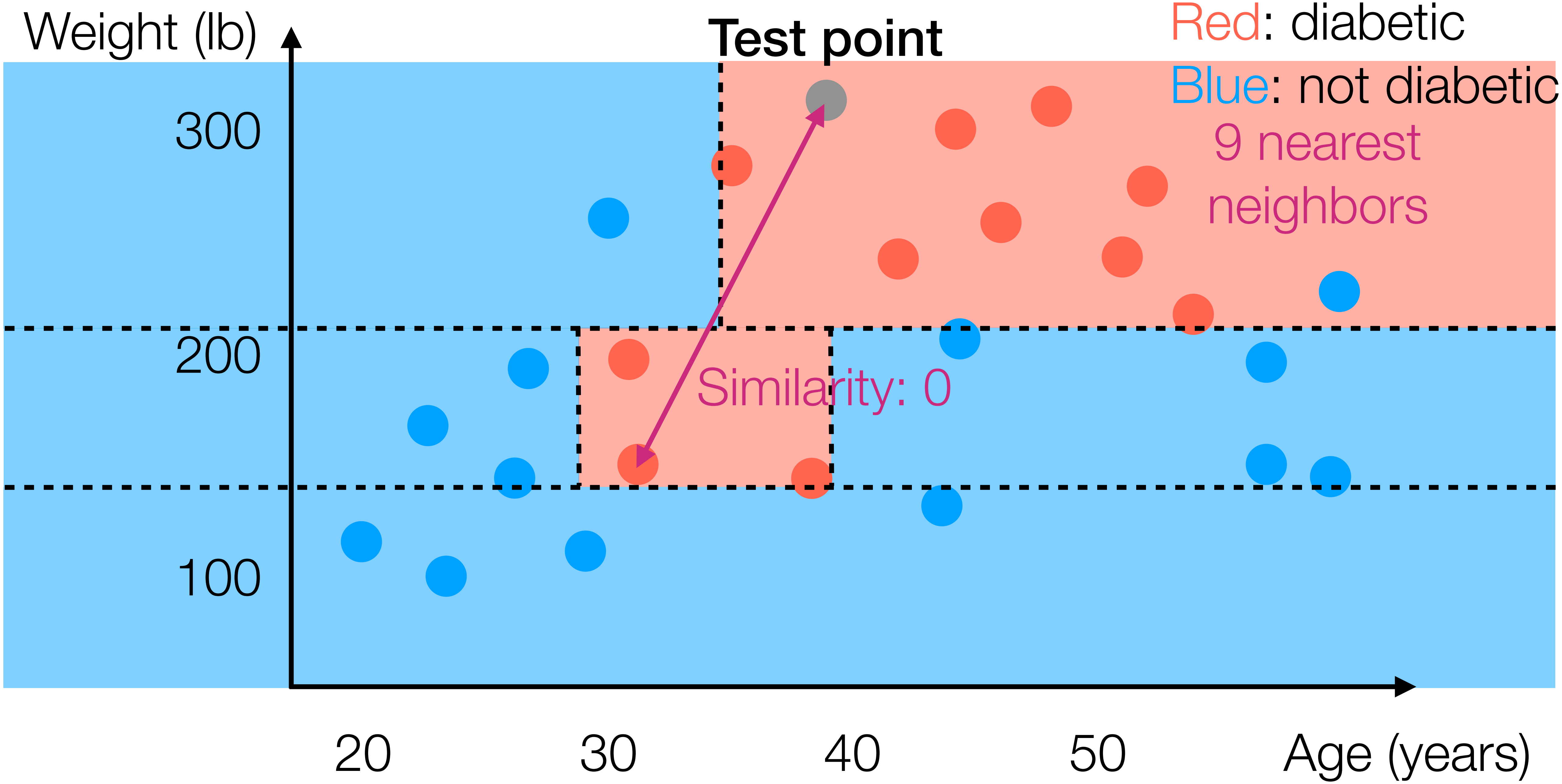


Prediction for test point: majority vote of training points in same leaf cell
(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"

Also: Any test data point lands in one leaf cell

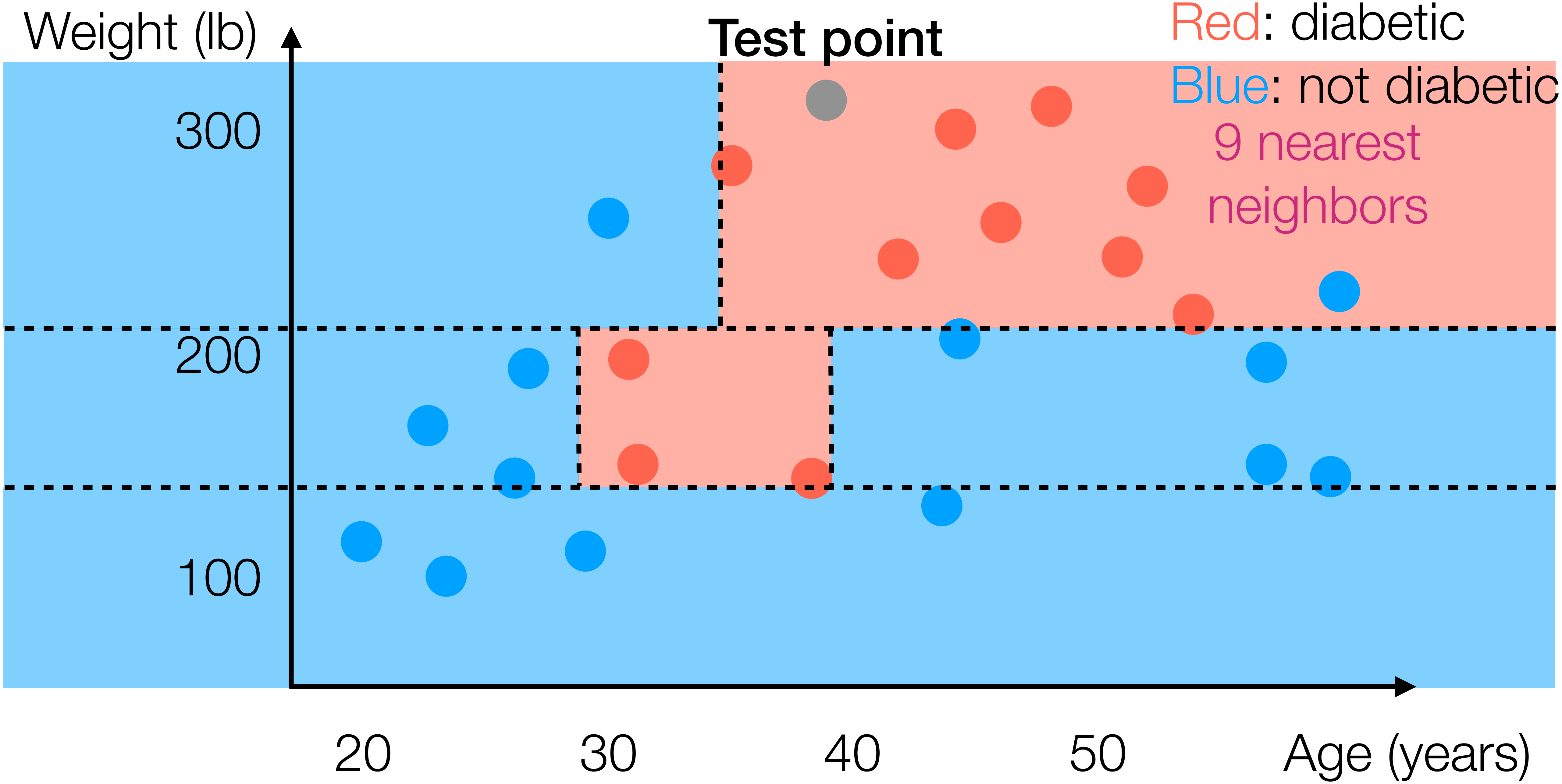


Prediction for test point: majority vote of training points in same leaf cell

(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

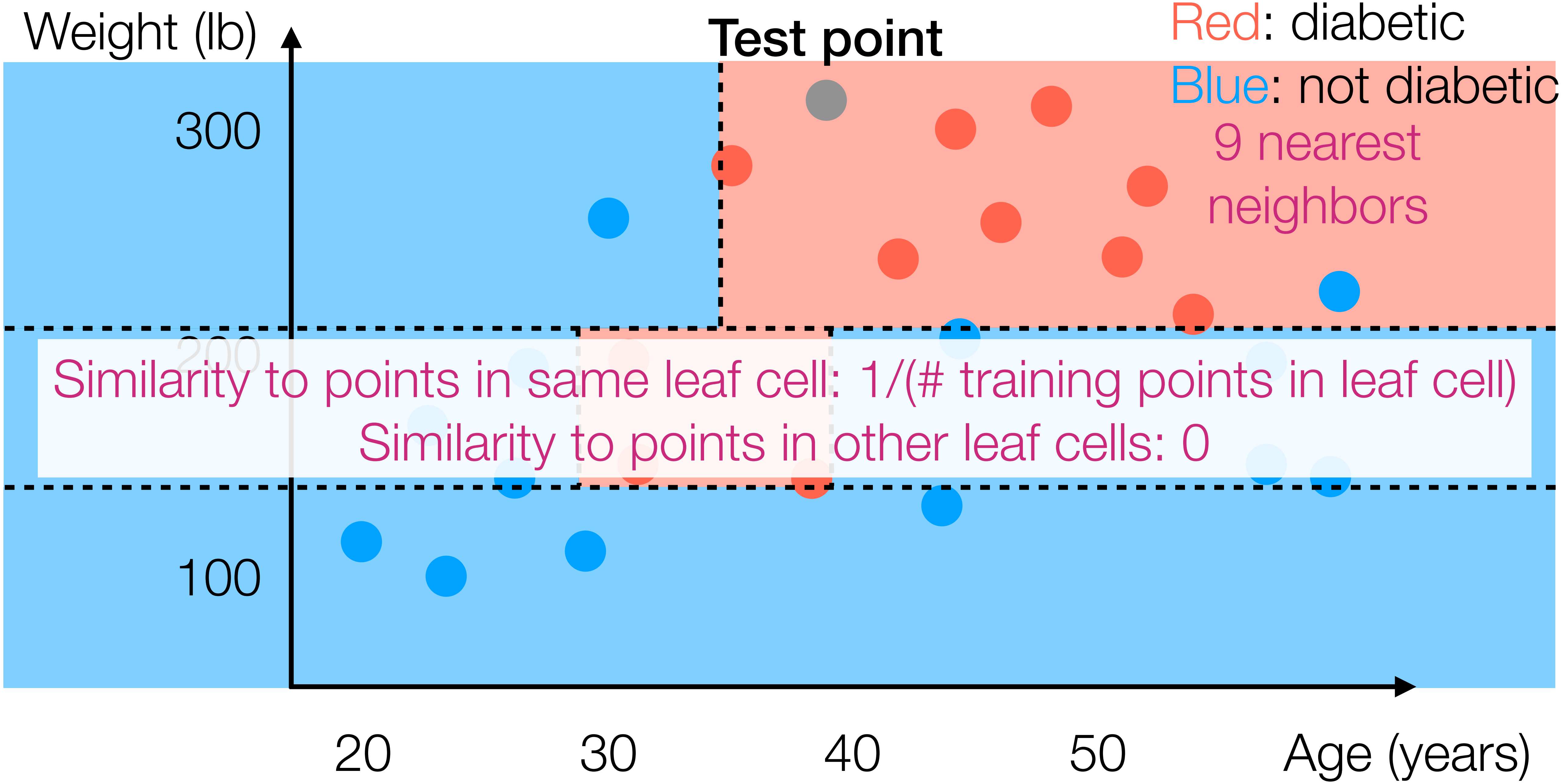
Note: Each training data point lands in one "leaf cell"
Also: Any test data point lands in one leaf cell



Prediction for test point: majority vote of training points in same leaf cell
(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”
Also: Any test data point lands in one leaf cell

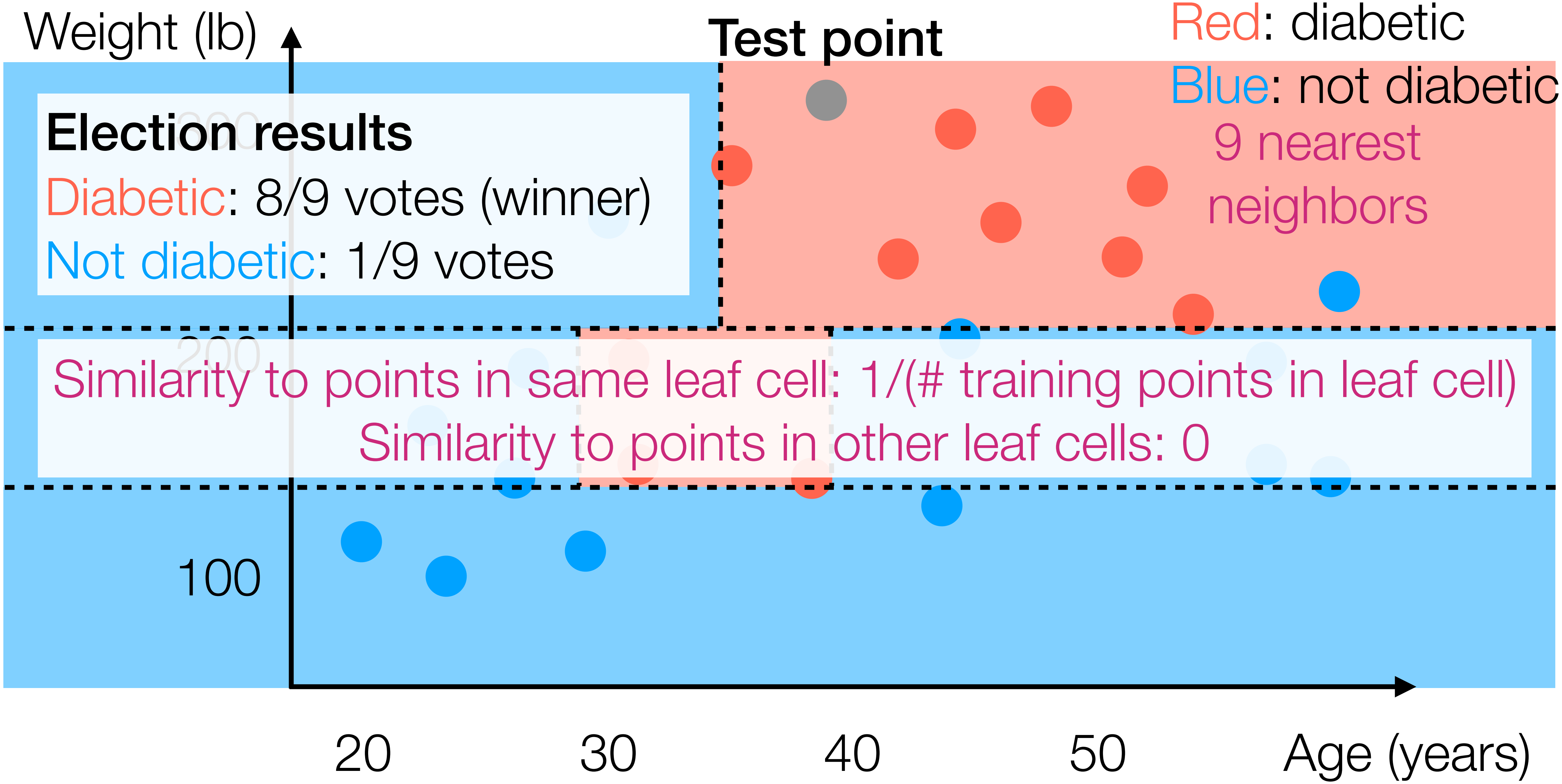


Prediction for test point: majority vote of training points in same leaf cell
(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one "leaf cell"

