

# Machine Learning Methods

## **P160B124**

### Bagging

### Random Forest, Boosting

assoc. prof. dr. Tomas Iešmantas

tomas.iesmantas@ktu.lt

Room 319

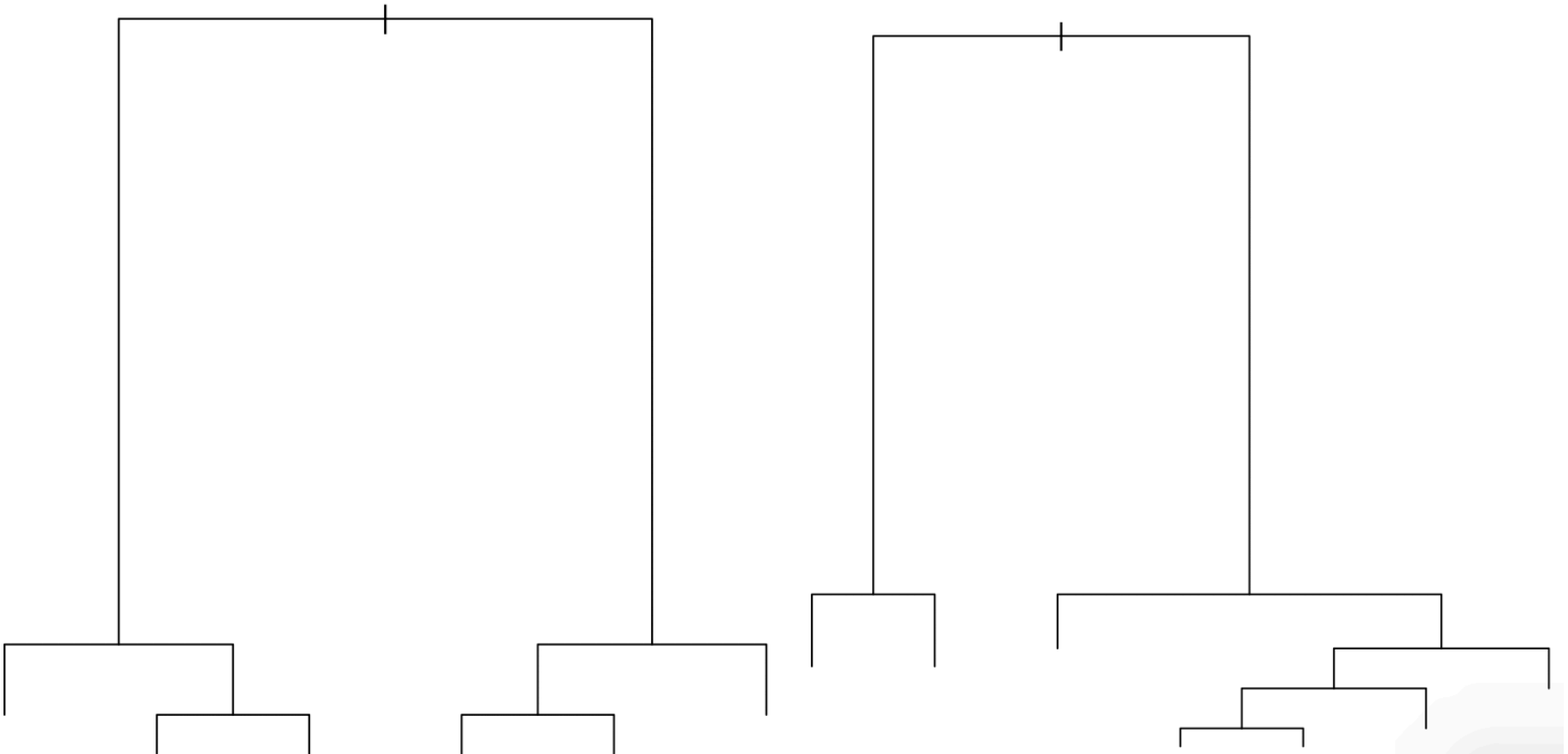
# Major issue of Decision Tree

- Decision trees are highly unstable – different training set (from the same population) will grow different trees. We don't want that!

93 %

High variance

79 %

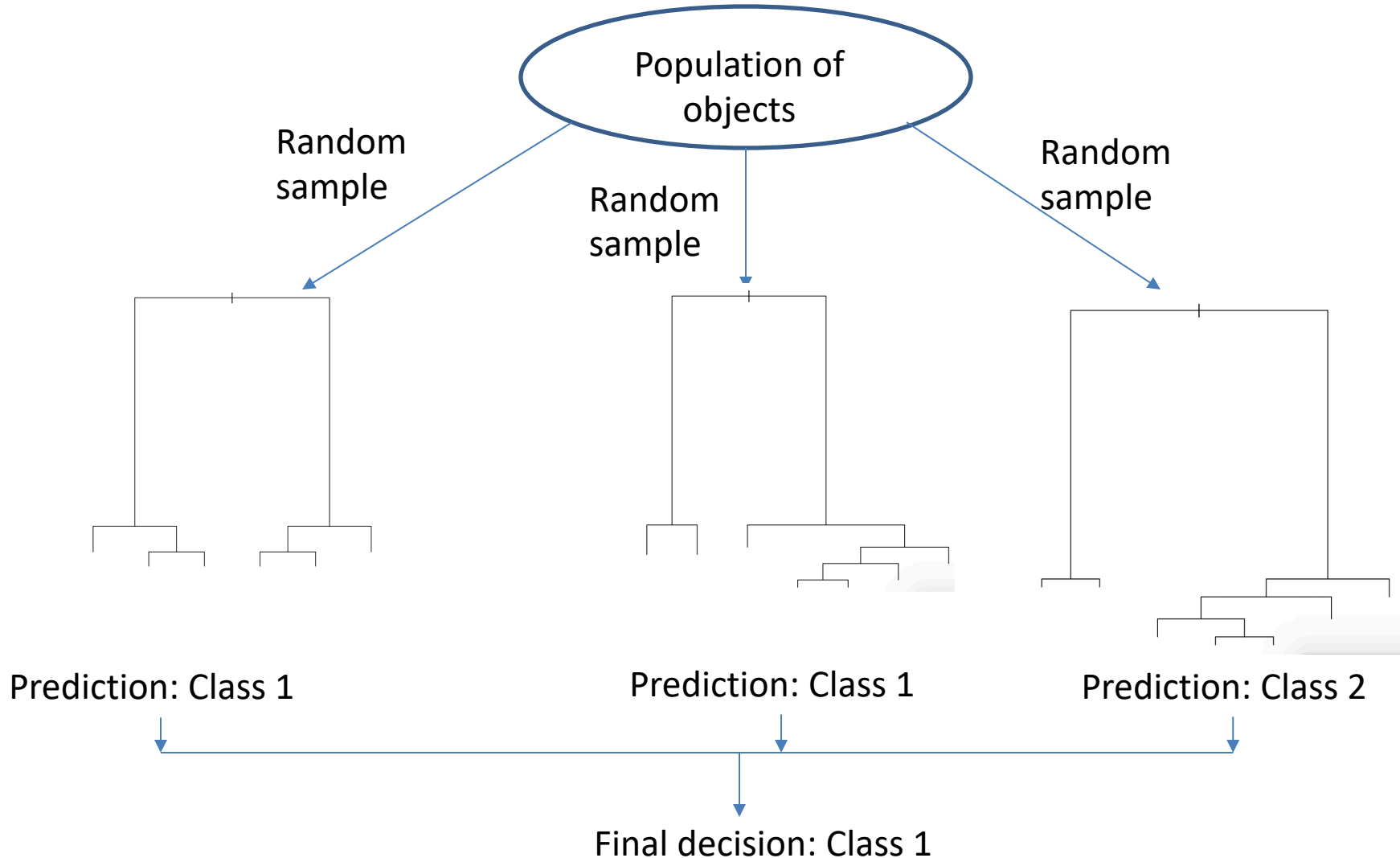


# Variance reduction

- A known fact: averaging more and more independent random variables will reduce the overall variance.
- Decision trees are also random variables because they are grown from random training sets.
- Thus, **a strategy**: draw  $T$  independent training sets from the population and build a tree for each training set. Use entire set of decision trees as a model;

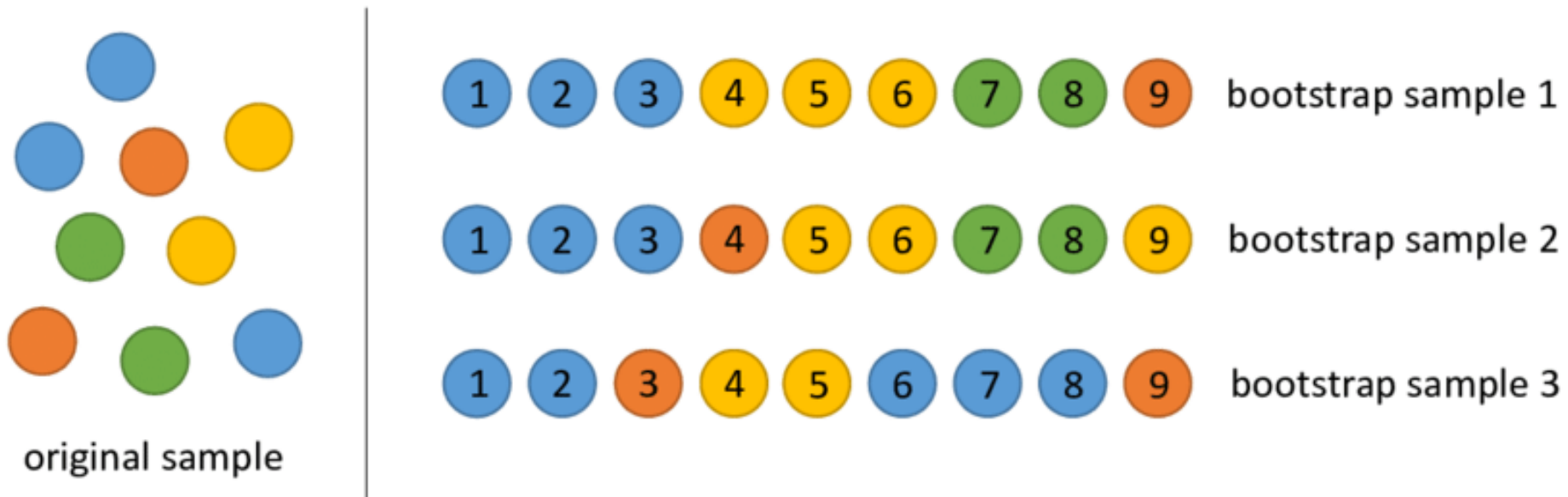
# Variance reduction

- In the regression setting, average the predictions; in the classification – use majority voting.



