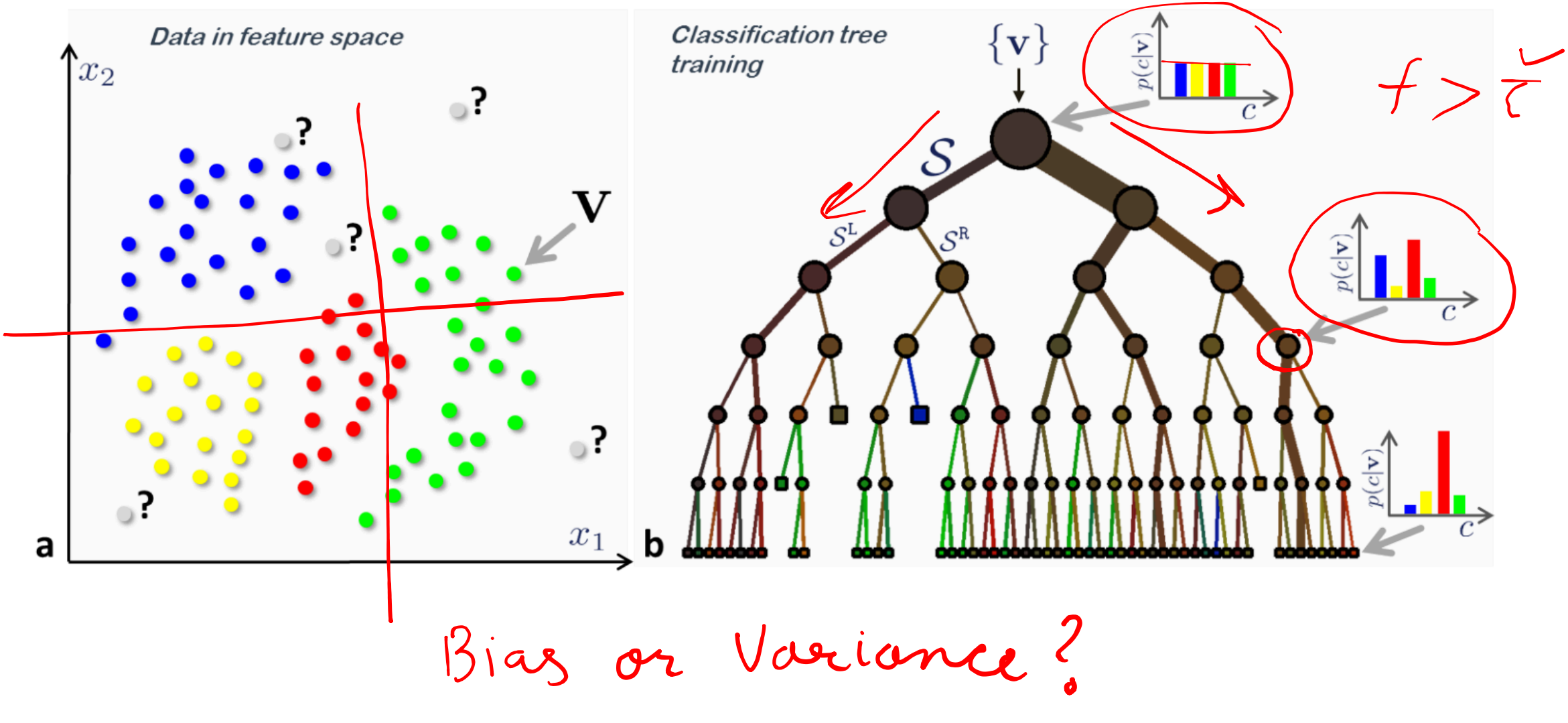
Statistical Methods in AI (CSE 471)
Ensemble Methods

Vineet Gandhi

Many slides and figures from: Dr. Markus Kalisch ETH Zurich, Criminisi et al. Microsoft Research, StatQuest with Josh Starmer

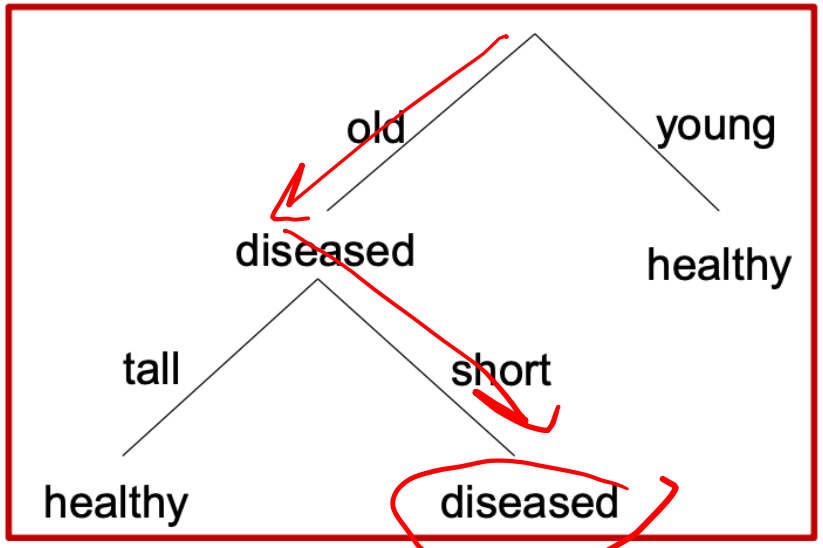
Single Tree



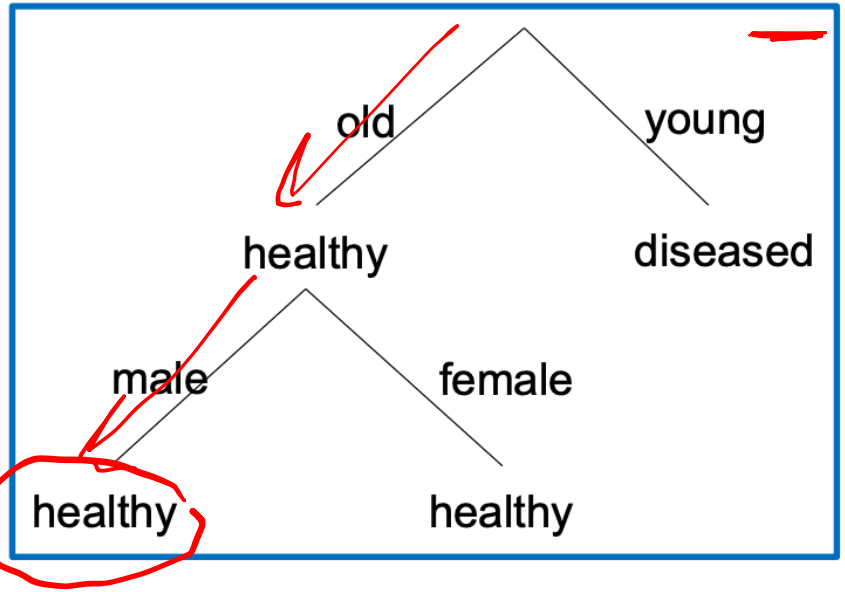
Bias or Variance?