Also: Any test data point lands in one leaf cell

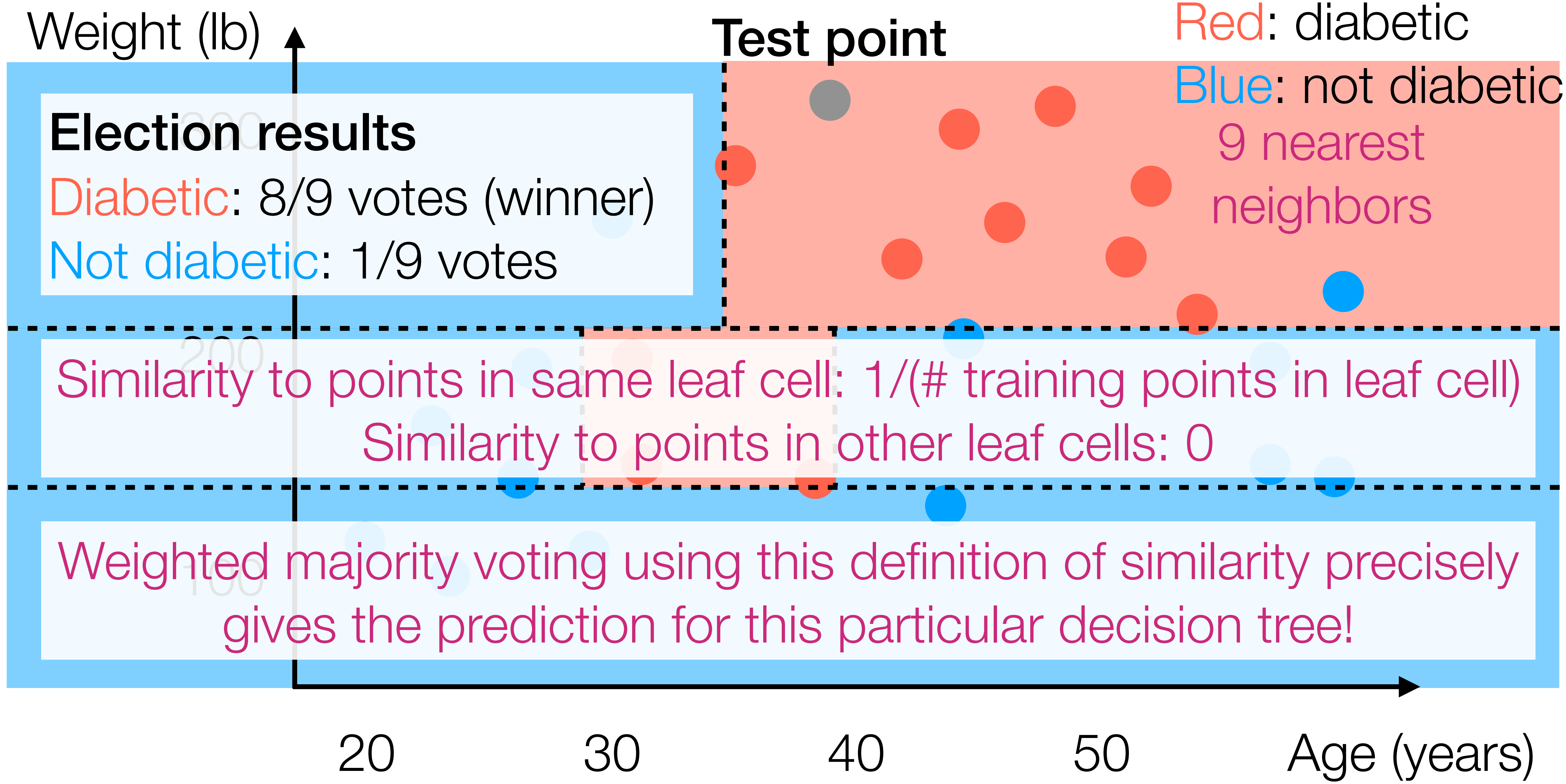


Prediction for test point: majority vote of training points in same leaf cell
(these training points act as nearest neighbors to the test point!)

Nearest Neighbor Interpretation

Note: Each training data point lands in one “leaf cell”

Also: Any test data point lands in one leaf cell



Prediction for test point: majority vote of training points in same leaf cell (these training points act as nearest neighbors to the test point!)

Decision Tree for Classification

Decision Tree for Classification

- Many ways to learn (some popular ways: CART, C4.5)

Decision Tree for Classification

- Many ways to learn (some popular ways: CART, C4.5)
- Extremely easy to interpret and to do prediction

Decision Tree for Classification

- Many ways to learn (some popular ways: CART, C4.5)
- Extremely easy to interpret and to do prediction
- Nearest neighbor interpretation:

Decision Tree for Classification

- Many ways to learn (some popular ways: CART, C4.5)
- Extremely easy to interpret and to do prediction
- Nearest neighbor interpretation:
  - For each test point, look at leaf cell it falls into to find its nearest neighbors among the training data
    (note: # of nearest neighbors varies!)

Decision Tree for Classification

- Many ways to learn (some popular ways: CART, C4.5)
- Extremely easy to interpret and to do prediction
- Nearest neighbor interpretation:
    - For each test point, look at leaf cell it falls into to find its nearest neighbors among the training data
      (note: # of nearest neighbors varies!)
    - Prediction for test point: majority vote of nearest neighbors' labels

Decision Tree for Classification

- Many ways to learn (some popular ways: CART, C4.5)
- Extremely easy to interpret and to do prediction
- Nearest neighbor interpretation:
  - For each test point, look at leaf cell it falls into to find its nearest neighbors among the training data
(note: # of nearest neighbors varies!)
  - Prediction for test point: majority vote of nearest neighbors' labels
- Learning a decision tree learns a similarity function (that depends on labels)

Decision Tree for ~~Classification~~ Regression

- Many ways to learn (some popular ways: CART, C4.5)
- Extremely easy to interpret and to do prediction
- Nearest neighbor interpretation:
  - For each test point, look at leaf cell it falls into to find its nearest neighbors among the training data
    (note: # of nearest neighbors varies!)
  - Prediction for test point: ~~majority vote~~ average of nearest neighbors' labels
- Learning a decision tree learns a similarity function (that depends on labels)

Decision Forest for Classification

Decision Forest for Classification

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

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!

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

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

Tree 1 Tree 2 Tree 3 $\cdots$ Tree T

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

Learn each tree separately using same training data

Tree 1 | Tree 2 | Tree 3 | $\cdots$ | Tree T

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

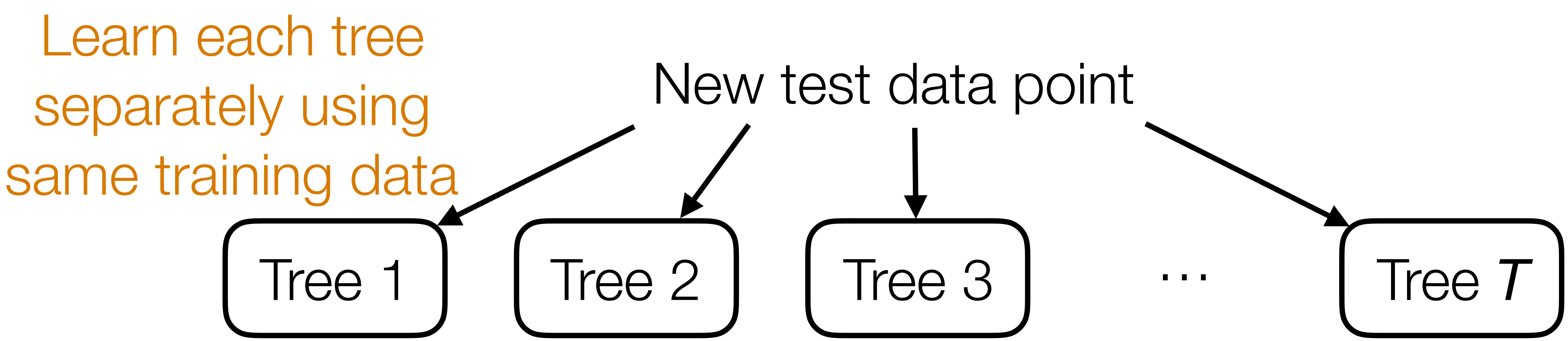
Learn each tree separately using same training data

