CLASSIFICATION—ESSENTIALS

Hari Sundaram

hs1@illinois.edu

http://sundaram.cs.illinois.edu

adapted from slides by Jiawei Han and Kevin Chang

BASIC CONCEPTS

Decision Trees Bayes Classification Rule-Based

Evaluation Ensemble Methods Summary

The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations

Supervised learning (classification)

New data is classified based on the training set

The class labels of training data is unknown

Unsupervised learning (**clustering**)

Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

predicts categorical class labels (discrete or nominal)

Classification

classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

Numeric Prediction

models continuous-valued functions, i.e., predicts unknown or missing values

Credit/loan approval

Medical diagnosis:
if a tumor is
cancerous or benign

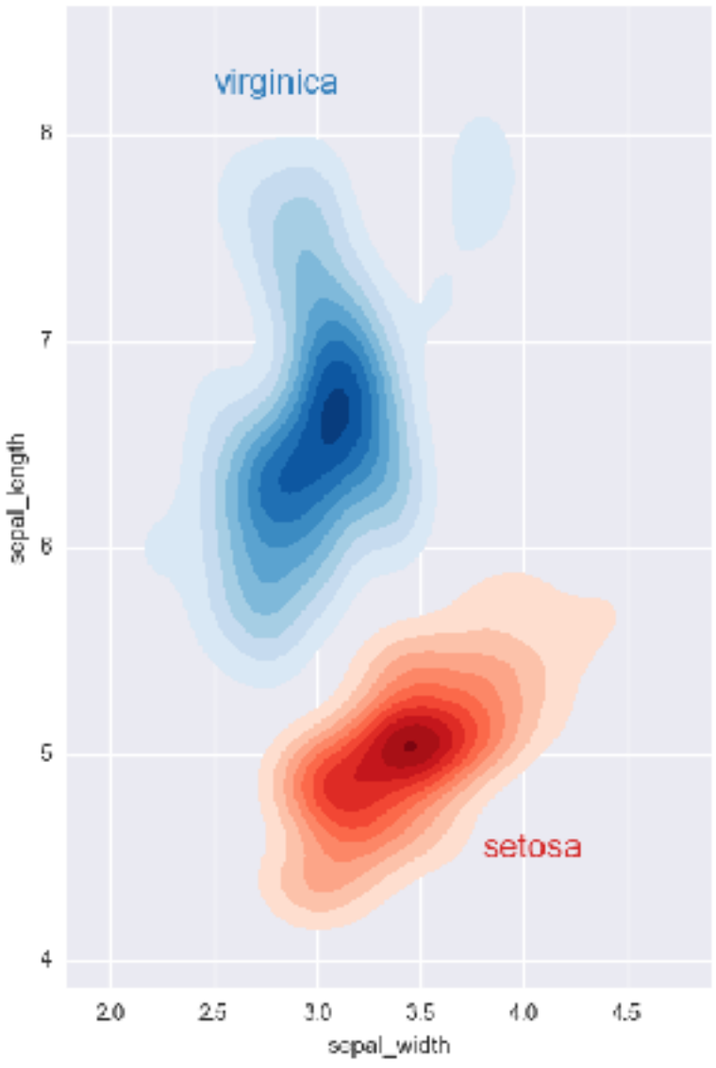
Applications

Fraud detection:
if a transaction is
fraudulent

Web page categorization

Classification—A Two-Step Process

MODEL CONSTRUCTION



Model construction: describing a set of predetermined classes

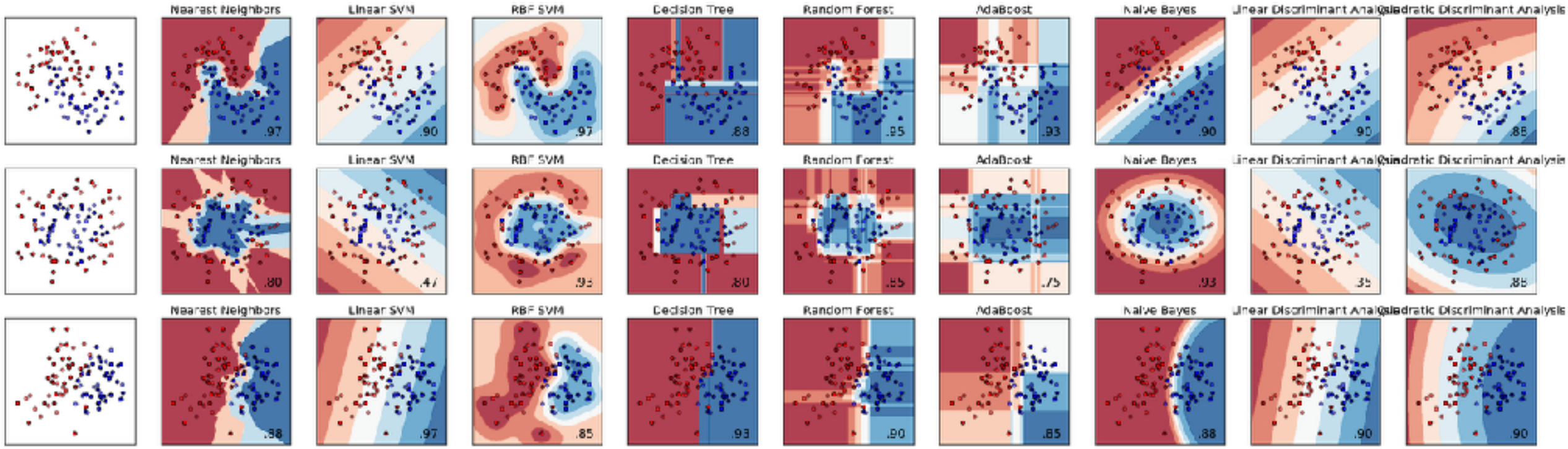
Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute

The set of tuples used for model construction is training set

The model is represented as classification rules, decision trees, or mathematical formulae

Model usage

Estimate accuracy of the model



Accuracy rate is the percentage of test set samples that are correctly classified by the model

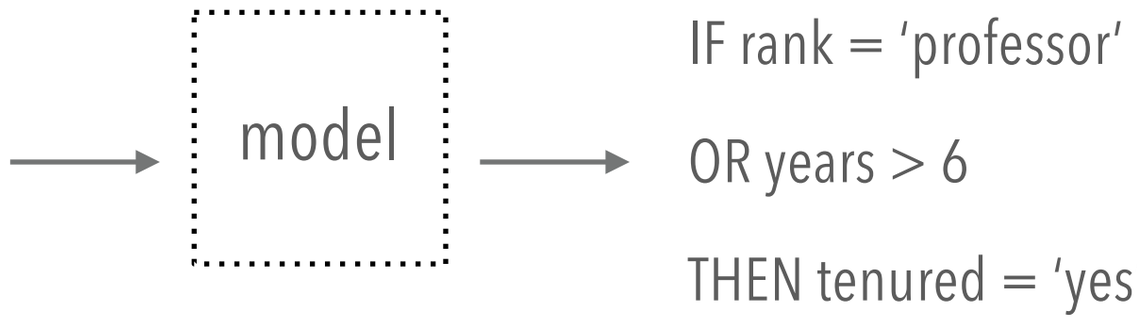
Test set is independent of training set (otherwise overfitting)

If the accuracy is acceptable, use the model to classify new data

Note: If the test set is used to select/refine models, it is called validation (test) set or development test set.

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no



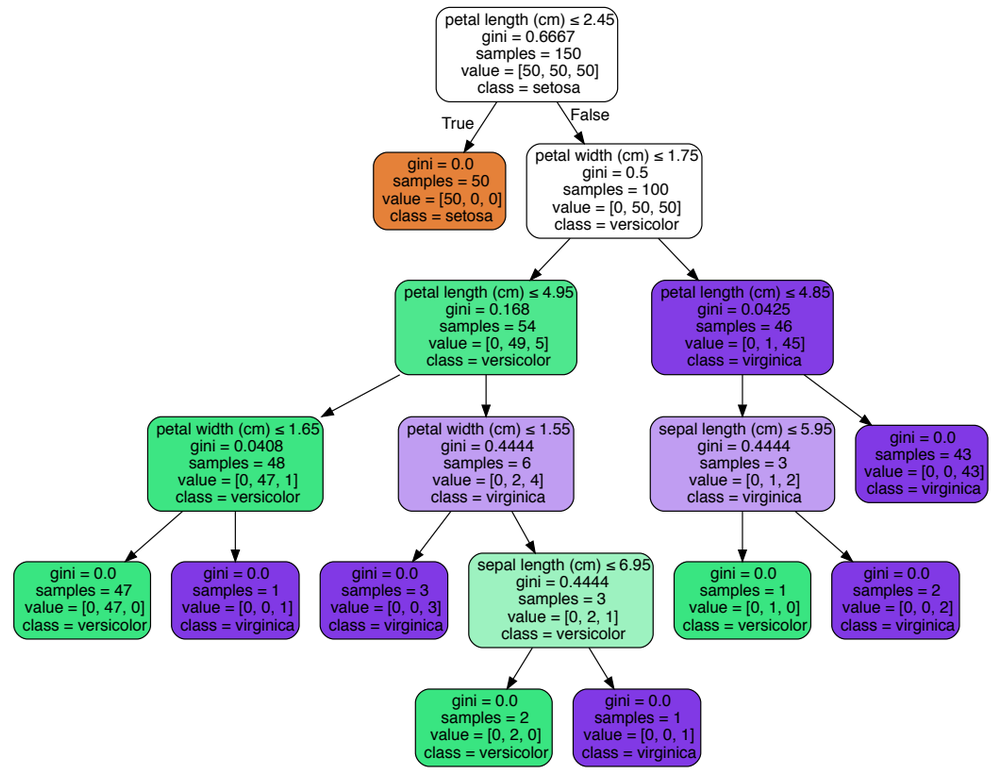
NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

test data

Jeff, Professor, 4→ Tenured?

yes!

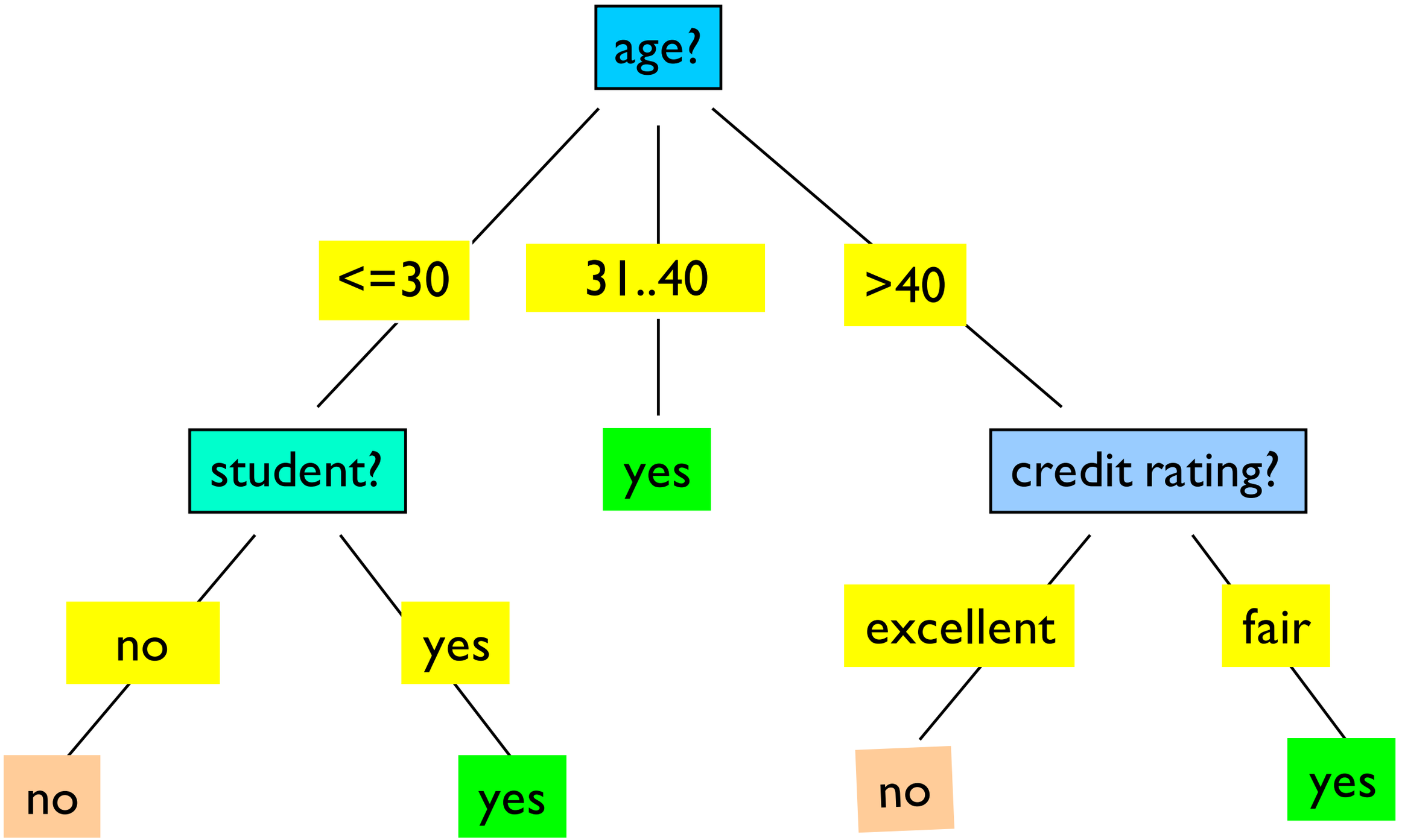
DECISION TREES



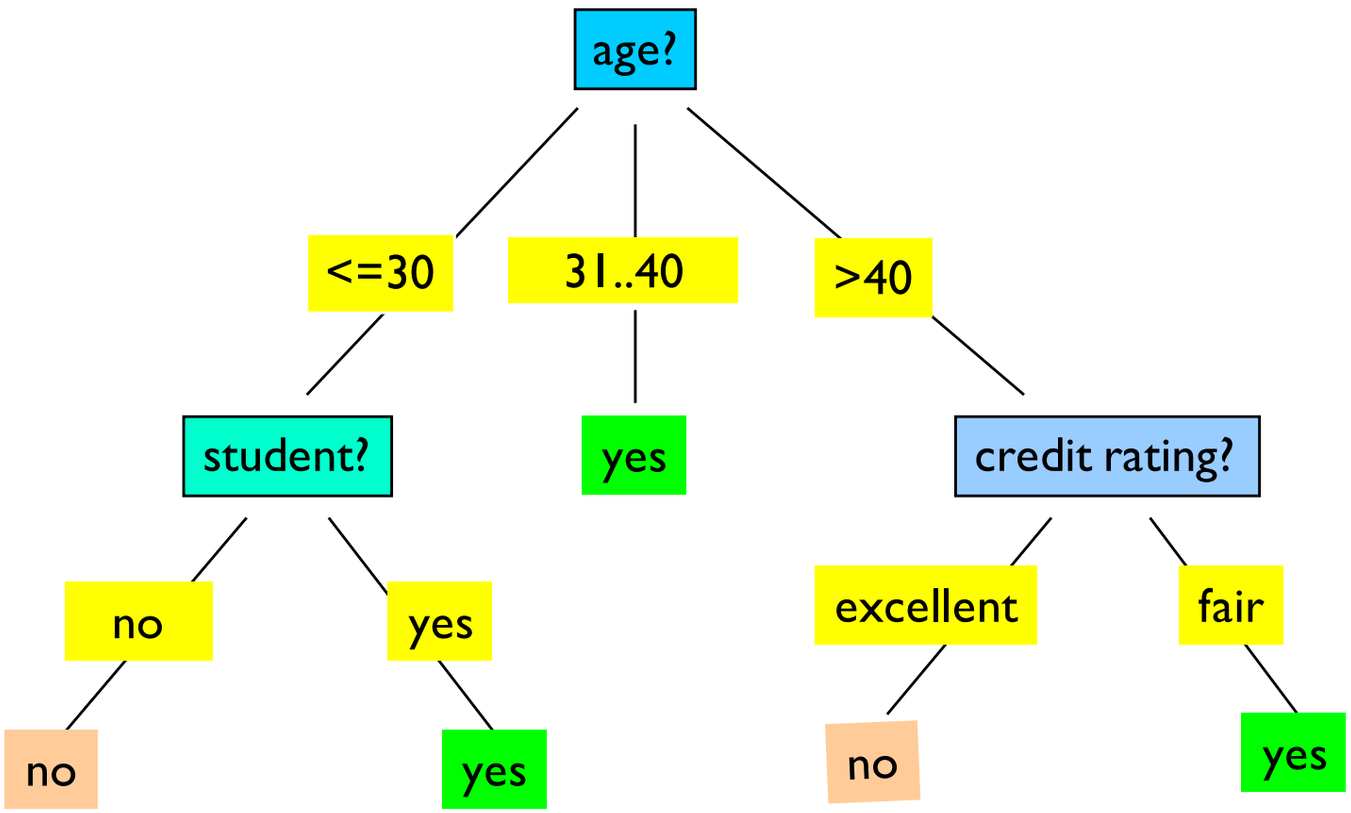
Basic Concepts Bayes Classification Rule-Based

Evaluation Ensemble Methods Summary

age	income	student	credit rating	buys computer
≤30	high	no	fair	no
≤30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
≤30	medium	no	fair	no
≤30	low	yes	fair	yes
>40	medium	yes	fair	yes
≤30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no


age?
<=30
31..40
>40
student?
yes
credit rating?
no
yes
excellent
fair
no
yes
no
yes

BASIC GREEDY ALGORITHM



Tree is constructed in a top-down recursive divide-and-conquer manner

At start, all the training examples are at the root

Attributes are categorical (if continuous-valued, they are discretized in advance)

Examples are partitioned recursively based on selected attributes

Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

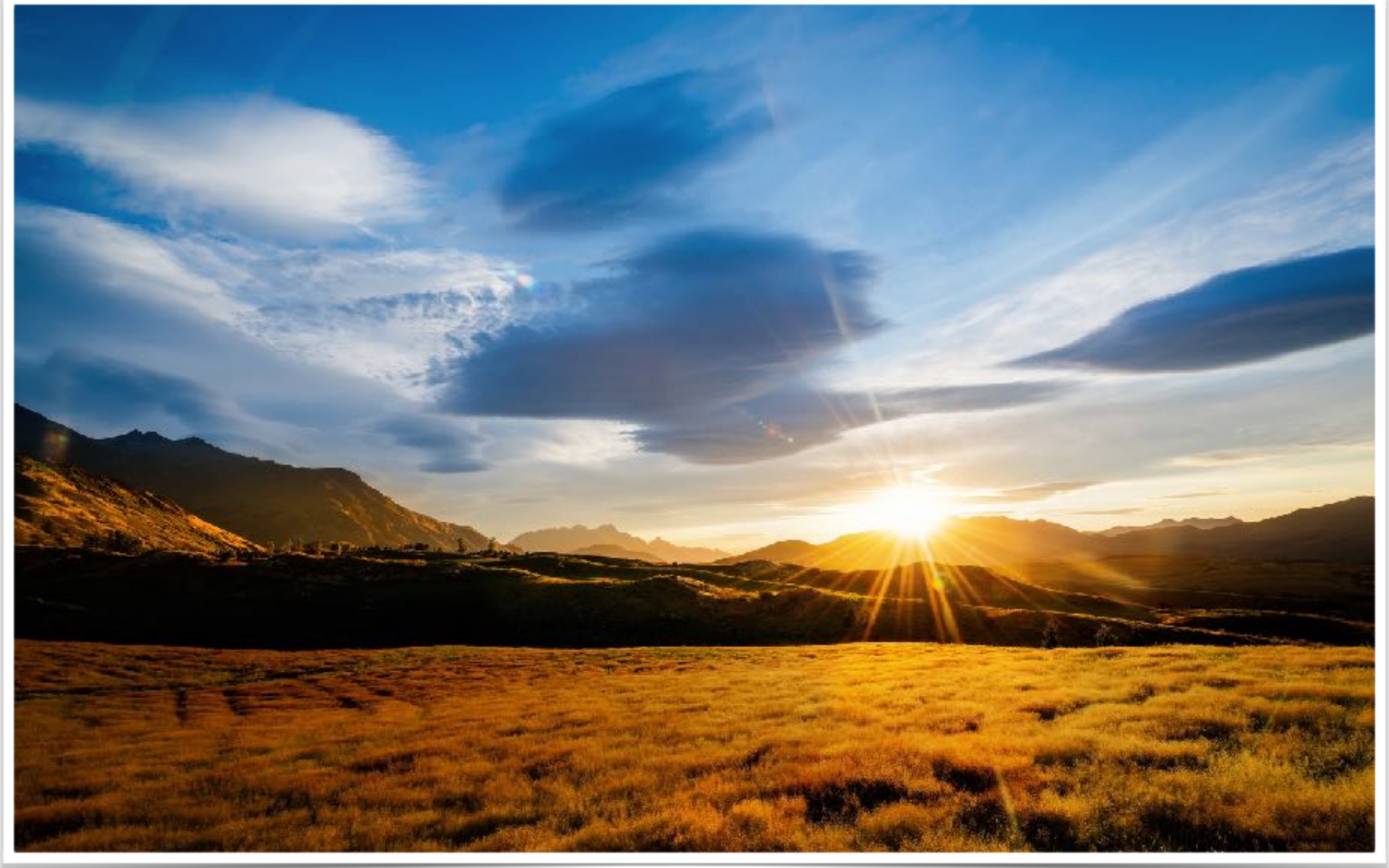
Conditions for stopping partitioning

- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
- There are no samples left


DETOUR

Will Congress ban Assault weapons in the current term?