Intuition of Random Forest

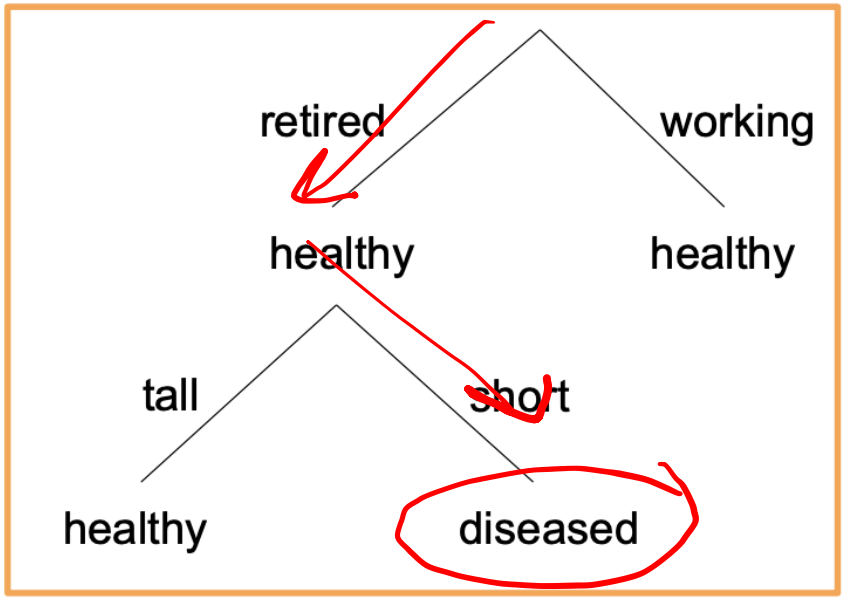
Tree 1



Tree 2



Tree 3



New sample:

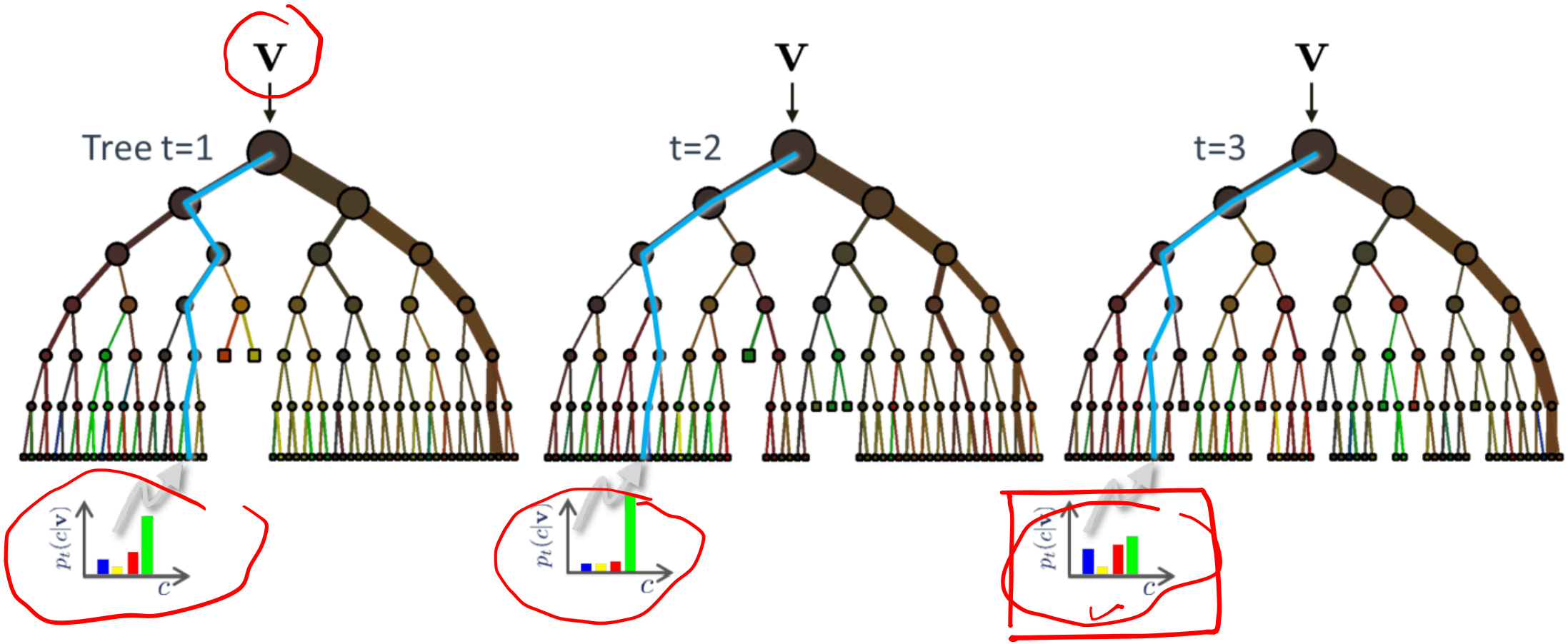
old, retired, male, short

Tree predictions:

diseased, healthy, diseased

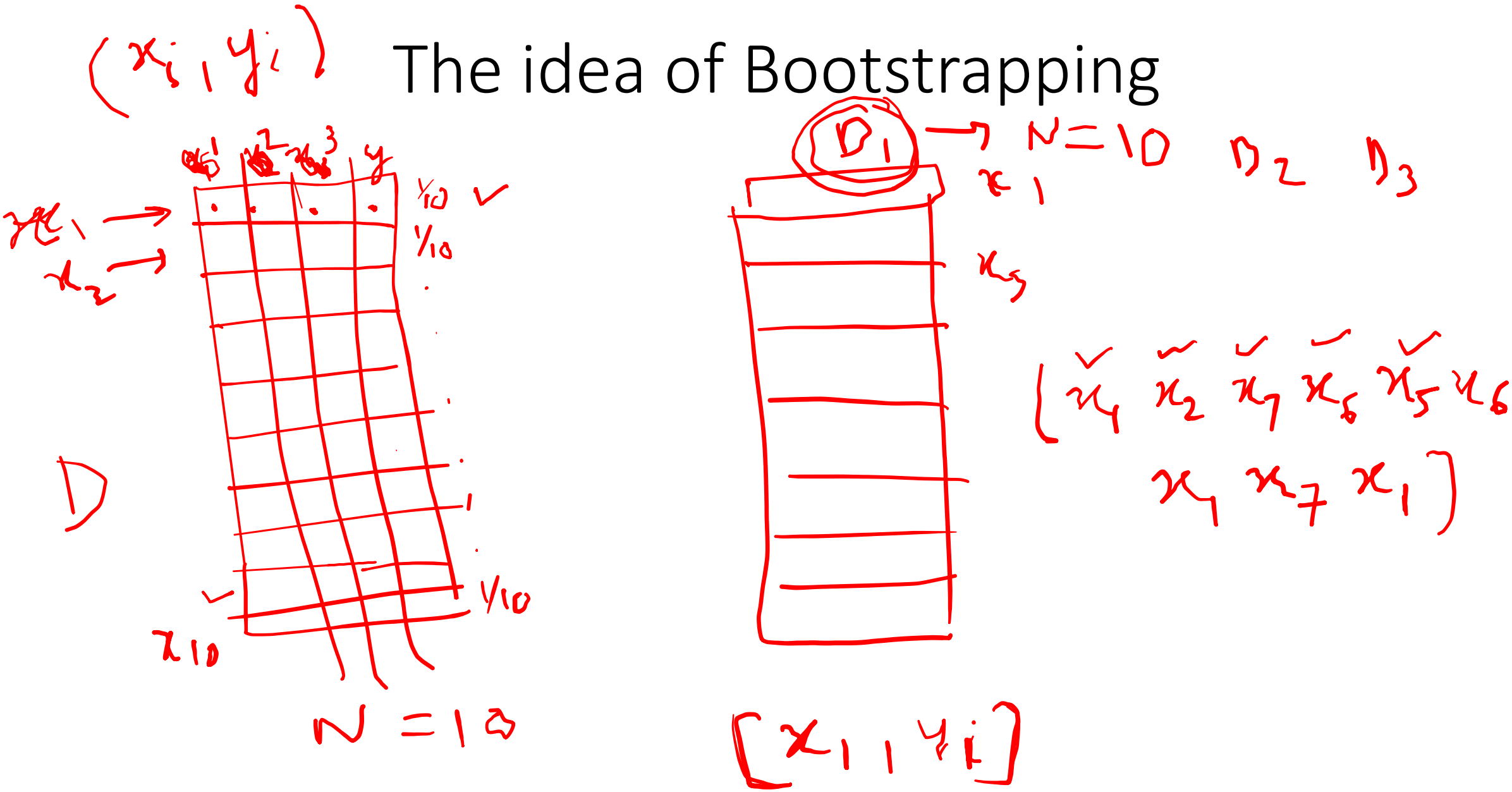
Majority rule:

diseased



$$p(c|\mathbf{v}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{v}).$$

The idea of Bootstrapping



Bootstrap aggregating = Bagging

- Generate many training sets by sampling with replacement from the given training data
- Given the original data of n points, we generate B number of training sets, each of size n' , by randomly sampling from the given dataset
- We learn model on each of the B training sets
- Final error could be average of errors of all the models