New test data point

Tree 1 | Tree 2 | Tree 3 | ⋯ | Tree T

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

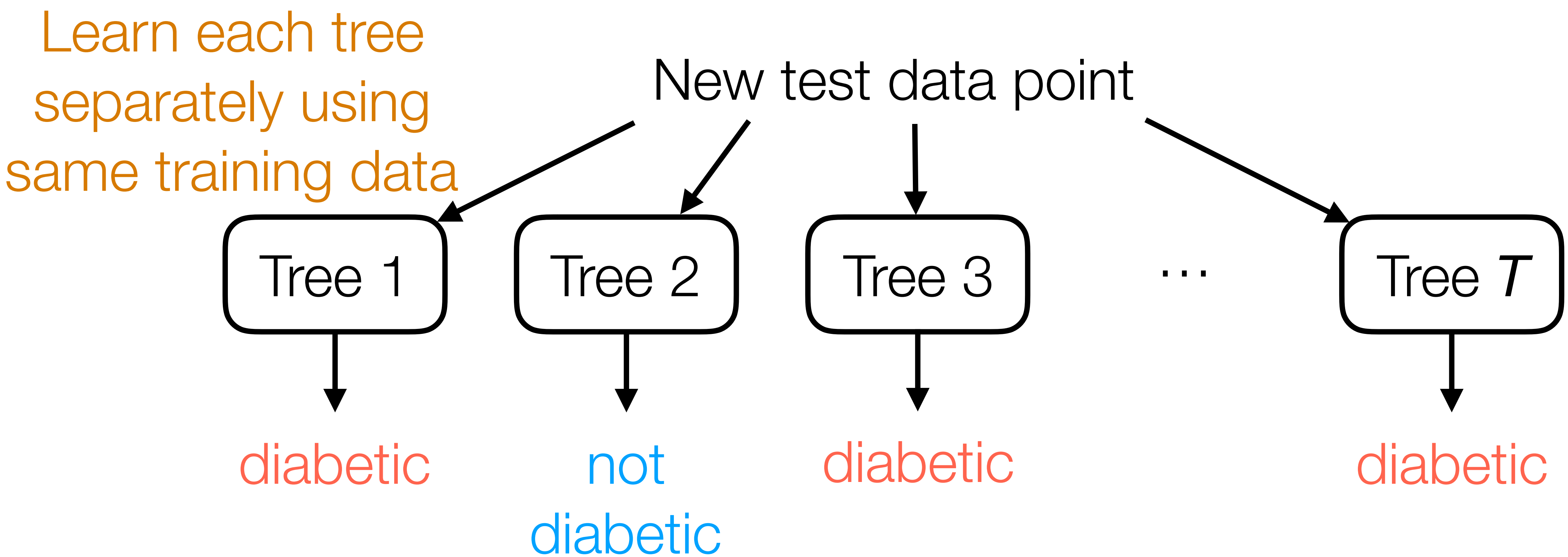


Decision Forest for Classification

- 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



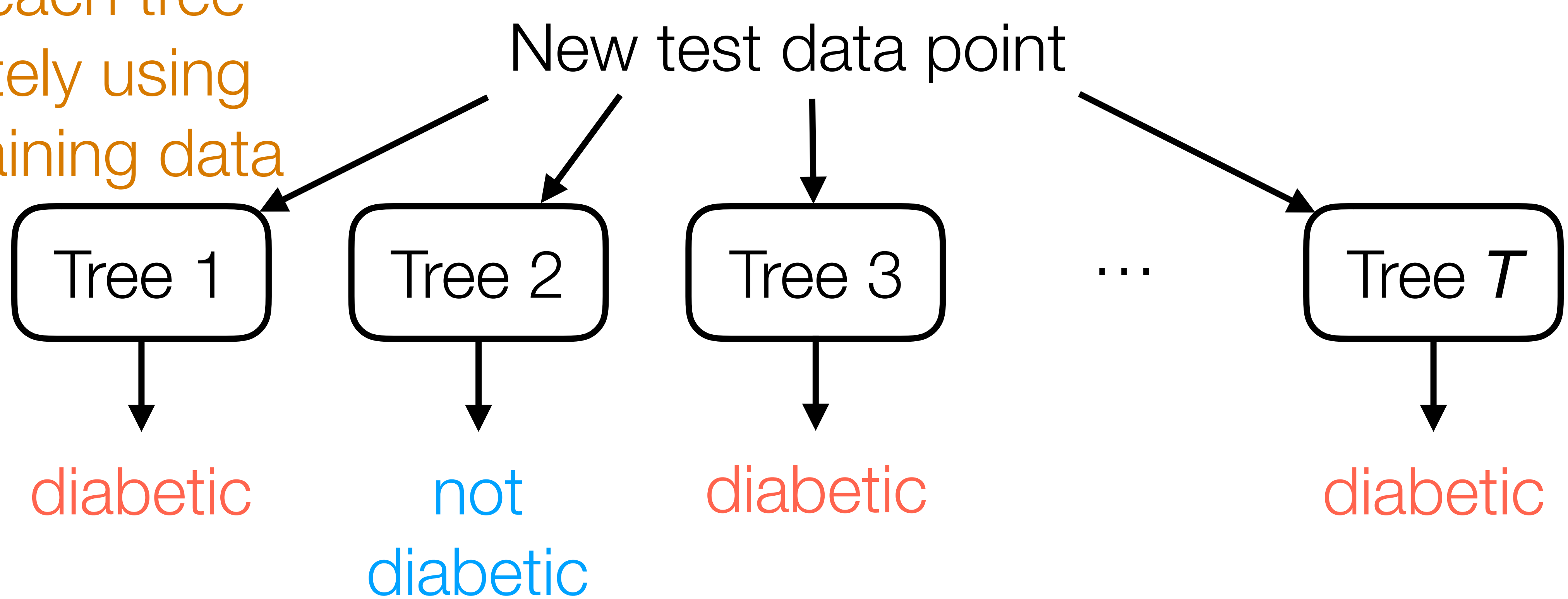
Decision Forest for Classification

- 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

Learn each tree separately using same training data



Final prediction: majority vote of the different trees' predictions

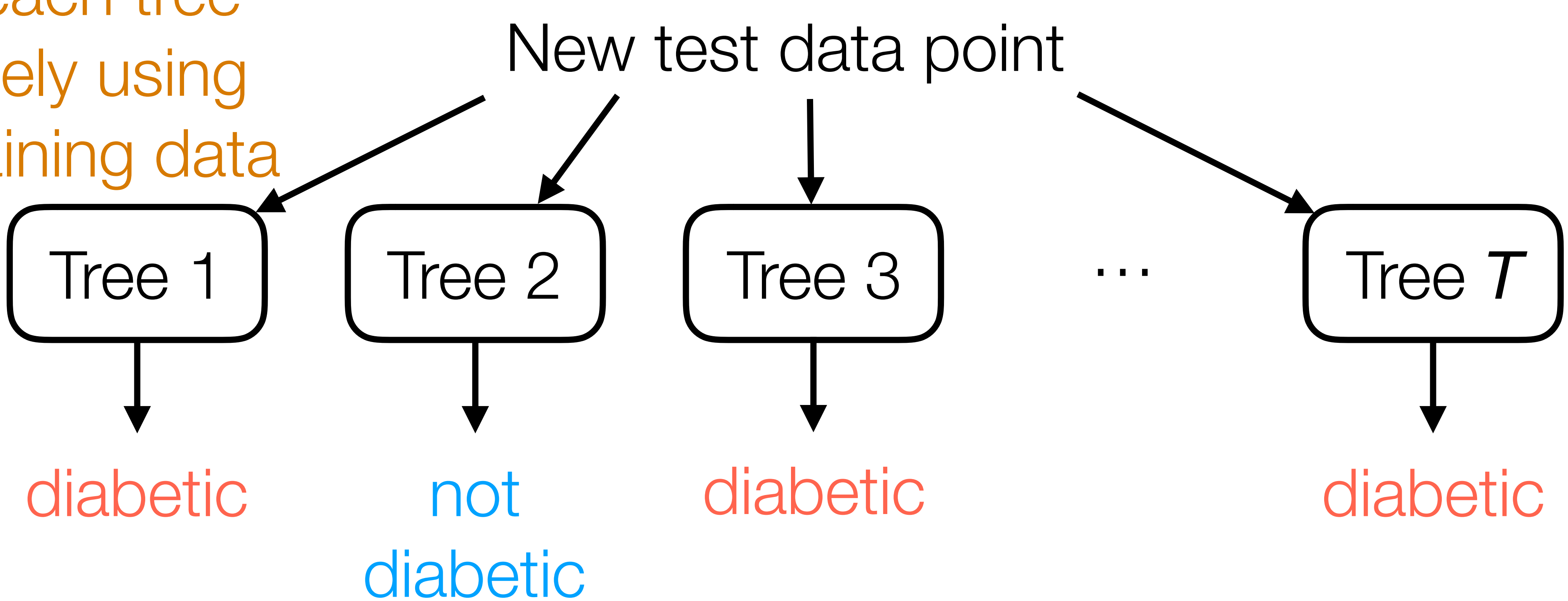
Decision Forest for Classification

- 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

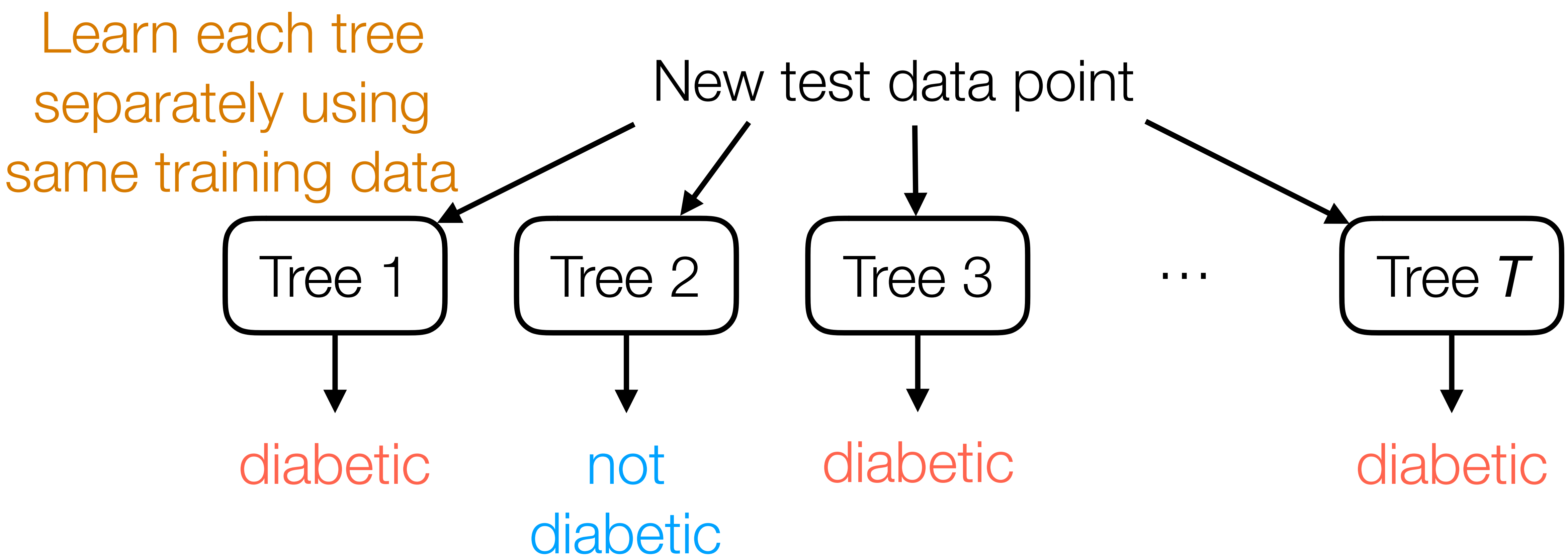
Learn each tree separately using same training data



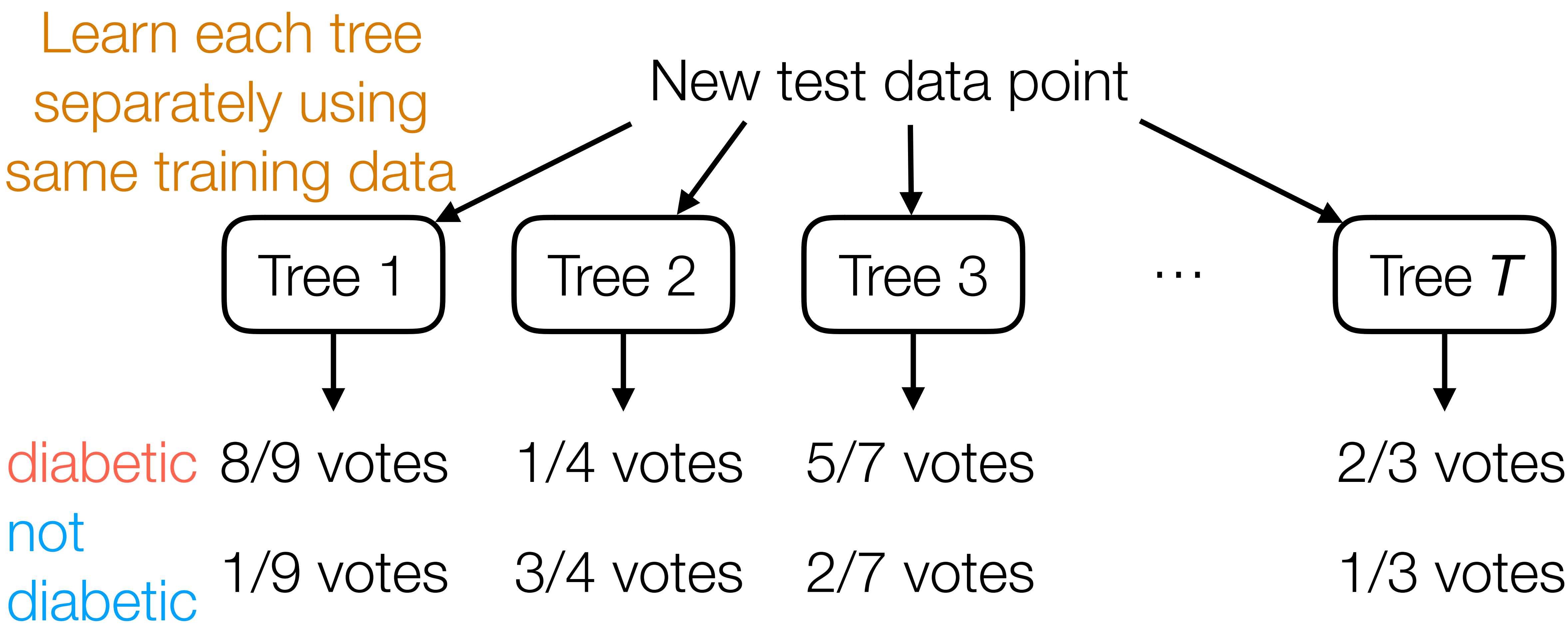
Final prediction: majority vote of the different trees' predictions

This is not the only way to aggregate predictions!

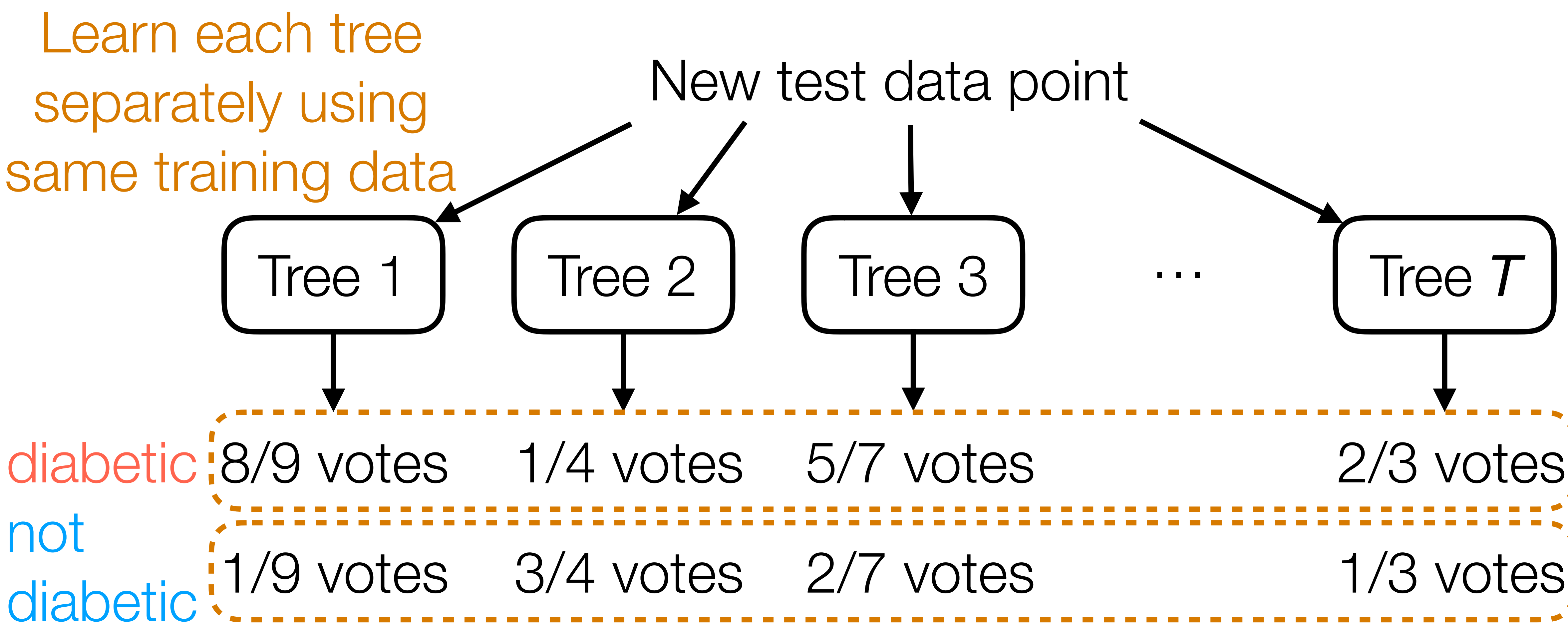
Decision Forest for Classification



Decision Forest for Classification

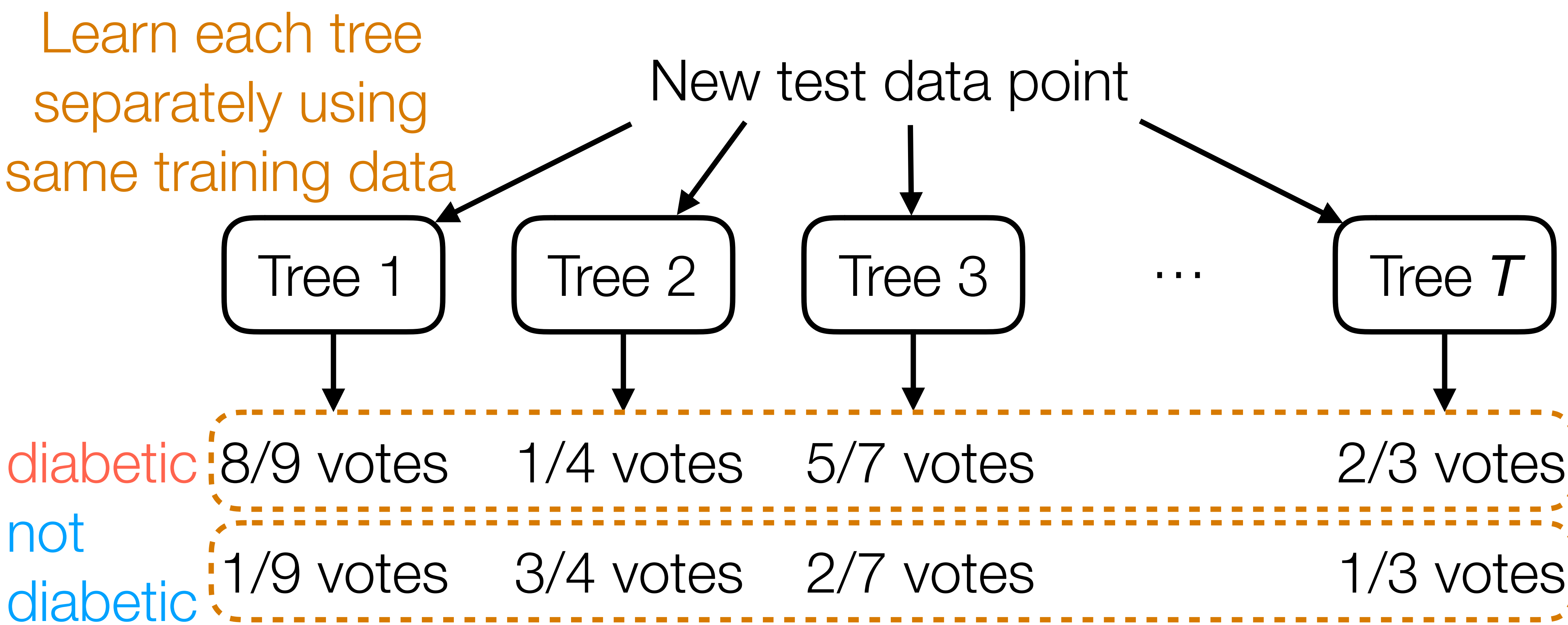


Decision Forest for Classification



Final prediction: sum up votes across trees to find winner of election!

