AI Course

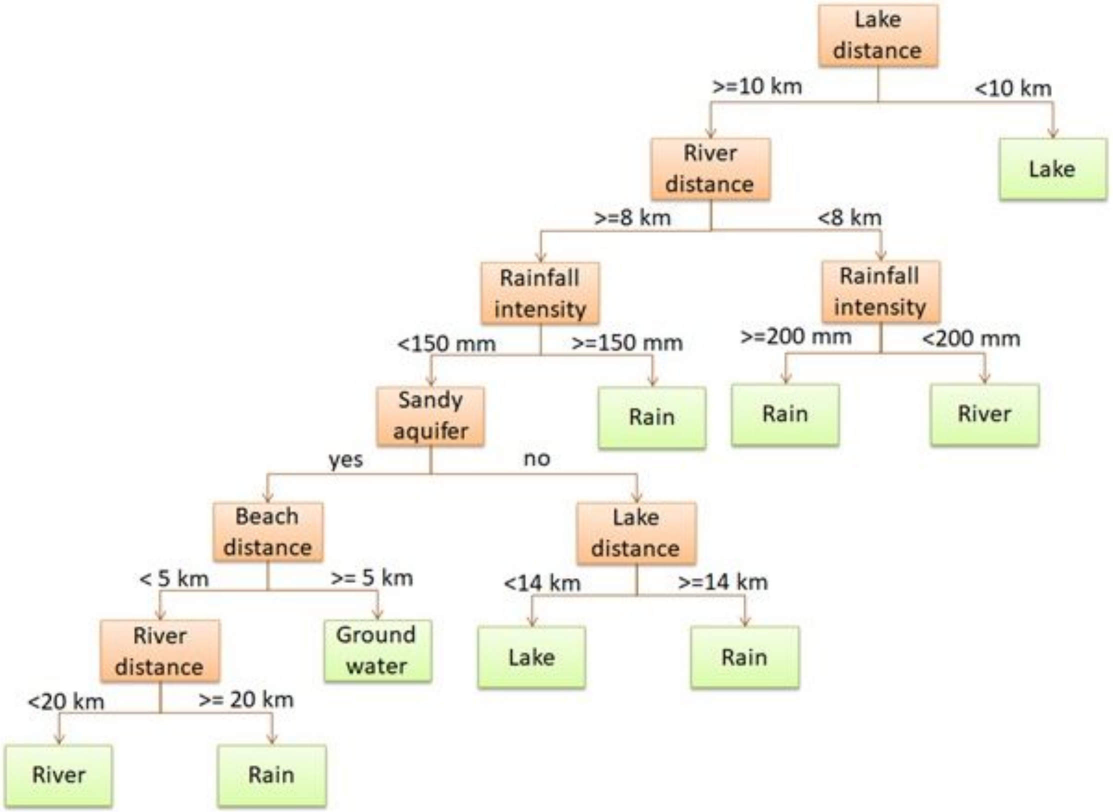
Dr. Mürsel Taşgın

Tree-based algorithms

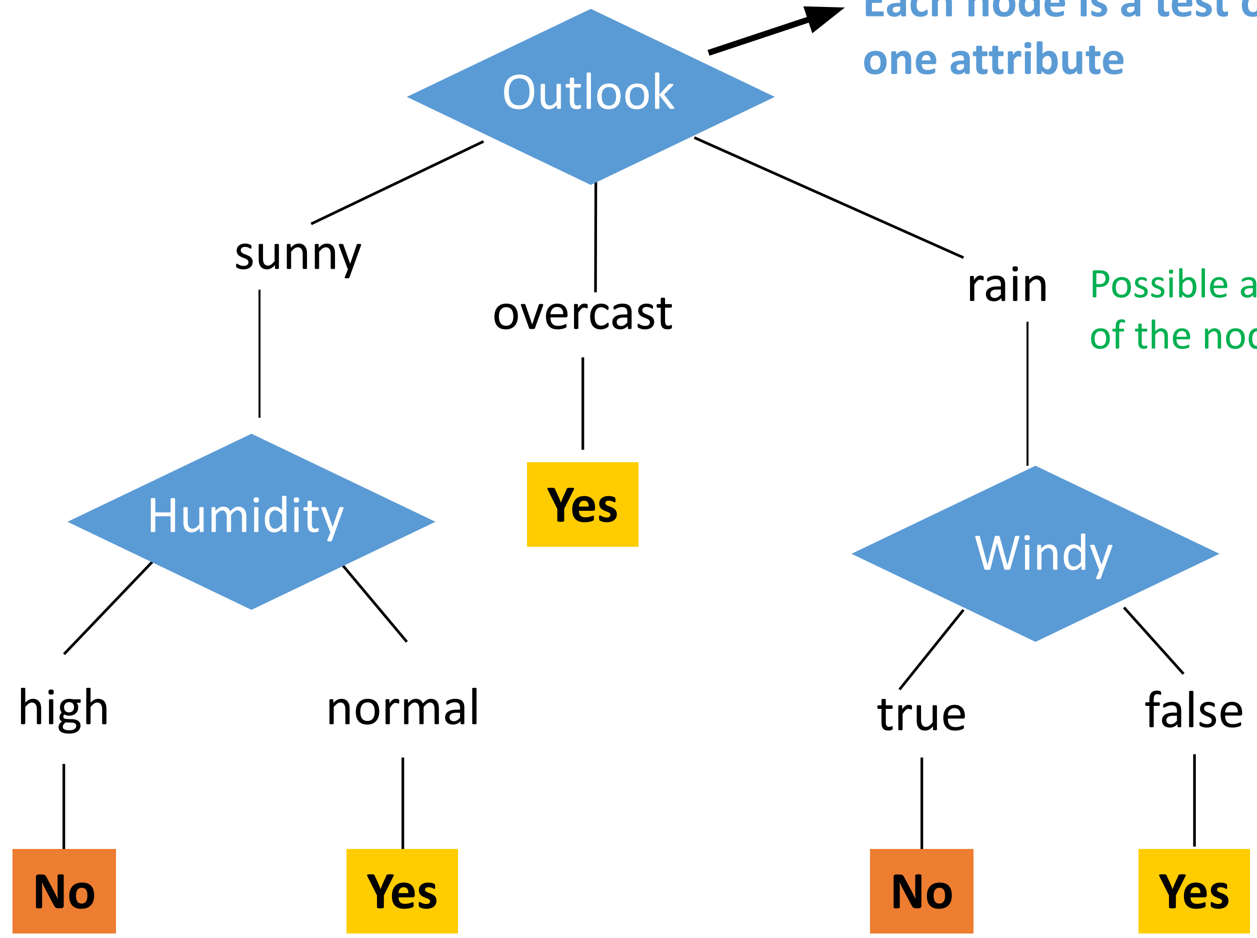
Tree-based algorithms

From observations about items, build a tree with branches and build conclusions on leaves