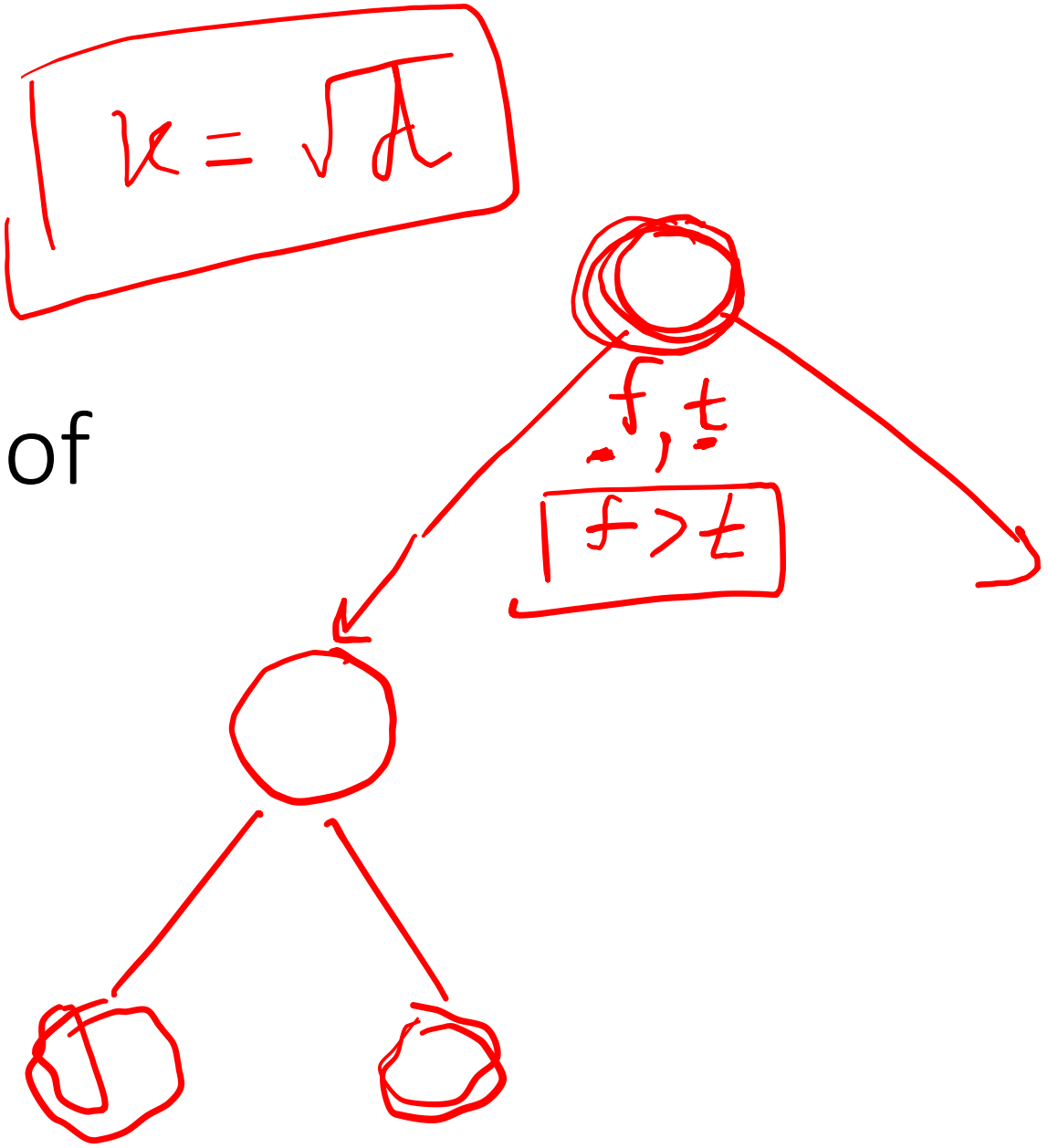
$h_1 \rightarrow B_1$

$h_2 \rightarrow D_2$

$$y_i = \frac{1}{M} \sum_{i=1}^{M} h_i(x_i)$$

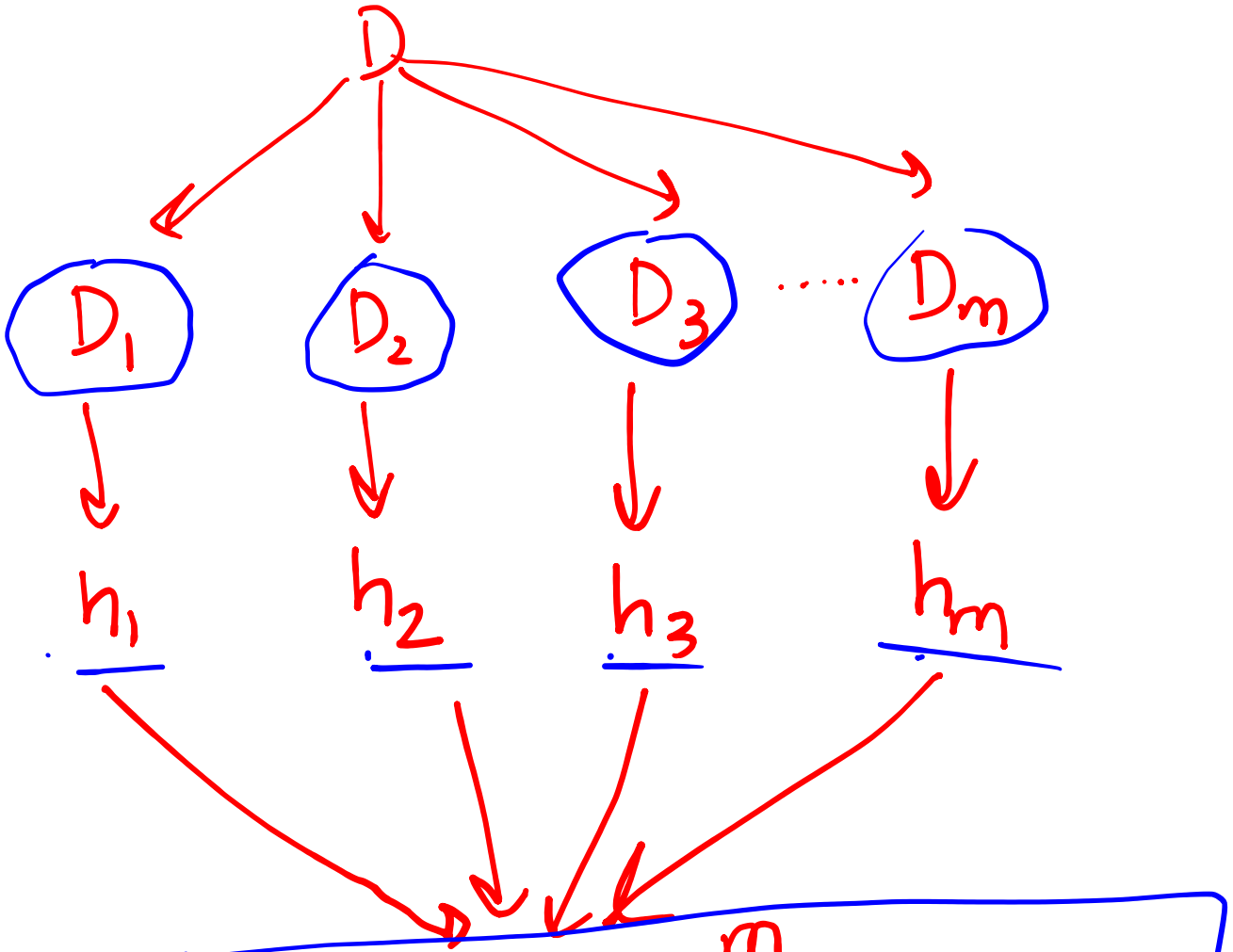
$\{-1, 1\}$

The idea of random variable selection

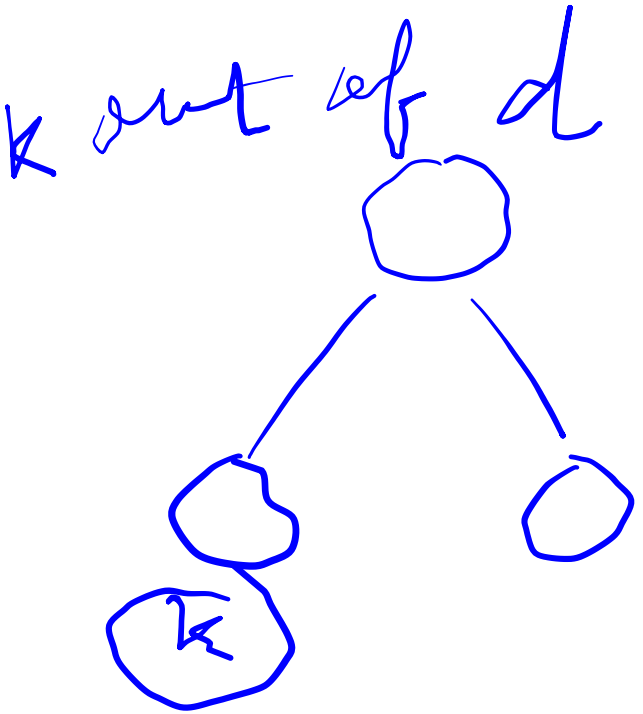


$k \ll d$

m | F x
- R | W
- B_a | A_g
S | T x



$$h(x) = \frac{1}{m} \sum_{j=1}^{m} h_j(x)$$



The Random Forest Algorithm

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

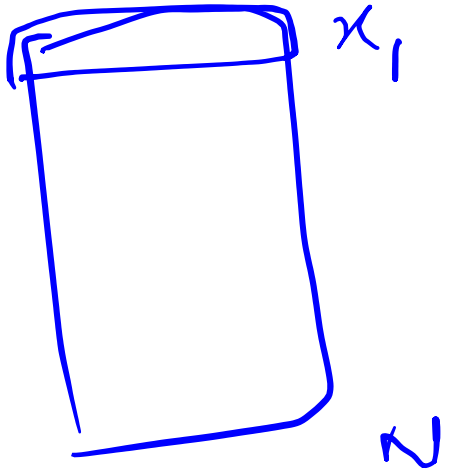
Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

Differences to standard tree

- Train each tree on bootstrap resample of data
  (Bootstrap resample of data set with N samples:
  Make new data set by drawing **with replacement** N samples; i.e., some samples will probably occur multiple times in new data set)
- For each split, consider only m randomly selected variables
- Don’t prune
- Fit B trees in such a way and use average or majority voting to aggregate results