Who will win the presidency in 2020?



what is the **common** theme?

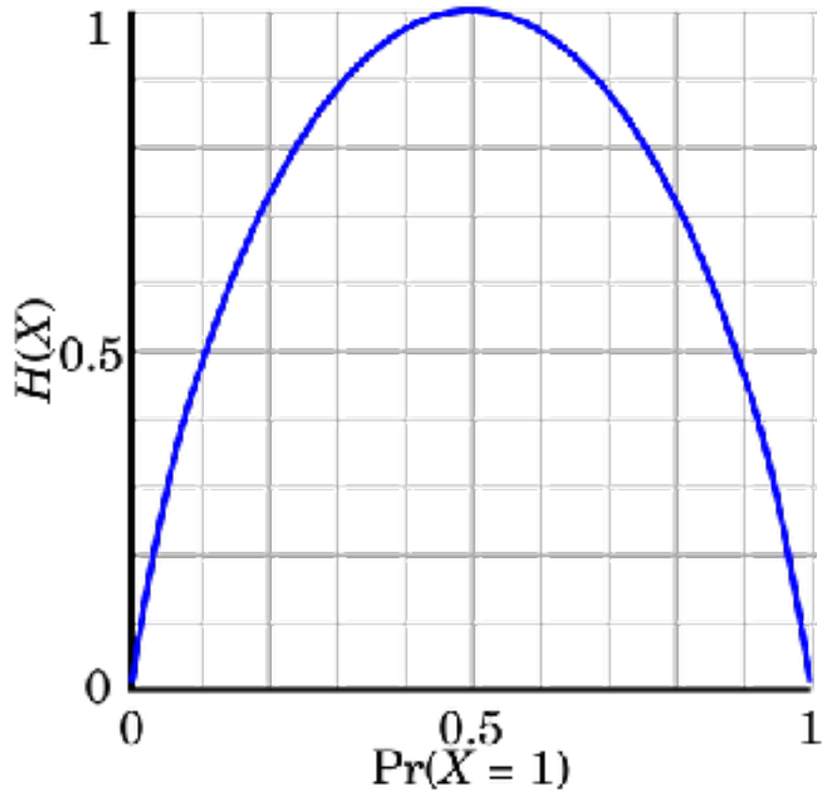
occurrence of a **rare event** is **more surprising**

Entropy is the expected Information

information of an event

$$H(X) = -\sum_i p_i \log p_i$$

$$H(X) = -\sum_i p_i \log p_i$$



$$H(Y/X) = -\sum_x p_x H(Y/X = x)$$

varying p, two state case

conditional entropy

$$H(Y/X) = -\sum_{x} p_x H(Y/X = x)$$

measure:

information gain

INFORMATION GAIN

Let p_i be the probability that an arbitrary tuple in D belongs to class C_i : $|C_{i,D}|/|D|$

Information Gain:

$$Info(D) = -\sum_i p_i \log p_i$$

Information needed (after using attribute A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} Info(D_j)$$

$$Gain(A) = Info(D) - Info_A(D)$$

how to select attributes?

age	income	student	credit rating	buys computer
≤30	high	no	fair	no
≤30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
≤30	medium	no	fair	no
≤30	low	yes	fair	yes
>40	medium	yes	fair	yes
≤30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940$$

looking at the age attribute

	"yes"	"no"	
Age	**P_i**	**N_i**	**$I(P_i, N_i)$**
≤30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$Gain_{Income} = 0.029$

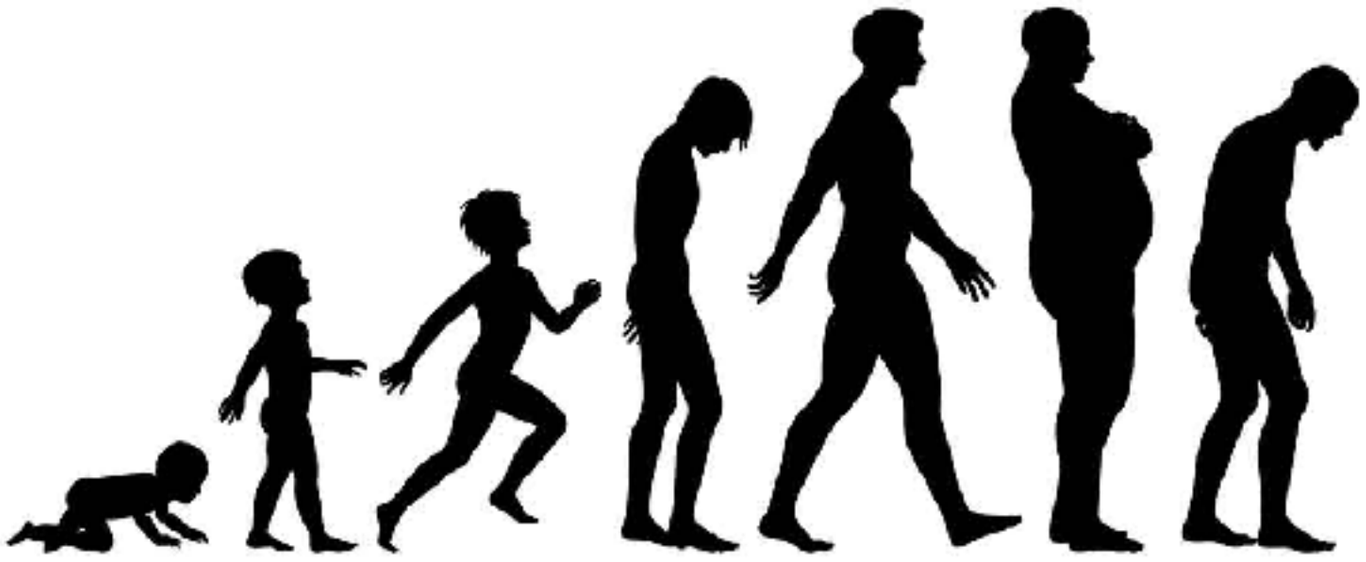
$Gain_{student} = 0.151$

$Gain_{credit\_rating} = 0.048$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} Info(D_j) = 0.694$$

$$Gain(A) = Info(D) - Info_{Age}(D) = 0.246$$

CONTINUOUS ATTRIBUTES



Let attribute A be a continuous-valued attribute

Must determine the best split point for A

- Sort the value A in increasing order
- Typically, the midpoint between each pair of adjacent values is considered as a possible split point
- $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
- The point with the minimum expected information requirement for A is selected as the split-point for A

Split:

- D_1 is the set of tuples in D satisfying A ≤ split-point, and D_2 is the set of tuples in D satisfying A > split-point

what is a **challenge with information gain?**

$$Gain(A) = Info(D) - Info_A(D)$$

GAIN RATIO (C4.5)

Information gain measure is biased towards attributes with a large number of values

C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

GainRatio(A) = Gain(A)/ SplitInfo(A)

Ex.

SplitInfo(Income)

GainRatio(income) = 0.029/1.557 = 0.019

The attribute with the maximum gain ratio is selected as the splitting attribute

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \log_2 \left(\frac{|D_j|}{|D|} \right)$$

GINI INDEX

If a data set D contains examples from n classes, gini index, gini(D) is defined as:

$$gini(D) = 1 - \sum_i p_i^2$$

If an attribute A is used to split the data:

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

reduction in impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

The attribute that provides the largest reduction in impurity is chosen to split the node

need to enumerate all the possible splitting points for each attribute

COMPARISONS

The three measures, in general, return good results but

Information gain:

- biased towards multivalued attributes

Gain ratio:

- tends to prefer unbalanced splits in which one partition is much smaller than the others

Gini index:

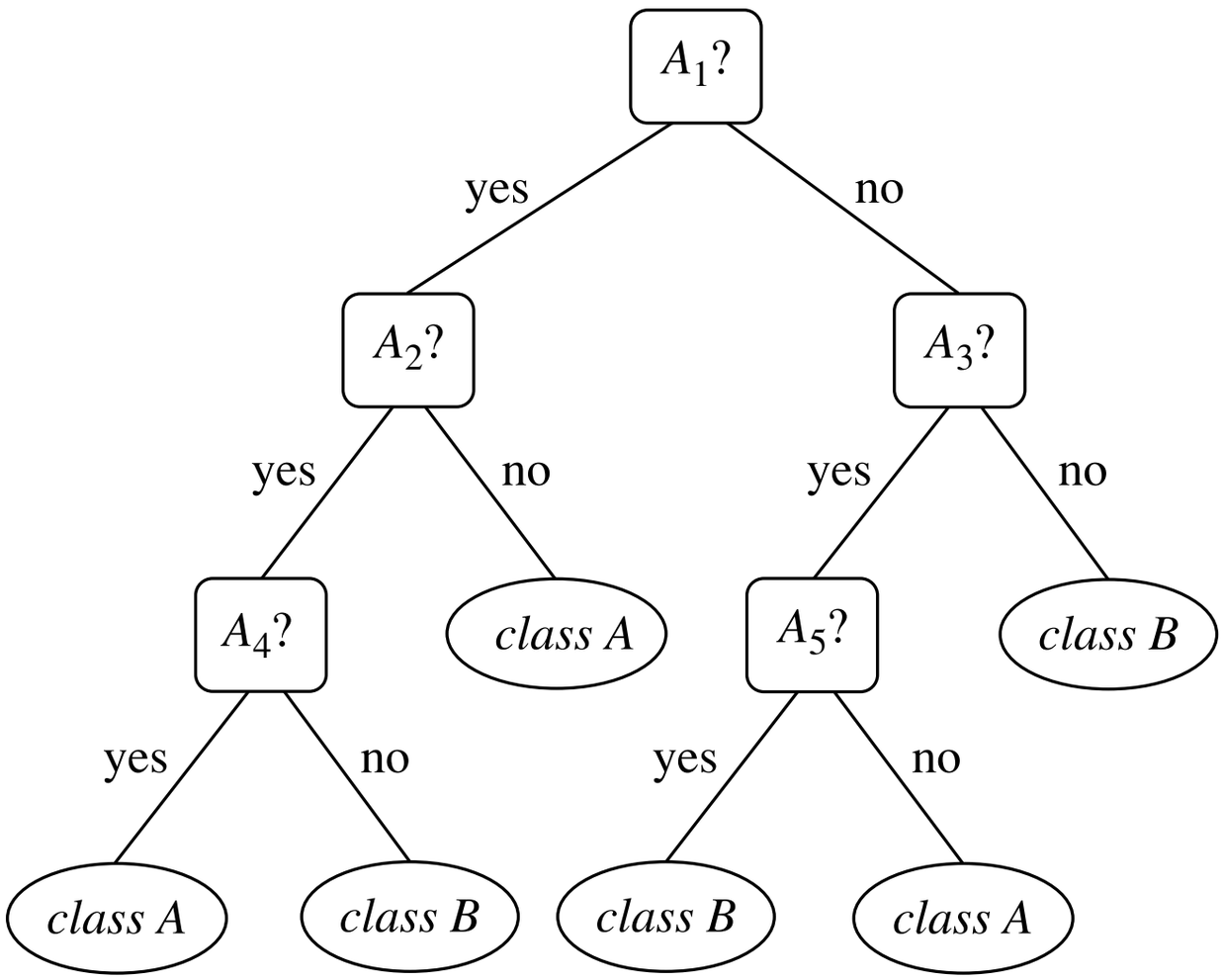
- biased to multivalued attributes
- has difficulty when # of classes is large
- tends to favor tests that result in equal-sized partitions and purity in both partitions

Too many branches, some may reflect anomalies due to noise or outliers

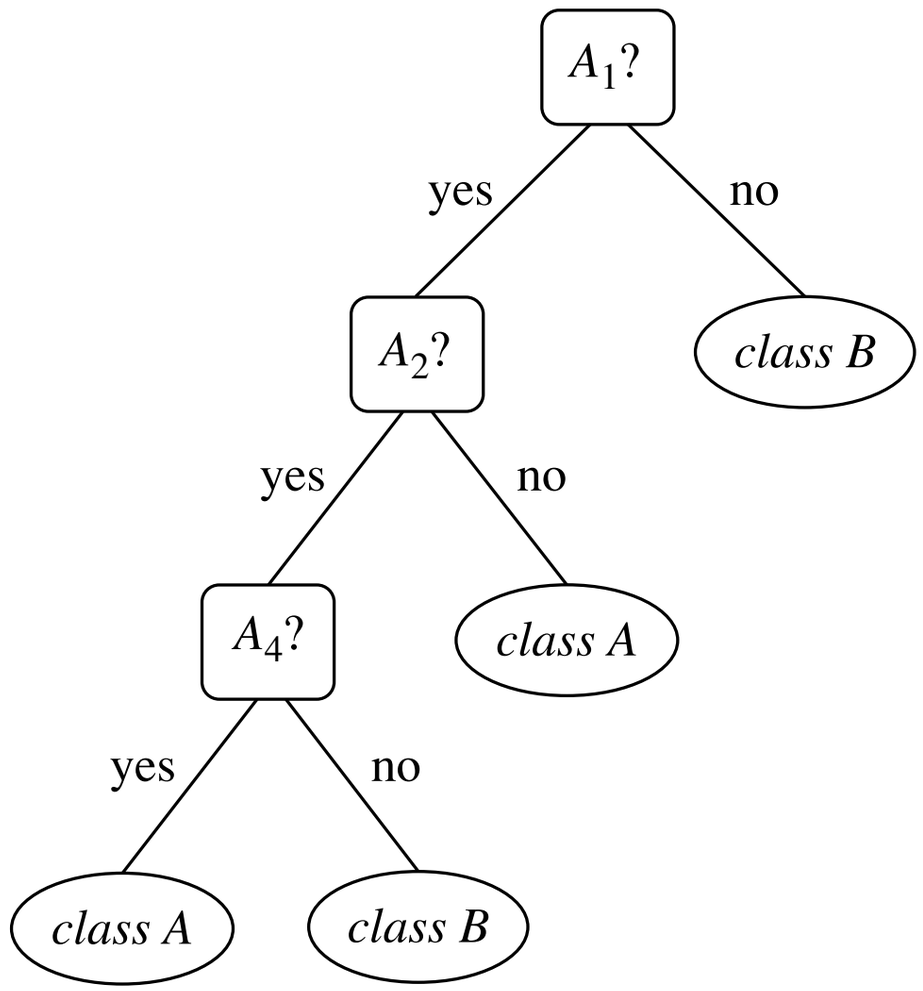
An induced tree may **overfit** the training data

Poor accuracy for unseen samples

Prepruning: Halt tree construction early– do not split a node if this would result in the goodness measure falling below a threshold



Postpruning: Remove branches from a "fully grown" tree–get a sequence of progressively pruned trees. Use a set of data different from the training data to decide which is the "best pruned tree"



"I'm not trying to change you—I'm trying to enhance you."