- Decision trees
- Classification trees
- Regression trees
- Boosted trees



Tree-based algorithms - *Anatomy of a decision tree*



Each node is a test on one attribute

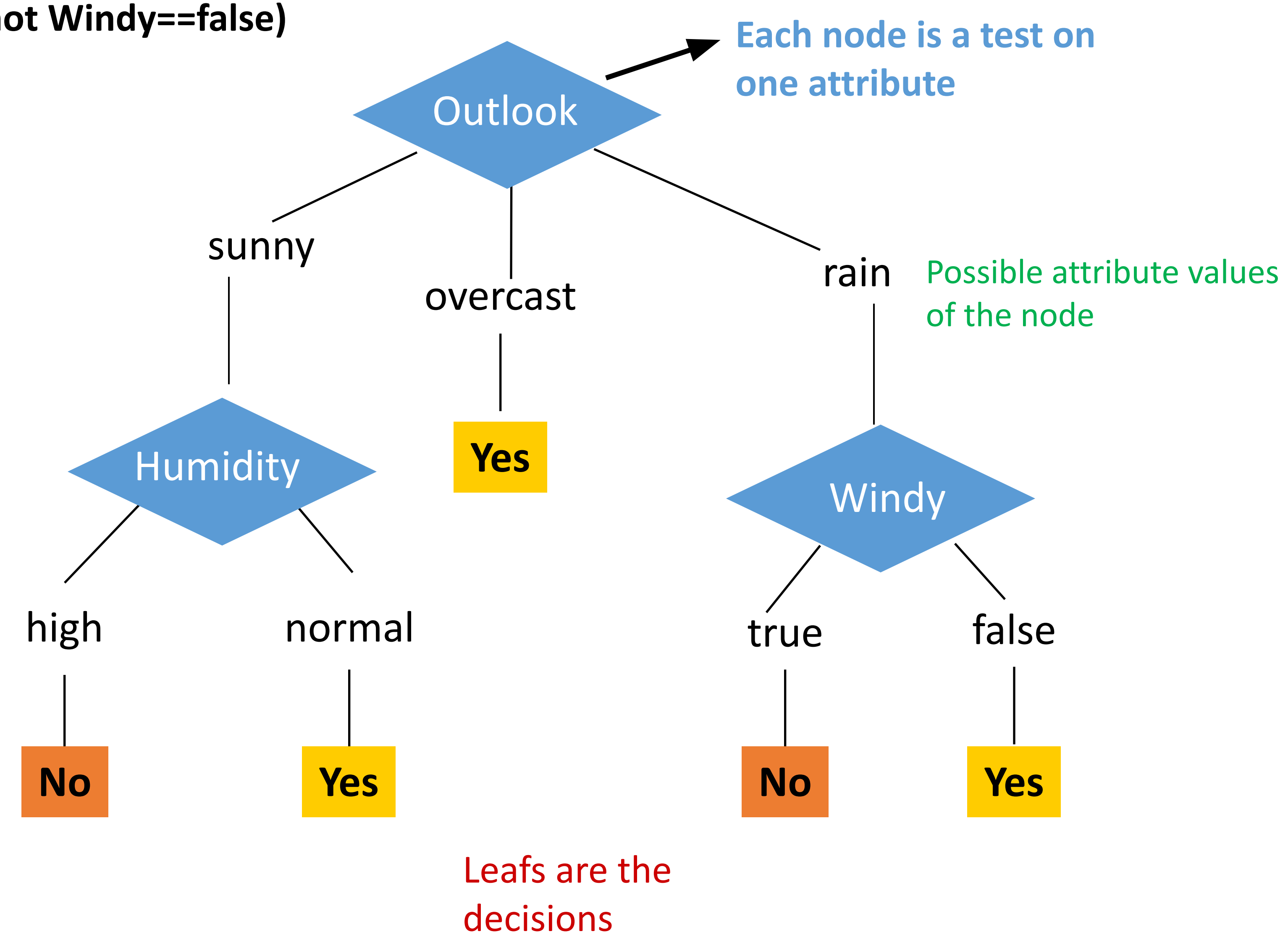
Possible attribute values of the node

Leafs are the decisions

Tree-based algorithms – *A decision example: Playing tennis*

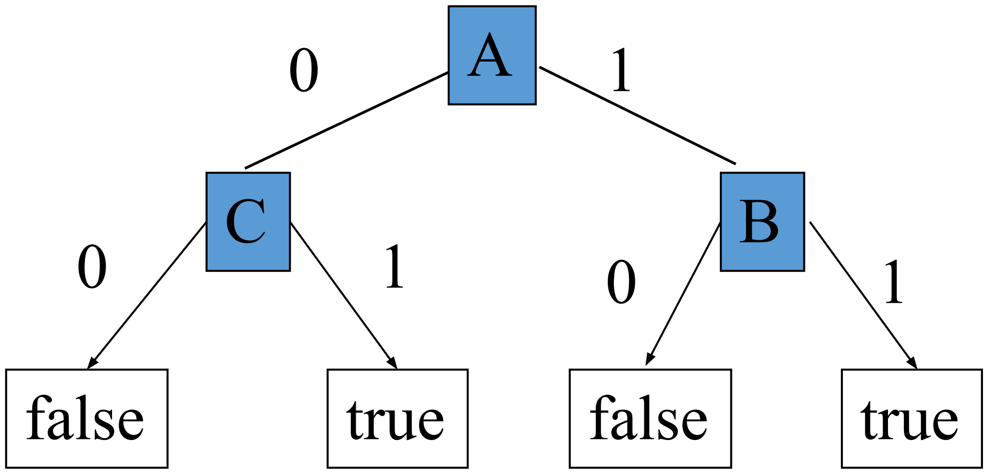
(Outlook==rain) and (not Windy==false)

Pass it on the tree
-> Decision is yes.



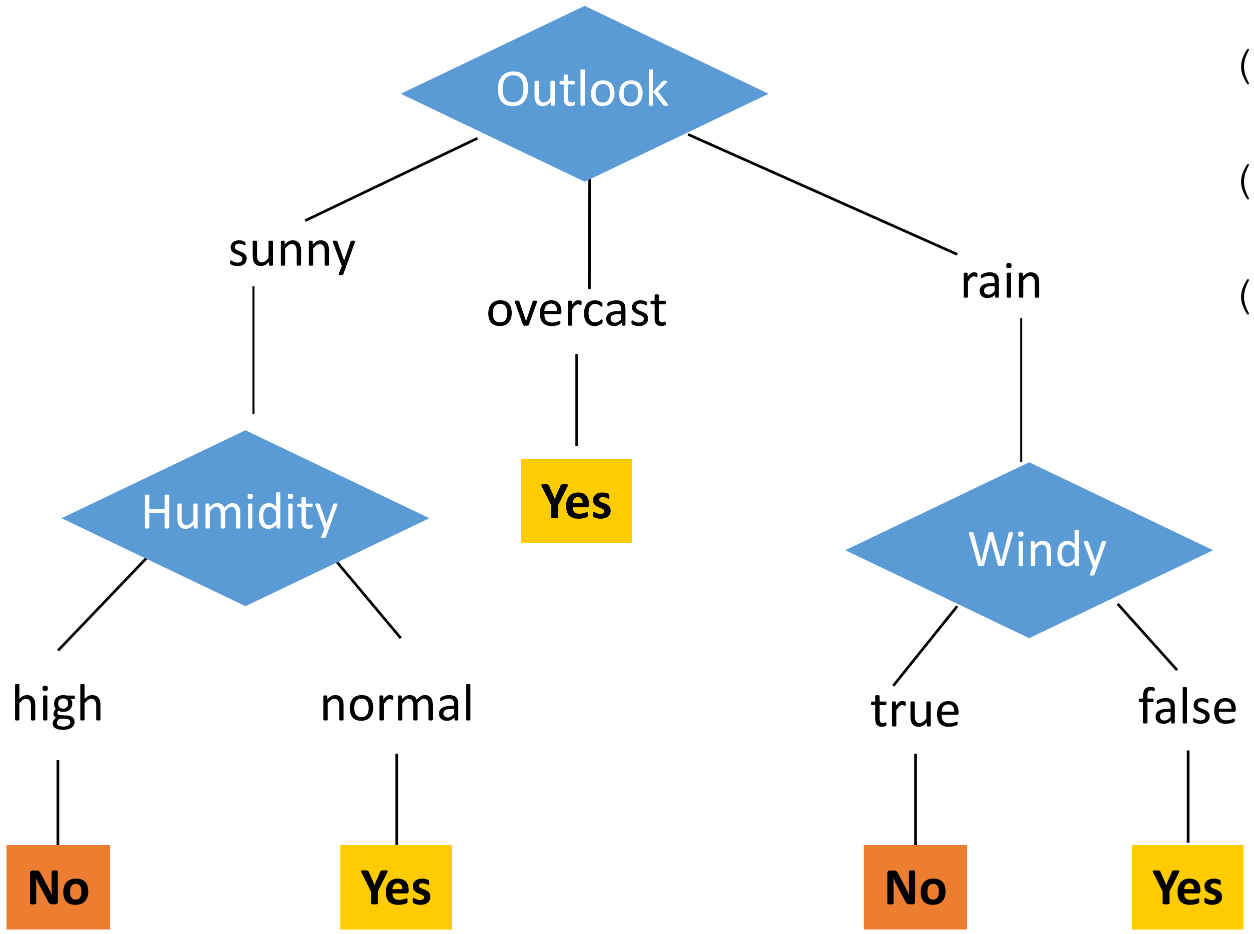
Tree-based algorithms – *Decision Trees*

Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.