The idea of Out of bag error

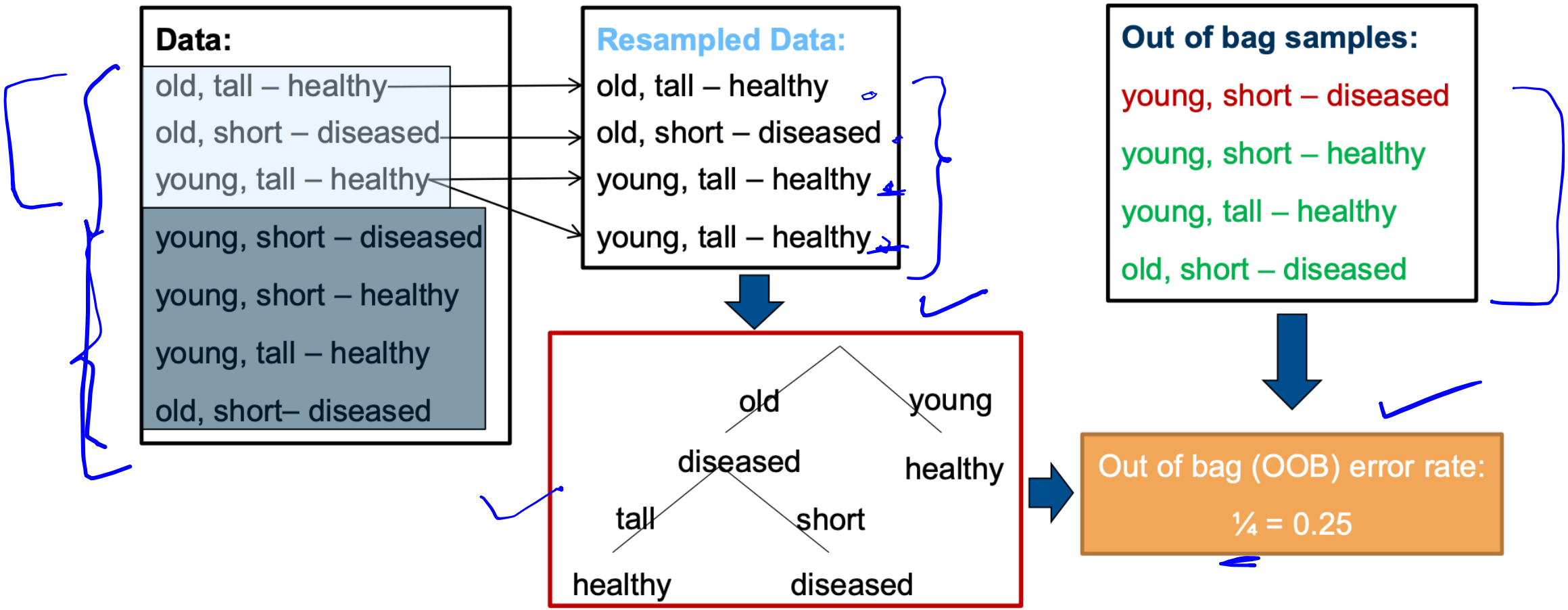


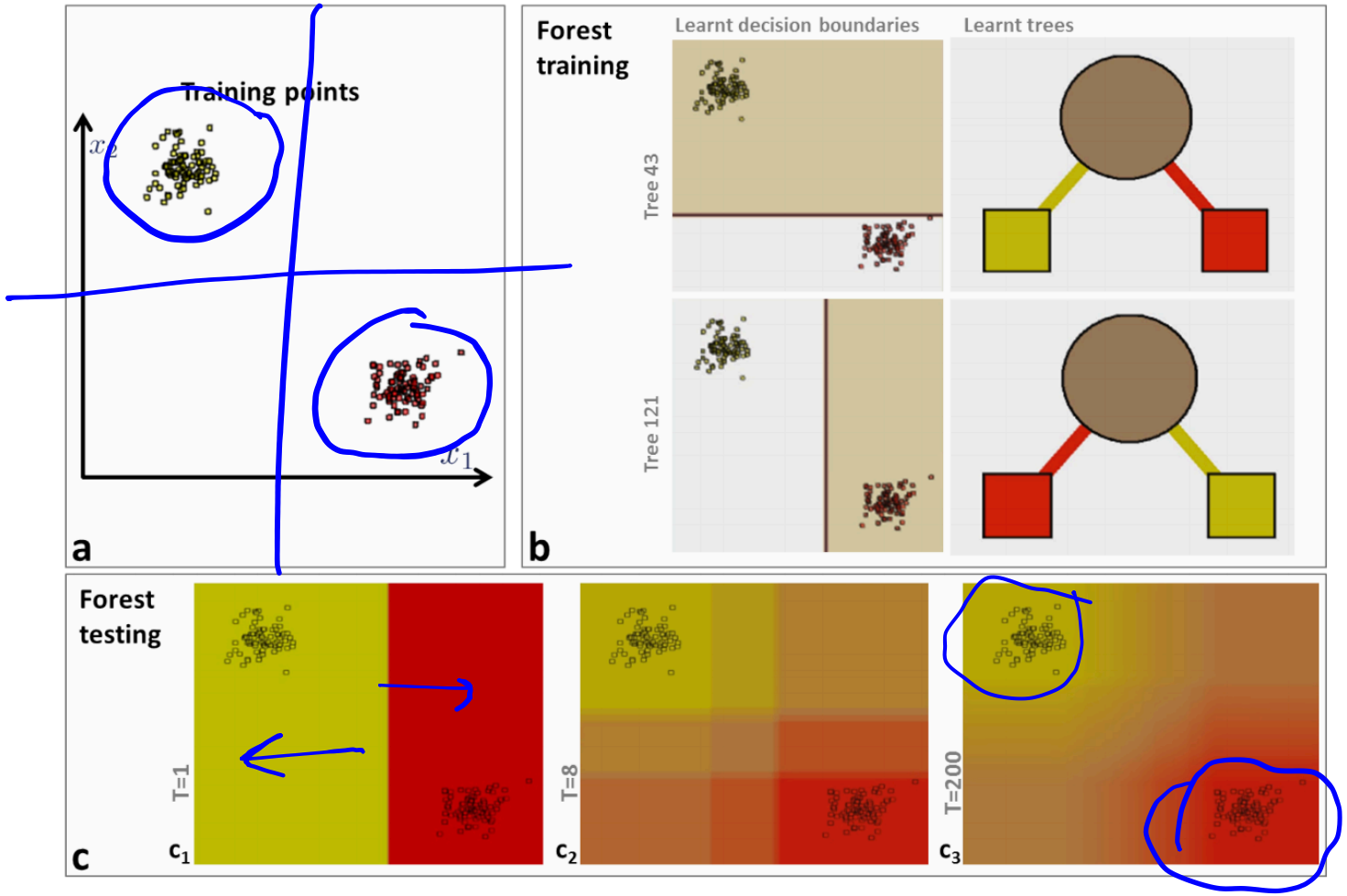
B_1 B_2 B_3 -- . . B_4 - - . . B_M

T_1 T_2 T_3 T_4 T_M

Estimating generalization error: Out-of bag (OOB) error

- Similar to leave-one-out cross-validation, but almost without any additional computational burden
- OOB error is a random number, since based on random resamples of the data





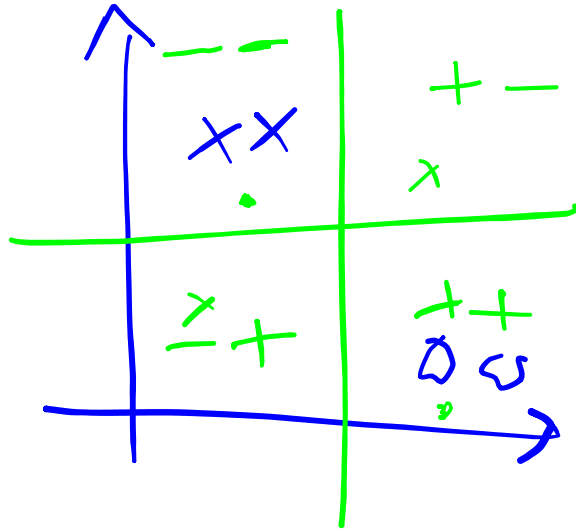
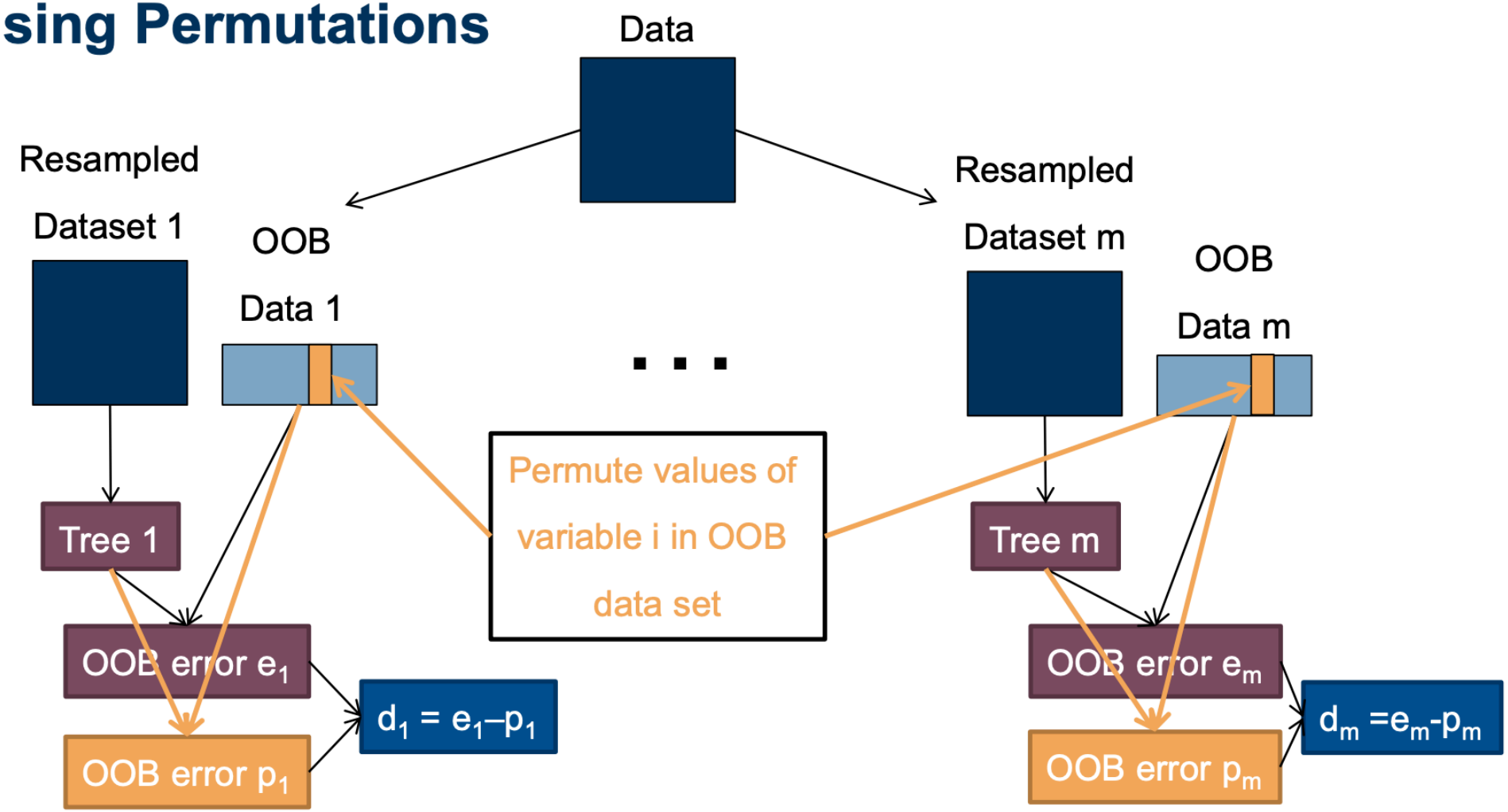


Fig. 3.3: **A first classification forest and the effect of forest size T.** **(a)** Training points belonging to two classes. **(b)** Different training trees produce different partitions and thus different leaf predictors. The colour of tree nodes and edges indicates the class probability of training points going through them. **(c)** In testing, increasing the forest size T produces smoother class posteriors. All experiments were run with $D = 2$ and axis-aligned weak learners. See text for details.