ENHANCEMENTS

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

BAYES CLASSIFICATION

A Bayes Classifier: **performs probabilistic prediction**, i.e., predicts class membership probabilities

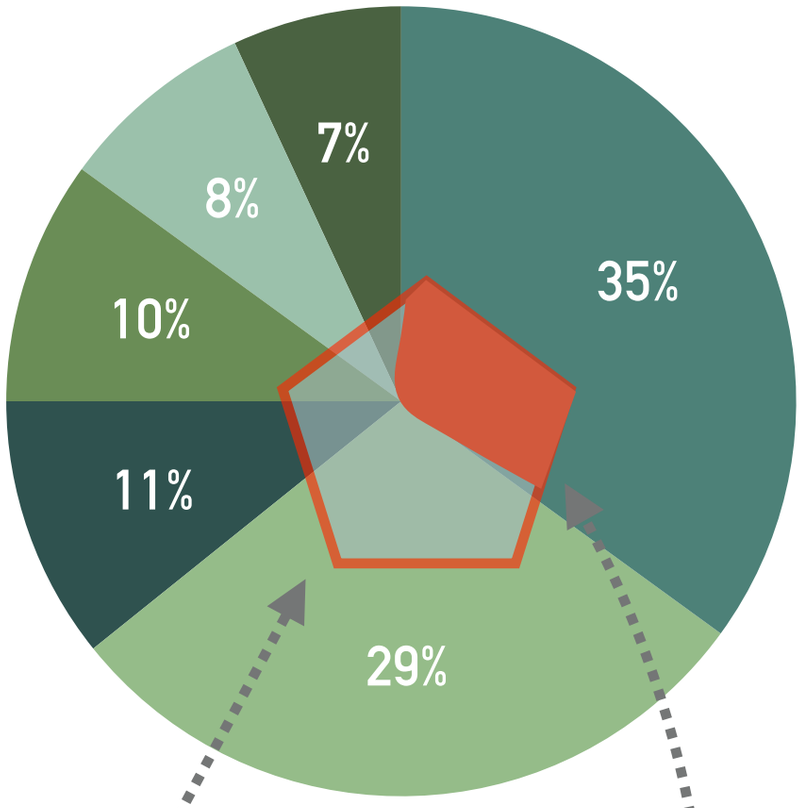
Bayes theorem

BAYES CLASSIFIER

Performance: A simple Bayesian classifier, naïve Bayesian classifier, has comparable performance with decision tree and selected neural network classifiers

Incremental: Each training example can incrementally increase/ decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data

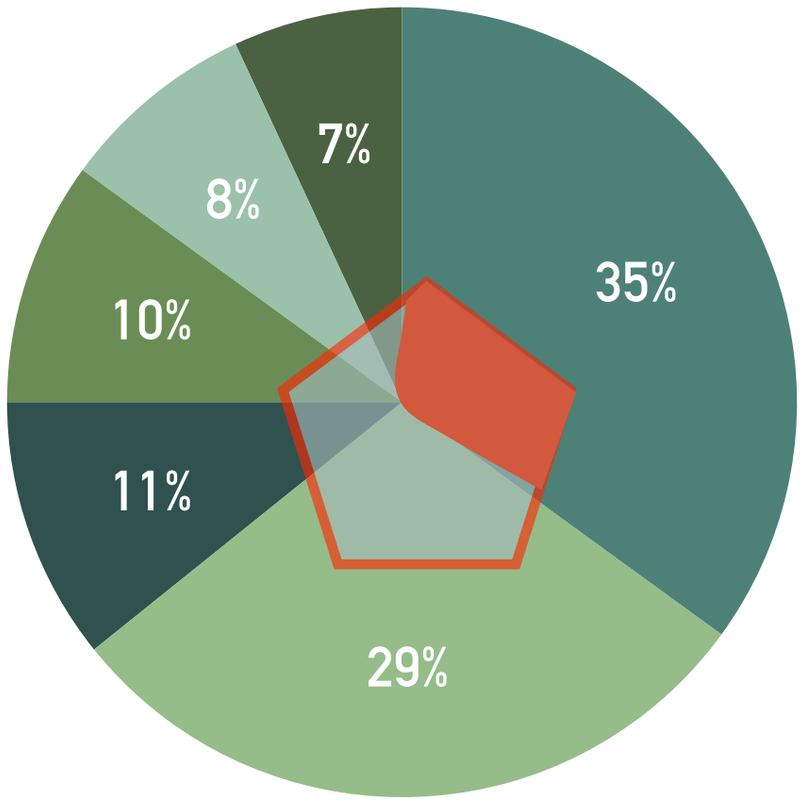
Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured



total probability

$$P(B) = \sum_i P(B|A_i)P(A_i)$$

partition

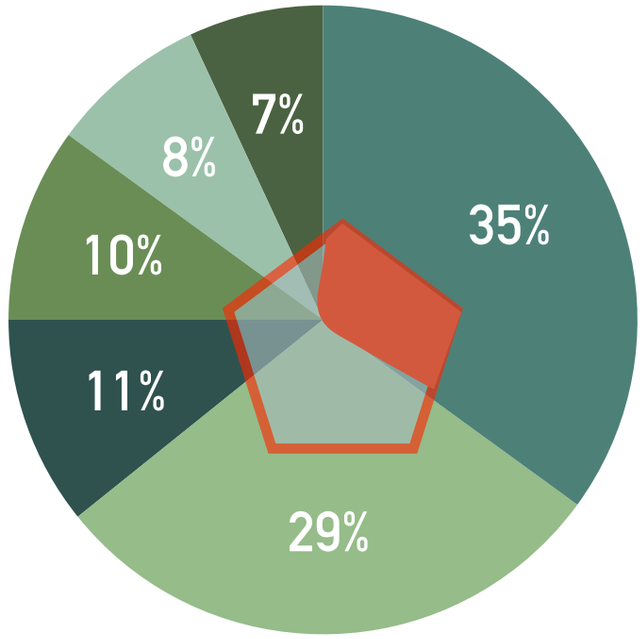


Bayes Theorem

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

"class"

"evidence"



$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

BASICS

Let X be a data sample ("evidence"): class label is unknown

Let H be a hypothesis that X belongs to class C

The classification goal is to determine P(H|X), (i.e., posteriori probability): the probability that the hypothesis holds given the observed data sample X

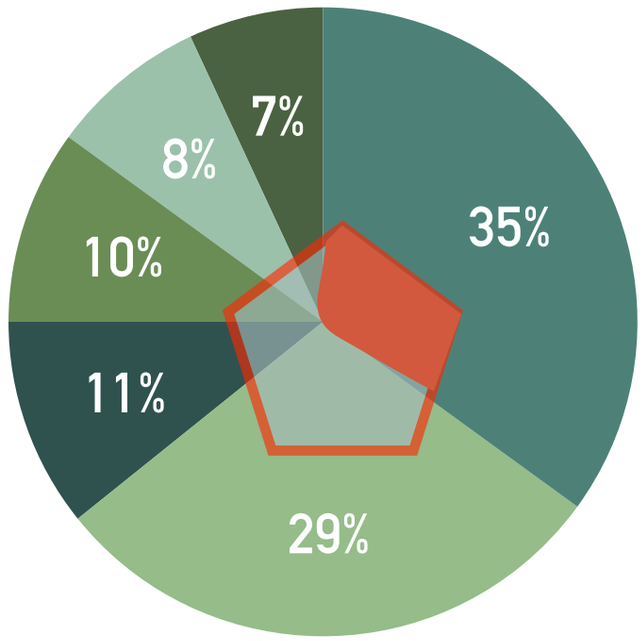
P(H) (prior probability): the initial probability

E.g., X will buy computer, regardless of age, income, …

P(X): probability that sample data is observed

P(X|H) (likelihood): the probability of observing the sample X, given that the hypothesis holds

E.g., Given that X will buy computer, the prob. that X is 31..40, medium income



$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

posterior = likelihood × prior / evidence

PREDICTION

Given training data X, posteriori probability of a hypothesis H, P(H|X), follows the Bayes' theorem

Informally, this can be viewed as posterior = likelihood x prior/ evidence

Predicts X belongs to C_i iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the k classes

Practical difficulty: It requires initial knowledge of many probabilities, involving significant computational cost

MAXIMUM APOSTERIORI CLASSIFIER

Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector $X = (x_1, x_2, \ldots, x_n)$

Suppose there are m classes $C_1, C_2, \ldots, C_m$.

Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|X)$

This can be derived from Bayes' theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

Since P(X) is constant for all classes, only

$$P(C_i|X) = P(X|C_i)P(C_i)$$

needs to be maximized

This greatly reduces the computation cost: Only counts the class distribution

Naïve Bayes

$$P(C_i|X) = P(X|C_i)P(C_i)$$

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

attributes are independent given class

THE PROBABILITIES

If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_i,D|$ (# of tuples of C_i in D)

If A_k is continuous, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(x_k|C_i)$ is

$$P(X|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

age	income	student	credit rating	buys computer
≤30	high	no	fair	no
≤30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
≤30	medium	no	fair	no
≤30	low	yes	fair	yes
>40	medium	yes	fair	yes
≤30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

X = (age ≤30, Income = medium, Student = yes, Credit_rating = Fair)

AN EXAMPLE

age	income	student	credit rating	buys computer
≤30	high	no	fair	no
≤30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
≤30	medium	no	fair	no
≤30	low	yes	fair	yes
>40	medium	yes	fair	yes
≤30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

P(Ci):

P(buys_computer = "yes") = 9/14 = 0.643

P(buys_computer = "no") = 5/14= 0.357

Compute P(X|Ci) for each class

P(age = "≤30"|buys_computer = "yes") = 2/9 = 0.222

P(age = "≤30"|buys_computer = "no") = 3/5 = 0.6

P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444

P(income = "medium" | buys_computer = "no") = 2/5 = 0.4

P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667

P(student = "yes" | buys_computer = "no") = 1/5 = 0.2

P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667

P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

AN EXAMPLE

age	income	student	credit rating	buys computer
≤30	high	no	fair	no
≤30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
≤30	medium	no	fair	no
≤30	low	yes	fair	yes
>40	medium	yes	fair	yes
≤30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

X = (age ≤30 , income=medium, student = yes, credit_rating = fair)

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X| buys_computer = "yes") * P(buys_computer ="yes") = 0.028

P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

AVOIDING THE ZERO PROBABILITY ISSUE

Naïve Bayesian prediction requires each conditional probability be non-zero. Else, the predicted probability will be 0

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)

Use Laplacian correction (or Laplacian estimator)

Adding 1 to each case

Prob(income = low) = 1/1003

Prob(income = medium) = 991/1003

Prob(income = high) = 11/1003

The "corrected" prob. estimates are close to their "uncorrected" counterparts

ISSUES

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.
    - Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier

exercise!

Given the following statistics, what is the **probability** that a woman over the age of 50 has cancer if she has a positive mammogram result?

p=0.1

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

- One percent of women over 50 have breast cancer.
- Ninety percent of women who have breast cancer test positive on mammograms.
- Eight percent of women will have false positives.

RULE BASED CLASSIFICATION

Basic Concepts Decision Trees Bayes

Evaluation Ensemble Methods Summary

IF age = youth AND student = yes THEN buys_computer = yes

IF-Then Rules

rule

n_{covers} = # of tuples covered by R

coverage

$$coverage(R) = n_{covers} / |D|$$

assessment

$n_{correct}$ = # of tuples correctly classified by R

accuracy

$$accuracy(R) = n_{correct} / n_{covers}$$

"We're fighting like—well, we're fighting."

how to resolve conflicts?

CONFLICT RESOLUTION

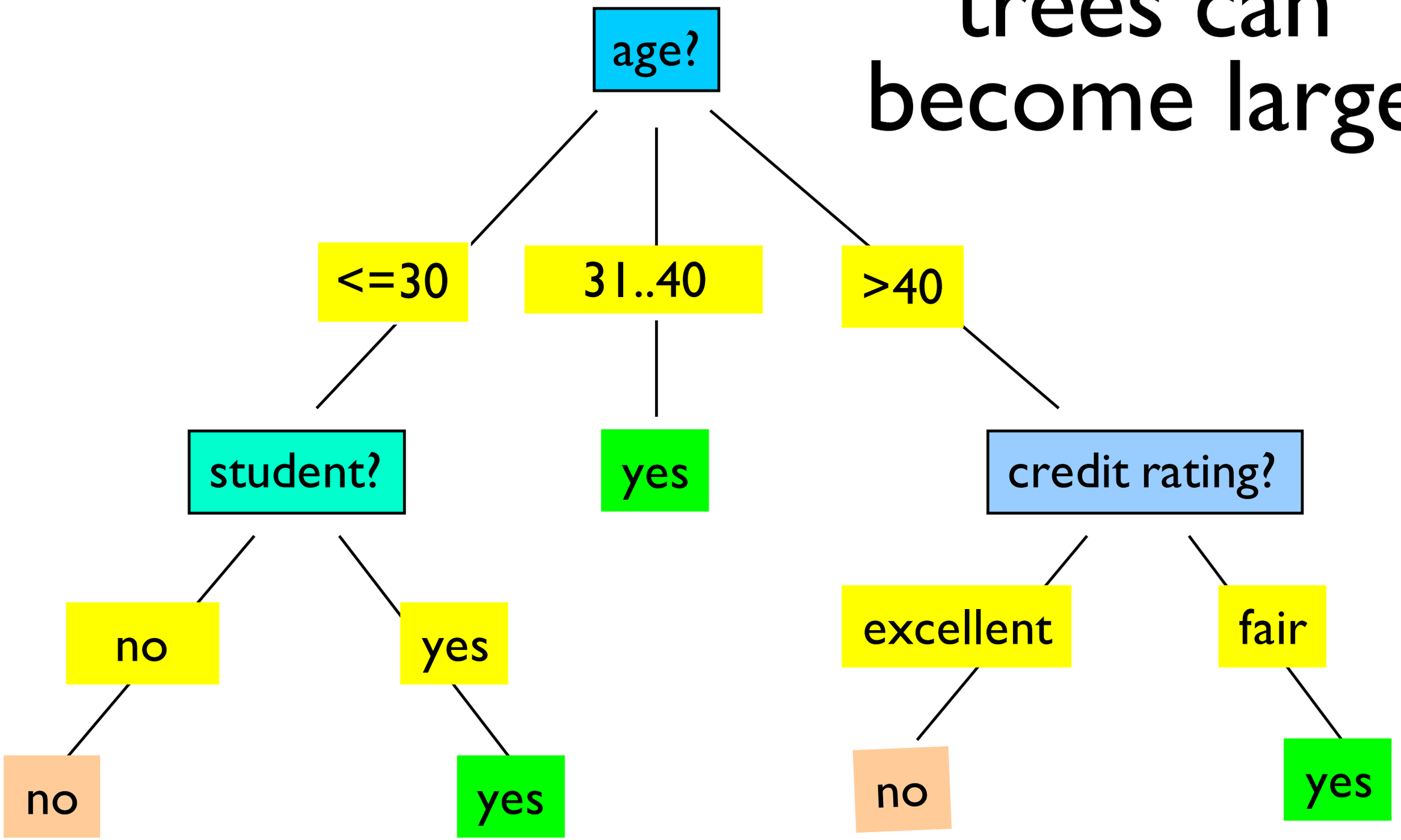
"We're fighting like—well, we're fighting."

Size ordering: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the most attribute tests)

Class-based ordering: decreasing order of prevalence or misclassification cost per class

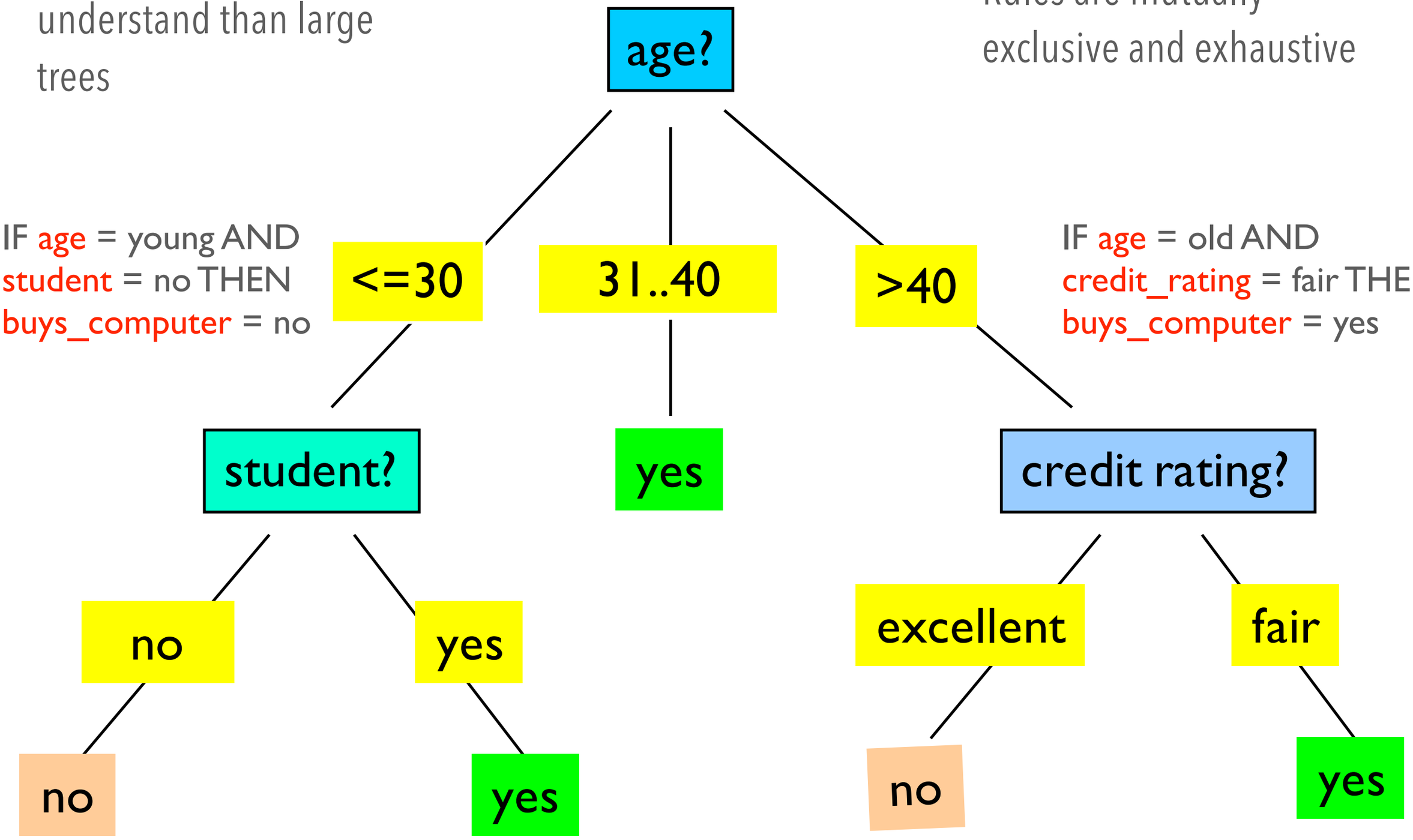
Rule-based ordering (decision list): rules are organized into one long priority list, according to some measure of rule quality or by experts

trees can become large



Rules are easier to understand than large trees

Rules are mutually exclusive and exhaustive



One rule is created for each path from the root to a leaf

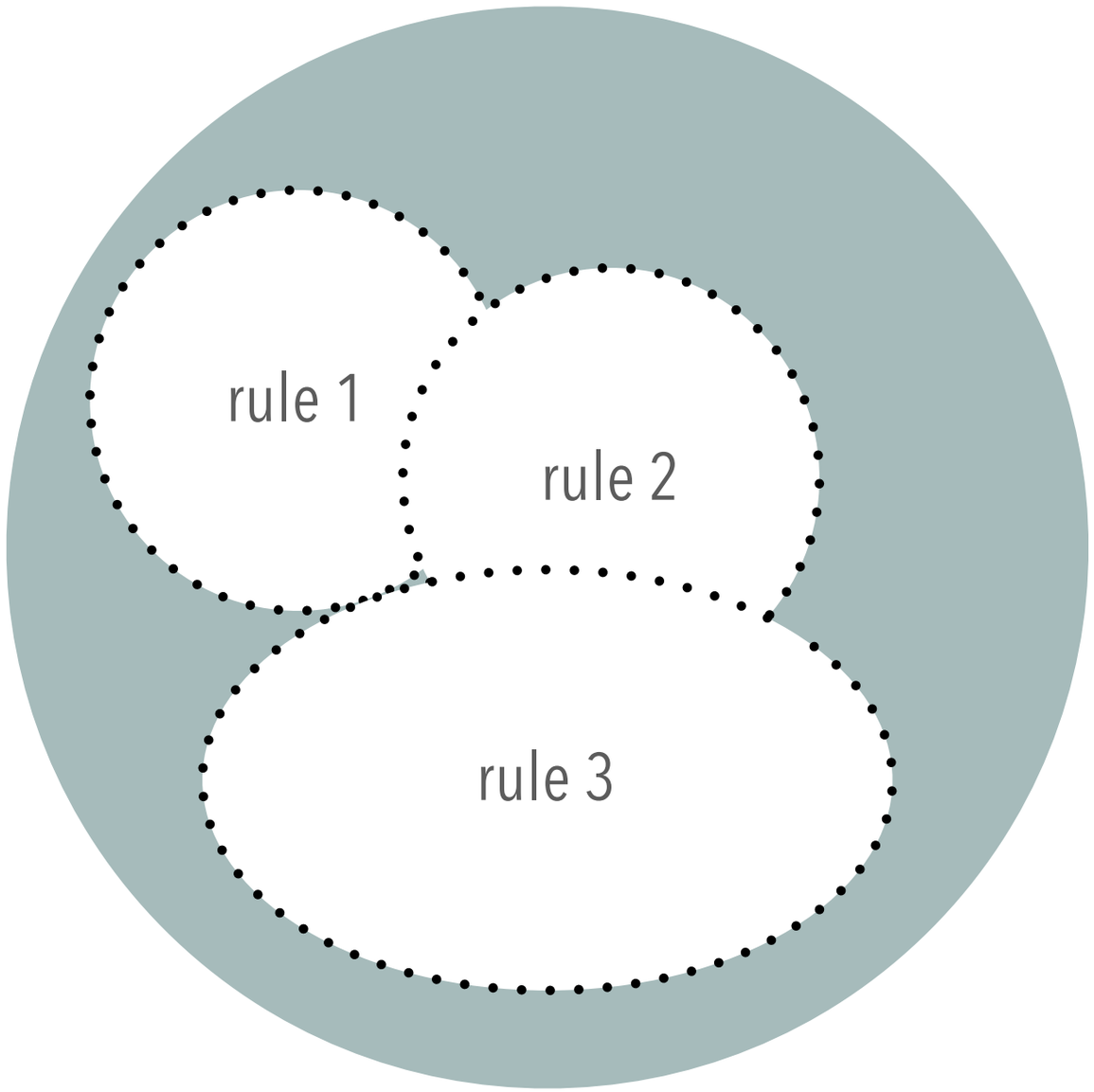
Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction

Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER

Sequential covering algorithm

Rules are learned sequentially, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes

Extracts rules directly from training data



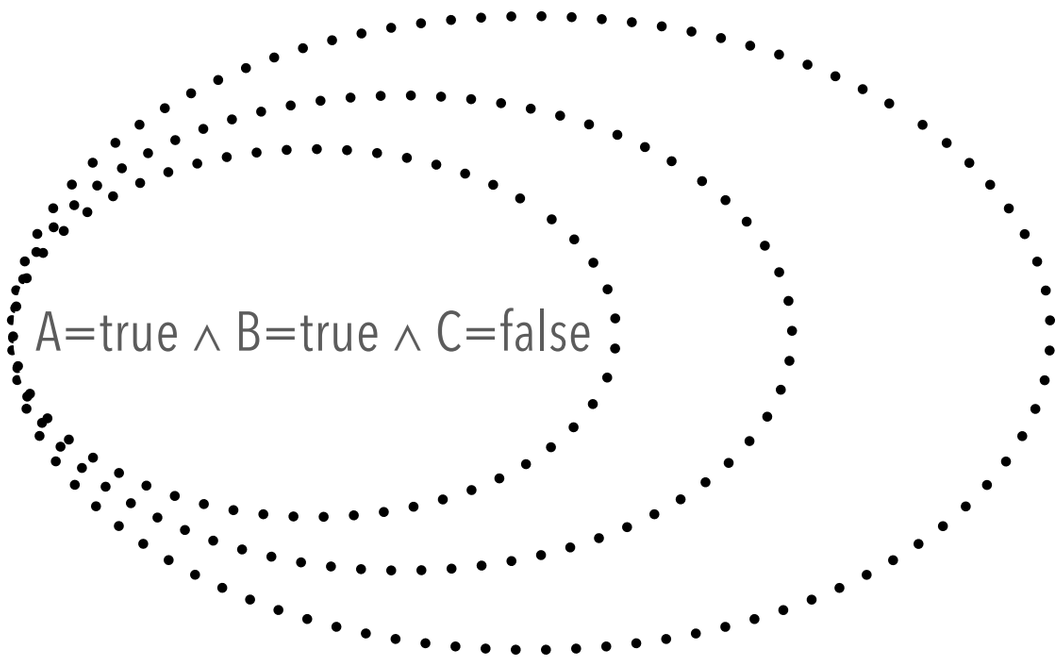
STEPS

1. Rules are learned one at a time
2. Each time a rule is learned, the tuples covered by the rules are removed
3. Repeat the process on the remaining tuples until termination condition, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold

how to learn one rule?

Adding new attributes by adopting a greedy depth-first strategy

Picks the one that most improves the rule quality



Start with the most general rule possible: condition = empty

RULE GENERATION

Rule-Quality measures: consider both coverage and accuracy

Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition

$$FOIL_{Gain} = pos' \times \left(\log_2 \frac{pos'}{pos' + neg'} - \log_2 \frac{pos}{pos + neg} \right)$$

favors rules that have high accuracy and cover many positive tuples

Rule pruning based on an independent set of test tuples

$$FOIL_{Prune}(R) = \frac{pos - neg}{pos + neg}$$

Pos/neg are # of positive/negative tuples covered by R.

If FOIL_Prune is higher for the pruned version of R, prune R

EVALUATION

Use validation test set of class-labeled tuples instead of training set when assessing accuracy

How can we measure accuracy?

Comparing classifiers:

- Confidence intervals
- Cost-benefit analysis and ROC Curves

Methods for estimating a classifier's accuracy:

- Holdout method, random subsampling
- Cross-validation
- Bootstrap

confusion matrix

Classes	*buys_computer = yes*	*buys_computer = no*	*Total*	*Recognition (%)*
buys_computer = yes	**6954**	**46**	7000	99.34
buys_computer = no	**412**	**2588**	3000	86.27
Total	7366	2634	10,000	95.42

Predicted class

Actual class	*yes*	*no*	Total
yes	*TP*	*FN*	*P*
no	*FP*	*TN*	*N*
Total	P'	N'	$P + N$

lots of jargon

Measure	*Formula*
accuracy, recognition rate	$\frac{TP+TN}{P+N}$
error rate, misclassification rate	$\frac{FP+FN}{P+N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP+FP}$
F, F_1, F-score, harmonic mean of precision and recall	$\frac{2 \times precision \times recall}{precision + recall}$
F_β, where β is a non-negative real number	$\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$

Classes	*buys_computer = yes*	*buys_computer = no*	*Total*	*Recognition (%)*
buys_computer = yes	**6954**	**46**	7000	99.34
buys_computer = no	**412**	**2588**	3000	86.27
Total	7366	2634	10,000	95.42

a mental image of the confusion matrix is pretty much all you need

ACCURACY, ERROR RATE

Classifier Accuracy, or recognition rate: percentage of test set tuples that are correctly classified

Accuracy = (TP + TN)/All

Error rate: 1 – accuracy, or

Error rate = (FP + FN)/All

Predicted class

		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$


how valuable
is a test with
an accuracy
of 99%?

CLASS IMBALANCE

One class may be rare, e.g. fraud, or HIV-positive

Significant majority of the negative class and minority of the positive class

Sensitivity: True Positive recognition rate

Sensitivity = TP/P

Specificity: True Negative recognition rate

Specificity = TN/N

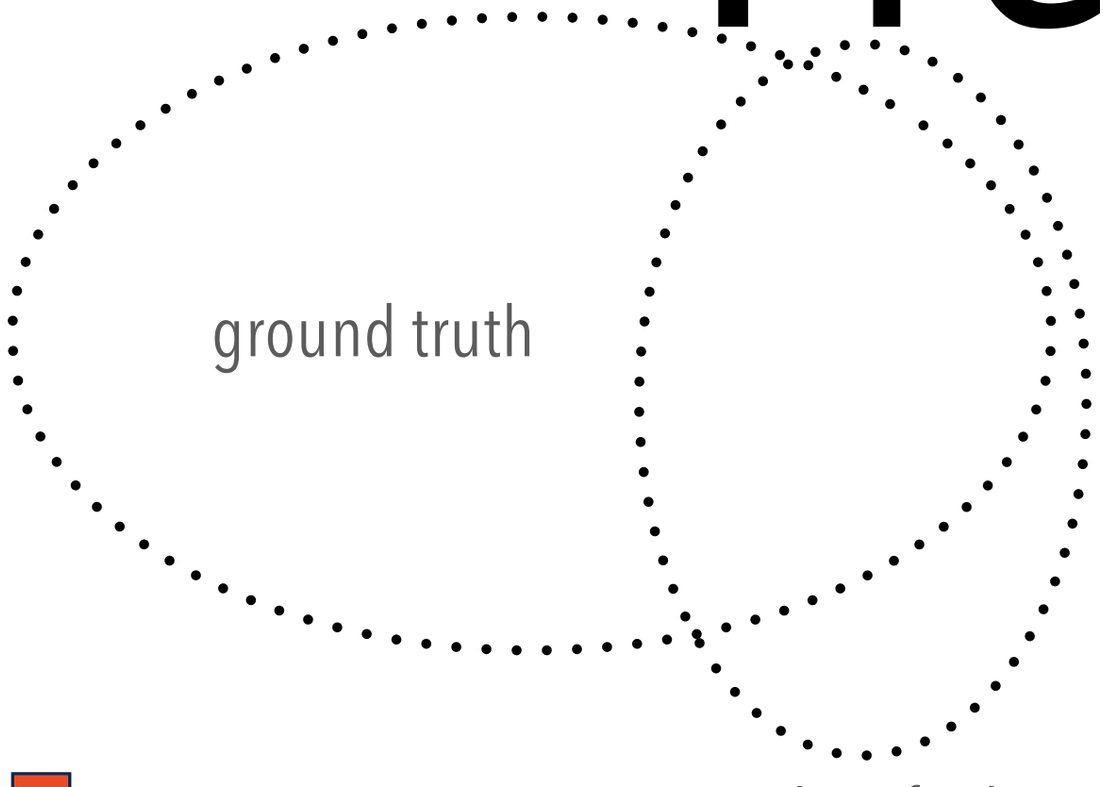
Predicted class

Actual class		*yes*	*no*	Total
	yes	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

Predicted class

Actual class	*yes*	*no*	Total
yes	*TP*	*FN*	*P*
no	*FP*	*TN*	*N*
Total	P'	N'	$P + N$

Precision



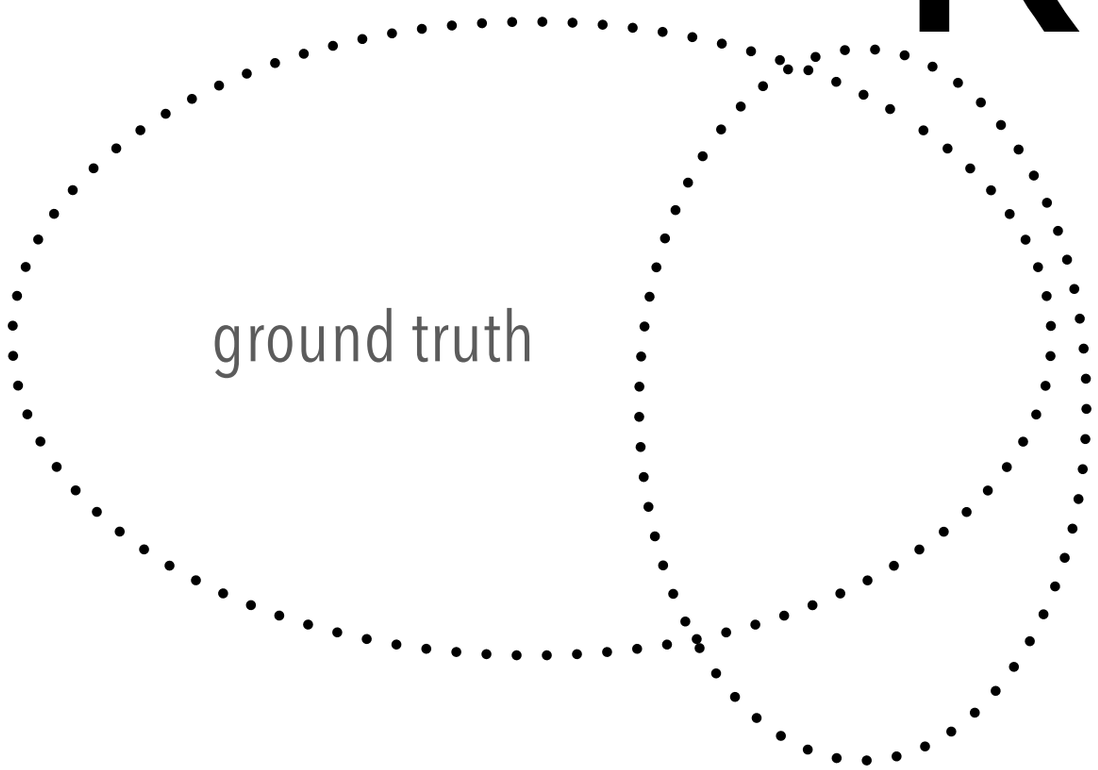
what % of tuples that the classifier labeled as positive are actually positive

$$P = \frac{TP}{TP + FP}$$

Predicted class

Actual class	*yes*	*no*	Total
yes	*TP*	*FN*	*P*
no	*FP*	*TN*	*N*
Total	P'	N'	$P + N$

Recall



what % of positive tuples did the classifier label as positive?

$$R = \frac{TP}{TP + FN}$$

inverse relationship between precision and recall

$$P = \frac{TP}{TP + FP} \qquad R = \frac{TP}{TP + FN}$$

F-score

harmonic mean of precision and recall

balanced $F_1 = \frac{2PR}{P+R}$

assigning β times more weight to recall $F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}$

compute precision and recall

Actual Class\Predicted class	cancer = yes	cancer = no	Total	Recognition(%)
cancer = yes	**90**	**210**	300	30.00 (*sensitivity*
cancer = no	**140**	**9560**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

Predicted class

Actual class	*yes*	*no*	Total
yes	*TP*	*FN*	*P*
no	*FP*	*TN*	*N*
Total	P'	N'	$P + N$

Training set (e.g., 2/3)
for model construction

Data is randomly
partitioned into two
independent sets

Test set (e.g., 1/3) for
accuracy estimation

holdout method

Random sampling: a variation of holdout

Repeat holdout k times,
accuracy = avg. of the
accuracies obtained

k-fold cross validation

Randomly partition the data into k mutually exclusive subsets, each approximately equal size

At i-th iteration, use D_i as test set and others as training set

Stratified cross-validation: folds are stratified so that class distribution in each fold is approximately the same as that in the initial data

Leave-one-out: k folds where k = # of tuples, for small sized data

bootstrap

Works well with small data sets

Samples the given training tuples uniformly with replacement

0.632 BOOTSTRAP

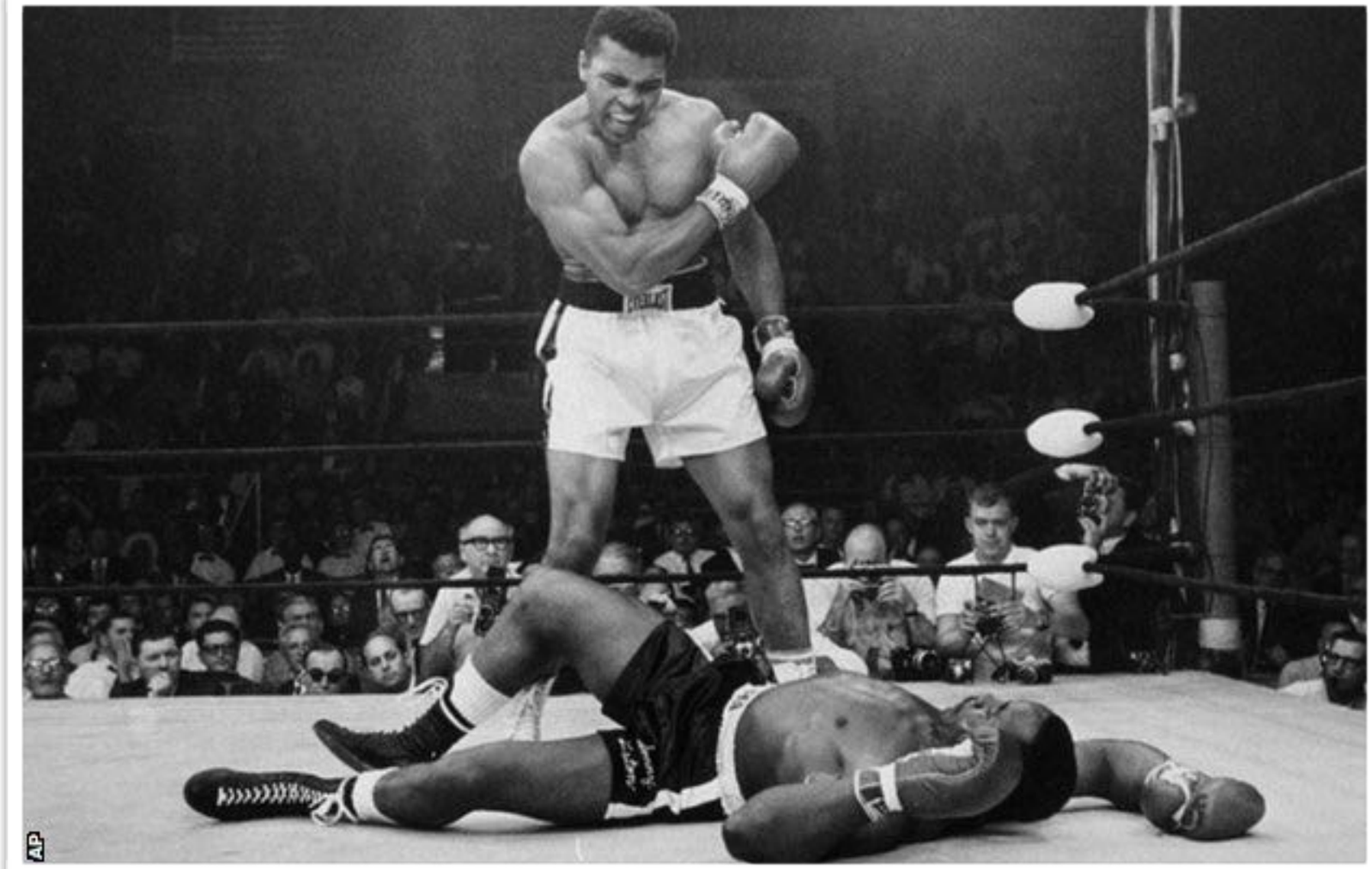
A data set with d tuples is sampled d times, with replacement, resulting in a training set of d samples.