Y=**((A and B) or ((not A) and C))**

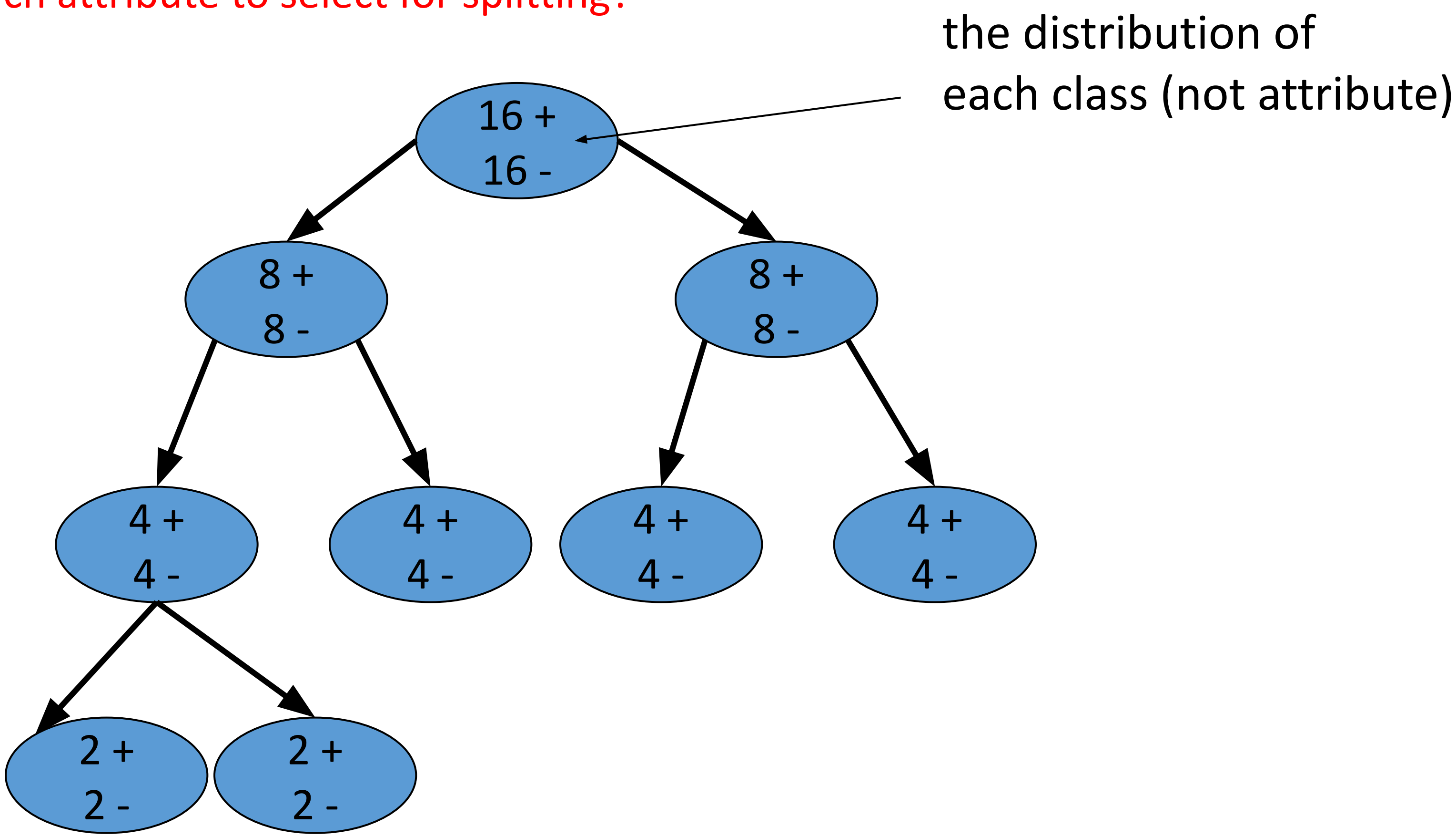
Tree-based algorithms – *Decision Trees*




```
(Outlook ==overcast)
 OR
((Outlook==rain) and (not Windy==false))
 OR
((Outlook==sunny) and (Humidity=normal))
 => yes play tennis
```

Tree-based algorithms – *Decision Trees*

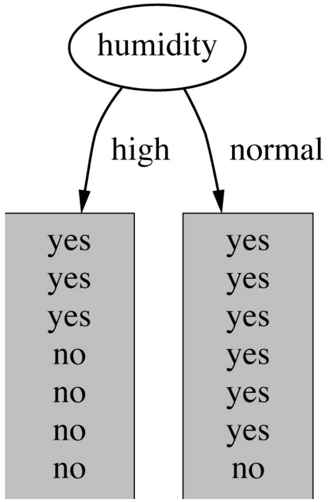
Which attribute to select for splitting?

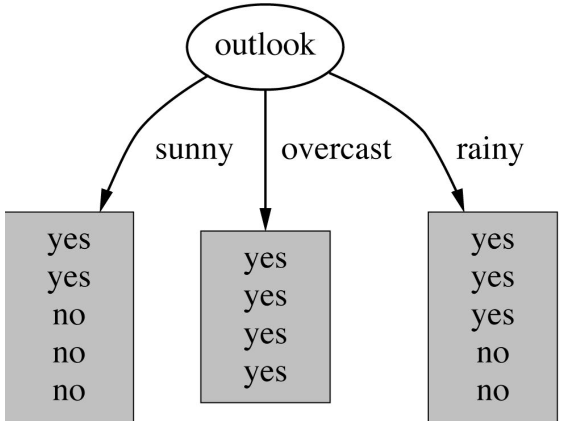


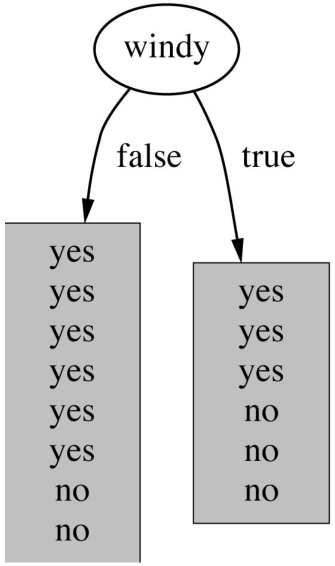
Decision Trees – *How do we choose the test?*

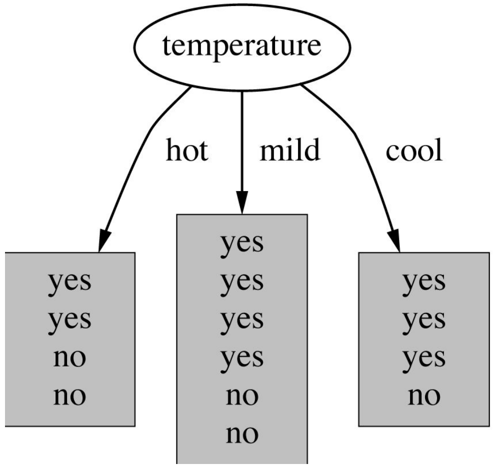
Which attribute should be used as the test?

Intuitively, you would prefer the one that *separates* the training examples as much as possible.









Decision Trees – *Information Gain*

Which attribute should be used as the test?

Information gain is one criteria to decide on the attribute.

Imagine:

1. Someone is about to tell you your own name
2. You are about to observe the outcome of a dice roll
2. You are about to observe the outcome of a coin flip
3. You are about to observe the outcome of a biased coin flip

Each situation have a different *amount of uncertainty* as to what outcome you will observe.

Decision Trees – *Information*

INFORMATION: Reduction in uncertainity *(amount of the surprise in the outcome)*

$$I(E) = \log_2 \frac{1}{p(x)} = -\log_2 p(x)$$

If the probability of this event happening is small and it happens the information is large.

Observing the outcome of a coin flip is head

$$I = -\log_2 1/2 = 1$$

Observe the outcome of a dice is 6

$$I = -\log_2 1/6 = 2.58$$

Decision Trees – *Entropy*

ENTROPY

The *expected amount of information* when observing the output of a random variable X.

It is the *number of bits required* to represent a randomly drawn even from the distribution, e.g. an average event.

$$H(X) = E(I(X)) = \sum_i p(x_i) I(x_i) = -\sum_i p(x_i) \log_2 p(x_i)$$

If X can have 8 outcomes and all are equally likely;

$$H(X) == -\sum_i 1/8 \log_2 1/8 = 3$$ → Entropy is 3 bits!

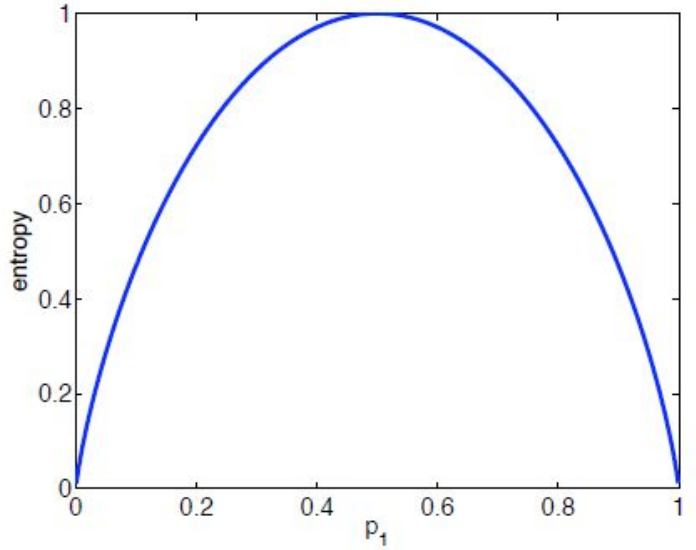
Decision Trees – *Entropy*

If there are *k* possible outcomes

$$H(X) \leq \log_2 k$$

Equality holds when all outcomes are equally likely

The more the probability distribution the deviates from uniformity, the ***lower the entropy***