Decision Forest for Classification

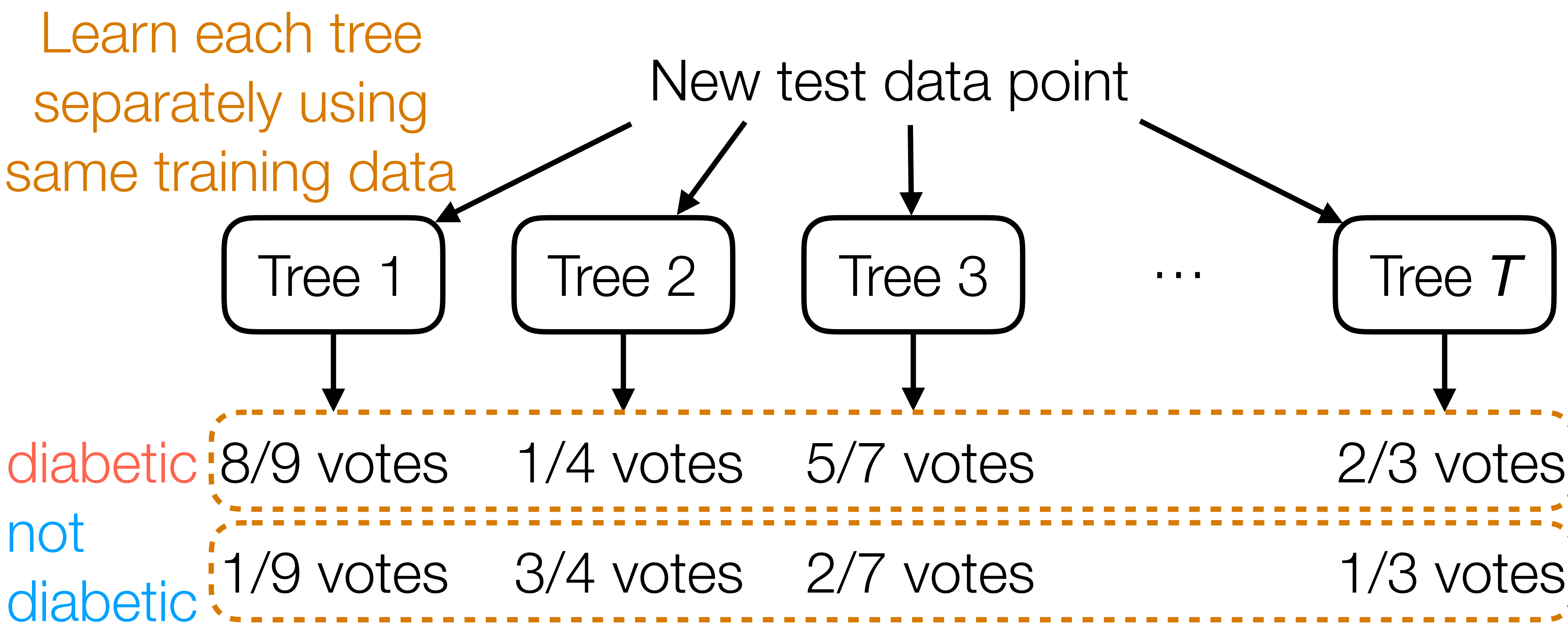


Final prediction: sum up votes across trees to find winner of election!

Nearest neighbor interpretation:

For a specific test data point x and training data point x_i

Decision Forest for Classification



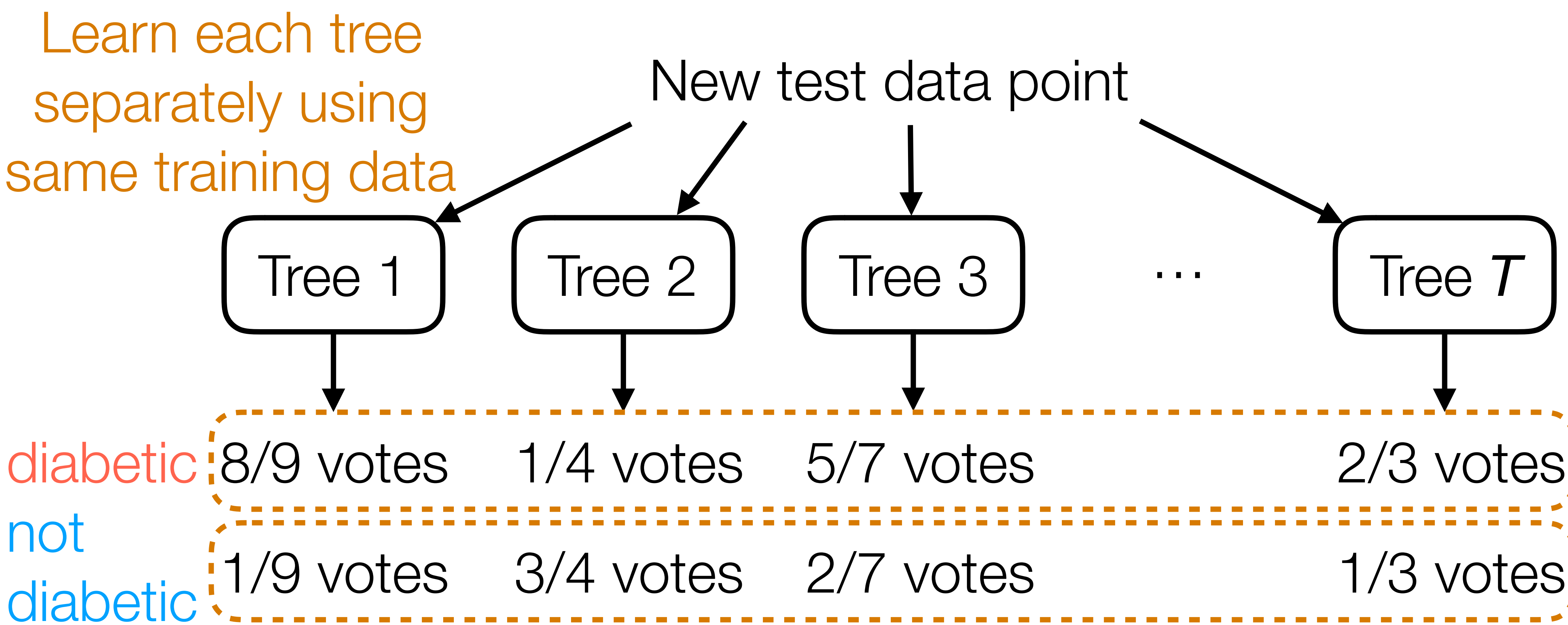
Final prediction: sum up votes across trees to find winner of election!

Nearest neighbor interpretation:

For a specific test data point x and training data point x_i

$$\text{similarity}(x, x_i) = \frac{1}{T} \sum_{t=1}^{T} \text{similarity}_t(x, x_i)$$

Decision Forest for Classification



Final prediction: sum up votes across trees to find winner of election!

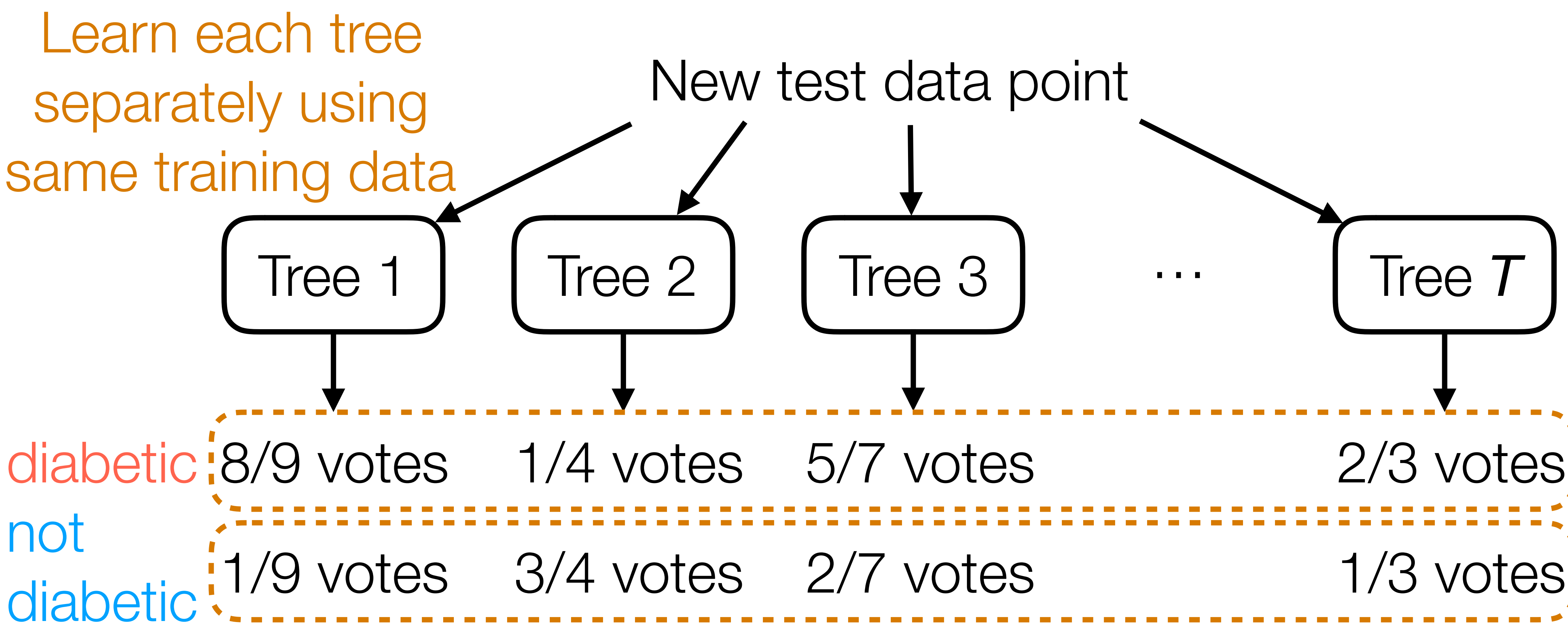
Nearest neighbor interpretation:

For a specific test data point x and training data point x_i

$$\text{similarity}(x, x_i) = \frac{1}{T}\sum_{t=1}^{T} \boxed{\text{similarity}_t(x, x_i)}$$

similarity function for t-th tree

Decision Forest for Classification



Final prediction: sum up votes across trees to find winner of election!

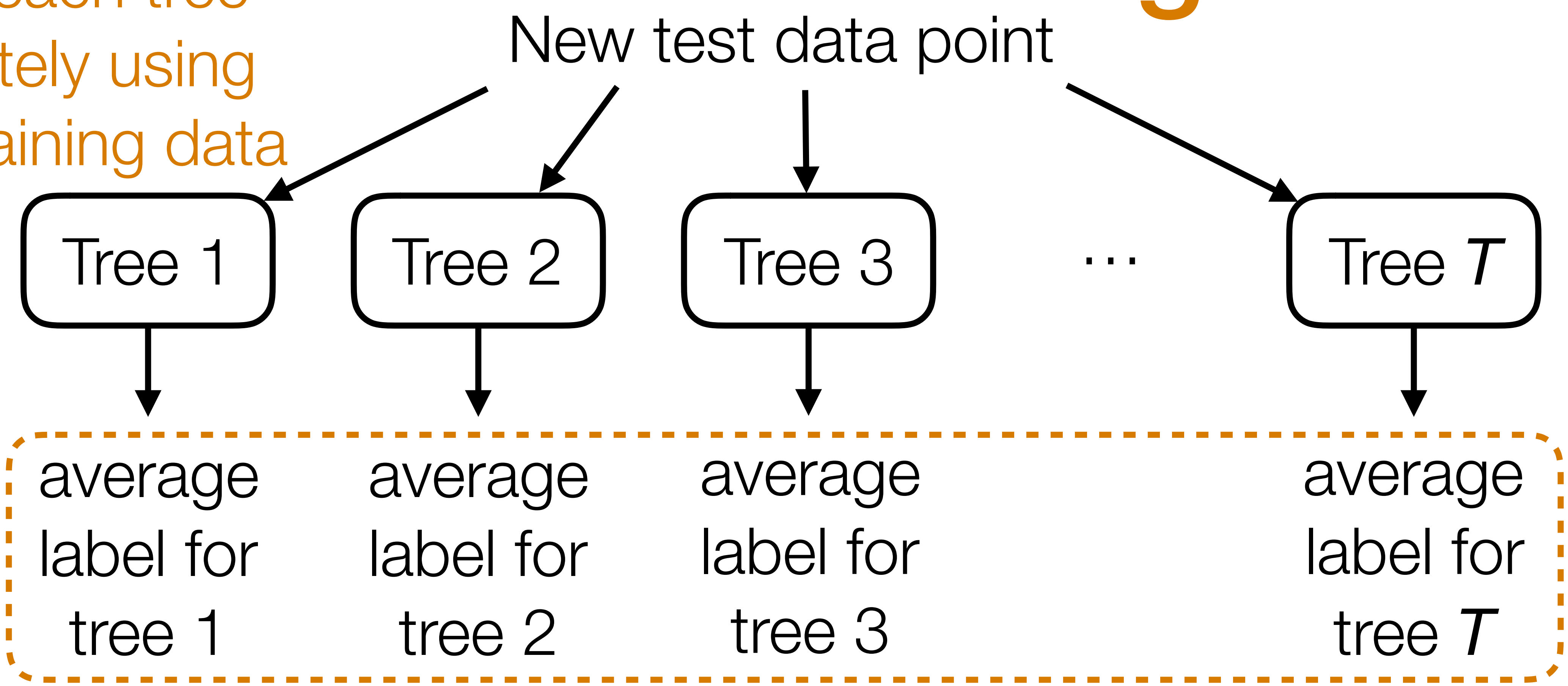
Nearest neighbor interpretation:

For a specific test data point x and training data point x_i

$$\text{similarity}(x, x_i) = \underbrace{\boxed{\frac{1}{T}}}_{\text{makes overall similarity between 0 and 1}} \sum_{t=1}^{T} \underbrace{\boxed{\text{similarity}_t(x, x_i)}}_{\text{similarity function for } t\text{-th tree}}$$

Decision Forest for ~~Classification~~ Regression

Learn each tree separately using same training data



Average these values to get final prediction

Nearest neighbor interpretation:

For a specific test data point x and training data point x_i

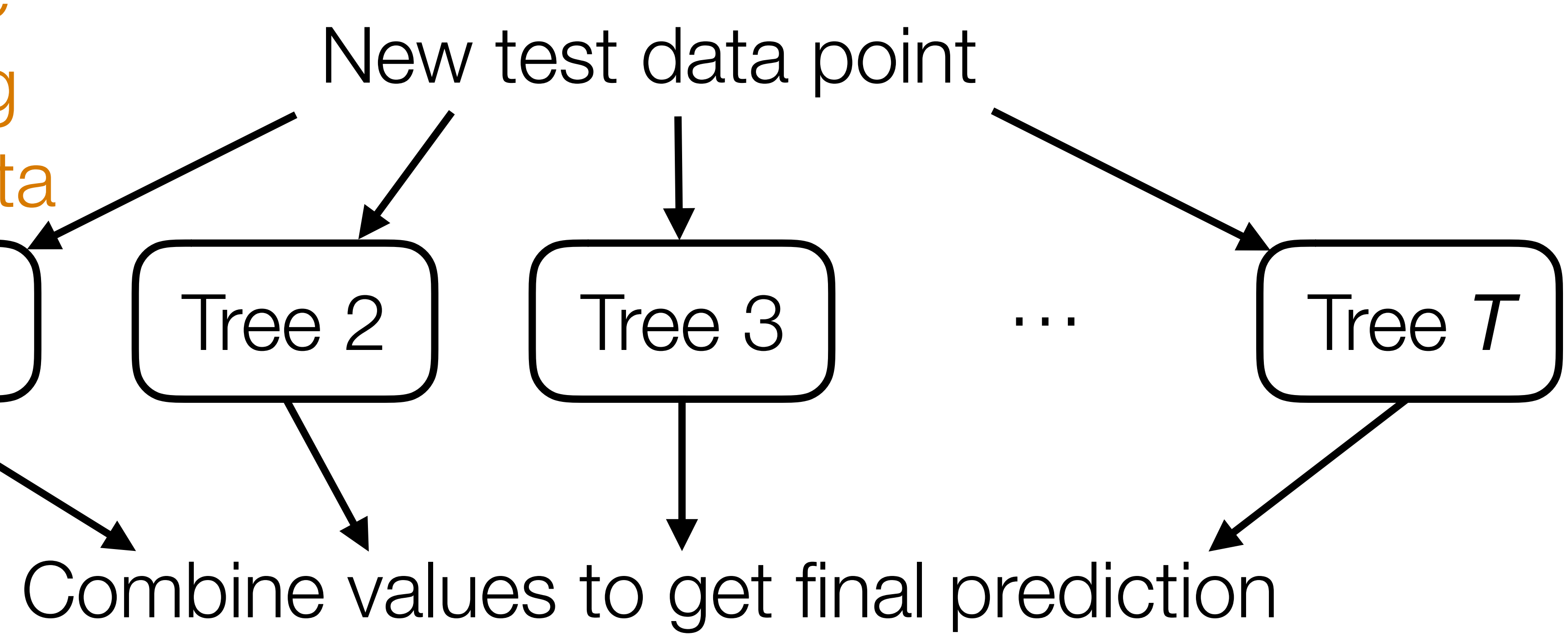
$$\text{similarity}(x, x_i) = \boxed{\frac{1}{T}} \sum_{t=1}^{T} \boxed{\text{similarity}_t(x, x_i)}$$

makes overall similarity between 0 and 1

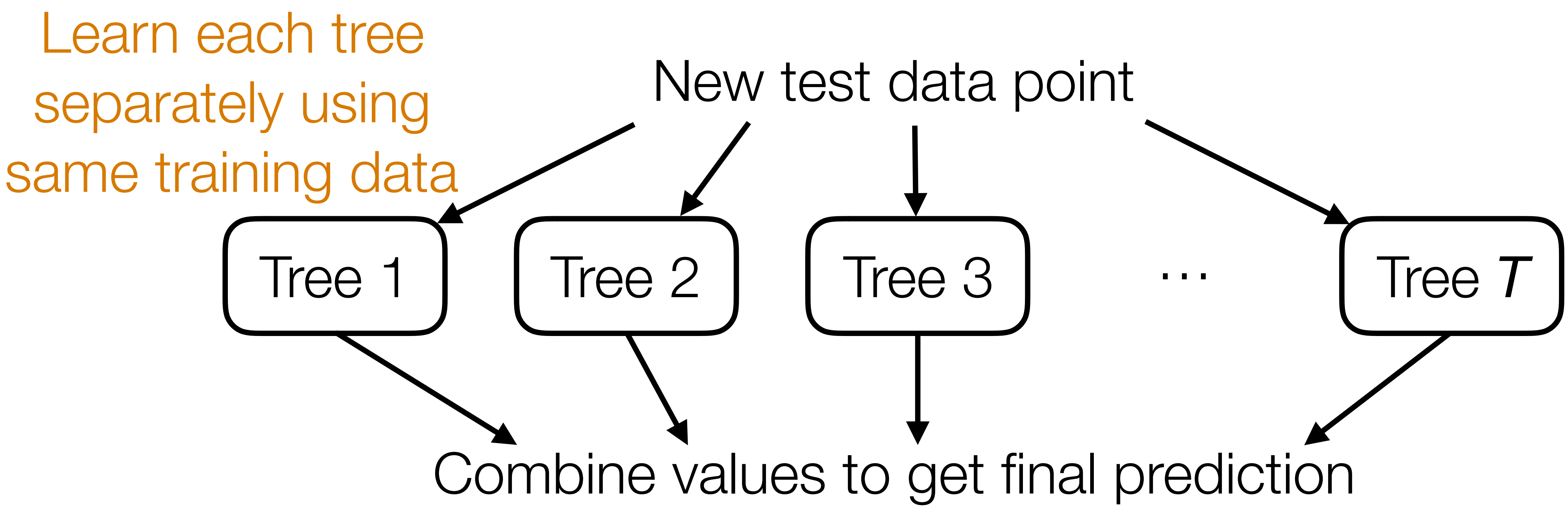
similarity function for t-th tree

Decision Forest

Learn each tree separately using same training data



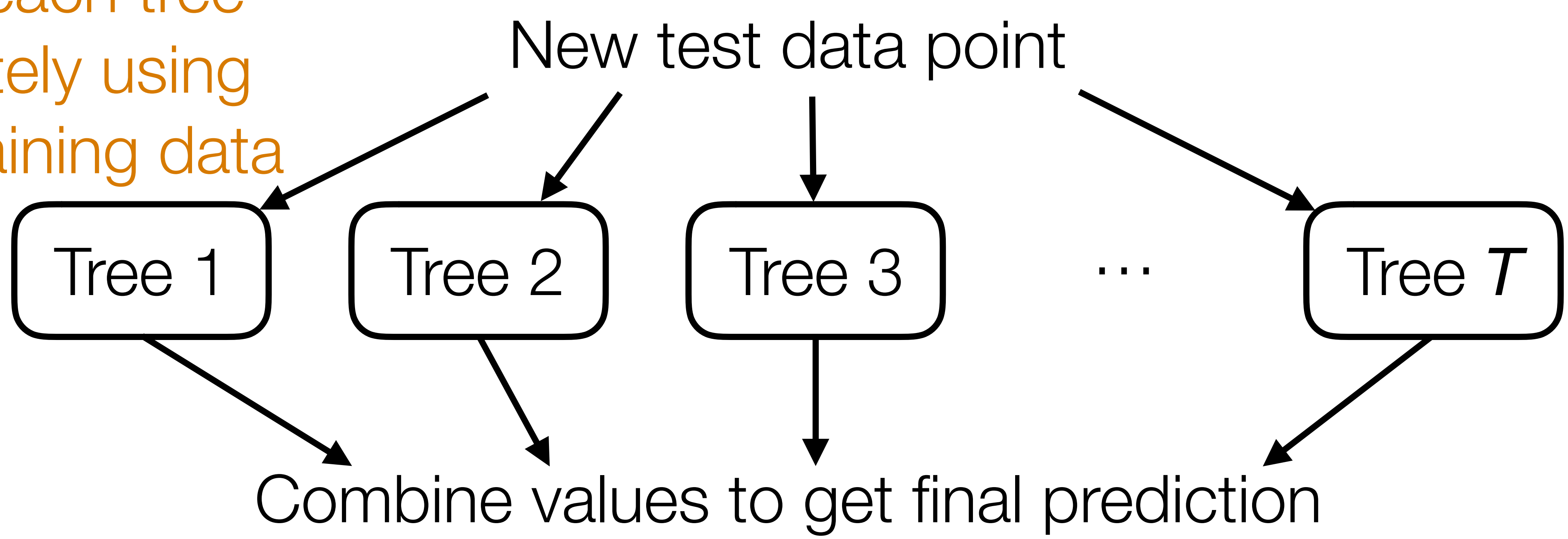
Decision Forest



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

Decision Forest

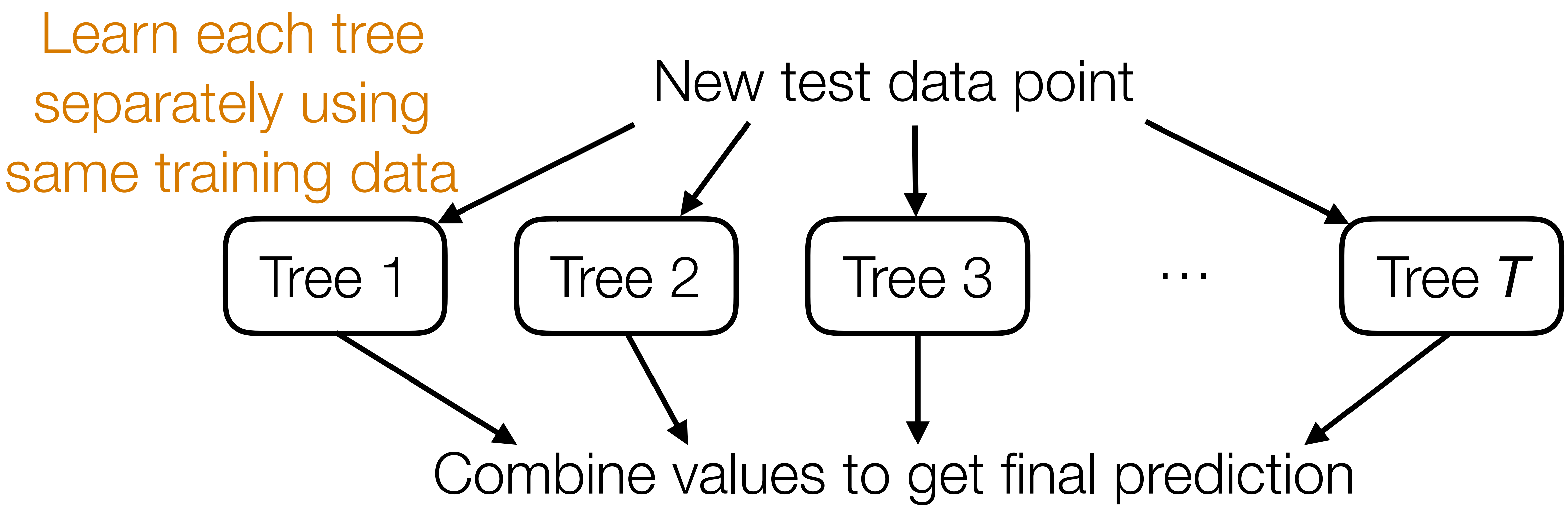
Learn each tree separately using same training data



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

Adding randomness can make trees more different!

Decision Forest

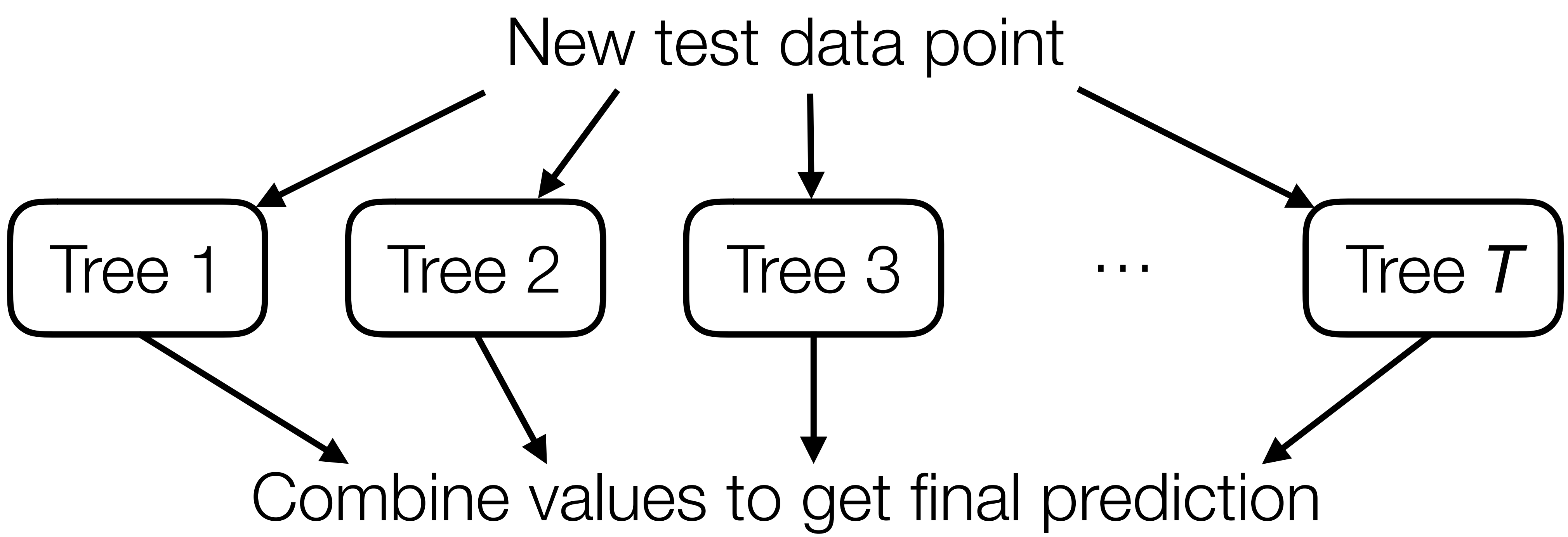


Question: What happens if all the trees are the same?