The data tuples that did not make it into the training set end up forming the test set.

About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx 1/e = 0.368$)

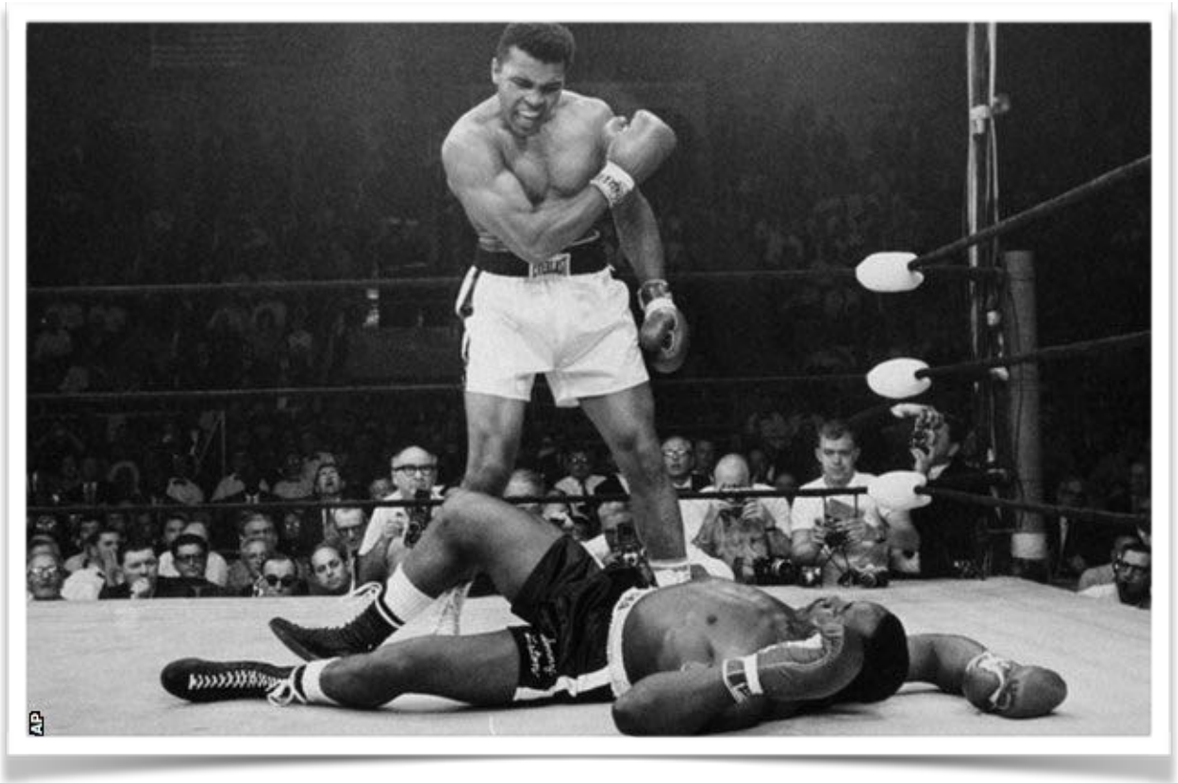
Repeat the sampling procedure k times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k}\sum_{i=1}^{k} 0.632 \times Acc(M_i)_{test\_set} + 0.368 \times Acc(M_i)_{train\_set}$$



how to choose between models M_1 and M_2?

EVALUATING TWO MODELS



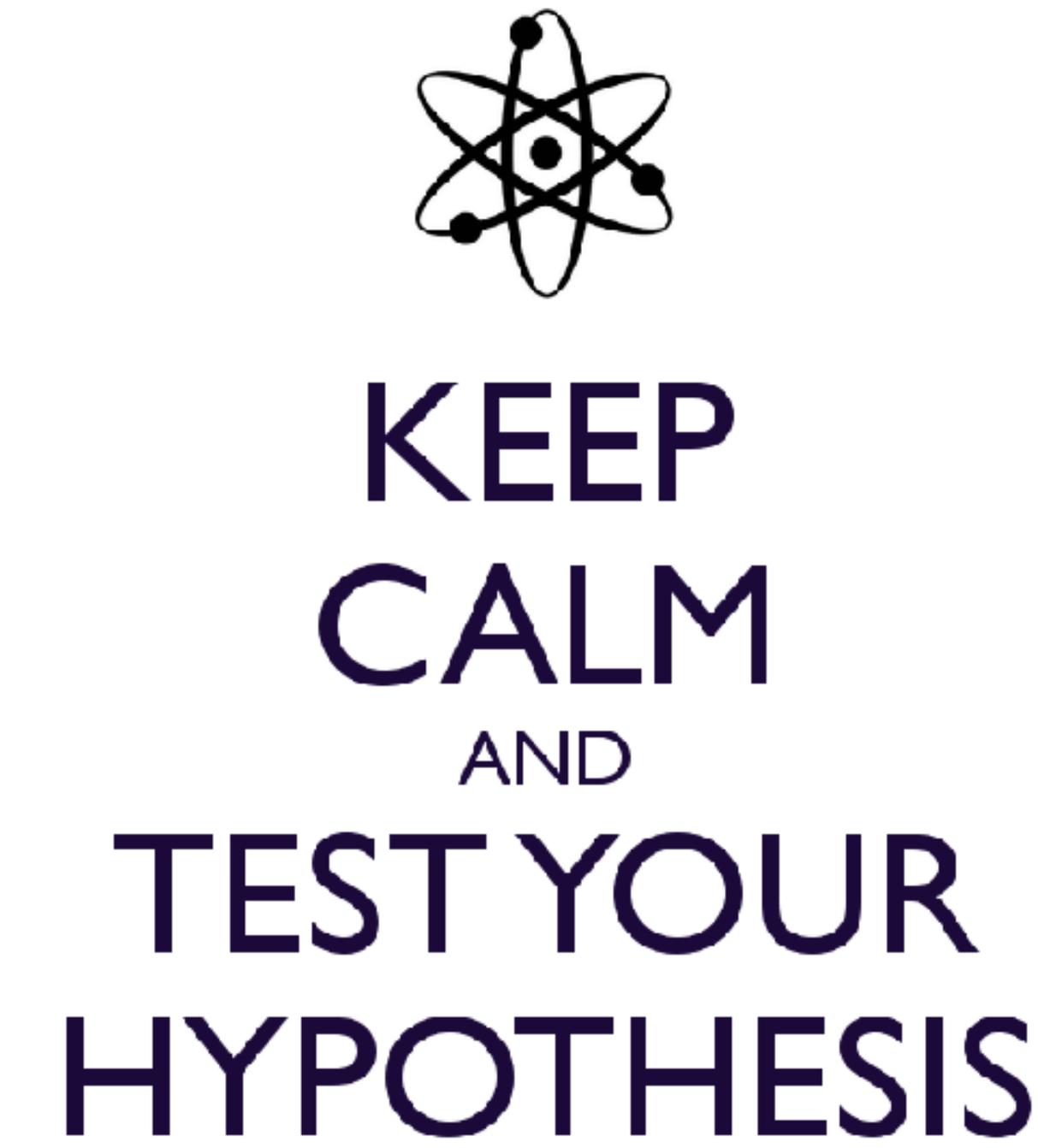
Suppose we have 2 classifiers, M_1 and M_2, which one is better?

Use 10-fold cross-validation to obtain **err**(M_1) and **err**(M_2)

These mean error rates are just estimates of error on the true population of future data cases

What if the difference between the two error rates is just attributed to chance?

- Use a test of statistical significance
- Obtain confidence limits for our error estimates



STATISTICAL TESTS

Perform 10-fold cross-validation

Assume samples follow a t distribution with k–1 degrees of freedom (here, k=10)

Use t-test (or Student's t-test)

Null Hypothesis: M_1 & M_2 are the same

If we can reject null hypothesis, then

- we conclude that the difference between M_1 & M_2 is statistically significant
- Chose model with lower error rate

T-TEST

If only 1 test set available: pairwise comparison

For i-th round of 10-fold cross-validation, the same cross partitioning is used to obtain **err**(M_1)i and **err**(M_2)i

Average over 10 rounds to get average **err**(M_1) and average **err**(M_2)

t-test computes t-statistic with k-1 degrees of freedom:

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}},$$

$$var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))]^2 .$$

T-TEST

If two test sets available: use non-paired t-test

t-test computes t-statistic with k-1 degrees of freedom:

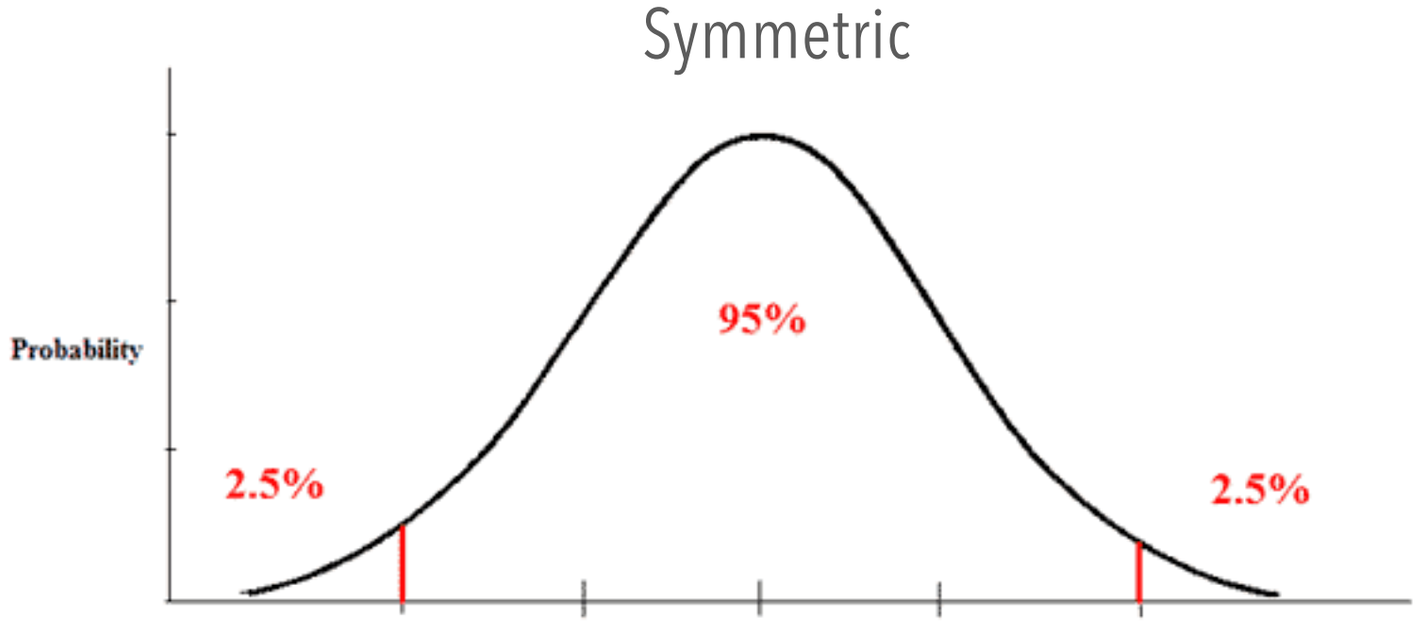
$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}},$$

$$var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}},$$

different number of rounds

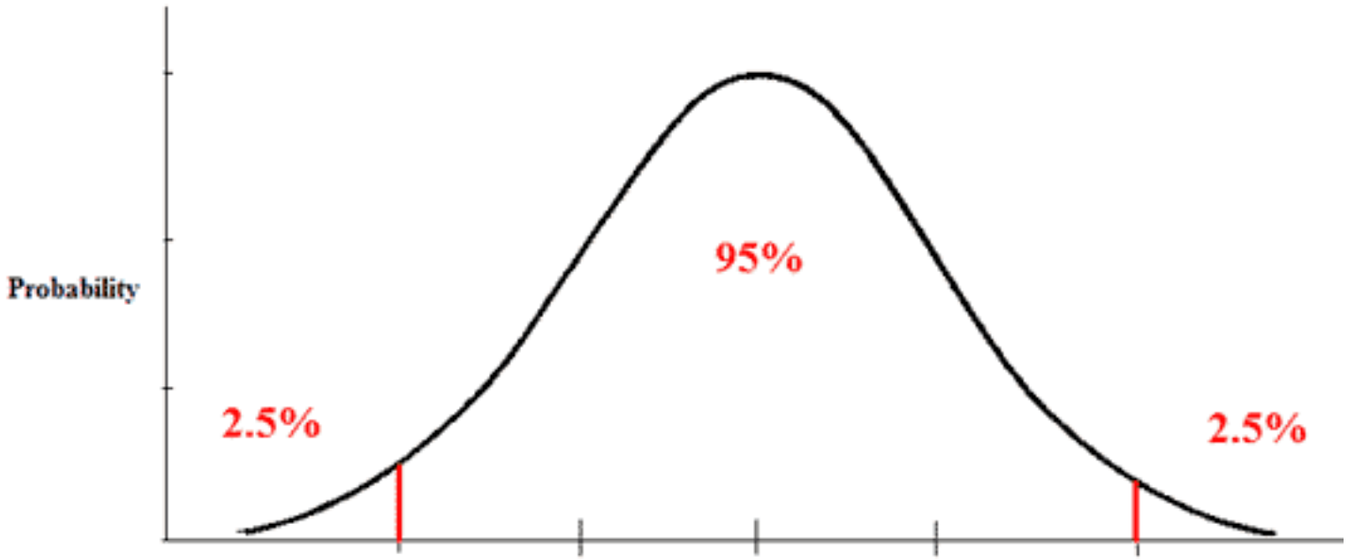
confidence intervals

Significance level, e.g., sig = 0.05 or 5% means M_1 & M_2 are significantly different for 95% of population



Confidence limit, z = sig/2

ESTIMATING CONFIDENCE INTERVALS



Compute t. Select significance level (e.g. sig = 5%)

Consult table for t-distribution: Find t value corresponding to k-1 degrees of freedom (here, k-1=9)

t-distribution is symmetric: typically upper % points of distribution shown → look up value for confidence limit z=sig/2 (here, 0.025, sig=5%)

If t > z or t < -z, then t value lies in rejection region:

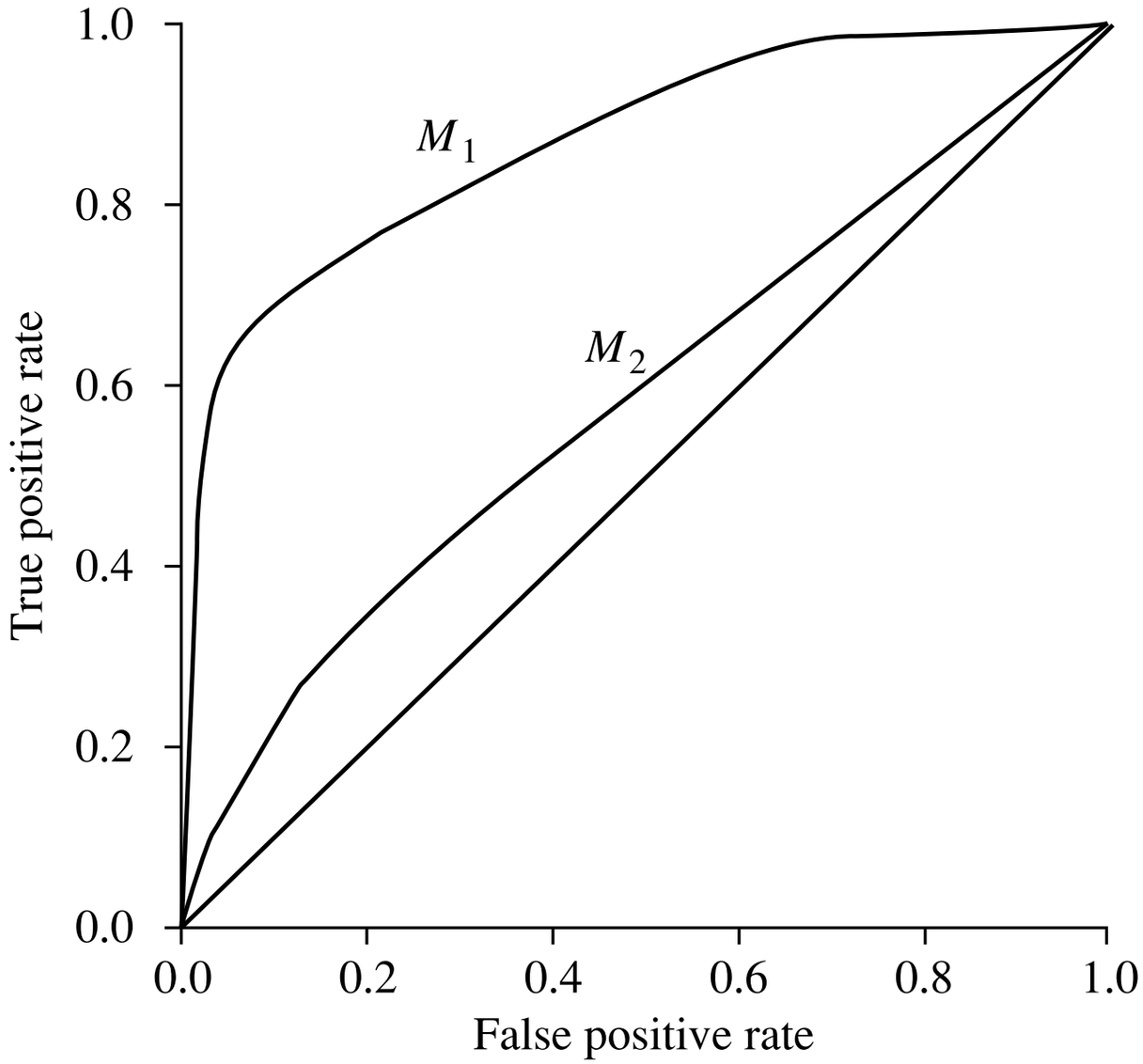
Reject null hypothesis that mean error rates of M_1 & M_2 are same

Conclude: statistically significant difference between M_1 & M_2

Otherwise, conclude that any difference is due to chance

Predicted class

Actual class		*yes*	*no*	Total
	yes	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$