Decision Trees – *Entropy, Purity*

Entropy measures the purity

The distribution is *less* uniform
Entropy is *lower*
The node is *purer*

Decision Trees – *Conditional Entropy*

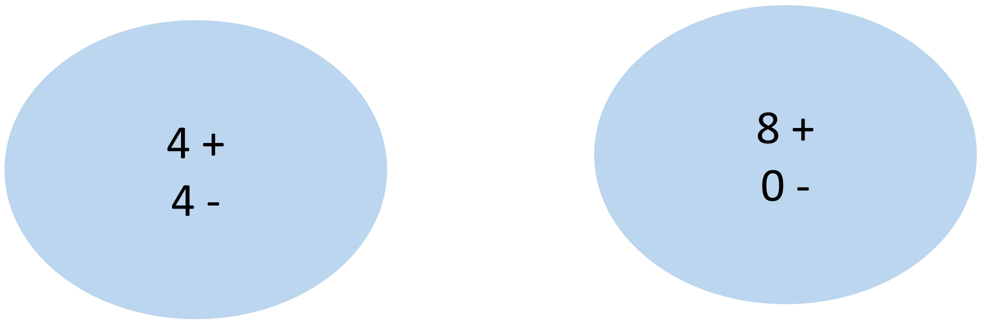
$$H(X) = -\sum_i p(x_i)\log_2 p(x_i)$$

$$H(X \mid Y) = -\sum_j p(y_j) H(X \mid Y = y_j)$$

$$= -\sum_j p(y_j) \sum_i p(x_i \mid y_j)\log_2 p(x_i \mid y_j)$$

Decision Trees – *Entropy, Purity*

Entropy measures the purity



The distribution is *less* uniform
Entropy is *lower*
The node is *purer*

Decision Trees – *Information Gain*

- $$IG(X,Y) = H(X) - H(X|Y)$$

Reduction in uncertainty by knowing *Y*

Information gain = (Information before split) – (Information after split)

Decision Trees – *Information Gain*

Example

Attributes Labels

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Which one do we choose X1 or X2?

$$IG(X1,Y) = H(Y) - H(Y|X1)$$

$$H(Y) = -(5/10)\log(5/10) - 5/10\log(5/10) = 1$$

$$H(Y|X1) = P(X1=T)H(Y|X1=T) + P(X1=F)\,H(Y|X1=F)$$

$$= 4/10\,(1\log 1 + 0\log 0) + 6/10\,(5/6\log 5/6 + 1/6\log 1/6)$$

$$= 0.39$$

Information gain (X1,Y)= 1 - 0.39 = 0.61

Decision Trees – *Information Gain*

Which one do we choose?

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

Information gain $(X1, Y) = 0.61$

Information gain $(X2, Y) = 0.12$

Pick the variable which provides the **most information gain** about Y → Pick X1

Decision Trees – *Information Gain*

Recursively build on branches and build the tree

X1	X2	Y	Count
T	T	+	2
T	F	+	2
F	T	-	5
F	F	+	1

One branch

The other branch

- The number of possible values influences the information gain.
- The more possible values, the higher the gain *(the more likely it is to form small, but pure partitions)*

Decision Trees – *Information Gain*

Purity (diversity) measures

- Gini (population diversity)
- Information Gain
- Chi-square Test

Decision Trees – *Overfitting*

- You can perfectly fit to any training data
- Zero bias, high variance

Avoiding overfitting:

1. Stop growing the tree when further splitting the data does not yield an improvement
2. Grow a full tree, then prune the tree, by eliminating nodes.

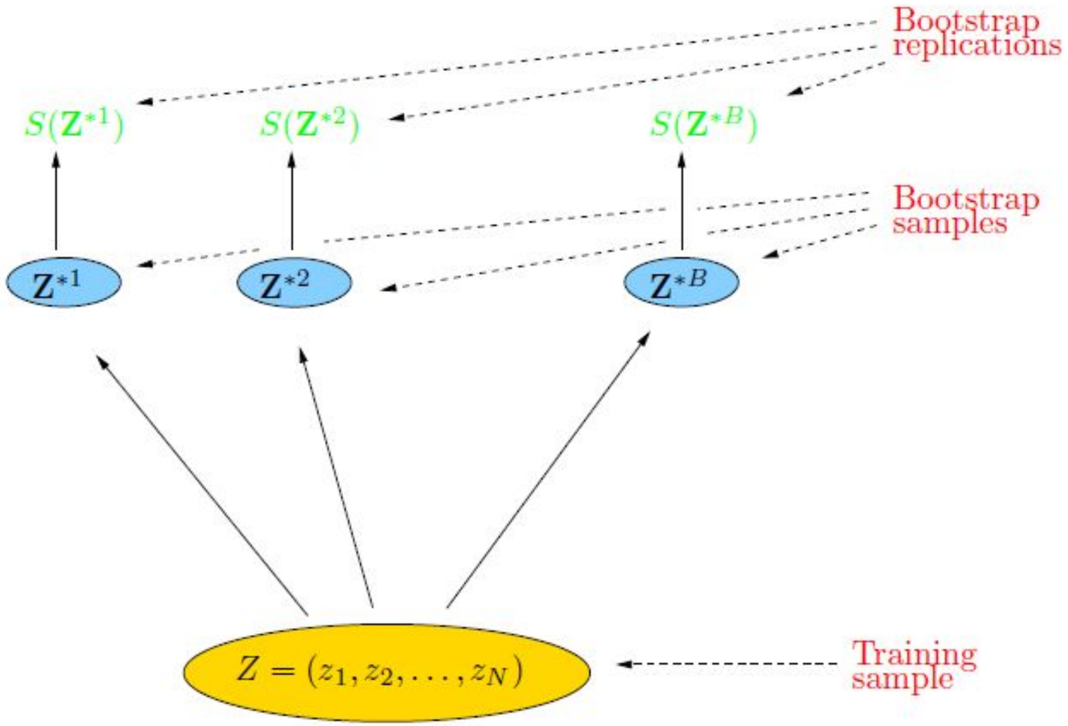
Bagging

- *Bagging* or *bootstrap aggregation* a technique for reducing the variance of an estimated prediction function.

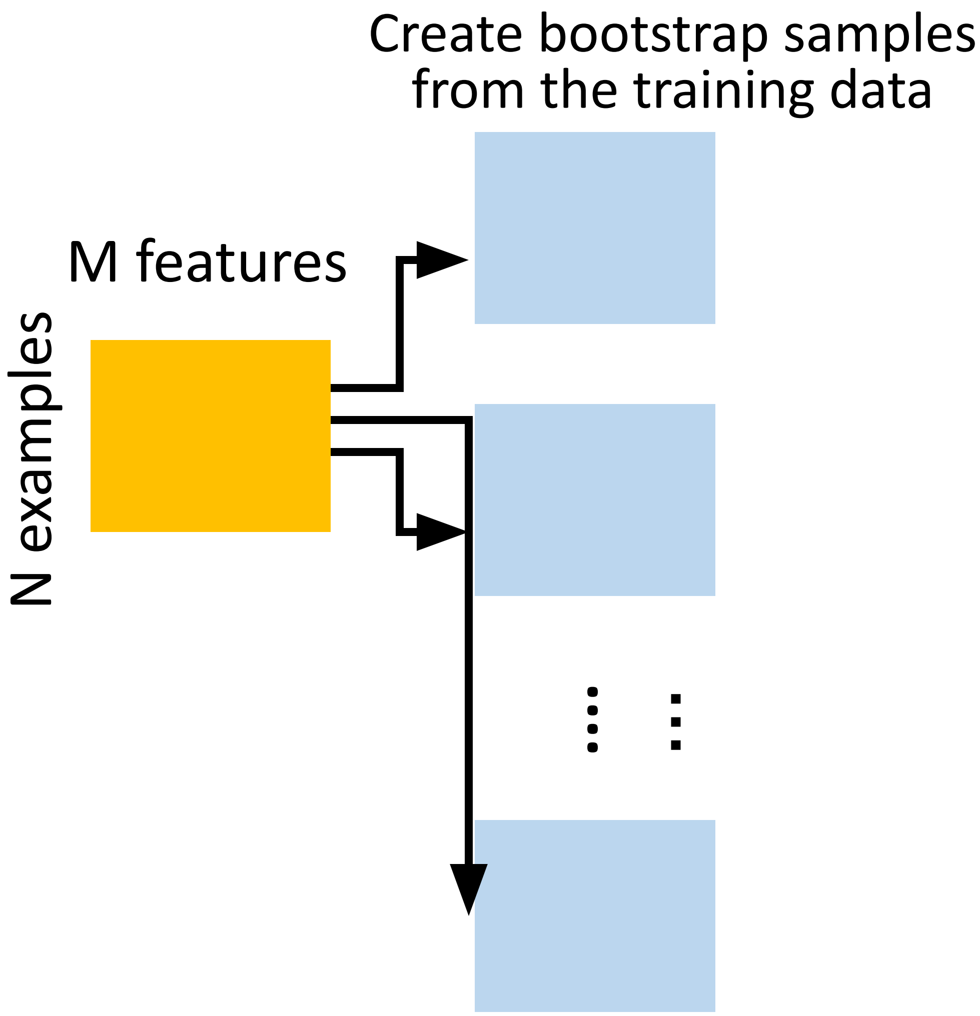
- For classification, a *committee* of trees each cast a vote for the predicted class.