# Bootstrapping

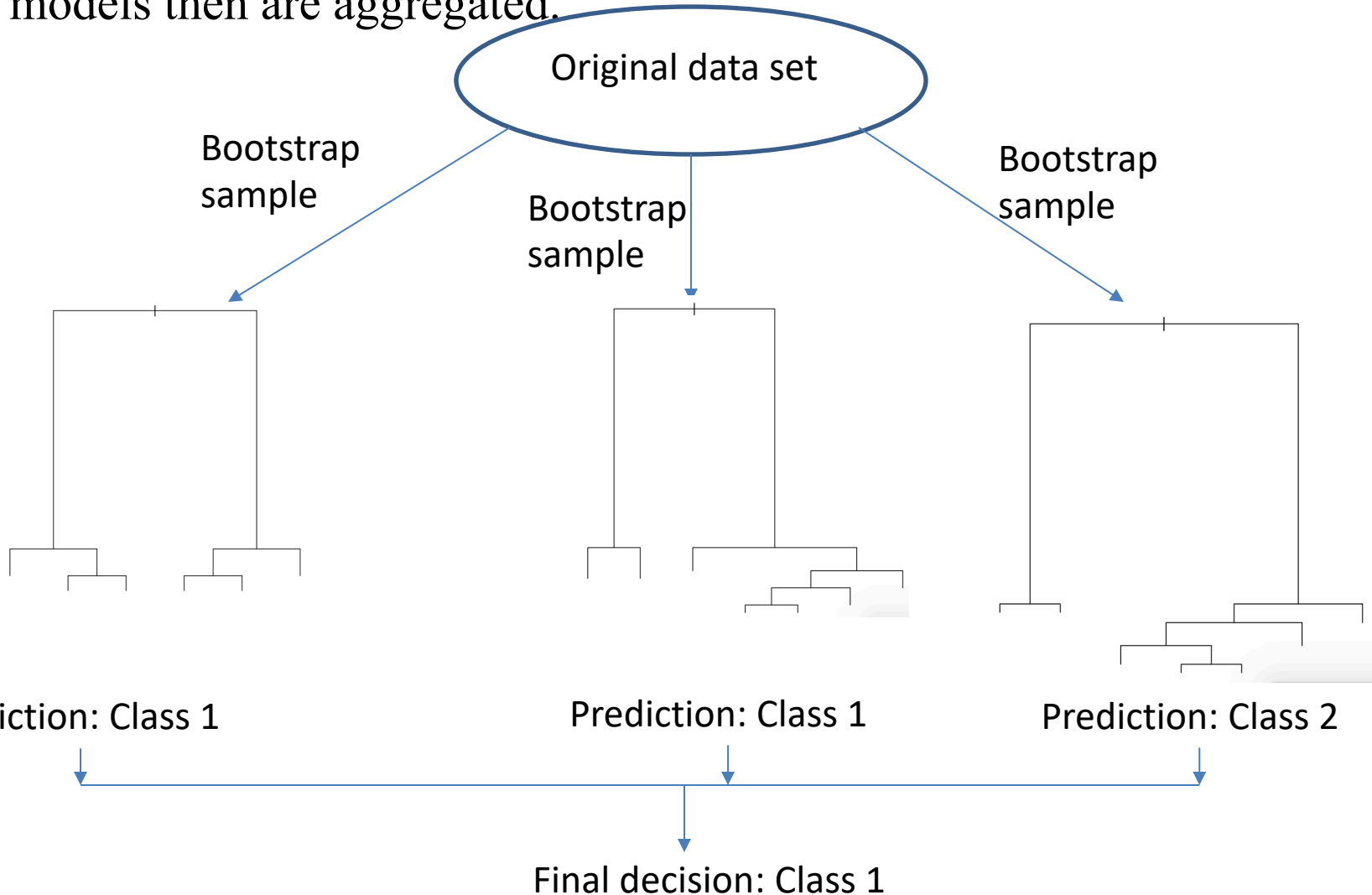
- Problem: we don't have an access to the entire population. Instead we have a finite dataset with  $N$  entries.
- Solution: sample training set of size  $N$  from the initial dataset. But sample with replacement. This is called a **bootstrap sample**.
- It's like picking at random  $N$  balls from a collection of  $N$  balls, but once you pick at random a ball you put it back.



- Some data points may not be picked at all in a single bootstrap sample.

# Bagging

- **Bootstrap aggregating** – **bagging** (Breiman, 1996). Bagging is a method when some number of models are trained on bootstrap samples and those models then are aggregated.



# Bagging

Bagging enjoys the benefits of:

- High expressiveness – by using full trees each model is able to approximate complex functions and decision boundaries;
- Low variance – averaging (*see note below*) the prediction of all the models reduces the variance in the final prediction, assuming that we choose a sufficiently large number of trees.

**Note:** averaging works as a variance reduction tool in regression, but in classification it's more complicated and not always true.

# Random Forest

- The concept of bagging works when the trees are independent of each other;
- However, since training samples are bootstrapped from the same original dataset this is not particularly true (also each tree is grown from the same set of features);
- Averaging dependent random variables may not reduce the variance, depending on the strength of the dependency.
- A decorrelation strategy is needed;



# Random Forest: decorrelation or trees

- In order for the trees to be as independent of each other as possible, a random subset of  $\sqrt{m}$  features is selected for node split.
- In the regular decision tree a node split is chosen from entire set of features. But this means that many trees in the bag will be similar to each other (at least in the lower branches).
- Instead, when splitting the node use a smaller and random subset of features to calculate the best split.

# Random Forest: the algorithm

For  $b = 1$  to  $B$ :

- a) Draw a bootstrap sample  $Z^*$  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:
  1. Select  $p$  variables at random from the  $m$  possible variables;
  2. Pick the best variable/split-point among the  $p$ ;
  3. Split the node into two child-nodes
- c) Output the ensemble of trees  $\{T_b\}_{b=\overline{1,B}}$ .

To make a prediction at a new  $x$ :

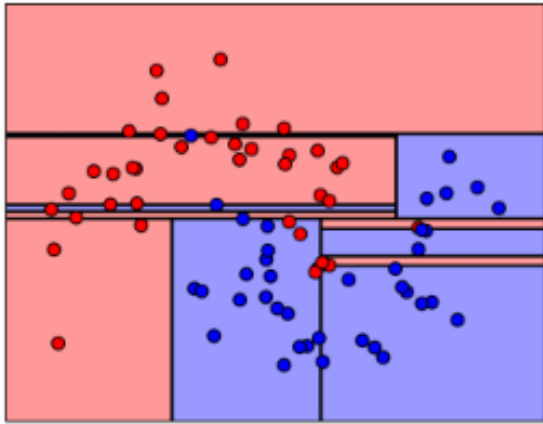
Let  $\hat{f}_b(x)$  be the class prediction of the  $b^{th}$  random-forest tree. Then

$$\hat{f}_{rf}(x) = \text{majority vote}\{\hat{f}_b(x)\}_{b=\overline{1,B}}$$

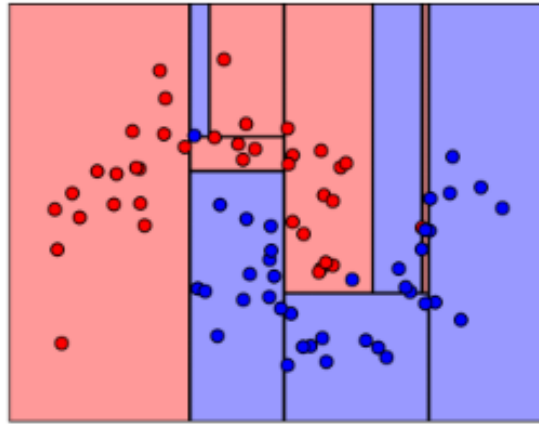
# Single tree vs random forest

Random forest produce a more regular decision boundary than a separate trees:

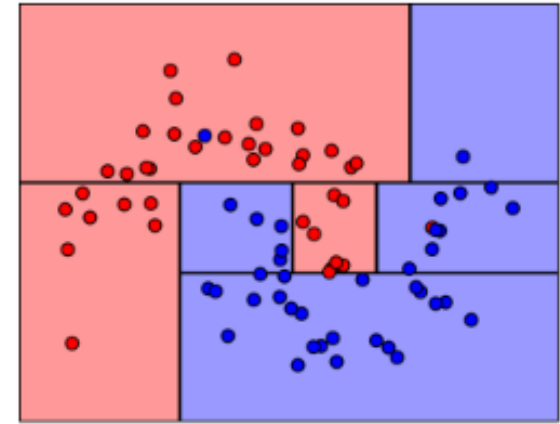
**tree 0**



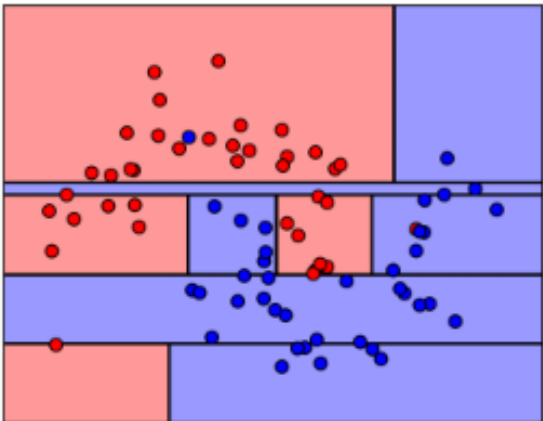
**tree 1**



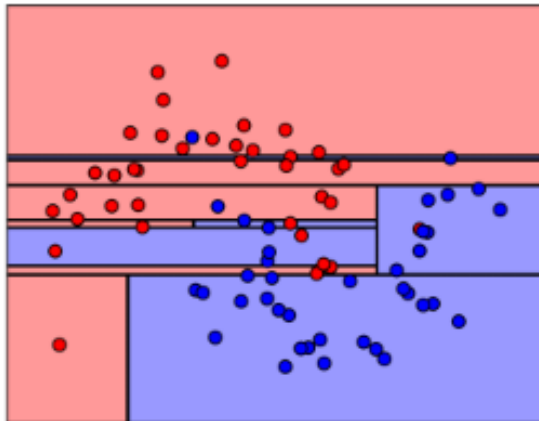
**tree 2**



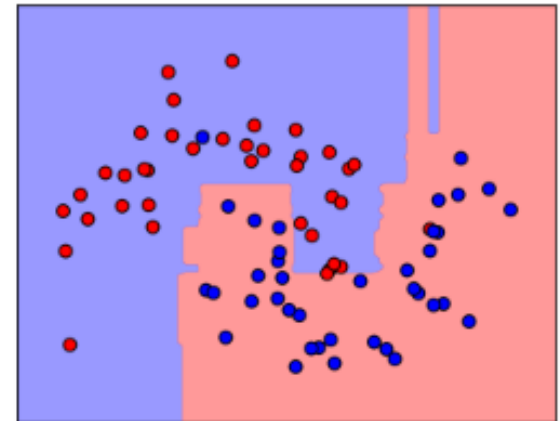
**tree 3**



**tree 4**

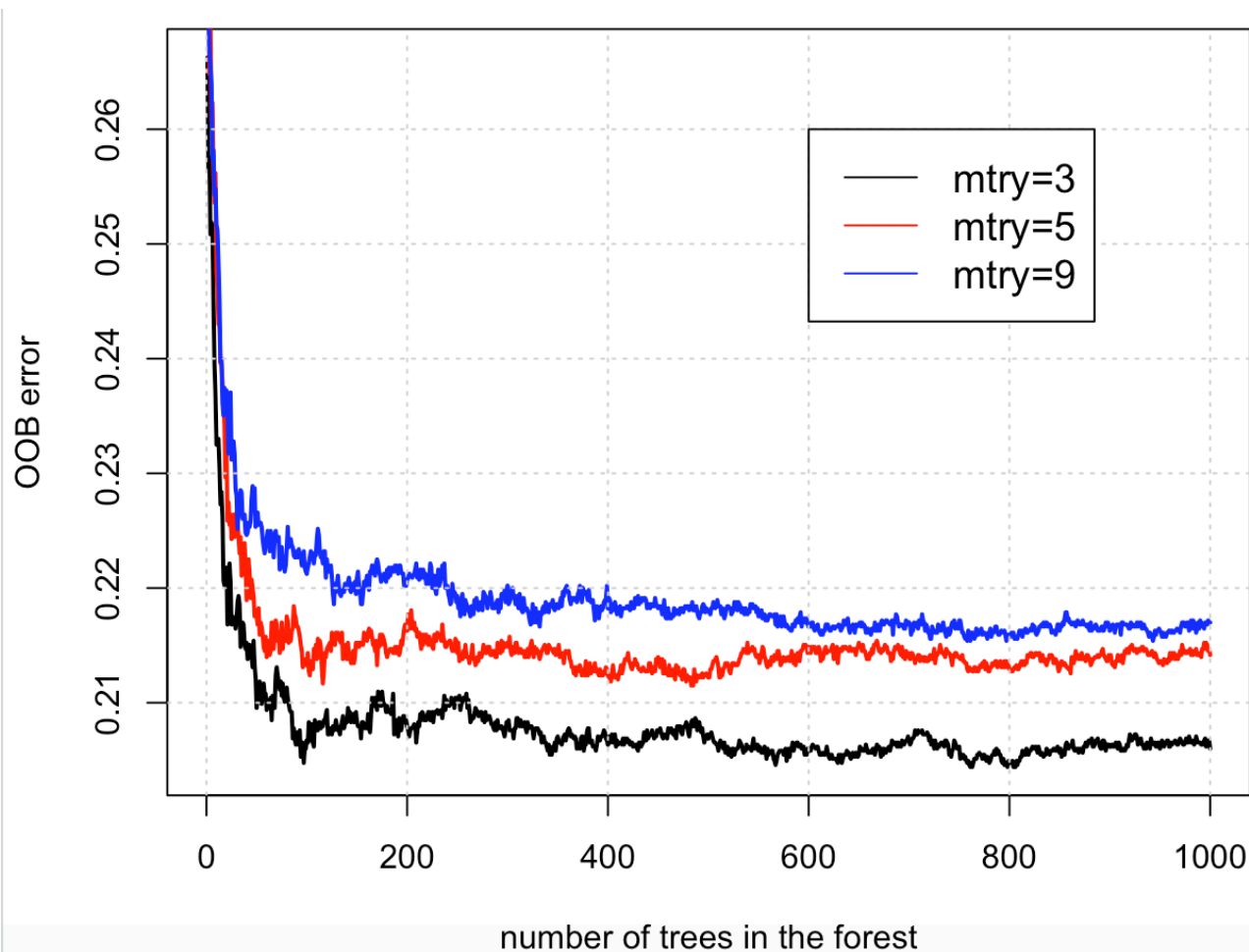


**random forest**



# Bagging: example

- Data set: customer churn data set from ANN lab.
- Overall error for a single decision tree is 27 %.
- Default choice of random subsample of variables is often  $\approx \sqrt{m}$ .



# Out-Of-Bag error

- Whenever we sample with replacement from a set, approximately 1/3 of the original data points are not picked.
- Not picked data points are called out of bag or OOB sample and it can be used to test the tree.



# Out-Of-Bag error

OOB error:

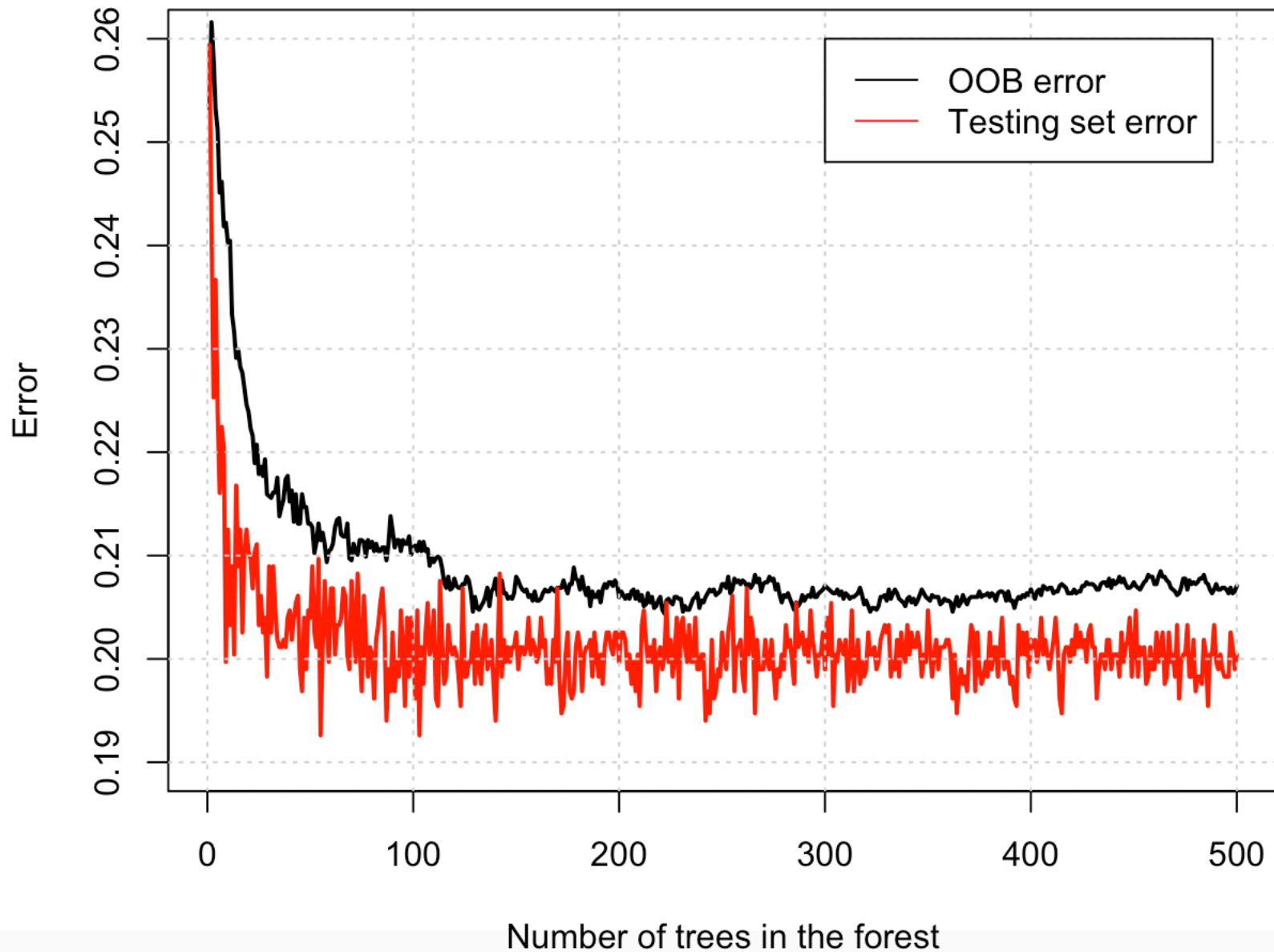
1. Test each  $x_i$  on those forest trees, which did not use this data point; calculate error estimate  $err_i$ .
2. Calculate average of error ( $err_1, \dots, err_N$ ).