ROC CURVES

ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models

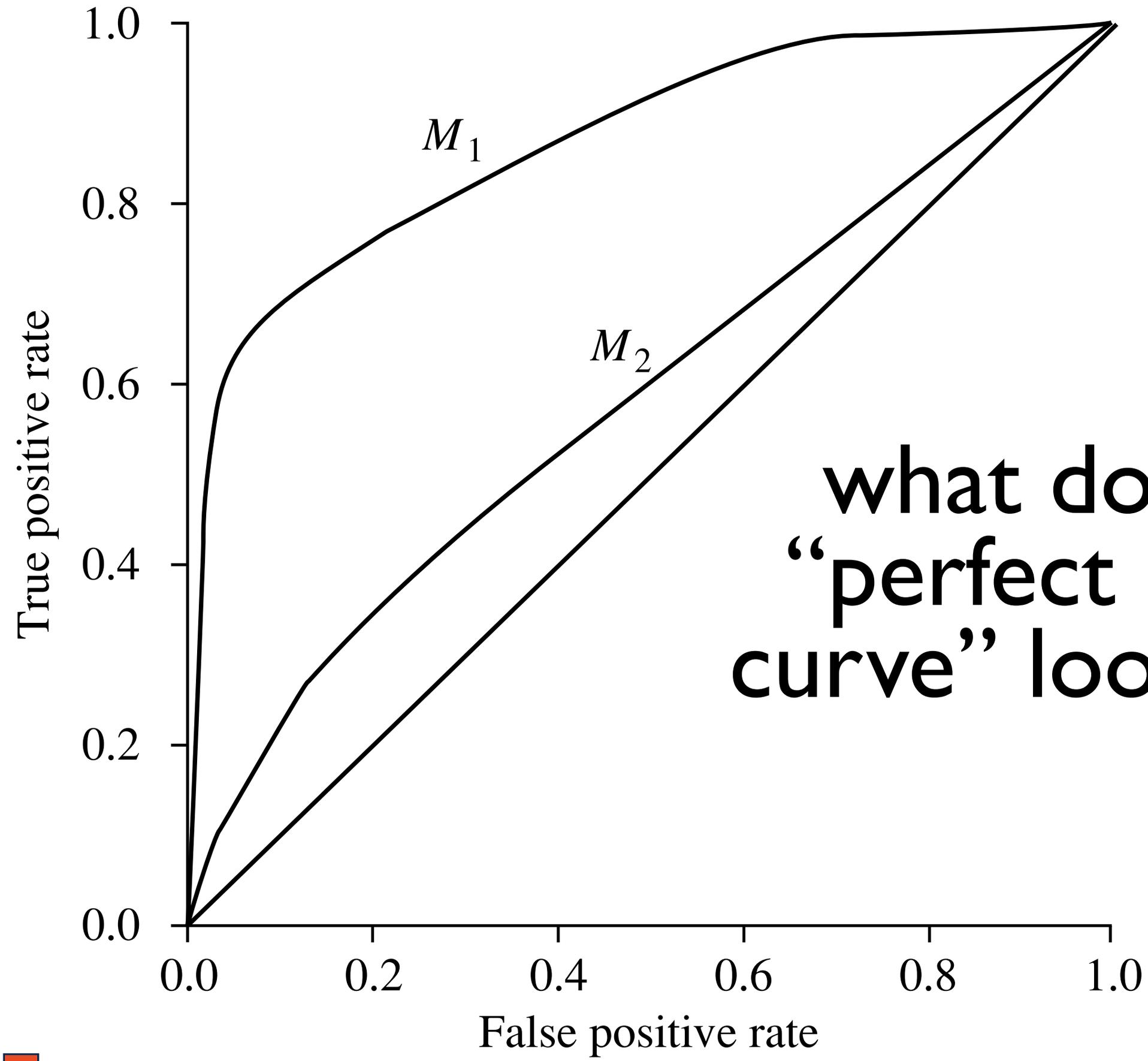
Originated from signal detection theory

Shows the trade-off between the true positive rate and the false positive rate

The area under the ROC curve is a measure of the accuracy of the model

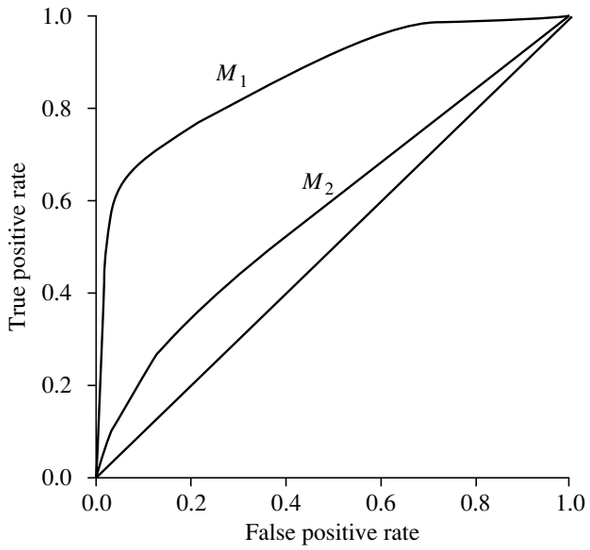
Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list

The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



what does a "perfect ROC curve" look like?

break data into training, test, validation



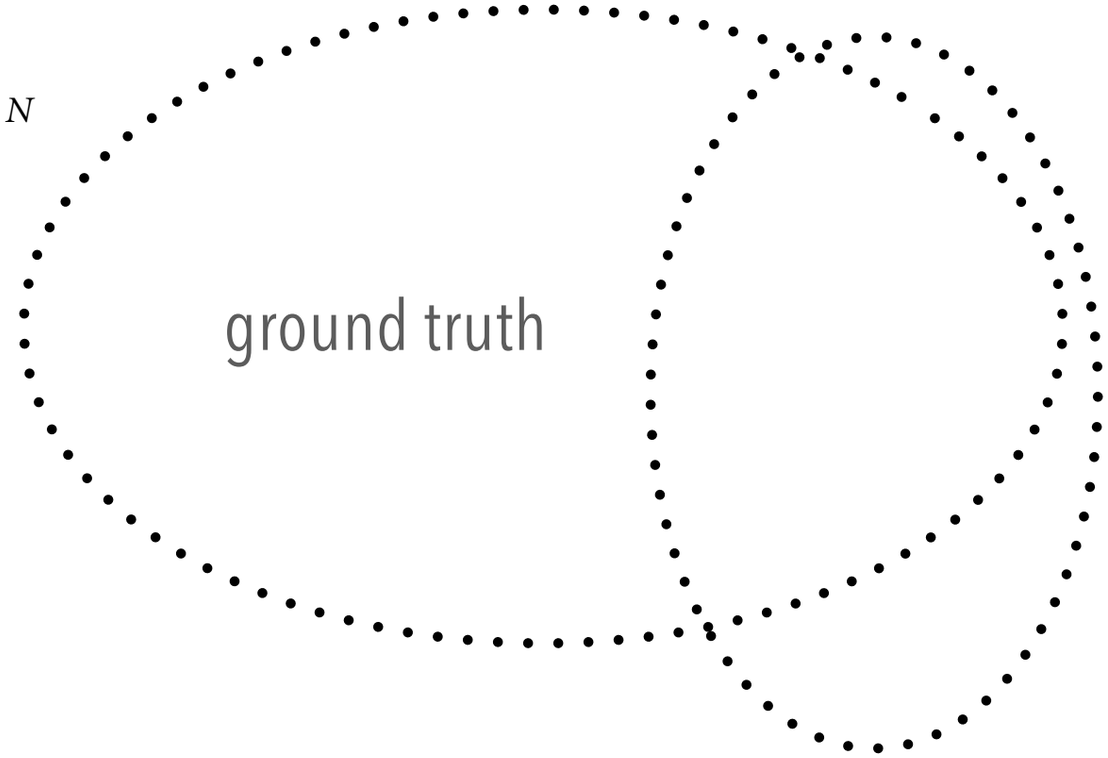
Methods for estimating a classifier's accuracy:

- Holdout method, random subsampling
- Cross-validation
- Bootstrap

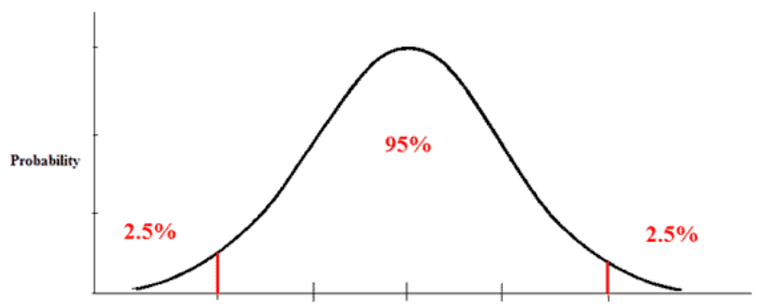
Predicted class

		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

SUMMARY



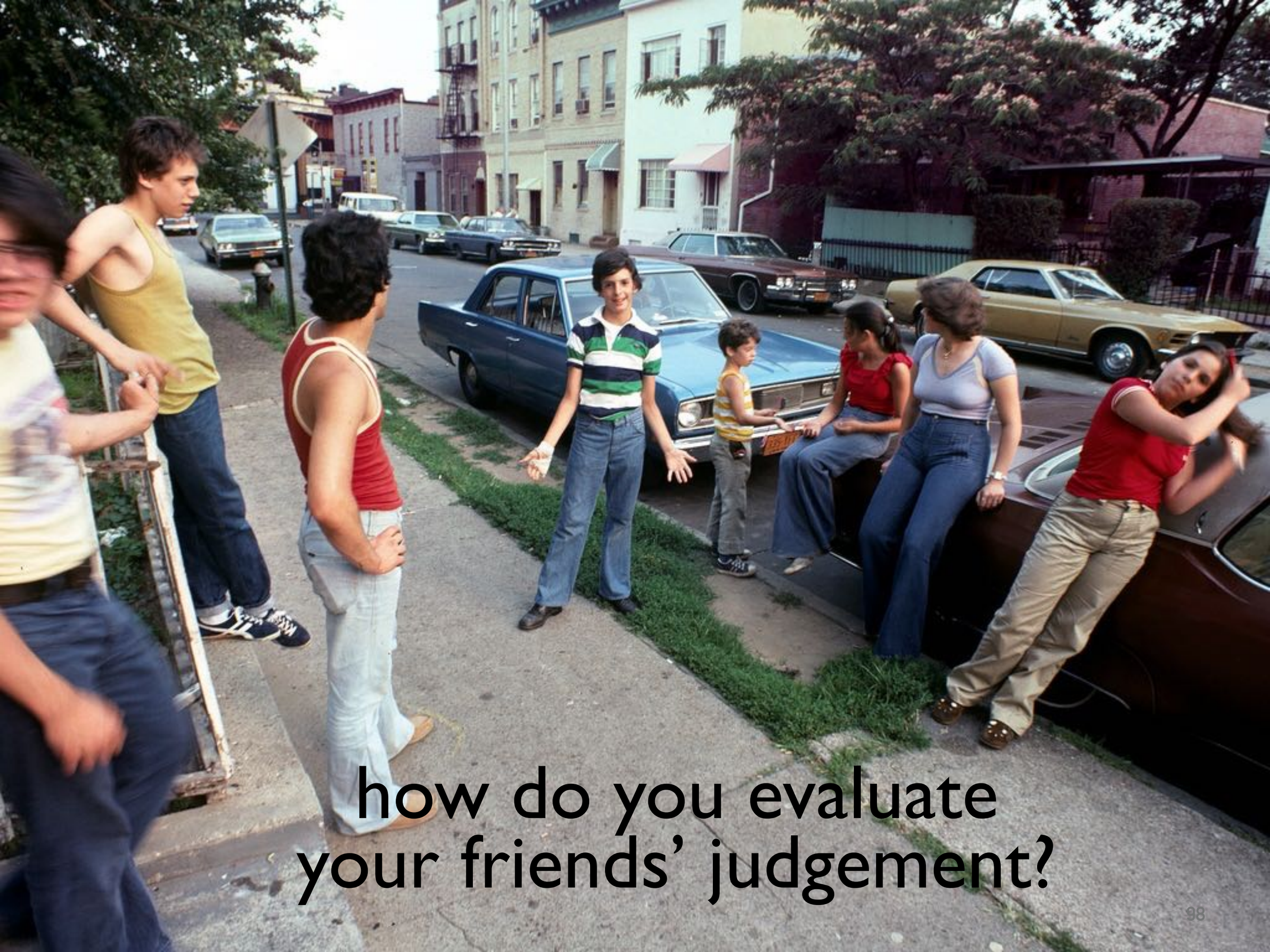
precision, recall F-score



ENSEMBLE METHODS

Basic Concepts Decision Trees Bayes

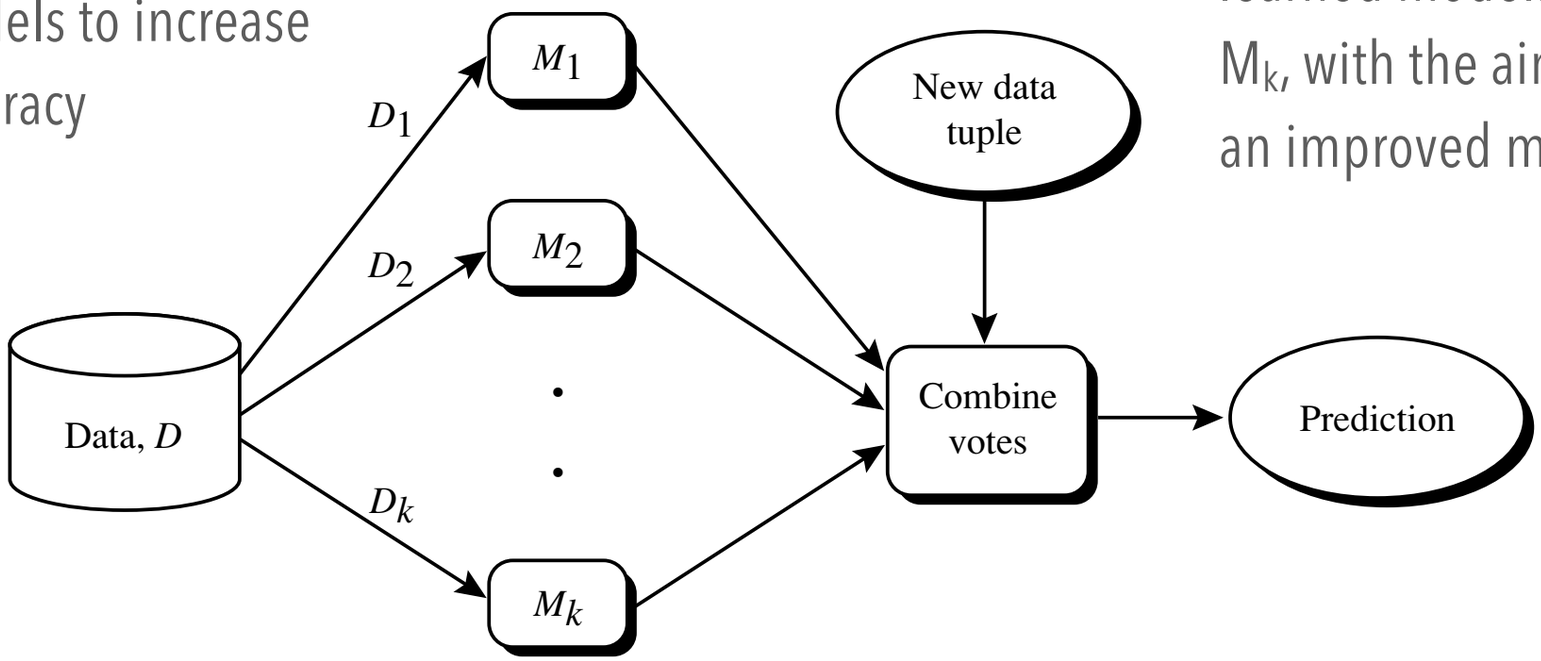
Rule-Based Summary



how do you evaluate your friends' judgement?

Use a combination of models to increase accuracy

Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*



ensemble methods

Bagging: averaging the prediction over a collection of classifiers

Boosting: weighted vote with a collection of classifiers

getting multiple opinions

BAGGING: BOOTSTRAP AGGREGATION

Given a set D of d tuples, at each iteration i, a training set D_i of d tuples is sampled with replacement from D (i.e., bootstrap)

A classifier model M_i is learned for each training set D_i

Classification: classify an unknown sample X

- Each classifier M_i returns its class prediction
- The bagged classifier M_* counts the votes and assigns the class with the most votes to X

BAGGING: BOOTSTRAP AGGREGATION

Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple

Accuracy:

Often significantly better than a single classifier derived from D

For noisy data: not considerably worse, more robust

Why does it work?