Bootstrap

The basic idea is to randomly draw datasets *with replacement* from the training data, each sample *the same size as the original training set*

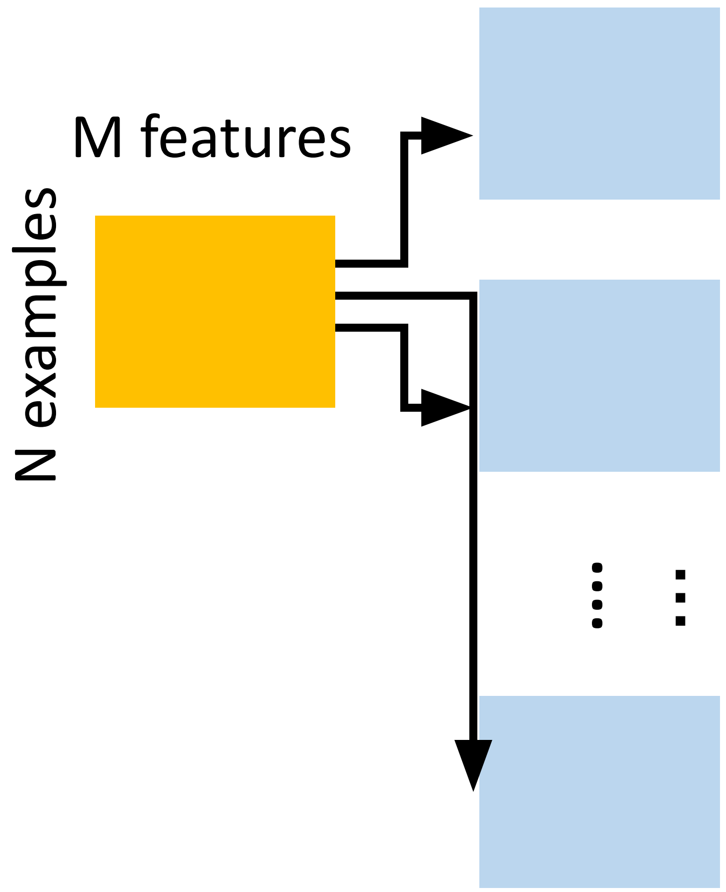


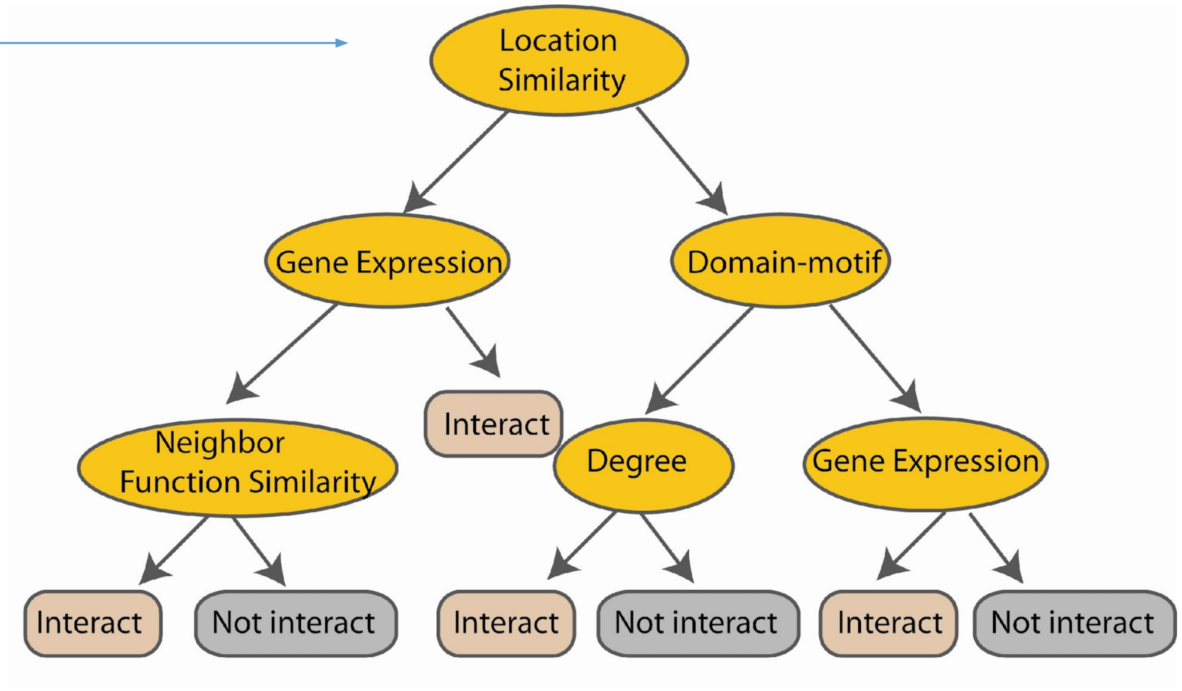
Bagging


Create bootstrap samples
from the training data
M features
N examples

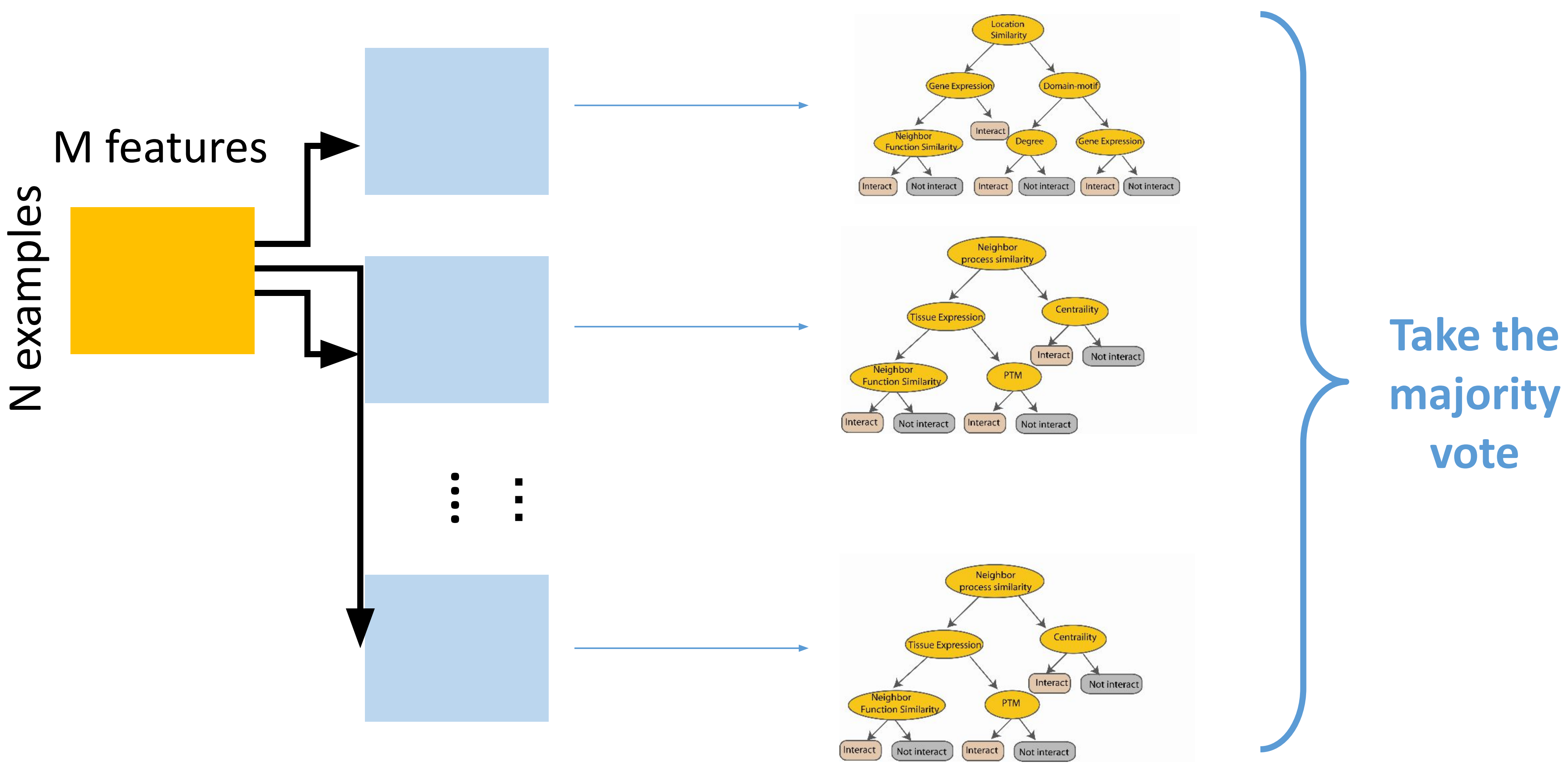
Bagging

Construct decision trees for each subset





Bagging


M features
N examples
Location Similarity
Gene Expression
Domain-motif
Neighbor Function Similarity
Interact
Degree
Gene Expression
Interact
Not interact
Interact
Not interact
Interact
Not interact
Neighbor process similarity
Tissue Expression
Centraility
Interact
Not interact
Neighbor Function Similarity
PTM
Interact
Not interact
Interact
Not interact
Neighbor process similarity
Tissue Expression
Centraility
Interact
Not interact
Neighbor Function Similarity
PTM
Interact
Not interact
Interact
Not interact
Take the majority vote