One feature selection

https://blog.datadive.net/selecting-good-features-part-iii-random-forests/

Variable Importance for variable i using Permutations



$$\bar{d} = \frac{1}{m}\sum_{i=1}^{m} d_i$$

$$s_d^2 = \frac{1}{m-1}\sum_{i=1}^{m}(d_i - \bar{d})^2$$

$$v_i = \frac{\bar{d}}{s_d}$$

Trees vs. Random Forest

+ Trees yield insight into decision rules
+ Rather fast
+ Easy to tune parameters

- Prediction of trees tend to have a high variance

+ RF as smaller prediction variance and therefore usually a better general performance
+ Easy to tune parameters

- Rather slow
- “Black Box”: Rather hard to get insights into decision rules

More reasons, why RF's are cool

- Works out of the box [: no feature scaling
  : categorical data]

- Only two main hyperparameters
  i.e. K and m

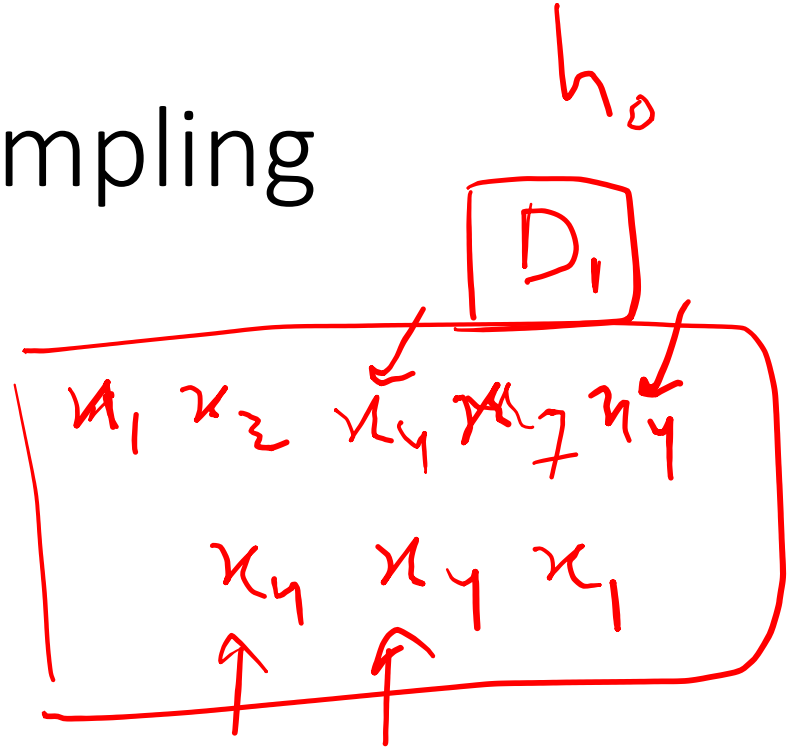
$K = \sqrt{d}$

$m \rightarrow$ as much as we can afford

Boosting

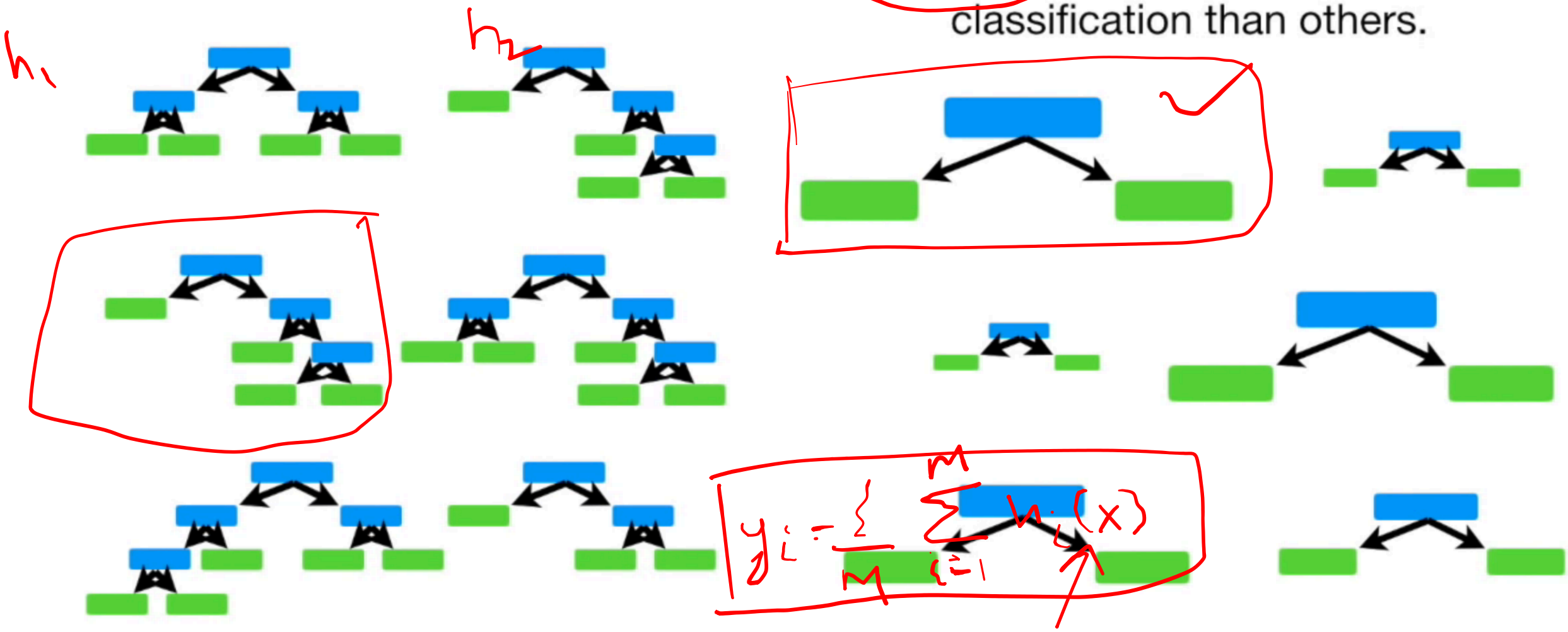
The idea of probabilistic sampling

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

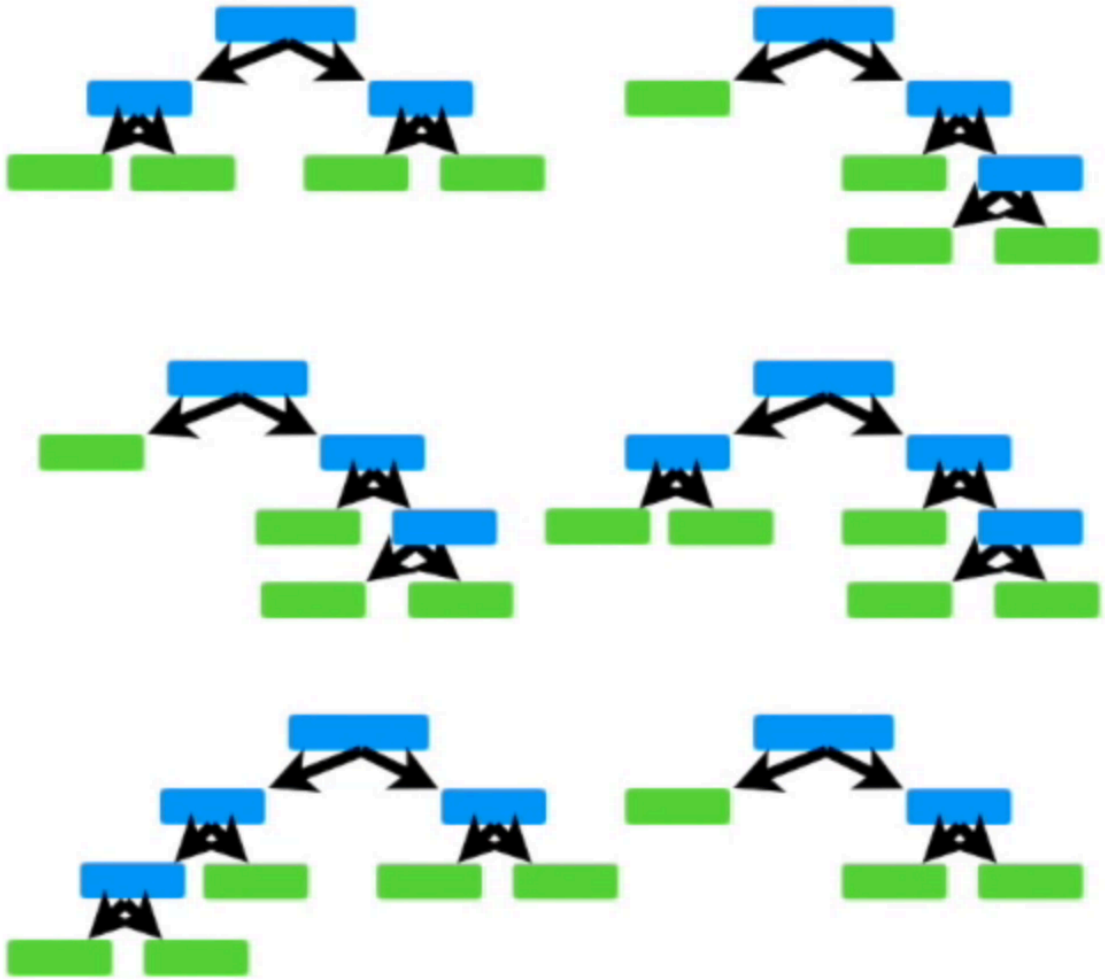


In a **Random Forest**, each tree has an equal vote on the final classification.