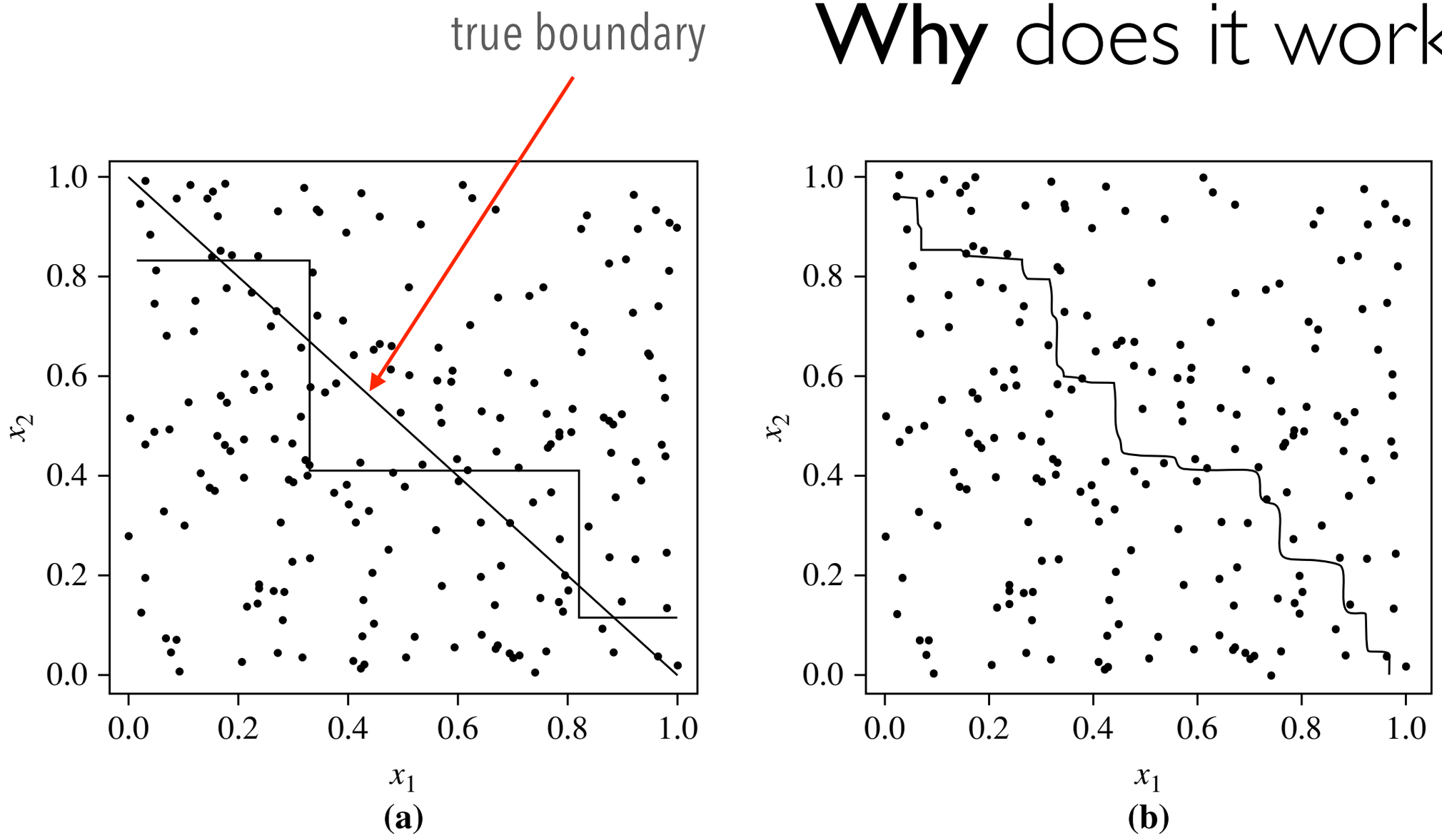
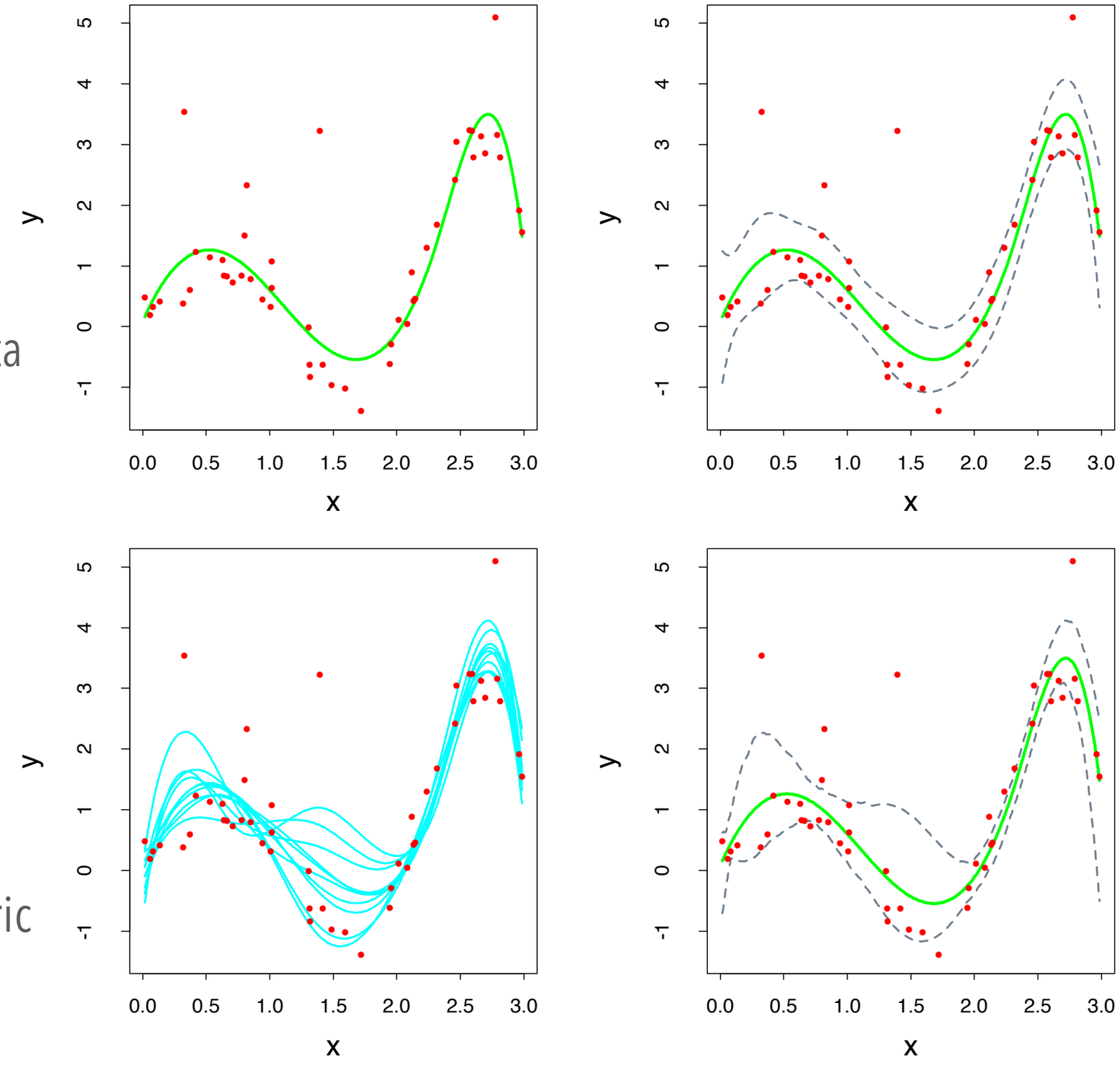


Figure 8.22 Decision boundary by (a) a single decision tree and (b) an ensemble of decision trees for a linearly separable problem (i.e., where the actual decision boundary is a straight line). The decision tree struggles with approximating a linear boundary. The decision boundary of the ensemble is closer to the true boundary. *Source:* From Seni and Elder [SE10]. © 2010 Morgan & Claypool Publishers; used with permission.

complete data

non parametric

samples drawn
from the data;
hence "model free"



parametric

samples
synthesized using
gaussian noise

FIGURE 8.2. *(Top left:) B-spline smooth of data. (Top right:) B-spline smooth plus and minus* $1.96\times$ *standard error bands. (Bottom left:) Ten bootstrap replicates of the B-spline smooth. (Bottom right:) B-spline smooth with* 95% *standard error bands computed from the bootstrap distribution.*

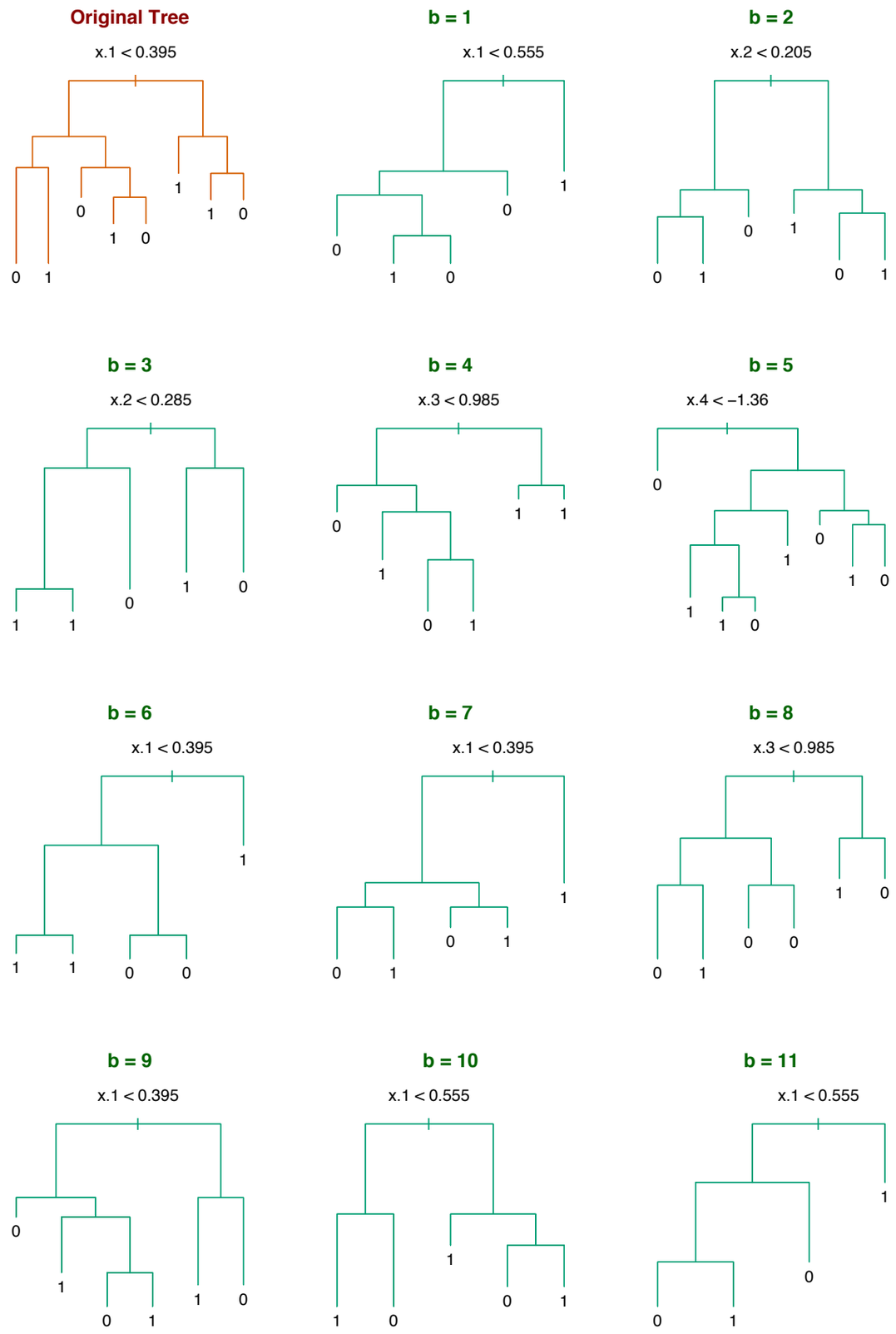


FIGURE 8.9. *Bagging trees on simulated dataset. The top left panel shows the original tree. Eleven trees grown on bootstrap samples are shown. For each tree, the top split is annotated.*

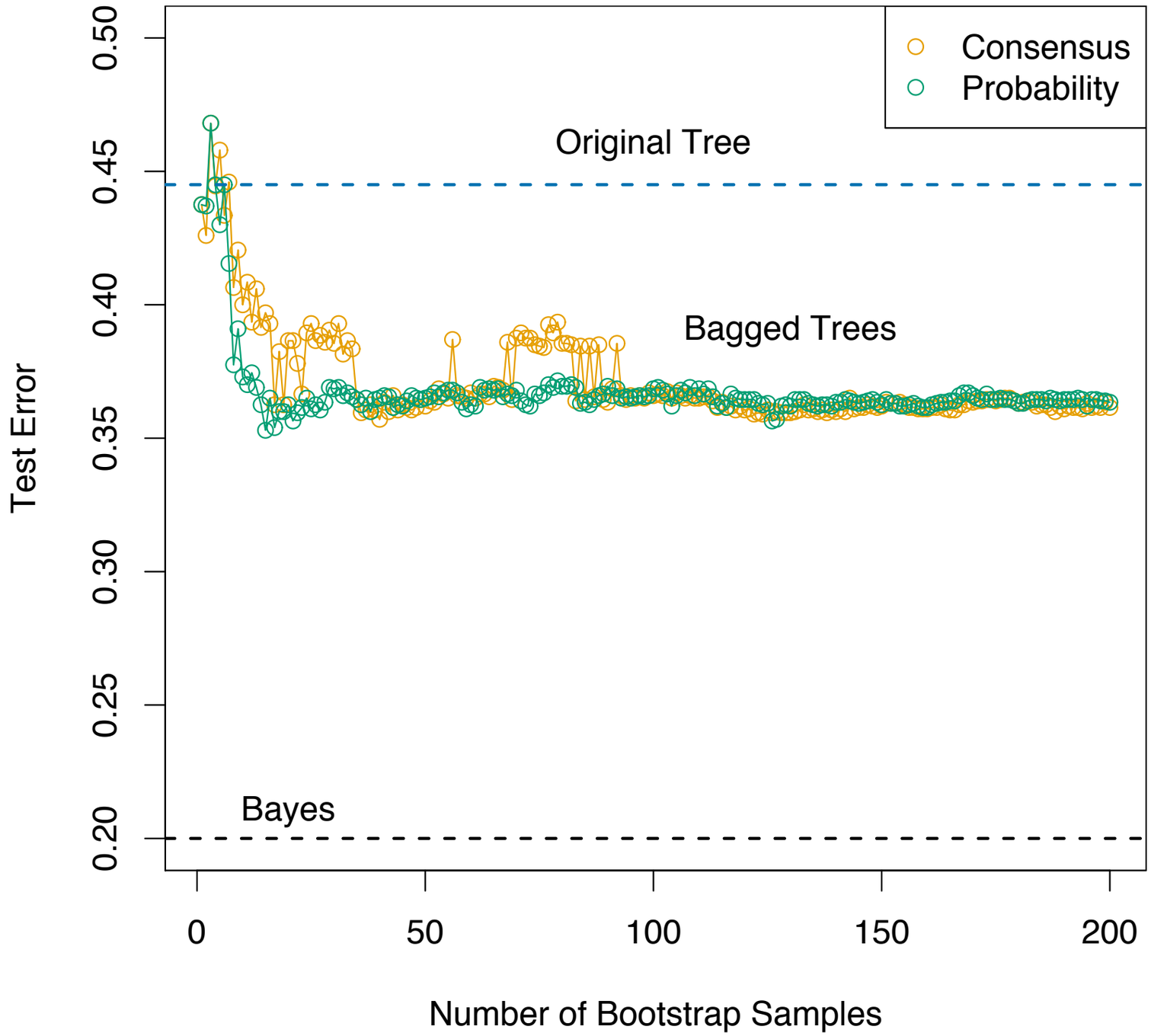
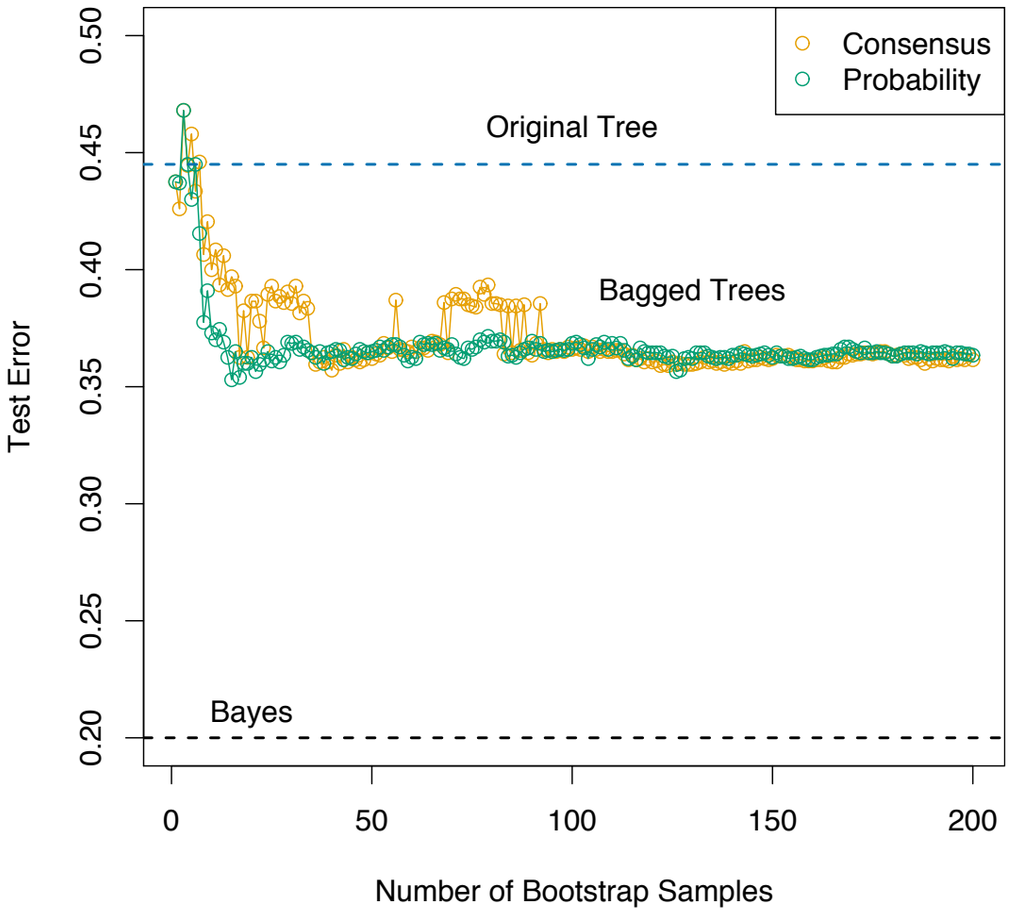


FIGURE 8.10. *Error curves for the bagging example of Figure 8.9. Shown is the test error of the original tree and bagged trees as a function of the number of bootstrap samples. The orange points correspond to the consensus vote, while the green points average the probabilities.*



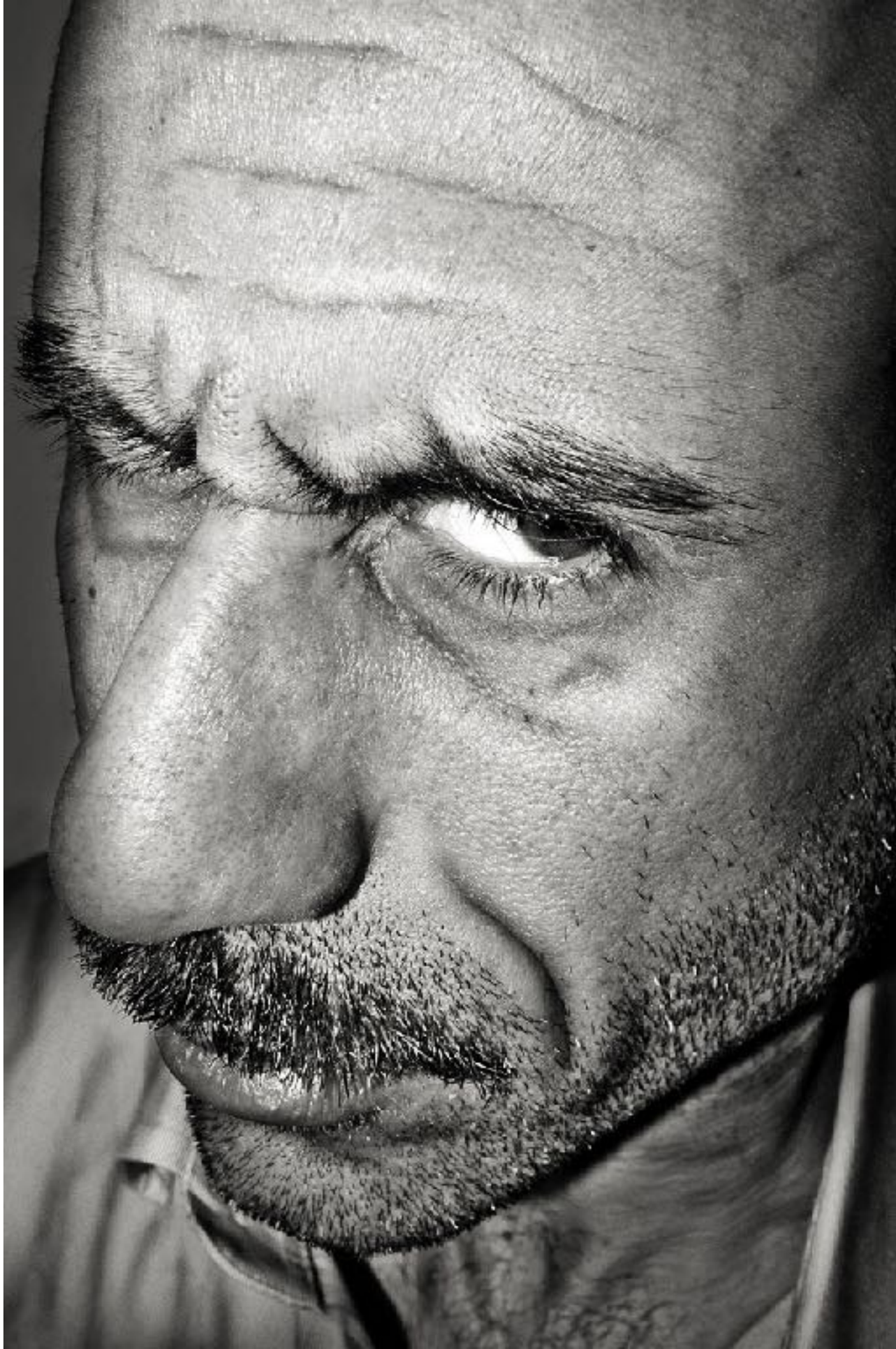
Often we require the class-probability estimates at x, rather than the classifications themselves. It is tempting to treat the voting proportions $p_k(x)$ as estimates of these probabilities. But these estimates are incorrect.