Bagging

$\mathbf{Z} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$

Z^{*b} where = 1,.., B..

The prediction at input x when bootstrap sample *b* is used for training

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x).$$

Bagging – An example

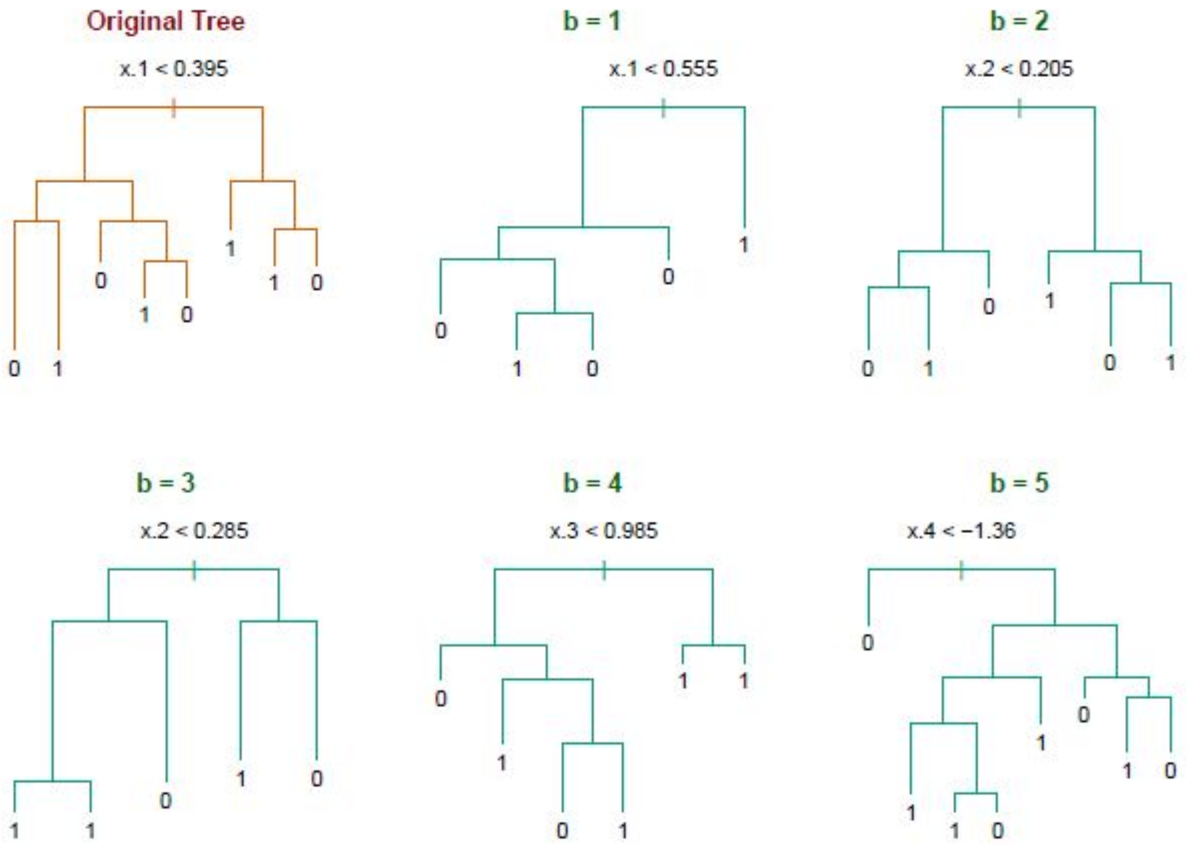
- Generated a sample of size $N = 30$, with two classes and $p = 5$ features, each having a standard Gaussian distribution with pairwise correlation 0.95.