This average error is called OOB error and is an estimate of test error.

- In Decision trees, SVM – 10 fold cross validation was used to estimate testing error and select hyperparameters. For random forest, hyperparameter selection is done by OOB error.



# OOB error vs. testing error



# Question 1

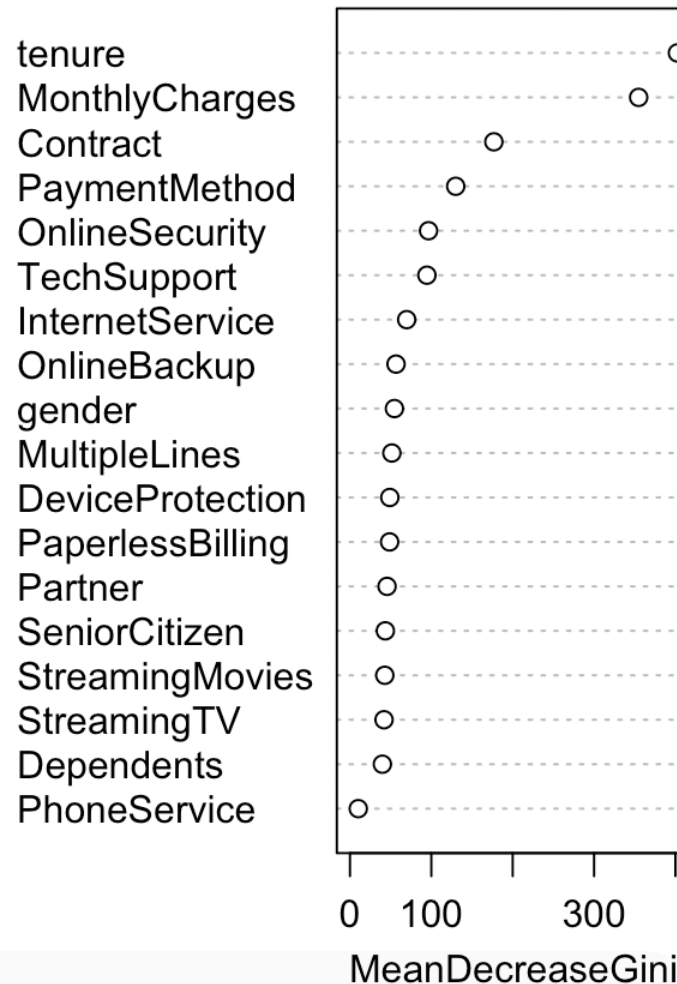
- A forest with 1000 trees was trained. Calculate the **OOB error** for the following training set:
- a) 32%;
  - b) 68%;
  - c) 10%;
  - d) Correct answer is not given

Feature vector	RF forest votes for class 1	RF forest votes for class 2	RF forest votes for class 3	True class
$x_1$	600	50	20	1
$x_2$	650	10	40	1
$x_3$	5	100	590	3
$x_4$	100	200	400	2
$x_5$	310	100	300	3



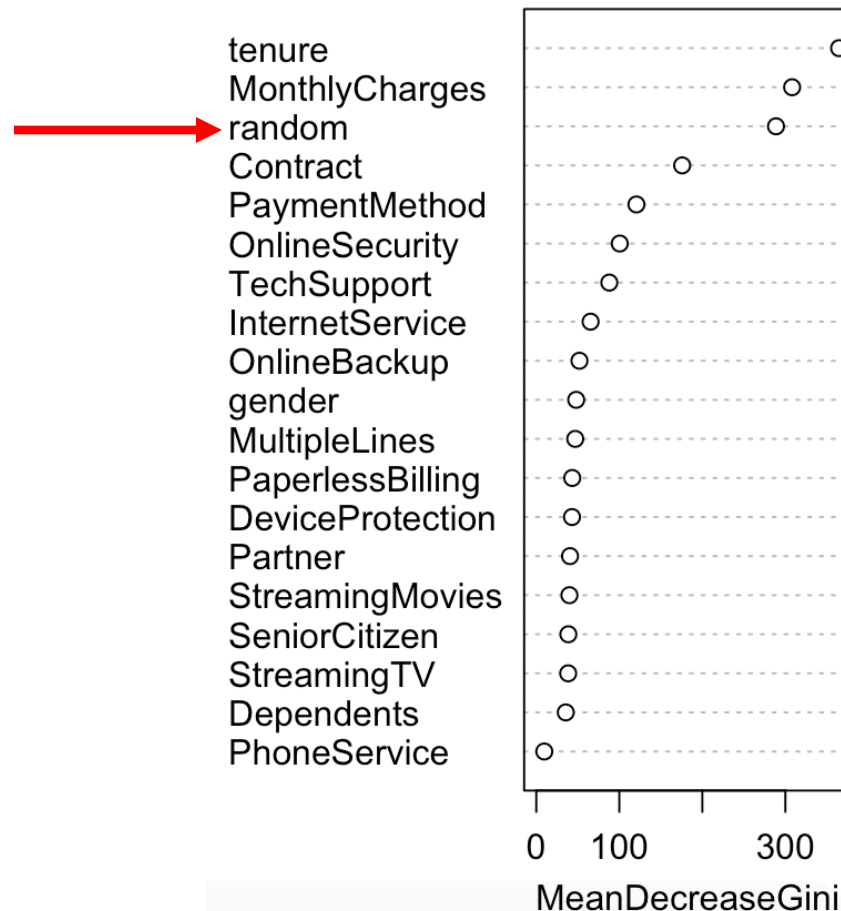
# Variable importance

- Just like in the case of a single decision tree, variable importance can be calculated for the entire forest.
- Mean Gini impurity decrease over entire forest.



# Why Gini importance can be misleading?

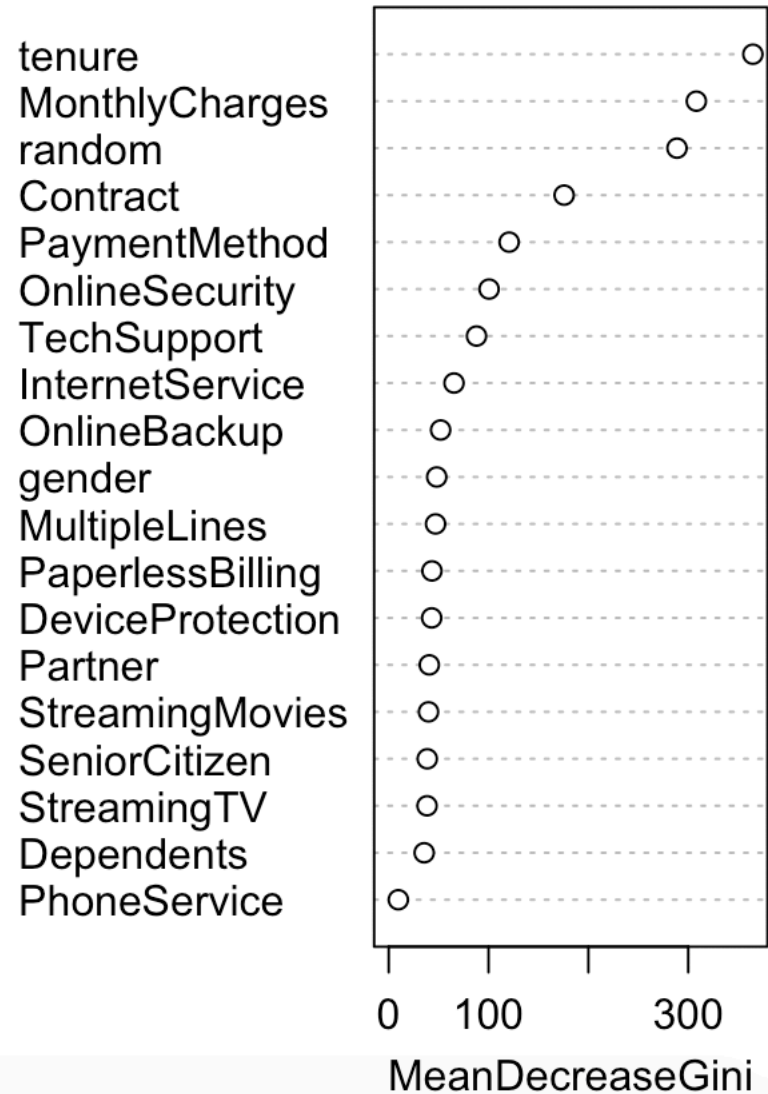
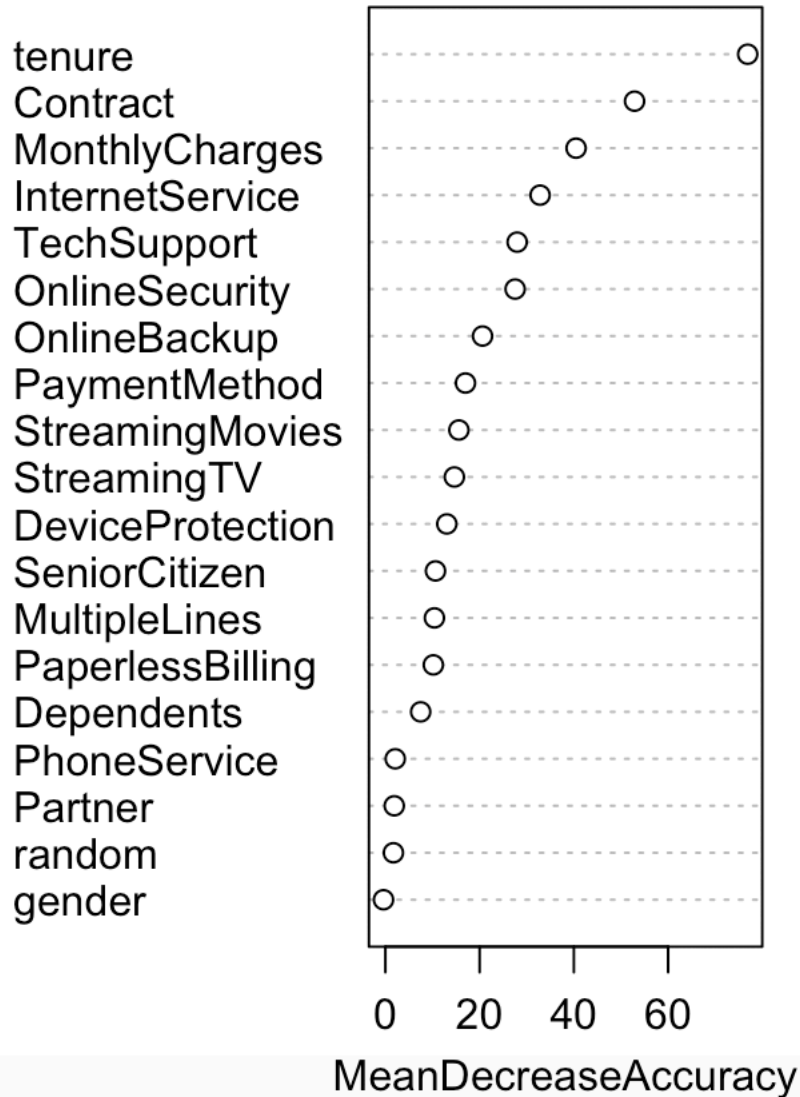
- Add a randomly generated column to the data. This column is completely uninformative since its values were generated independently of the class.
- What happened? Gini impurity is biased towards continuous variables or variables with many different categories.