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

Decision Forest



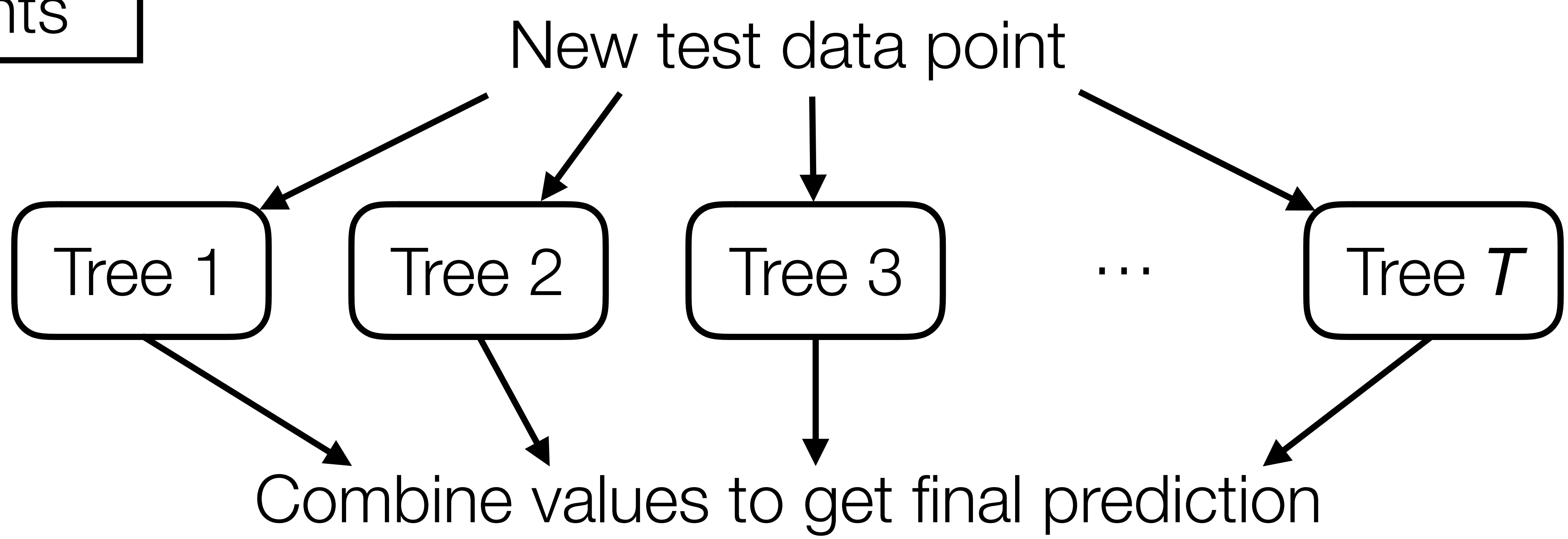
Question: What happens if all the trees are the same?

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

Decision Forest

n training data points

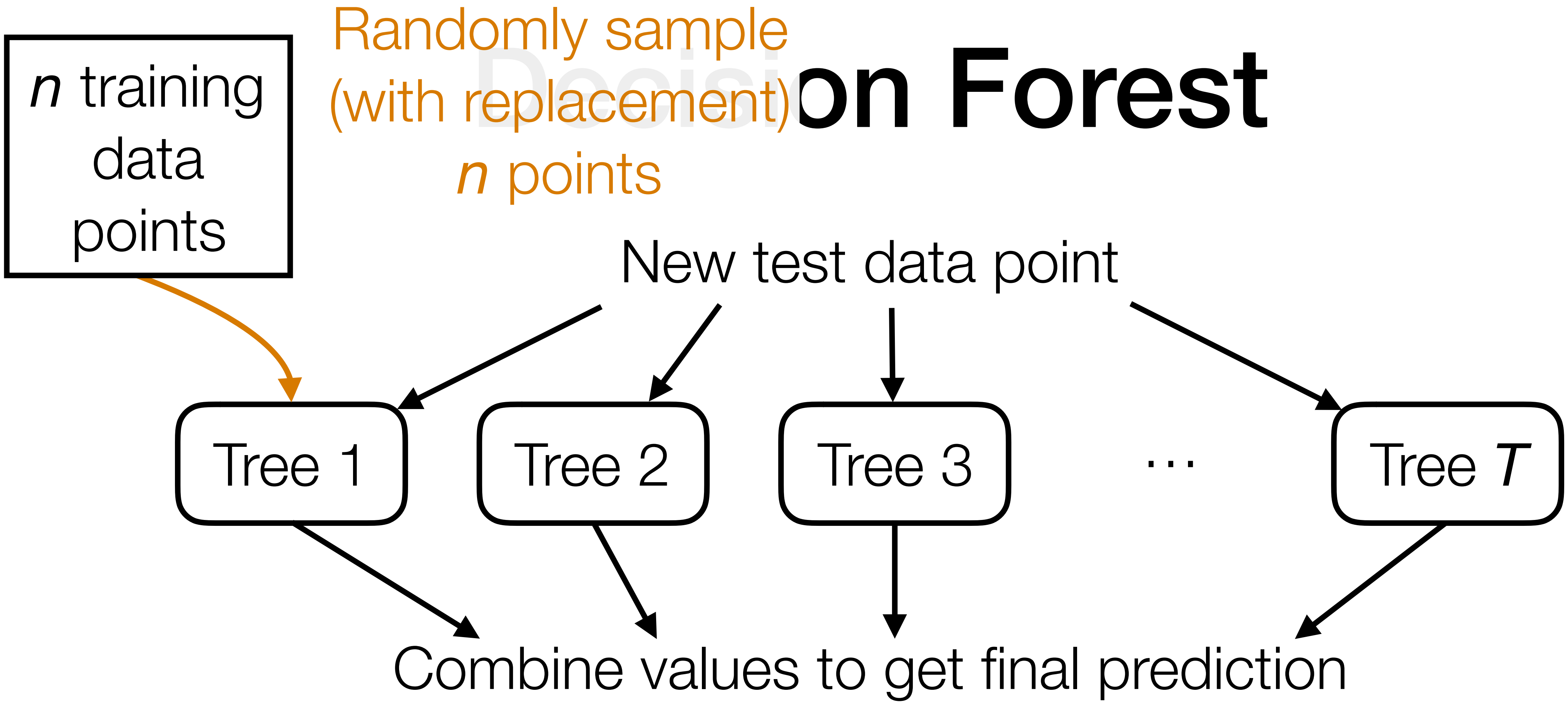


Question: What happens if all the trees are the same?

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

Decision Forest

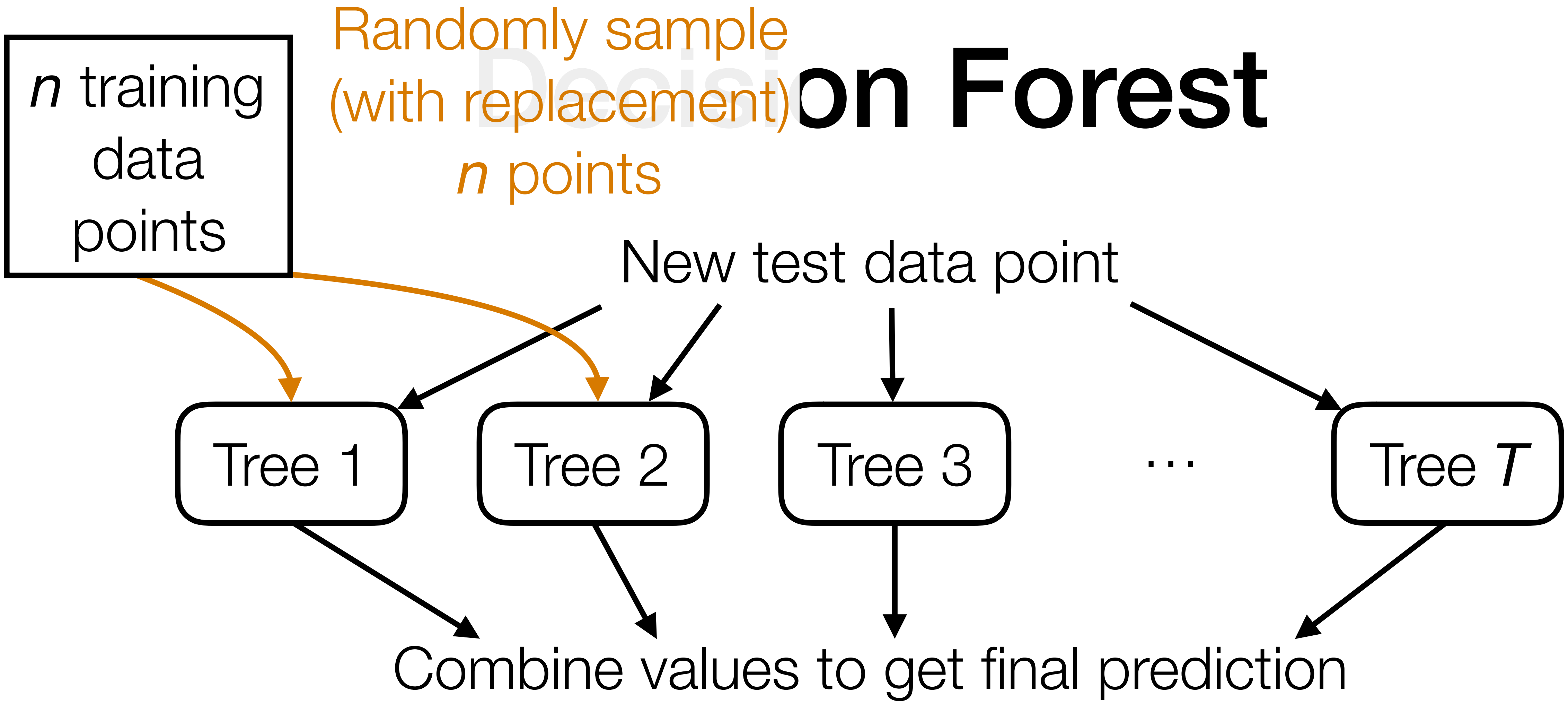


Question: What happens if all the trees are the same?

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

Decision Forest

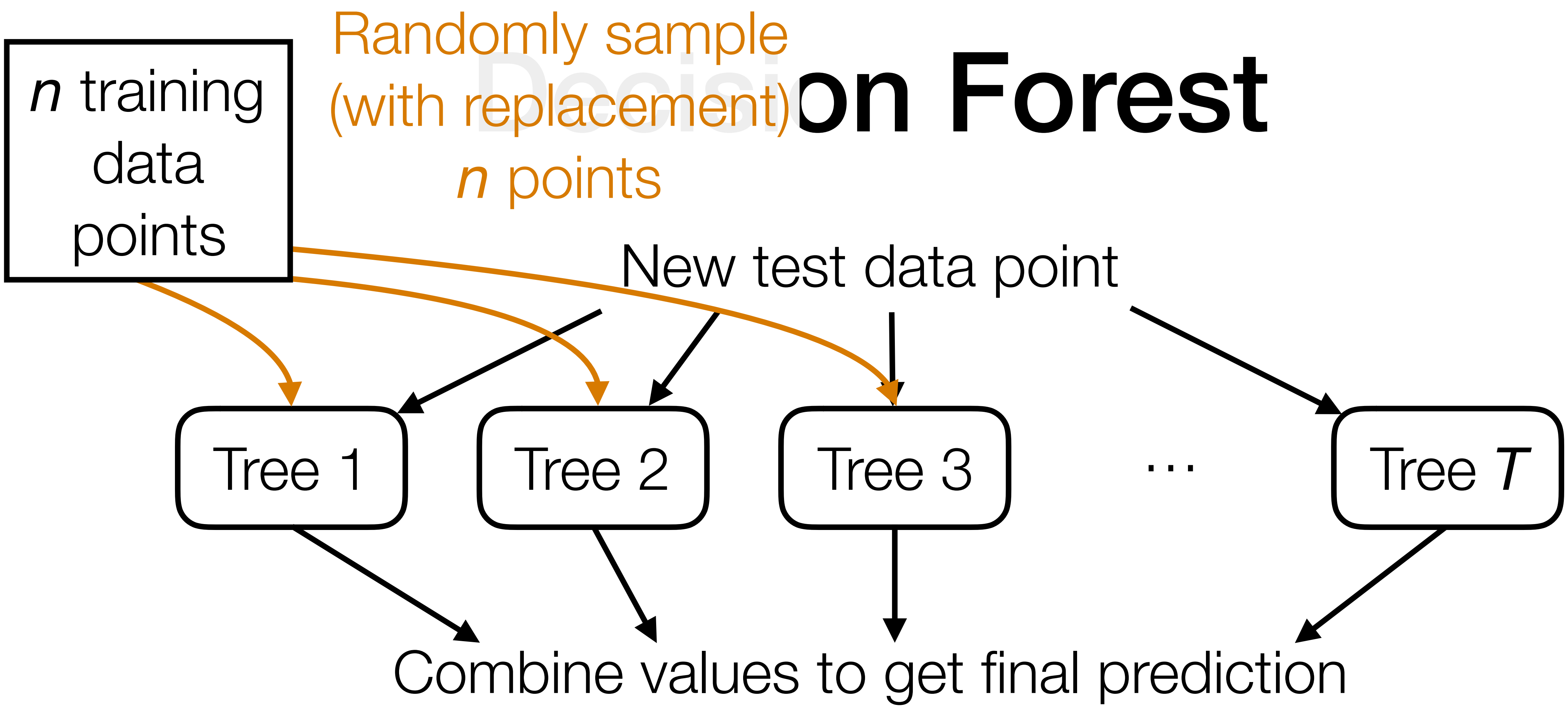


Question: What happens if all the trees are the same?