The response Y was generated according to

- $\Pr(Y = 1 | x1 \leq 0.5) = 0.2,$
- $\Pr(Y = 0 | x1 > 0.5) = 0.8.$

Bagging – An example

Notice the bootstrap trees are different than the original tree



Bagging – An example

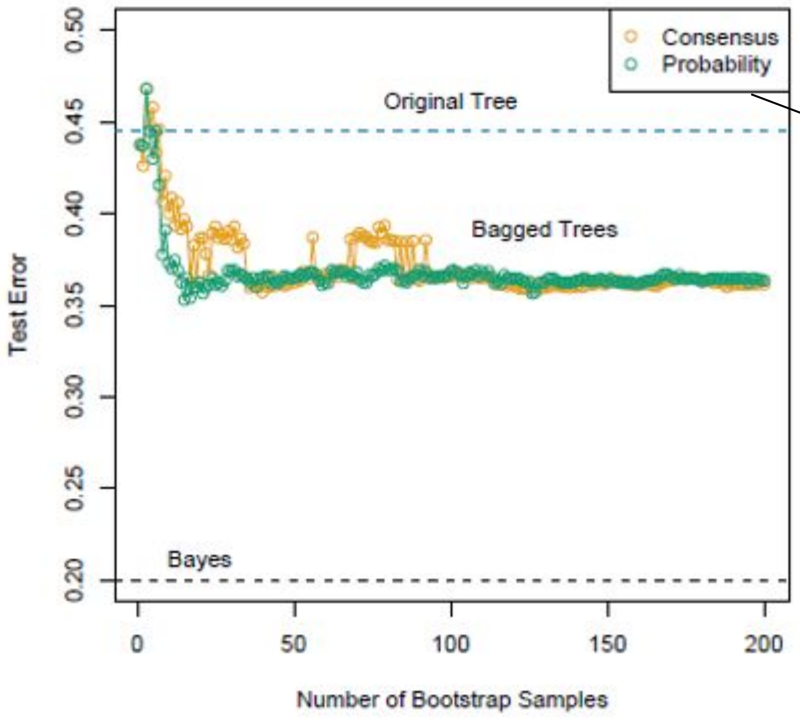


FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*

Treat the voting Proportions as probabilities

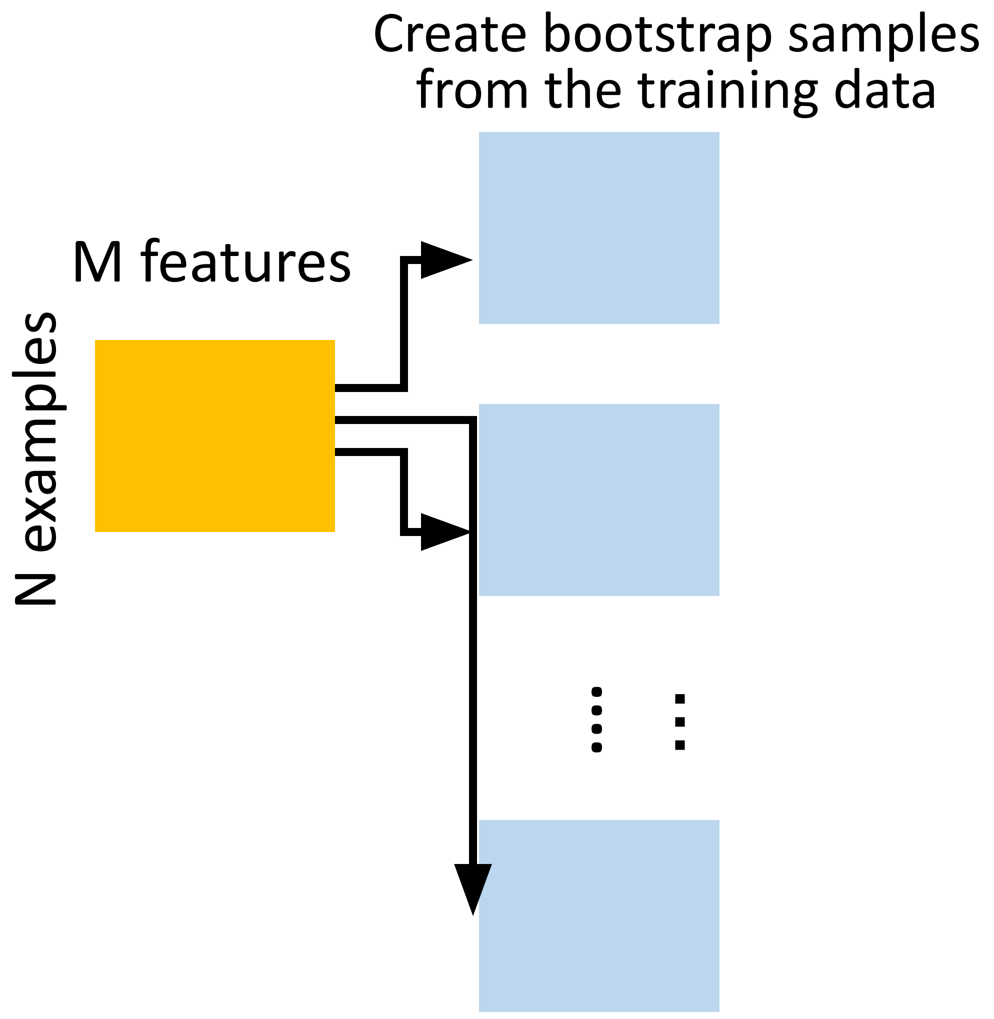
Hastie

http://www-stat.stanford.edu/~hastie/Papers/ESLII.pdf Example 8.7.1

Random Forest Classifier

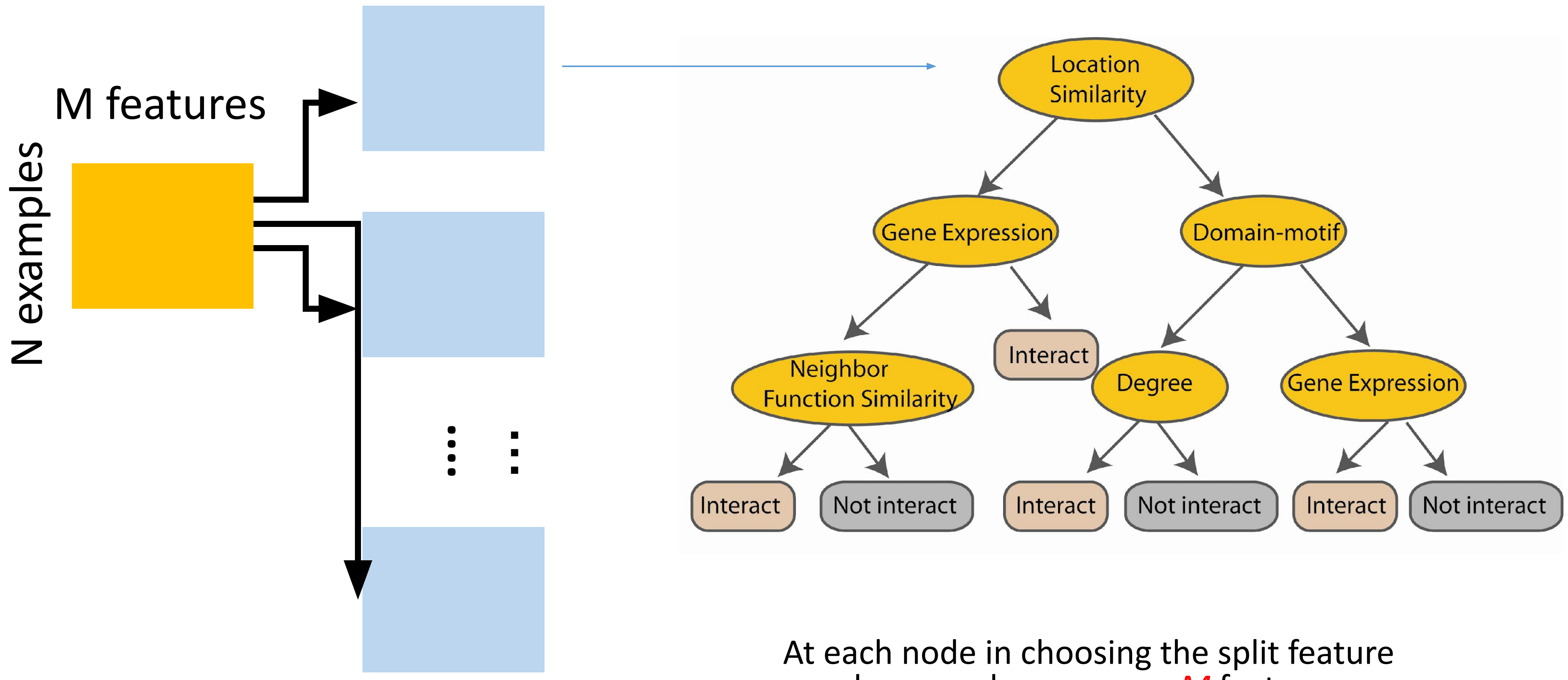
Random forest classifier, an extension to bagging which uses *de-correlated* trees.