# Why Gini importance can be misleading?

- Instead, use permutation importance: take one feature and randomly shuffle its values. Obtain the OOB error and calculate how much it decreased.
- Shuffling randomly feature values breaks its connection to the class variable – if the decrease in accuracy is large, then this feature is important.

# Why Gini importance can be misleading?



# Random forests do not overfit\*

- You can often find (especially in unreliable blogs, videos and so on) a saying that “random forests does not overfit”. **This is incorrect.**
- Correct formulation: random forests does not overfit when more and more trees are added.
- This means that when we grow larger and larger forest, the test error (or OOB error) will not increase.

# Random Forest: summary

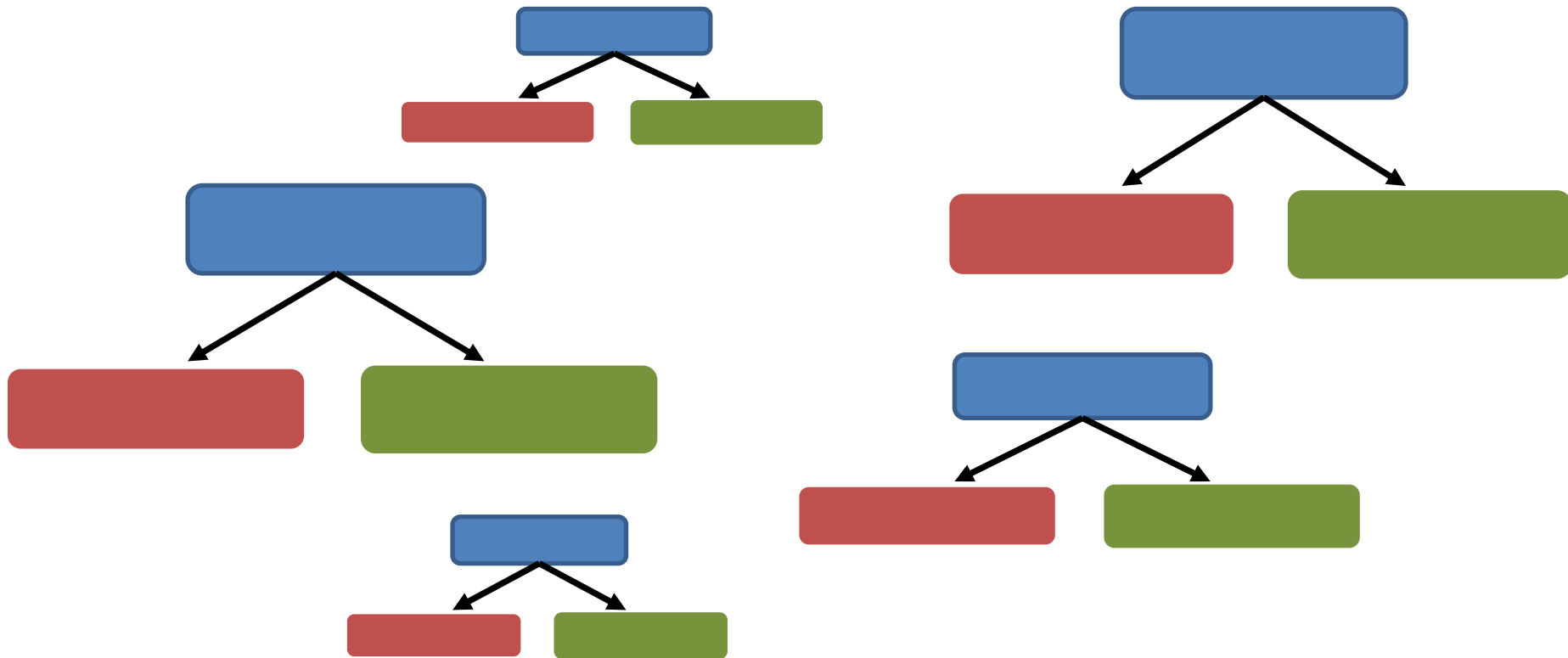
- RFs are a collection of decision trees, where the final decision is based on majority vote for classification or average value for regression;
- RFs use double randomization: bootstrap samples from the training dataset and random selection of subset of variables for split consideration;
- Double randomization serves as a decorrelation strategy in order to get as diverse forest as possible;
- RFs works by decreasing variance of a highly unstable model. RF for robust models does not improve accuracy.
- RF training process generates error approximation by calculating OOB errors;
- RFs variable importance is better assessed by the permutation importance;
- RFs does not overfit when increasing the complexity.

# Boosting

- Random forest can be thought as giving equal importance/contribution to each grown tree;
- Also, each tree is grown to a large size and thus generally the bias (how well the tree fits the data) is small. Random forest tries to reduce variance but not the bias.
- **Adaptive Boosting (AdaBoost)** tries to reduce the bias by creating an ensemble of weak learners by repeatedly emphasizing mispredicted instances.

# Weak learner

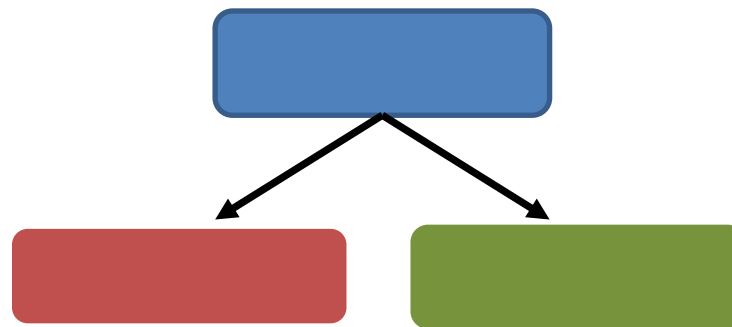
- A weak learner is a classifier which predicts only slightly better than random guessing.
- A stump – one node and two leaves – is such a classifier.
- AdaBoost is a forest of stumps (but each stump has different weights)