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

Decision Forest

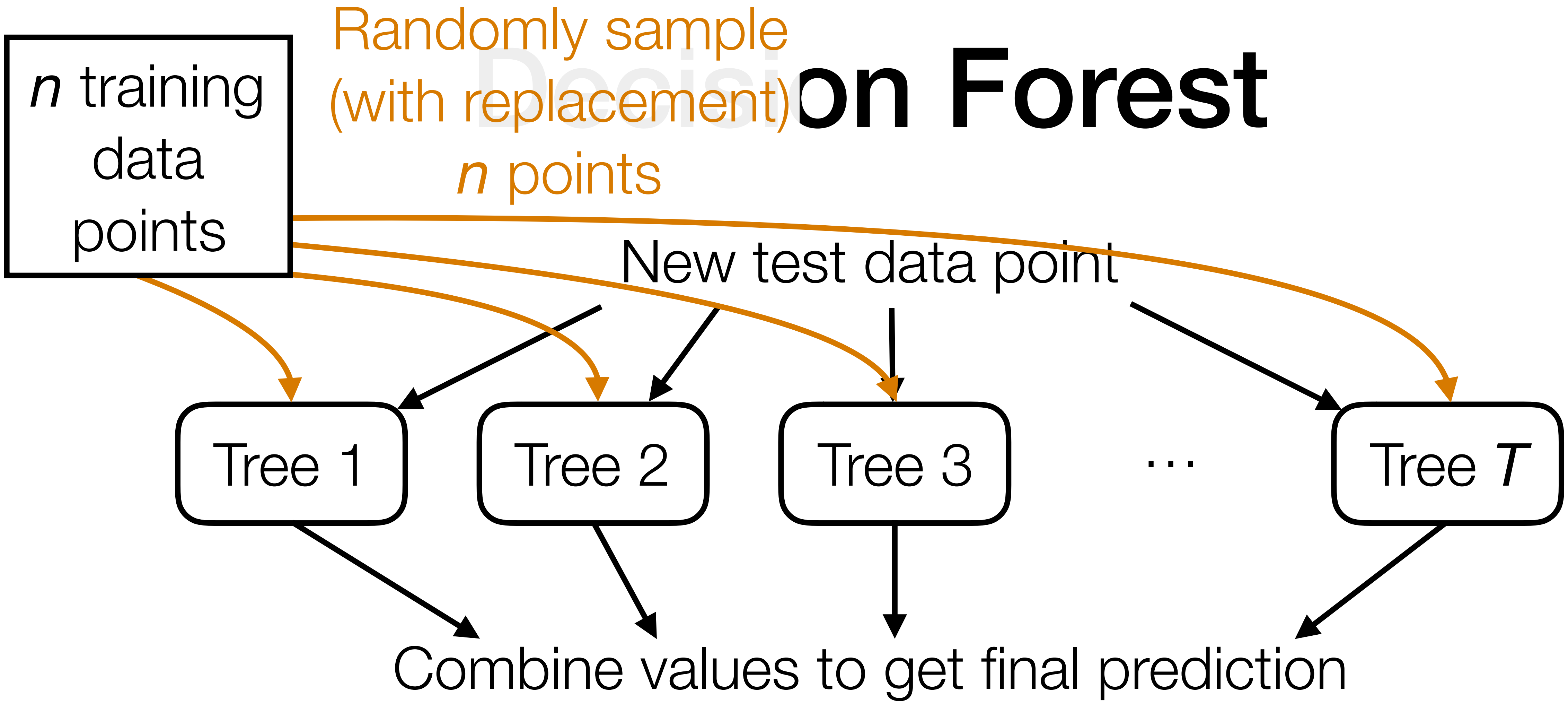


Question: What happens if all the trees are the same?

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

Decision Forest

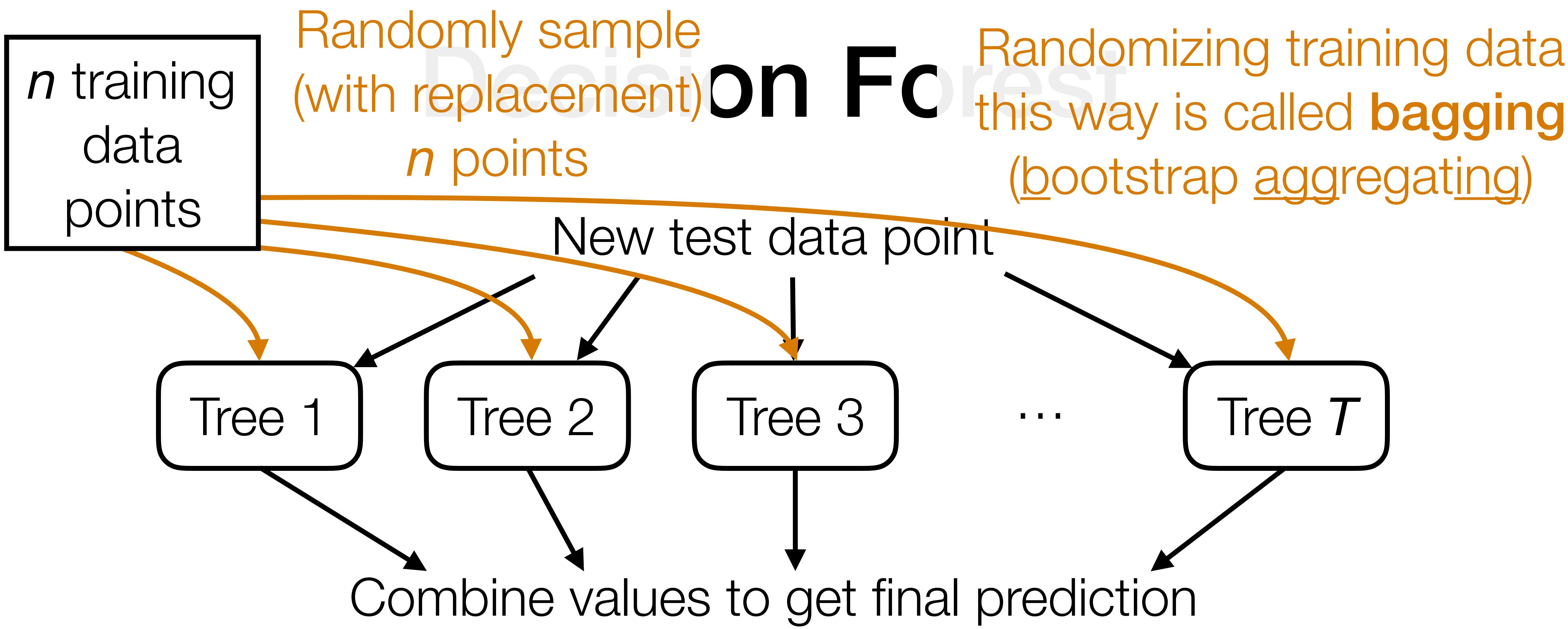


Question: What happens if all the trees are the same?

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

Decision Forest

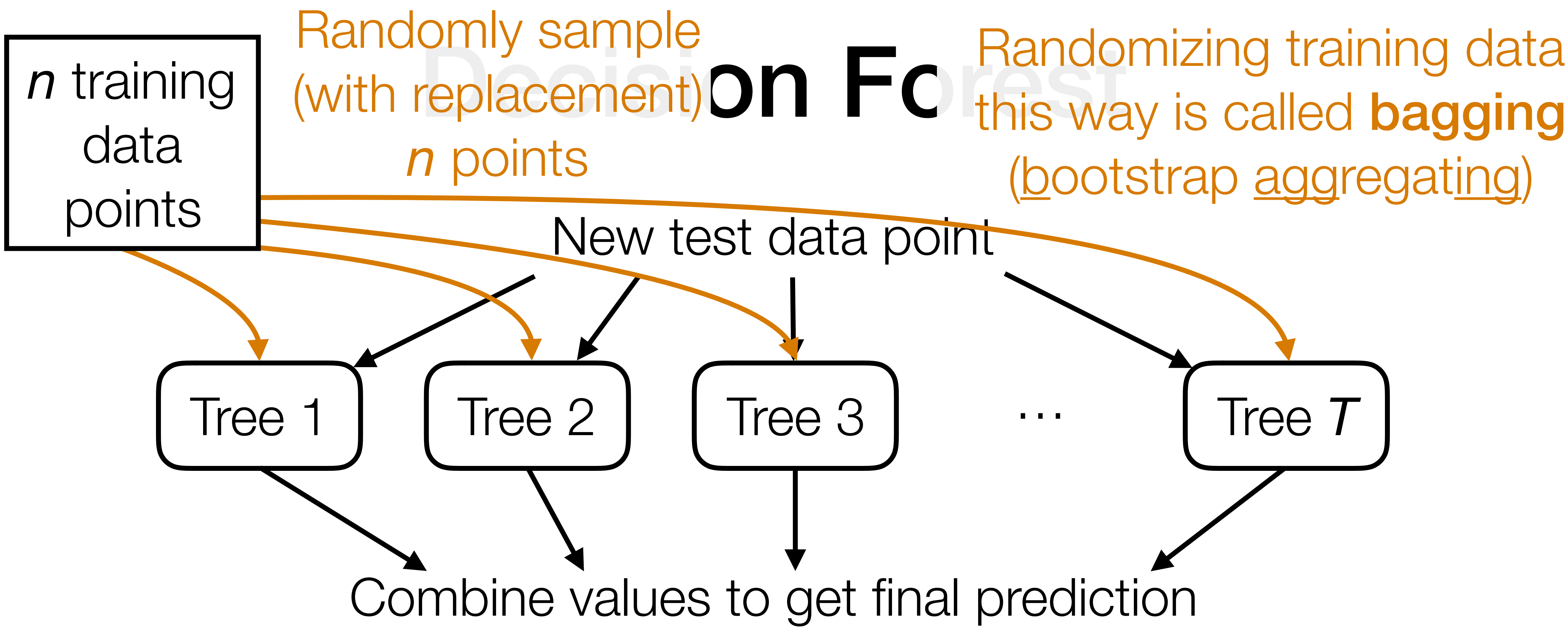


Question: What happens if all the trees are the same?

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

Decision Forest



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

Boosting

Boosting

I’ll only sketch the general idea

Boosting

I’ll only sketch the general idea

Random decision forests learned each tree separately

Boosting

I'll only sketch the general idea

Random decision forests learned each tree separately

If some trees are bad, we still weight them equally

Boosting

I'll only sketch the general idea

Random decision forests learned each tree separately

Boosting: learn trees *sequentially*, and learn from previous trees' mistakes

If some trees are bad, we still weight them equally

Boosting

I'll only sketch the general idea

Random decision forests learned each tree separately

Boosting: learn trees *sequentially*, and learn from previous trees' mistakes

If some trees are bad, we still weight them equally

Boosting: weight trees unequally so bad trees are down-weighted

Boosting

Boosting

Tree 1

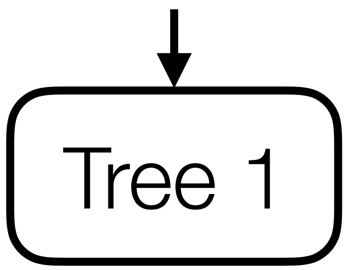
Boosting

Training data

Tree 1

Boosting

Training data



Boosting

Training data

Tree 1

Boosting

Training data

↓

Tree 1

↓

Predicted: cat, dog, shark

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples
to emphasize them more when
learning the next tree

Boosting

Training data

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Training data

Tree 2

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

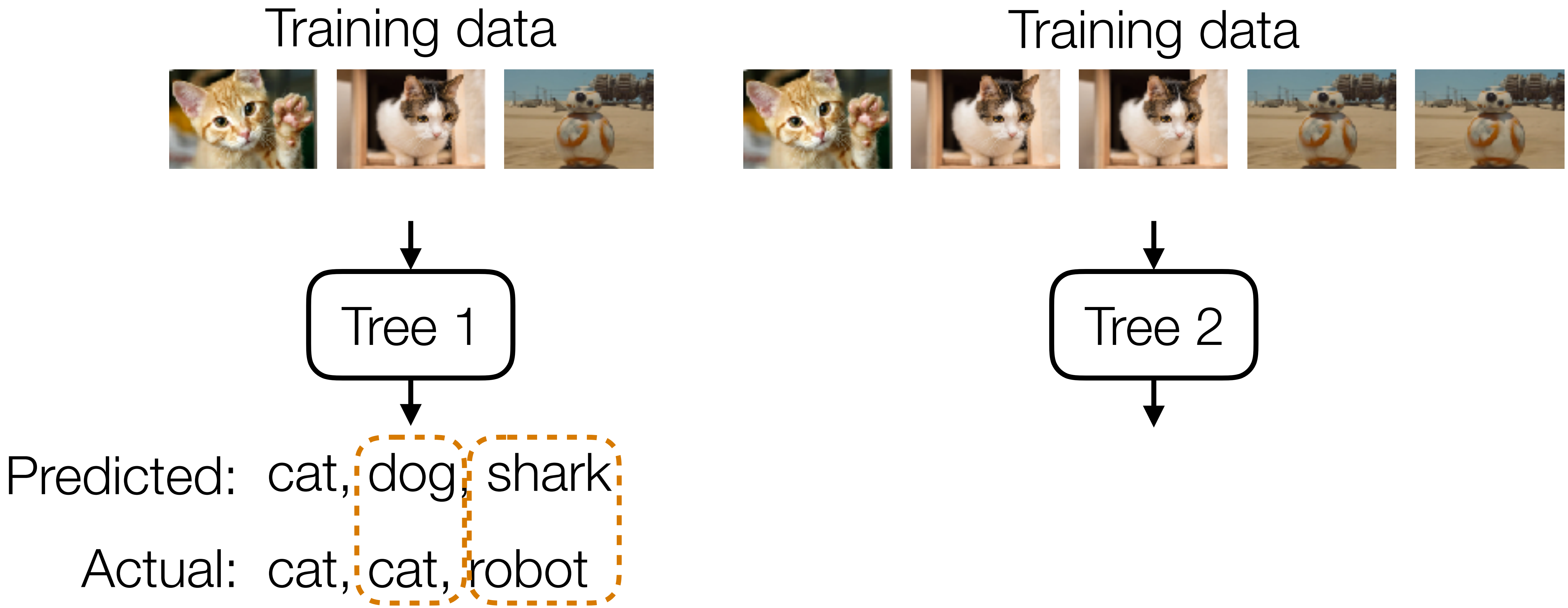
Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Training data

Tree 2

Boosting



Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Training data

Tree 2

Predicted: cat, cat, donkey

Actual: cat, cat, robot

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Training data

Tree 2

Predicted: cat, cat, donkey

Actual: cat, cat, robot

Where did the errors appear?