Random Forest Classifier



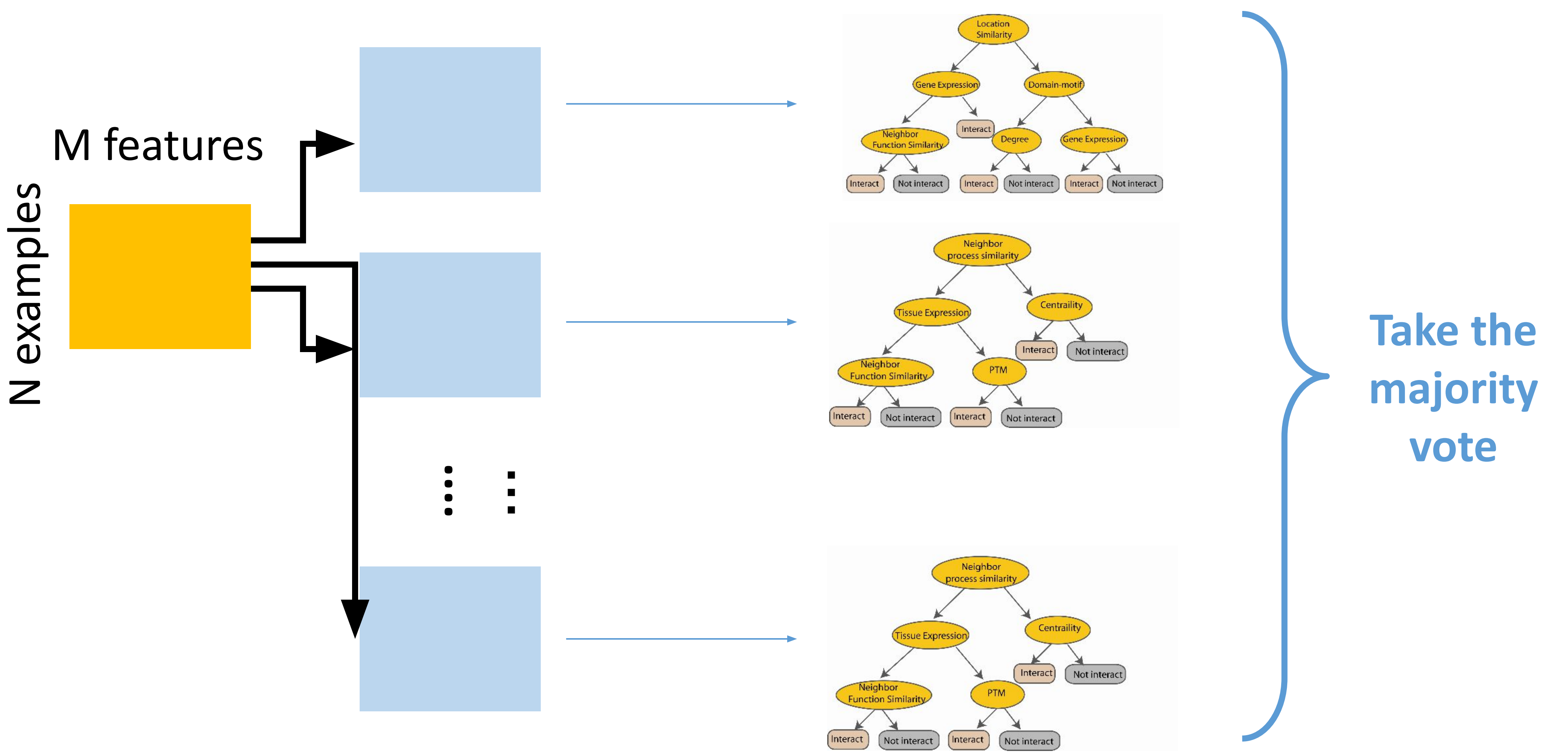
Random Forest Classifier

Construct decision trees for each subset



At each node in choosing the split feature choose only among *m<M* features

Random Forest Classifier



Random Forest Classifier

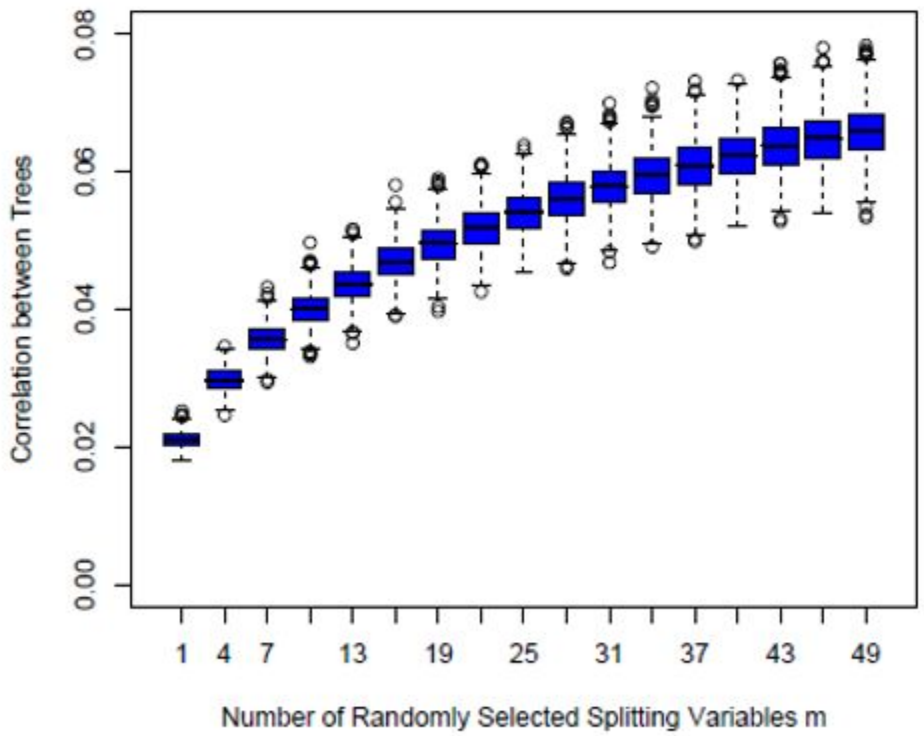


FIGURE 15.9. *Correlations between pairs of trees drawn by a random-forest regression algorithm, as a function of m. The boxplots represent the correlations at 600 randomly chosen prediction points x.*

Random Forests 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 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.

Output the ensemble of trees.

To make a prediction at a new point *x* we do:

For regression: average the results

For classification: majority vote

Random Forests Tuning

Recommendations:

- For classification, the default value for m is $\sqrt{p}$ and the minimum node size is one.
- For regression, the default value for m is $p/3$ and the minimum node size is five.

In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters.

Like with Bagging, we can use out-of-bag (OOB) and therefore Random Forest can be fit in one sequence, with cross-validation being performed along the way. Once the OOB error stabilizes, the training can be terminated.

Out of bag (OOB) score is a way of validating the Random forest model

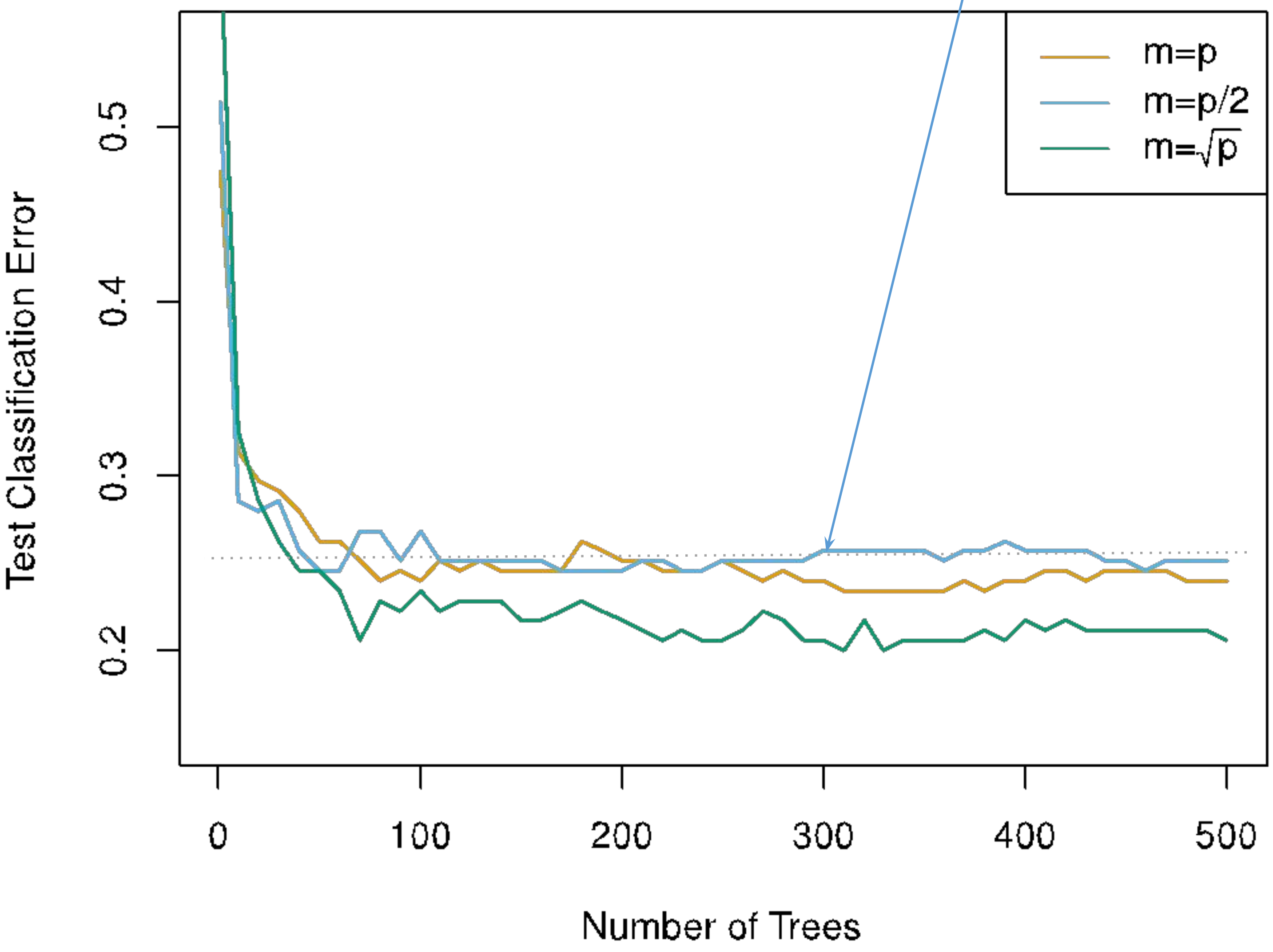
Example

- 4,718 genes measured on tissue samples from 349 patients.
- Each gene has different expression
- Each of the patient samples has a qualitative label with 15 different levels: either normal or 1 of 14 different types of cancer.

Use random forests to predict cancer type based on the 500 genes that have the largest variance in the training set.

Example

Null choice (Normal)



Random Forests Issues

When the number of variables is large, but the fraction of relevant variables is small, random forests are likely to perform poorly when m is small

Why?

Because: At each split the chance can be *small* that the relevant variables will be selected

For example, with 3 relevant and 100 not so relevant variables the probability of any of the relevant variables being selected at any split is ~0.25

Boosting

Boosting is a general approach that can be applied to many statistical learning methods for regression or classification.

Bagging: Generate multiple trees from bootstrapped data and average the trees.

Recall bagging results in i.d. trees and not i.i.d.

RF produces i.i.d (or more independent) trees by randomly selecting a subset of predictors at each step

Boosting

Boosting works very differently.

1. Boosting does not involve bootstrap sampling
2. Trees are grown sequentially: each tree is grown using information from previously grown trees
3. Like bagging, boosting involves combining a large number of decision trees, $f^1, \ldots, f^B$

Boosting for regression

1. Set $f(x)=0$ and $r_i = y_i$ for all i in the training set.

2. For $b=1,2,\ldots,B$, repeat:

   a. Fit a tree with d splits (+1 terminal nodes) to the training data (X, r).

   b. Update the tree by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda\, \hat{f}^b(x)$$

   c. Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$