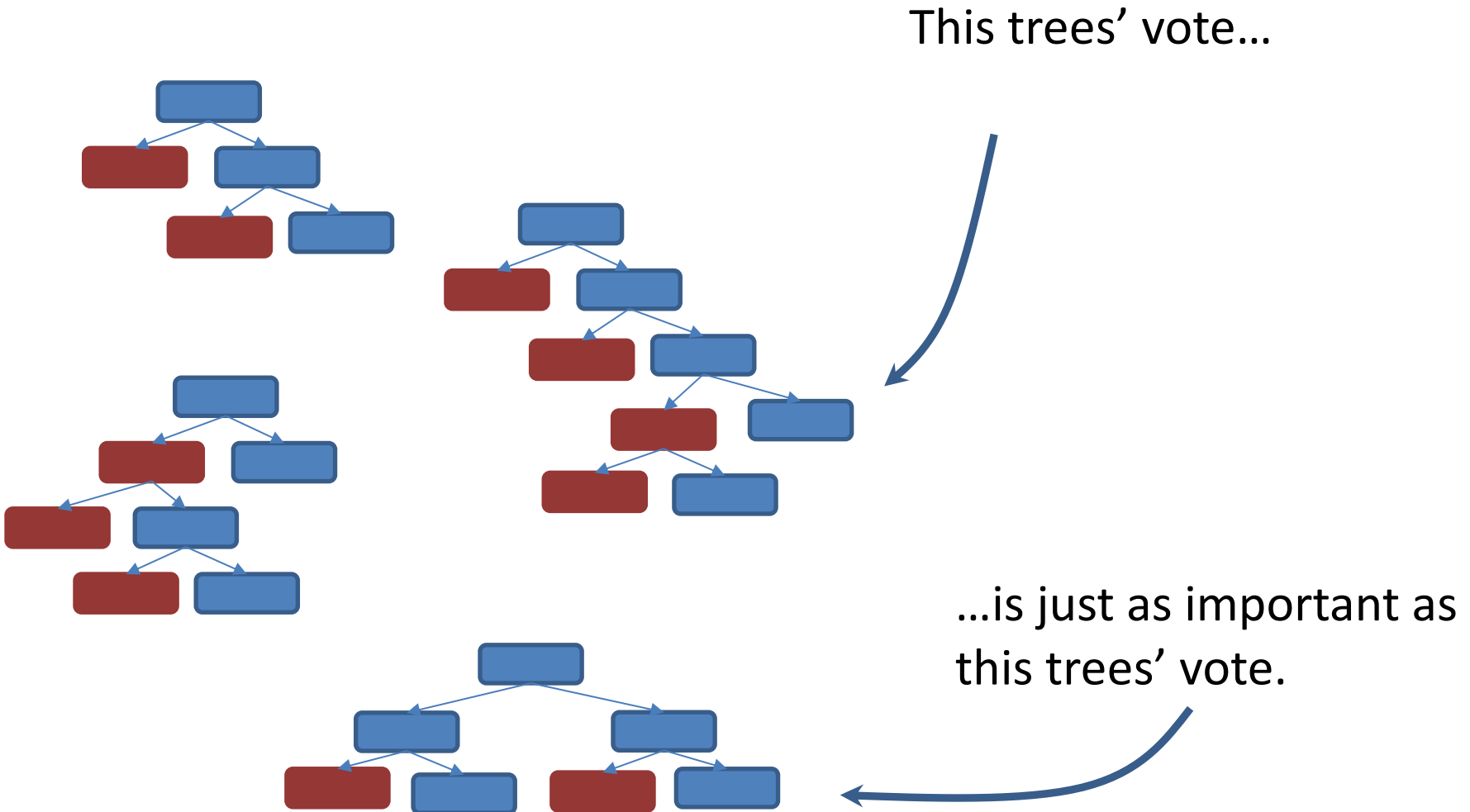
# Weak learner

- A weak learner is a classifier which predicts only slightly better than random guessing.
- A stump – one node and two leaves – is such a classifier.
- AdaBoost is a forest of stumps (but each stump has different weights)
- A stump is a poor classifier with high bias, i.e. it cannot fit to the training data very well. AdaBoost tries to reduce the bias.



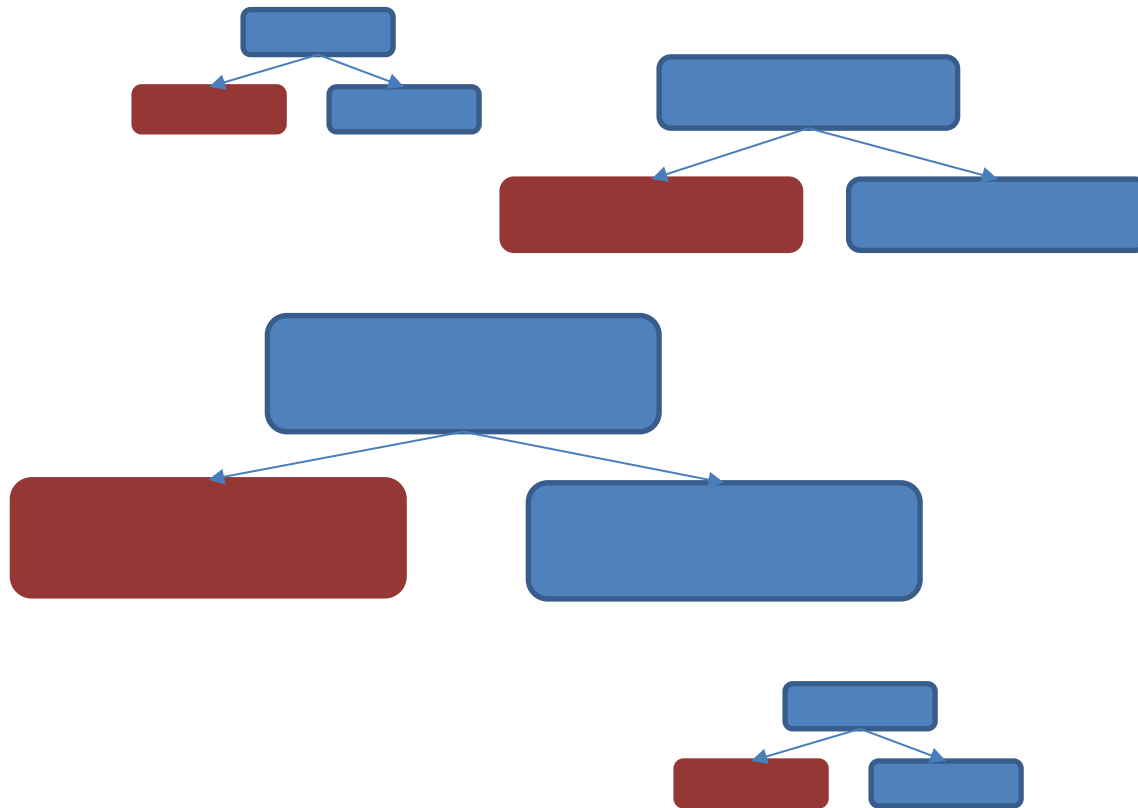
# AdaBoost algorithm

- Random forest puts equal importance on each decision tree:



# AdaBoost algorithm

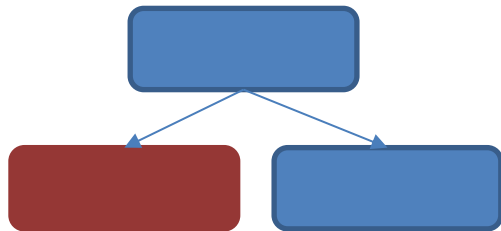
- Random forest puts equal importance on each decision tree, while AdaBoost uses different weights for each of the weak classifier.



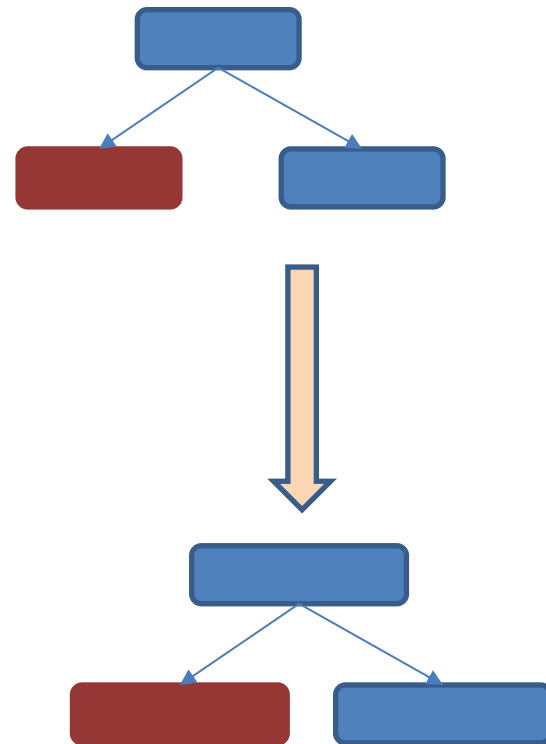
# AdaBoost algorithm

- In random forest, each tree is grown independently of each other, while in AdaBoost there is a sequence of stumps, each depending on the previous stumps.

The errors, that the first stump makes...



... influences how the second stump is made.



And the errors that the second stump makes,  
Influences how the third stump is made,  
etc...

# AdaBoost algorithm

Main ideas:

1. AdaBoost combines a lot of “weak learners” to make classification. Most often those are stumps.
2. Some stumps get to say more in the classification than others;
3. Each stump is made by taking the previous stumps’ mistakes into account.