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Training data

Tree 2

Predicted: cat, cat, donkey

Actual: cat, cat, robot

Where did the errors appear?

Boosting

Training data

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Training data

Tree 2

Predicted: cat, cat, donkey

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Boosting

Training data

$w_1^{(1)} = 1$ $w_2^{(1)} = 1$ $w_3^{(1)} = 1$

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Training data

Tree 2

Predicted: cat, cat, donkey

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Boosting

Training data

$w_1^{(1)} = 1$ $w_2^{(1)} = 1$ $w_3^{(1)} = 1$

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Training data

Tree 2

Predicted: cat, cat, donkey

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Boosting

Training data

$w_1^{(1)} = 1$ $w_2^{(1)} = 1$ $w_3^{(1)} = 1$

Tree 1

Predicted: cat, dog, shark

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Training data

$w_1^{(2)} = 1$ $w_2^{(2)} = 2$ $w_3^{(2)} = 2$

Tree 2

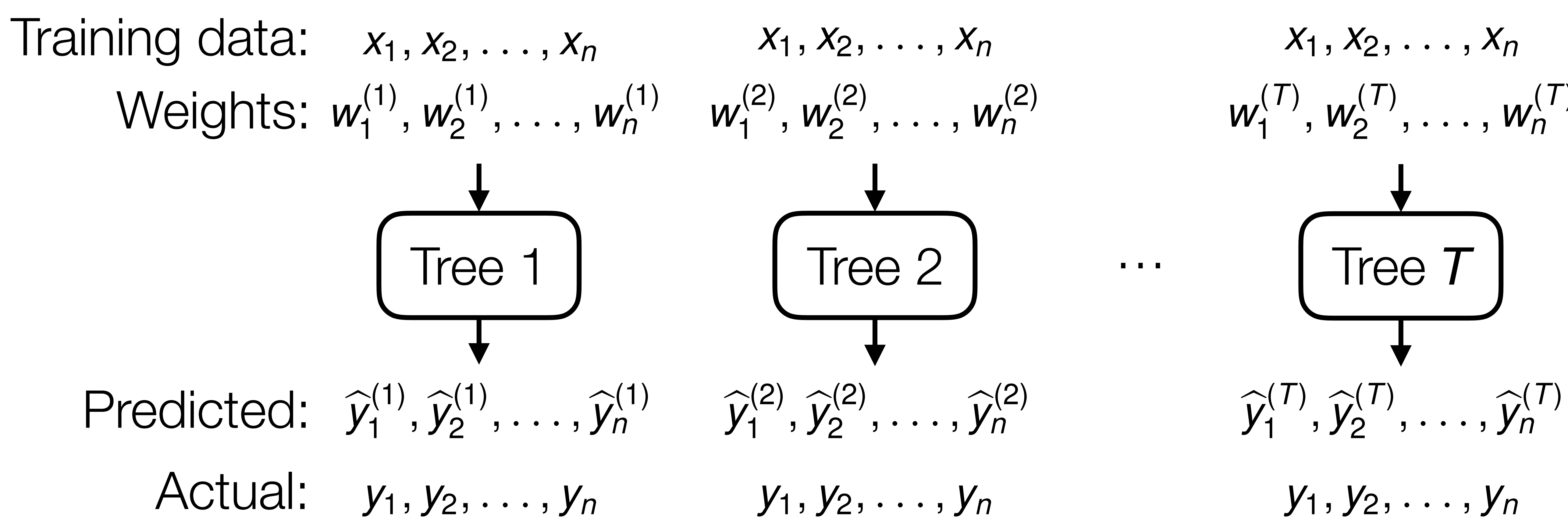
Predicted: cat, cat, donkey

Actual: cat, cat, robot

Where did the errors appear?

Duplicate these training examples to emphasize them more when learning the next tree

Boosting


Training data:
$x_1, x_2, \ldots, x_n$
$x_1, x_2, \ldots, x_n$
$x_1, x_2, \ldots, x_n$
Weights:
$w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}$
$w_1^{(2)}, w_2^{(2)}, \ldots, w_n^{(2)}$
$w_1^{(T)}, w_2^{(T)}, \ldots, w_n^{(T)}$
Tree 1
Tree 2
$\cdots$
Tree T
Predicted:
$\widehat{y}_1^{(1)}, \widehat{y}_2^{(1)}, \ldots, \widehat{y}_n^{(1)}$
$\widehat{y}_1^{(2)}, \widehat{y}_2^{(2)}, \ldots, \widehat{y}_n^{(2)}$
$\widehat{y}_1^{(T)}, \widehat{y}_2^{(T)}, \ldots, \widehat{y}_n^{(T)}$
Actual:
$y_1, y_2, \ldots, y_n$
$y_1, y_2, \ldots, y_n$
$y_1, y_2, \ldots, y_n$

Boosting

Learn trees *sequentially* accounting for mistakes made previously

Training data:	$x_1, x_2, \ldots, x_n$	$x_1, x_2, \ldots, x_n$		$x_1, x_2, \ldots, x_n$
Weights:	$w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}$	$w_1^{(2)}, w_2^{(2)}, \ldots, w_n^{(2)}$		$w_1^{(T)}, w_2^{(T)}, \ldots, w_n^{(T)}$
	$\downarrow$	$\downarrow$		$\downarrow$
	Tree 1	Tree 2	$\cdots$	Tree T
	$\downarrow$	$\downarrow$		$\downarrow$
Predicted:	$\widehat{y}_1^{(1)}, \widehat{y}_2^{(1)}, \ldots, \widehat{y}_n^{(1)}$	$\widehat{y}_1^{(2)}, \widehat{y}_2^{(2)}, \ldots, \widehat{y}_n^{(2)}$		$\widehat{y}_1^{(T)}, \widehat{y}_2^{(T)}, \ldots, \widehat{y}_n^{(T)}$
Actual:	$y_1, y_2, \ldots, y_n$	$y_1, y_2, \ldots, y_n$		$y_1, y_2, \ldots, y_n$

Boosting

Learn trees *sequentially* accounting for mistakes made previously

Training data:	$x_1, x_2, \ldots, x_n$	$x_1, x_2, \ldots, x_n$		$x_1, x_2, \ldots, x_n$
Weights:	$w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}$	$w_1^{(2)}, w_2^{(2)}, \ldots, w_n^{(2)}$		$w_1^{(T)}, w_2^{(T)}, \ldots, w_n^{(T)}$
	$\downarrow$	$\downarrow$		$\downarrow$
	Tree 1	Tree 2	$\cdots$	Tree T
	$\downarrow$	$\downarrow$		$\downarrow$
Predicted:	$\widehat{y}_1^{(1)}, \widehat{y}_2^{(1)}, \ldots, \widehat{y}_n^{(1)}$	$\widehat{y}_1^{(2)}, \widehat{y}_2^{(2)}, \ldots, \widehat{y}_n^{(2)}$		$\widehat{y}_1^{(T)}, \widehat{y}_2^{(T)}, \ldots, \widehat{y}_n^{(T)}$
Actual:	$y_1, y_2, \ldots, y_n$	$y_1, y_2, \ldots, y_n$		$y_1, y_2, \ldots, y_n$

Adjust for how much each tree's votes count

Boosting

Learn trees *sequentially* accounting for mistakes made previously

Training data: $x_1, x_2, \ldots, x_n$ $\quad$ $x_1, x_2, \ldots, x_n$ $\quad$ $x_1, x_2, \ldots, x_n$

Weights: $w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}$ $\quad$ $w_1^{(2)}, w_2^{(2)}, \ldots, w_n^{(2)}$ $\quad$ $w_1^{(T)}, w_2^{(T)}, \ldots, w_n^{(T)}$

$\downarrow$ $\quad$ $\downarrow$ $\quad$ $\downarrow$

Tree 1 $\quad$ Tree 2 $\quad \cdots \quad$ Tree T

$\downarrow$ $\quad$ $\downarrow$ $\quad$ $\downarrow$

Predicted: $\widehat{y}_1^{(1)}, \widehat{y}_2^{(1)}, \ldots, \widehat{y}_n^{(1)}$ $\quad$ $\widehat{y}_1^{(2)}, \widehat{y}_2^{(2)}, \ldots, \widehat{y}_n^{(2)}$ $\quad$ $\widehat{y}_1^{(T)}, \widehat{y}_2^{(T)}, \ldots, \widehat{y}_n^{(T)}$

Actual: $y_1, y_2, \ldots, y_n$ $\quad$ $y_1, y_2, \ldots, y_n$ $\quad$ $y_1, y_2, \ldots, y_n$

Adjust for how much each tree's votes count

$$\text{similarity}(x, x_i) = \sum_{t=1}^{T} \alpha_t \text{similarity}_t(x, x_i)$$

Boosting

Learn trees *sequentially* accounting for mistakes made previously

	Tree 1	Tree 2	⋯	Tree T
Training data:	$x_1, x_2, \ldots, x_n$	$x_1, x_2, \ldots, x_n$		$x_1, x_2, \ldots, x_n$
Weights:	$w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}$	$w_1^{(2)}, w_2^{(2)}, \ldots, w_n^{(2)}$		$w_1^{(T)}, w_2^{(T)}, \ldots, w_n^{(T)}$
Predicted:	$\widehat{y}_1^{(1)}, \widehat{y}_2^{(1)}, \ldots, \widehat{y}_n^{(1)}$	$\widehat{y}_1^{(2)}, \widehat{y}_2^{(2)}, \ldots, \widehat{y}_n^{(2)}$		$\widehat{y}_1^{(T)}, \widehat{y}_2^{(T)}, \ldots, \widehat{y}_n^{(T)}$
Actual:	$y_1, y_2, \ldots, y_n$	$y_1, y_2, \ldots, y_n$		$y_1, y_2, \ldots, y_n$

Adjust for how much each tree's votes count

$$\text{similarity}(x, x_i) = \sum_{t=1}^{T} \alpha_t \, \text{similarity}_t(x, x_i)$$

weight for tree ***t***

Boosting

Learn trees *sequentially* accounting for mistakes made previously

Training data:	$x_1, x_2, \ldots, x_n$	$x_1, x_2, \ldots, x_n$		$x_1, x_2, \ldots, x_n$
Weights:	$w_1^{(1)}, w_2^{(1)}, \ldots, w_n^{(1)}$	$w_1^{(2)}, w_2^{(2)}, \ldots, w_n^{(2)}$		$w_1^{(T)}, w_2^{(T)}, \ldots, w_n^{(T)}$
	$\downarrow$	$\downarrow$		$\downarrow$
	Tree 1	Tree 2	$\cdots$	Tree T
	$\downarrow$	$\downarrow$		$\downarrow$
Predicted:	$\widehat{y}_1^{(1)}, \widehat{y}_2^{(1)}, \ldots, \widehat{y}_n^{(1)}$	$\widehat{y}_1^{(2)}, \widehat{y}_2^{(2)}, \ldots, \widehat{y}_n^{(2)}$		$\widehat{y}_1^{(T)}, \widehat{y}_2^{(T)}, \ldots, \widehat{y}_n^{(T)}$
Actual:	$y_1, y_2, \ldots, y_n$	$y_1, y_2, \ldots, y_n$		$y_1, y_2, \ldots, y_n$

Adjust for how much each tree's votes count

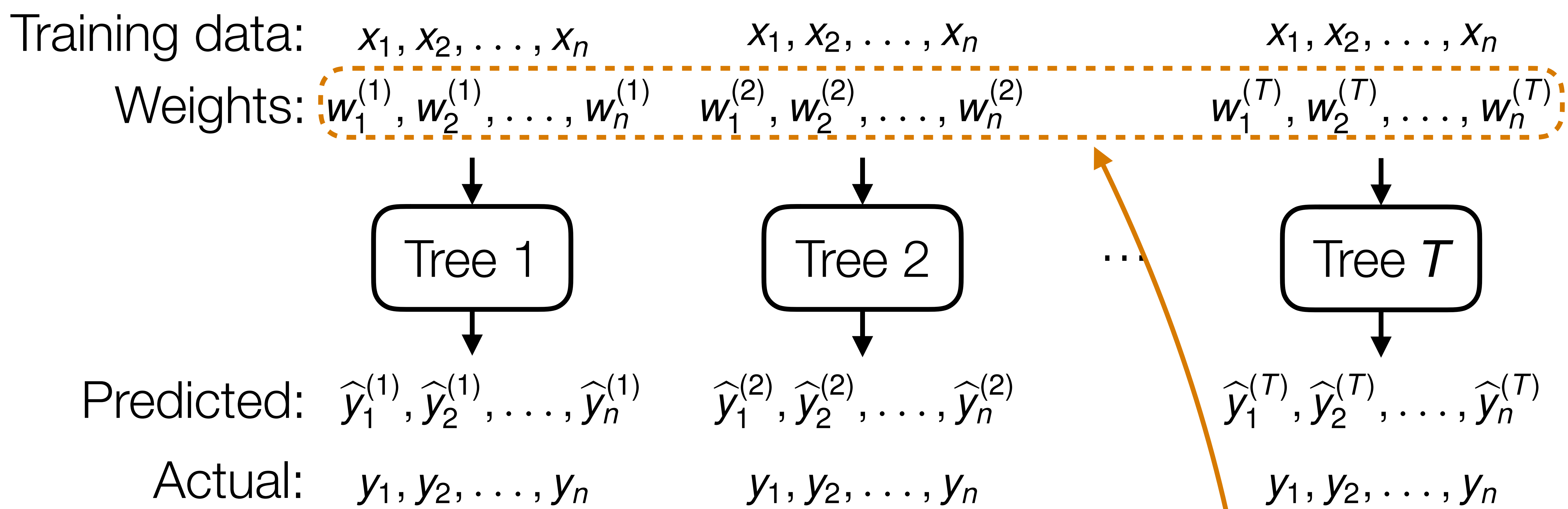
$$\text{similarity}(x, x_i) = \sum_{t=1}^{T} \alpha_t \text{similarity}_t(x, x_i)$$

weight for tree ***t***

Still an adaptive NN method!

Boosting

Learn trees *sequentially* accounting for mistakes made previously



Adjust for how much each tree's votes count

$$\text{similarity}(x, x_i) = \sum_{t=1}^{T} \alpha_t \text{similarity}_t(x, x_i)$$

weight for tree ***t***

Still an adaptive NN method!

Different ways to choose weights yield different boosting methods (e.g., AdaBoost, gradient tree boosting)