In contrast, in a **Forest of Stumps** made with **AdaBoost**, some stumps get more say in the final classification than others.

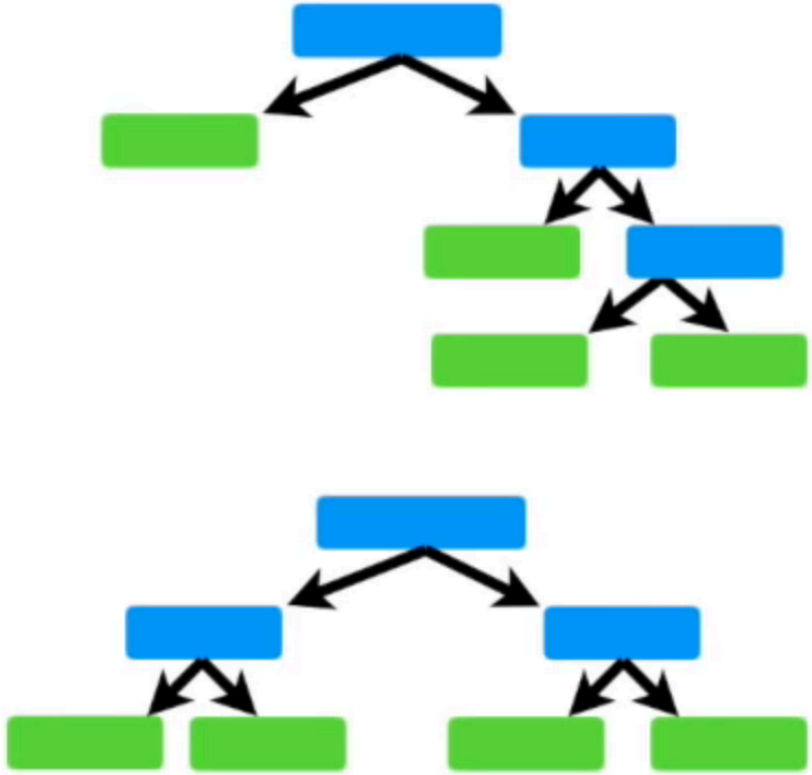


Lastly, in a **Random Forest**, each decision tree is made independently of the others.

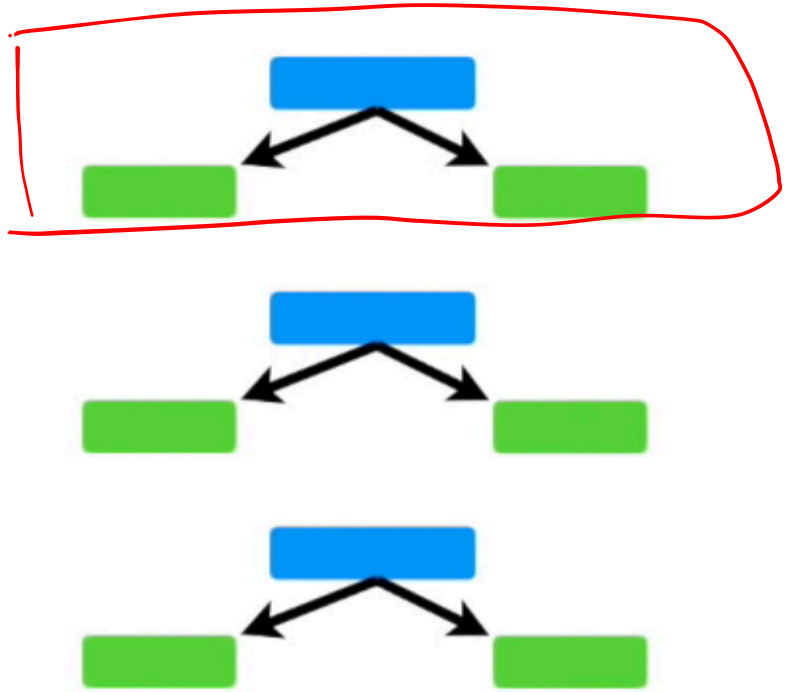


In contrast, in a **Forest of Stumps** made with **AdaBoost**, order is important.

In a **Random Forest**, each time you make a tree, you make a full sized tree.



In contrast, in a **Forest of Trees** made with **AdaBoost**, the trees are usually just a **node** and two **leaves**.



Adaboost

- In Adaboost we assign (non-negative) weights to points in the data set, which are then normalised so that they sum to one
- Iteratively learn new classifier
- In each iteration, we generate a training set by sampling from the data using the weights
- After learning the current classifier, we increase the (relative) weights of the data points which are misclassified by the current classifier
- The final classifier is the weighted majority voting by all classifiers

Adaboost

- Let $\{(X_1, y_1),\ldots,(X_n,y_n)\}$ be the data. We take y_i in $\{-1, +1\}$
- Let $w_i(k)$ denote the weight for the ith data point at kth iteration
- Let h_k be the classifier learnt at kth iteration, we take $h_k(X)$ in $\{-1,+1\}$
- We assume error rate of each classifier on its training data is less than 0.5

Adaboost algorithm

1. Initialize: $w_i(1) = 1/n$
2. For m =1:M do
   A. Generate a training set by sampling with $\{w_i(m)\}$
   B. Learn classifier h_m using the training set
   C. Let