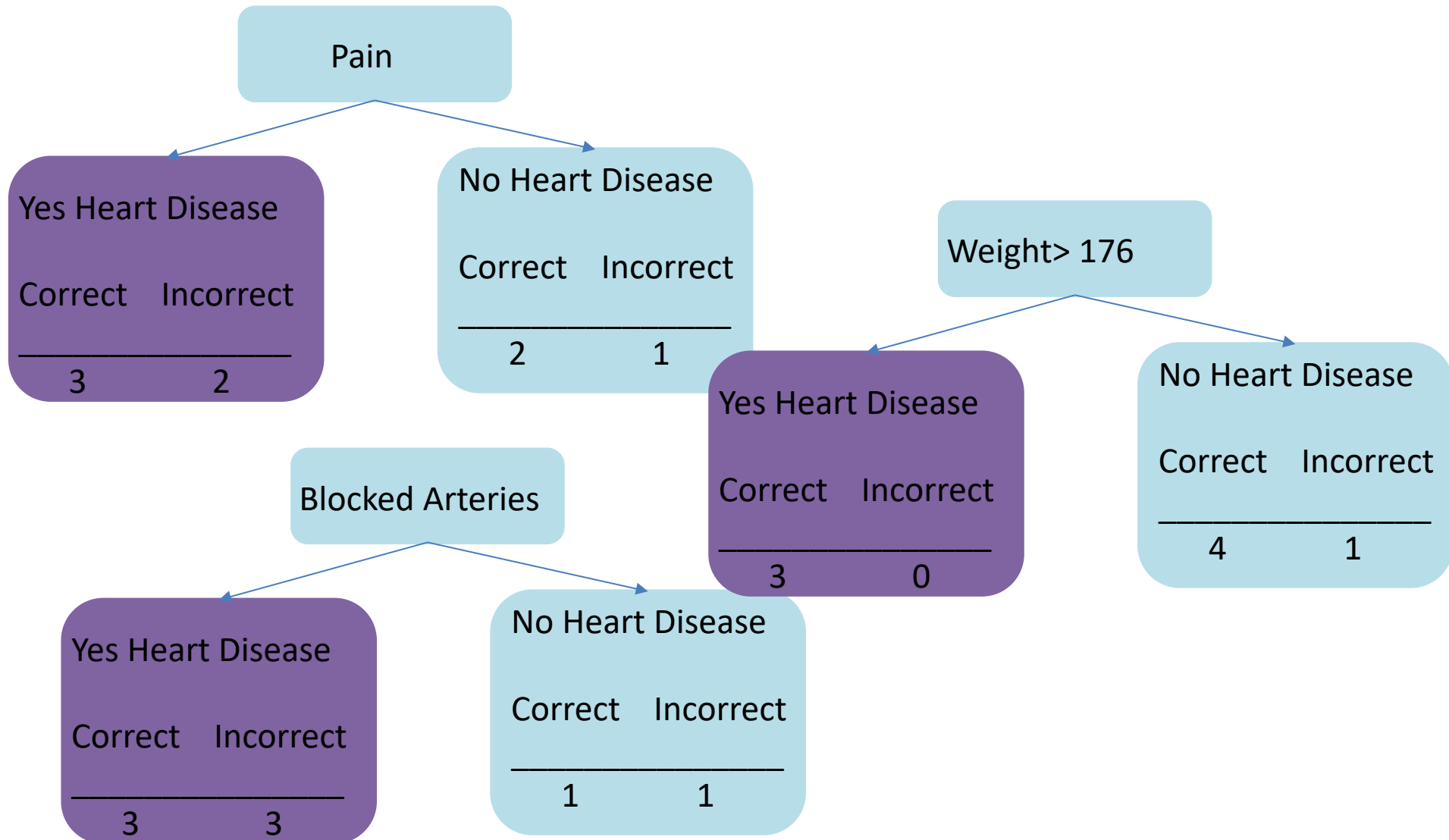
# AdaBoost by example

- First, we give each sample a weight that indicates how important it is to be correctly classified.
- Initially, all weights are equal and must sum to 1.

Chest pain	Blocked arteries	Weight (lbs)	Heart Disease	Sample weight
Yes	Yes	205	Yes	
No	Yes	180	Yes	
Yes	No	210	Yes	
Yes	Yes	167	Yes	
No	Yes	156	No	
No	Yes	125	No	
Yes	No	168	No	
Yes	Yes	172	No	

# AdaBoost by example

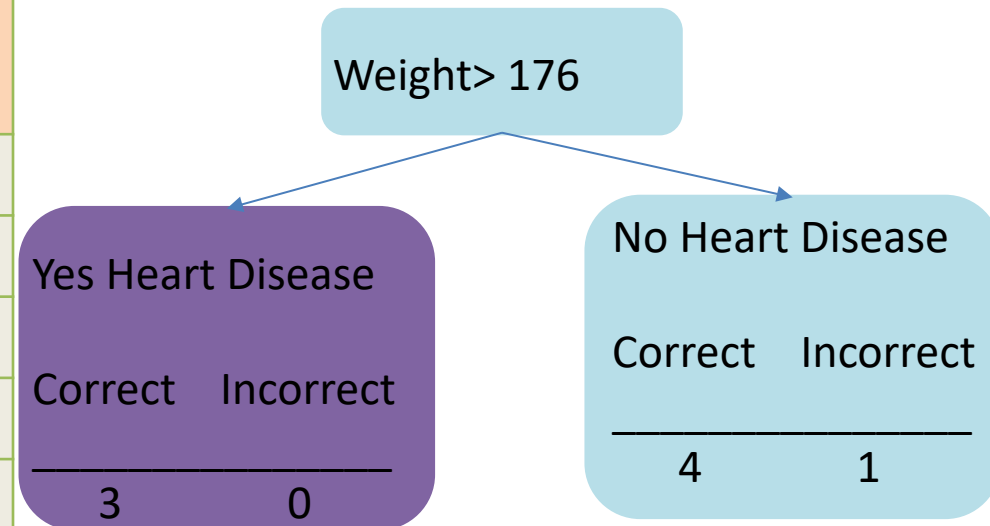
- Find the best stump of all possible stumps:



# AdaBoost by example

- The error of the stump is obtained by summing weight of incorrectly classified training data points

Chest pain	Blocked arteries	Weight (lbs)	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



Total error: 1/8



# AdaBoost by example

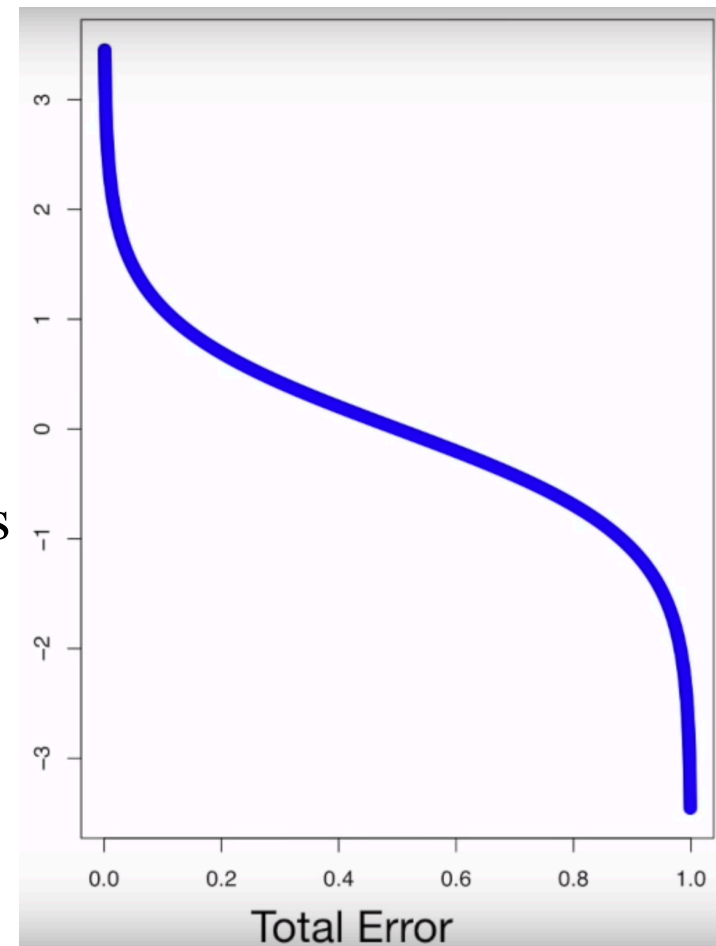
- The amount of say is calculated using the expression:

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

- When the error is small, learners weight is large and vice versa.

- $\text{Amount of Say} = \frac{1}{2} \ln \left( \frac{1 - \frac{1}{8}}{\frac{1}{8}} \right) \approx 0.973$

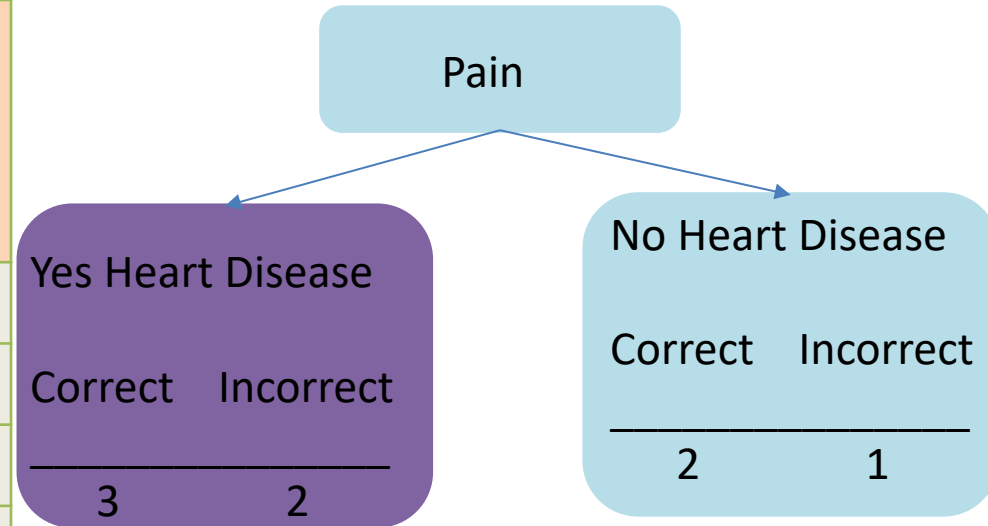
- Thus, in the final classifier the weight of this stump will be 0.973.



# Question 2

- Calculate the weight (amount of say) of the following stump:

Chest pain	Blocked arteries	Weight (lbs)	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



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

- a) 0.2554128;
- b) 0.5493061;
- c) 0.9729551;
- d) Correct answer is not given

# AdaBoost by example

- Now we need to take into account the mistakes that previous stump made.
- This is done by *reducing the weights of correctly classified examples and increasing the weights of incorrectly classified examples*.

1. For incorrectly classified examples:

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

$$\text{New Sample Weigh} = \frac{1}{8} \times e^{0.97} = 0.33$$

2. For correctly classified examples:

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

$$\text{New Sample Weigh} = \frac{1}{8} \times e^{-0.97} = 0.05$$

- Note: we have to normalize weights to sum to 1!

# AdaBoost by example

After recalculating weights:

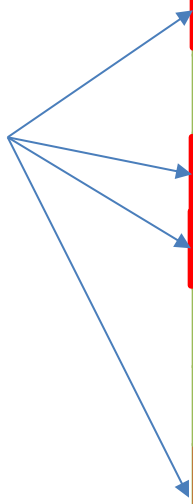
Chest pain	Blocked arteries	Weight (lbs)	Heart Disease	Sample weight	Weights	Weights (sum to 1)
Yes	Yes	205	Yes	1/8	0.05	0.07
No	Yes	180	Yes	1/8	0.05	0.07
Yes	No	210	Yes	1/8	0.05	0.07
Yes	Yes	167	Yes	1/8	0.33	0.49
No	Yes	156	No	1/8	0.05	0.07
No	Yes	125	No	1/8	0.05	0.07
Yes	No	168	No	1/8	0.05	0.07
Yes	Yes	172	No	1/8	0.05	0.07

- Form a new bootstrap training sample using the weights as probabilities.
- Incorrectly classified example is  $\frac{0.49}{0.07} = 7$  times more likely to be chosen!