$$\xi_m = \sum_{i=1}^{n} w_i(m) I_{[y_i \neq h_m(X_i)]}$$

We assume it be less than 0.5

   D. Set

$$\alpha_m = \ln\left(\frac{1-\xi_m}{\xi_m}\right) \qquad \alpha_m > 0$$

   E. Update Weights

$$w_i'(m+1) = w_i(m) exp(\alpha_m I_{[y_i \neq h_m(X_i)]})$$

$$= w_i(m) \cdot exp(-\alpha_m \cdot y_i \cdot h(x_i))$$

$$w_i(m+1) = \frac{w'(m+1)}{\sum_i w'(m+1)}$$

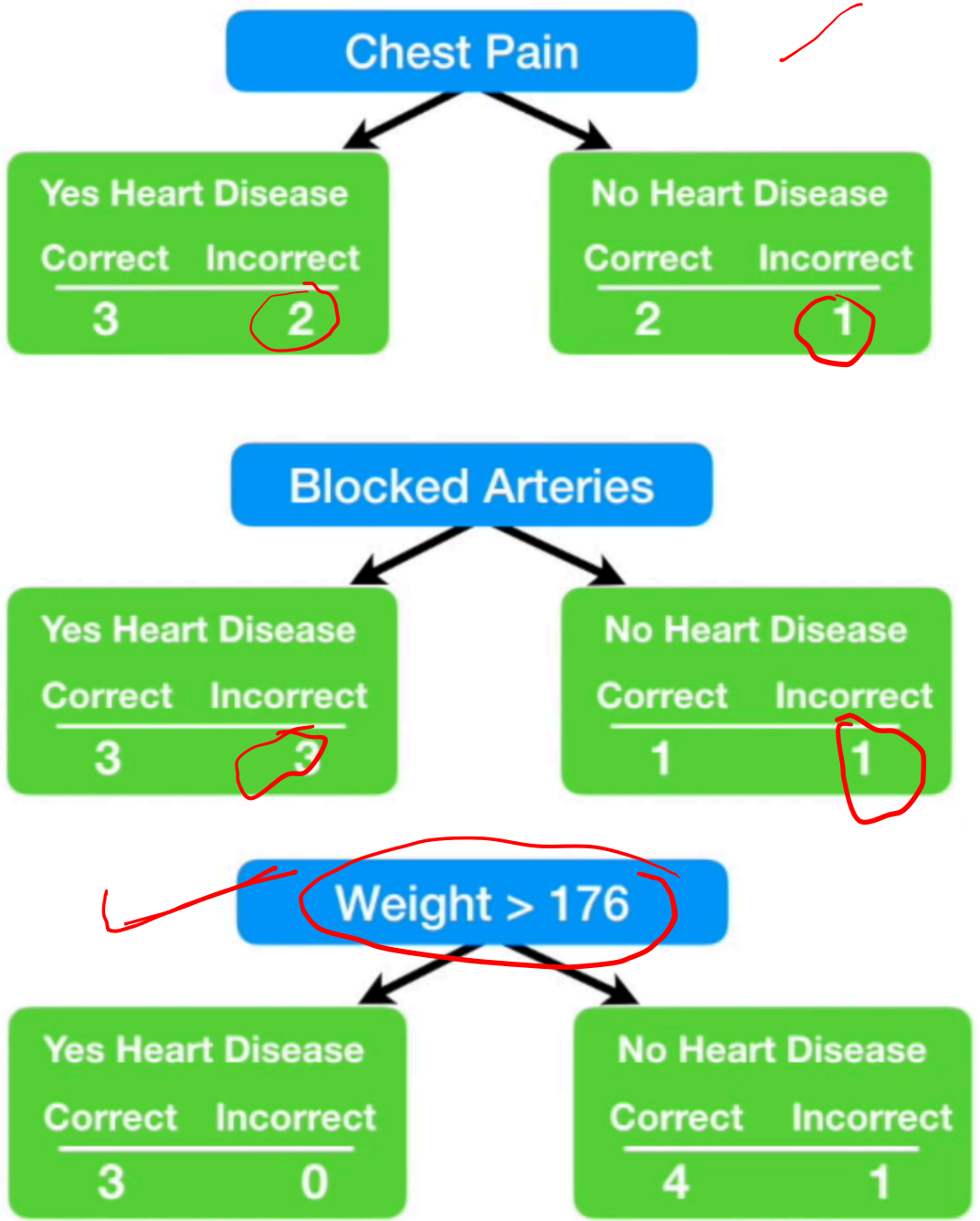
Adaboost algorithm

3. Final classifier

$$h(X) = sgn(\sum_{m=1}^{M} \alpha_m h_m(X))$$

Lets take an example

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8



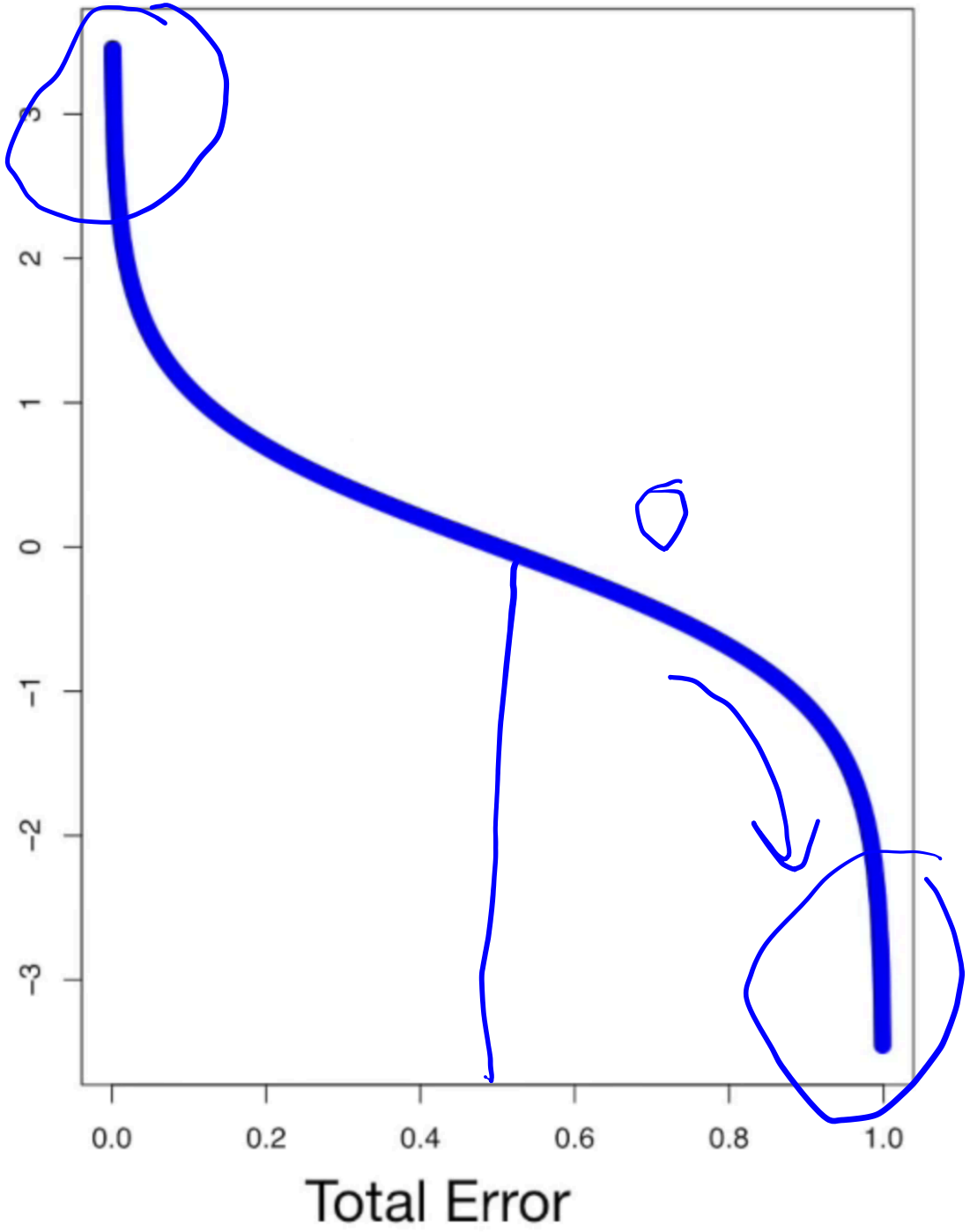
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

Thus, in this case, the **Total Error** is **1/8**.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

Thus, in this case, the **Total Error** is **1/8**.

$$\text{Amount of Say} = \frac{1}{2}\log_2\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$



$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$$

$$\text{Amount of Say} = \frac{1}{2}\log(7) = 0.97$$

$\alpha_m = 0$

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

$$\text{New Sample Weight} = \text{sample weight} \times e^{\text{amount of say}}$$

$$= \frac{1}{8} e^{0.97} = \frac{1}{8} \times 2.64 = 0.33$$

$$\text{New Sample Weight} = \text{sample weight} \times e^{-\text{amount of say}}$$

$$= \frac{1}{8} e^{-0.97} = \frac{1}{8} \times 0.38 = 0.05$$

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight	New Weight	Norm. Weight
Yes	Yes	205	Yes	1/8	0.05	0.07
No	Yes	180	Yes	1/8	0.05	0.07
Yes	No	210	Yes	1/8	0.05	0.07
Yes	Yes	167	Yes	1/8	0.33	0.49
No	Yes	156	No	1/8	0.05	0.07
No	Yes	125	No	1/8	0.05	0.07
Yes	No	168	No	1/8	0.05	0.07
Yes	Yes	172	No	1/8	0.05	0.07

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease
No	Yes	156	No
Yes	Yes	167	Yes
No	Yes	125	No
Yes	Yes	167	Yes
Yes	Yes	167	Yes
Yes	Yes	172	No
Yes	Yes	205	Yes
Yes	Yes	167	Yes

Ultimately, the patient is classified as **Has Heart Disease** because this is the larger sum.

Has Heart Disease

Total = 2.7

Total = 1.23

Does Not Have Heart Disease

Amount of Say

0.97

0.32

0.78

0.63

Amount of Say

0.41

0.82