# AdaBoost by example

Bootstrap sample:

It will be difficult for the classifier to incorrectly classify this example!



The diagram shows a table with 8 rows and 4 columns. The first three rows are highlighted with red borders. Blue arrows point from the text 'It will be difficult for the classifier to incorrectly classify this example!' to the first, second, and third rows of the table.

Chest pain	Blocked arteries	Weight (lbs)	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

# AdaBoost prediction

- Assume that there are  $T$  weak classifiers  $C_t(x)$  in the model with weights  $\alpha_t$ .
- The entire AdaBoost classifier then has the following form:

$$C(x) = \text{sign} \left( \sum_{t=1}^T \alpha_t C_t(x) \right).$$

- $C_t$  classifier predict class either  $+1$  or  $-1$ .
- For example, let 3 stumps produce classes  $+1, +1, -1$ , weight of those stumps are 2, 0.1, 1. Then the prediction is done like this:

$$C(x) = \text{sign}(2 \cdot (+1) + 0.1 \cdot (+1) + 1 \cdot (-1)) = \text{sign}(1.1) = +1$$

# Question 3

Consider the following stumps:

$$C_1(\text{e-mail}) = \begin{cases} +1 & \text{if e-mail contains word "money"} \\ -1 & \text{otherwise} \end{cases}$$

$$C_2(\text{e-mail}) = \begin{cases} +1 & \text{if e-mail contains word "free"} \\ -1 & \text{otherwise} \end{cases}$$

$$C_3(\text{e-mail}) = \begin{cases} +1 & \text{if e-mail contains word "order"} \\ -1 & \text{otherwise} \end{cases}$$

$$C_4(\text{e-mail}) = \begin{cases} +1 & \text{if e-mail contains word "credit"} \\ -1 & \text{otherwise} \end{cases}$$

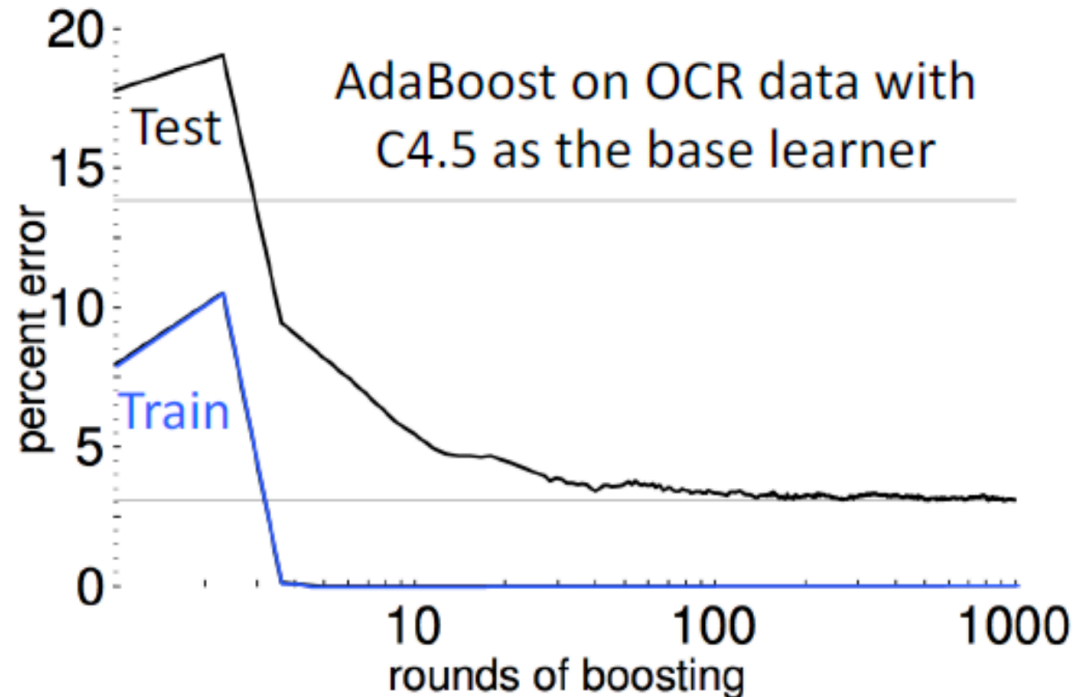
AdaBoost produced weights 0.2, 0.1, 0.4, 0.3. Suppose that your email box uses this algorithm to filter spam (+1) into not-spam (−1).

You received an email: “Dear customer, we kindly inform you that you have missed your last payment of the credit.” Calculate the value of  $\sum_{t=1}^T \alpha_t C_t(x)$  and the prediction of AdaBoost algorithm.

- a) -0.4, Spam;
- b) -0.4 Not spam;
- c) 0.4, Spam;
- d) Correct answer is not given.

# AdaBoost

- If a point is repeatedly misclassified:
  1. Its weight is increased every time;
  2. Eventually AdaBoost will find a weak classifier which correctly classifies this point.
- In practice AdaBoost does not typically overfit.
- Often it continues to drive down test error even after the training error reaches zero!

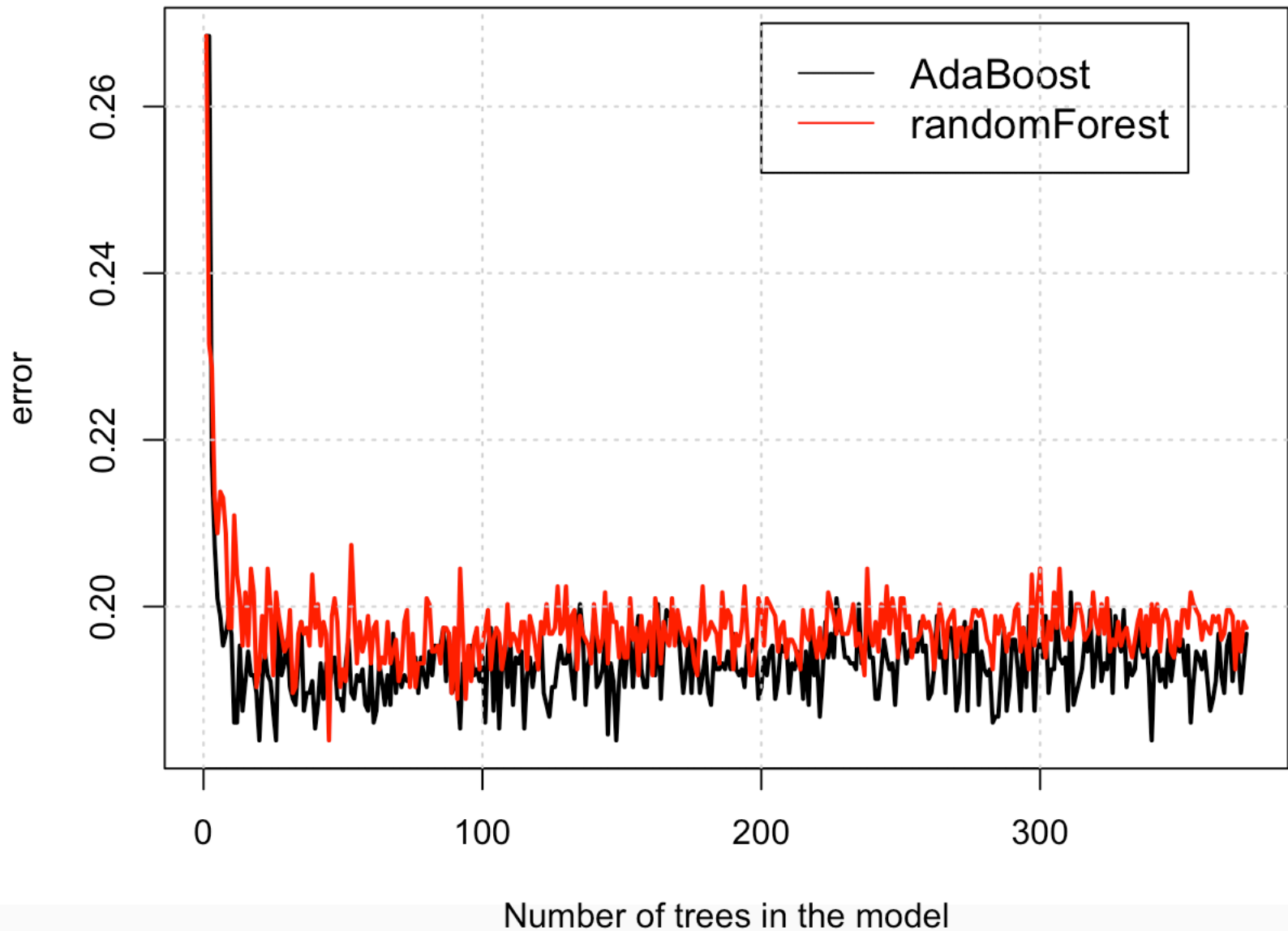




# AdaBoost

- AdaBoost works best with weak learners:
  1. Stumps;
  2. Depth-limited decision trees;
  3. Linear classifiers (logistic regression)
- No parameters to tune, except for the number of weak learners;
- Boosting can fail:
  1. Given insufficient data;
  2. Overly complex weak learners;
  3. Can be susceptible to noise

# AdaBoost Churn example



# Žodynas

- Bootstrap aggregating – bagging – bagingas;
- Bootstrap sample – maišos būdu gauta imtis;
- bootstrapping - maiša;
- random forest – atsitiktinis miškas;
- Out-of-bag error – OOB paklaida;
- permutation importance - perstatos svarbumas;
- boosting – stiprinimas;
- adaptive boosting – adaptyvus stiprinimas;
- stump – kelmas;