Instead, for trees we can use the class proportions in the terminal node. Other classifiers produce probability estimates.

An alternative bagging strategy is to average these probabilities instead, rather than the vote indicator vectors. Not only does this produce improved estimates of the class probabilities, but it also tends to produce bagged classifiers with lower variance, especially for small B

but not all doctors are good are they?

weight assigned based on the previous diagnosis accuracy



BOOSTING

Weights are assigned to each training tuple

A series of k classifiers is iteratively learned

After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i

The final M^* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy

Boosting algorithm can be extended for numeric prediction

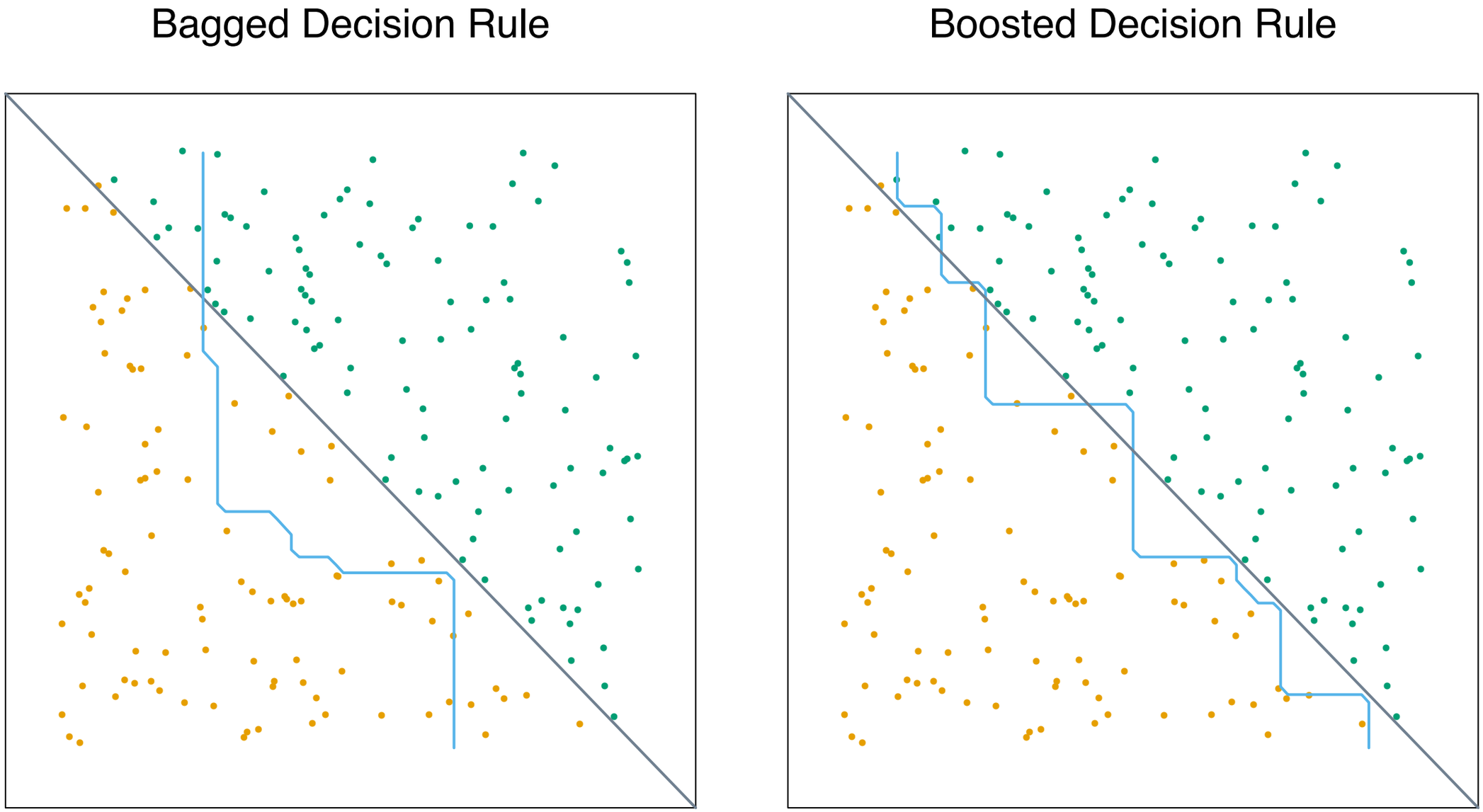
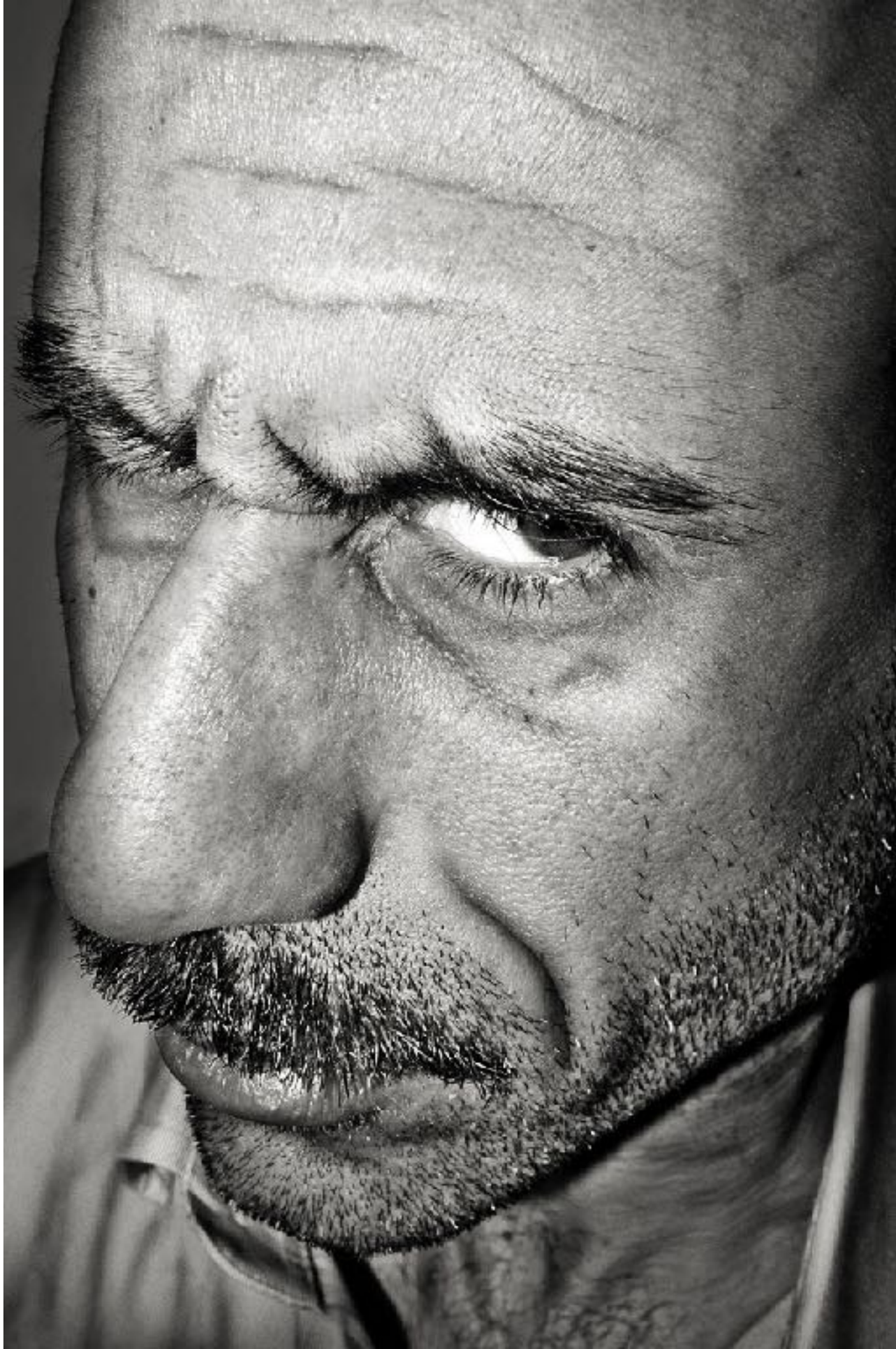


FIGURE 8.12. *Data with two features and two classes, separated by a linear boundary. (Left panel:) Decision boundary estimated from bagging the decision rule from a single split, axis-oriented classifier. (Right panel:) Decision boundary from boosting the decision rule of the same classifier. The test error rates are* 0.166, *and* 0.065, *respectively.*

this sounds too good to be true!

Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data



AdaBoost

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119 – 139, 1997.

ADABOOST

Given a set of d class-labeled tuples, $(X_1, y_1), \ldots, (X_d, y_d)$

Initially, all the weights of tuples are set the same (1/d)

Generate k classifiers in k rounds. At round i,

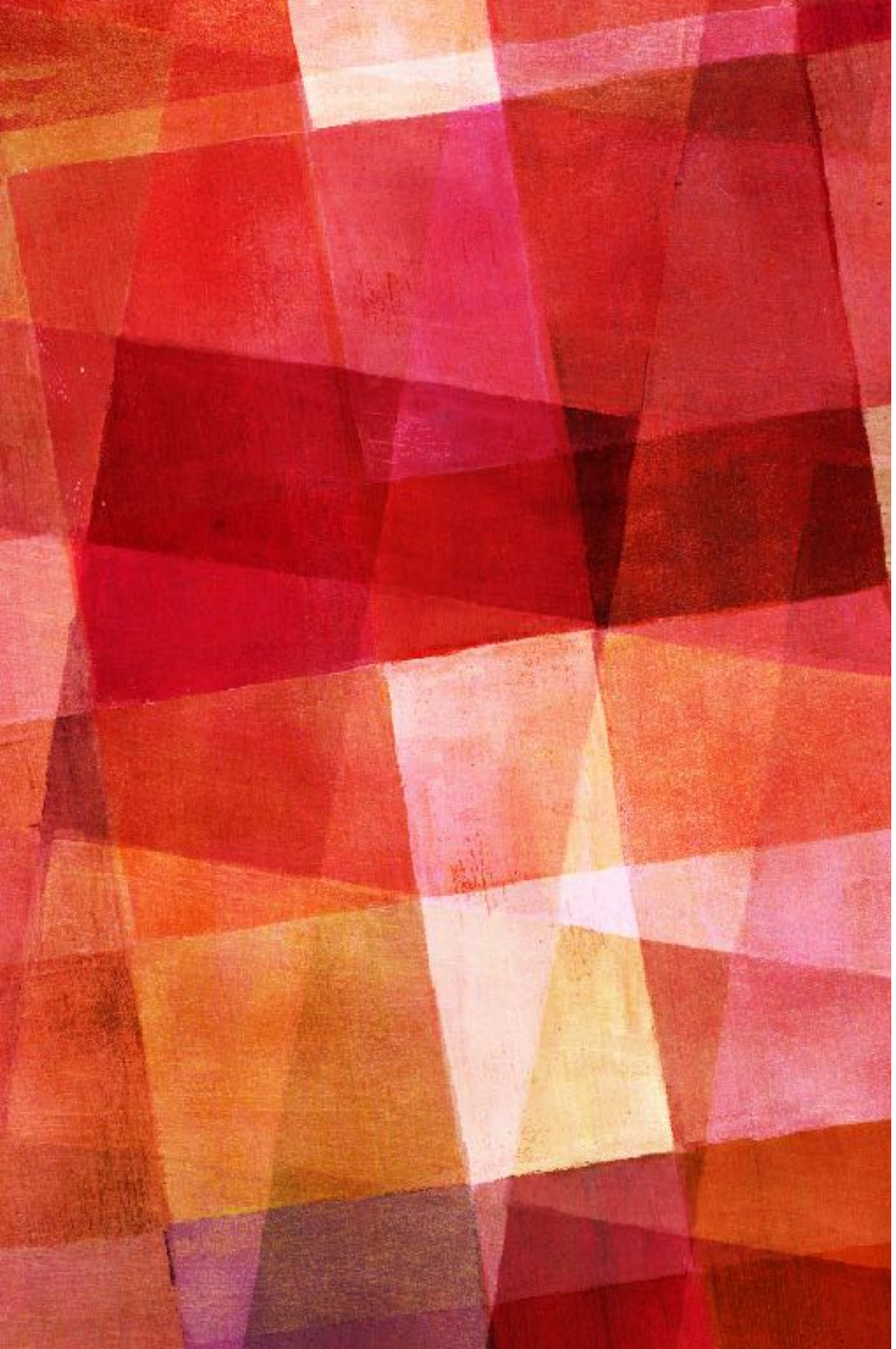
Tuples from D are sampled (with replacement) to form a training set D_i of the same size

Each tuple's chance of being selected is based on its weight

A classification model M_i is derived from D_i

Its error rate is calculated using D_i as a test set

If a tuple is misclassified, its weight is increased, otherwise it is decreased



ADABOOST

Error rate: **err**(X_j) is the misclassification error of tuple X_j. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

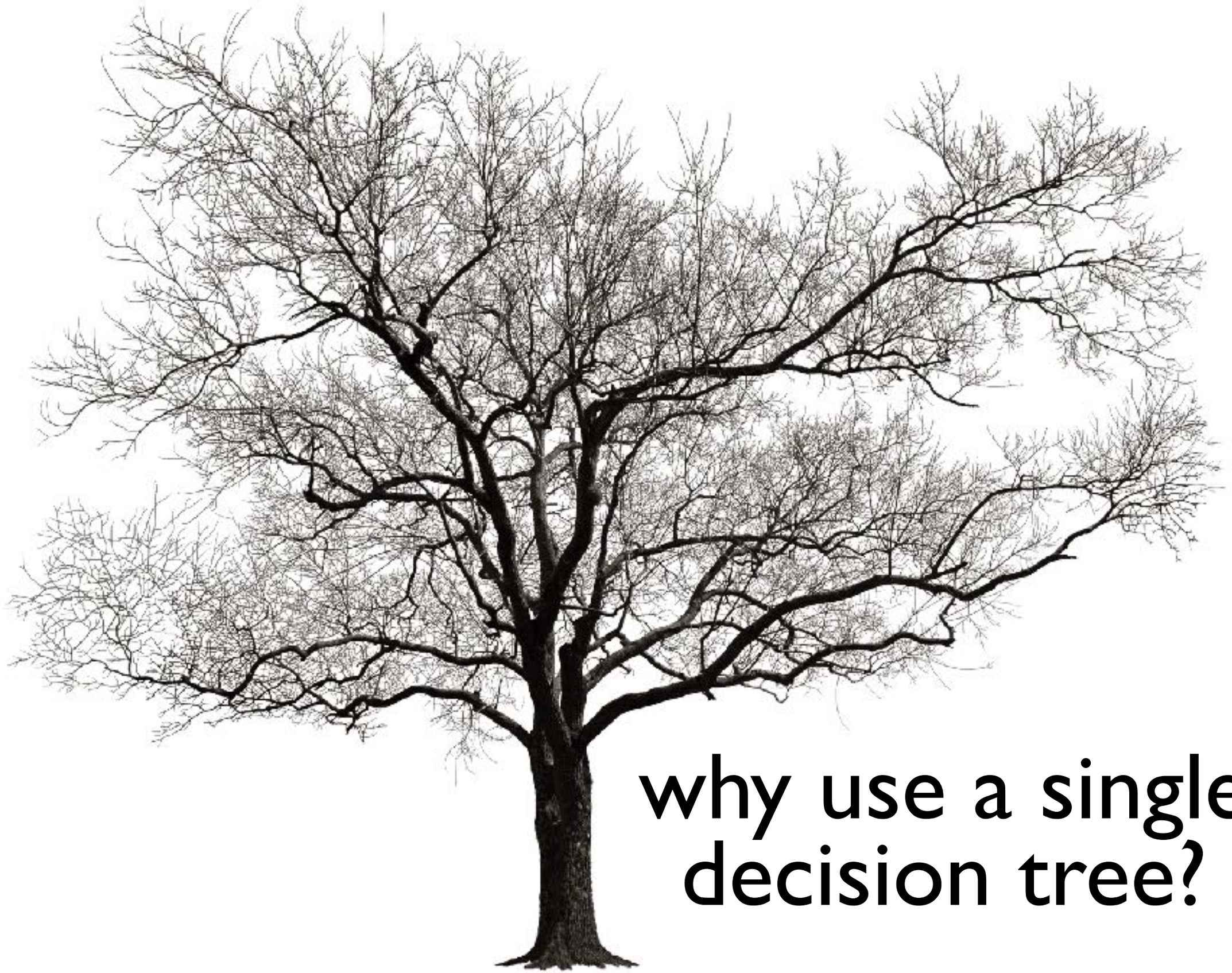
$$error(M_i) = \sum_{j=1}^{d} w_j \times err(X_j)$$

All correctly classified tuples' weights are updated by multiplying them with $\frac{error(M_i)}{1 - error(M_i)}$

All tuples' weights are renormalized

The weight of classifier M_i's vote for a tuple is

$$\log \frac{1 - error(M_i)}{error(M_i)}$$



why use a single decision tree?

Leo Breiman. Random forests. Machine Learning, 45(1):5–32, October 2001.

RANDOM FORESTS

Each classifier in the ensemble is a decision tree classifier and is generated using a random selection of attributes at each node to determine the split

During classification, each tree votes and the most popular class is returned

CONSTRUCTING RANDOM FORESTS

Forest-RI (random input selection): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size

Forest-RC (random linear combinations): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)

Comparable in accuracy to Adaboost, but more robust to errors and outliers

Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

classifiers assume balanced classes

not true in reality!

multi class imbalance is hard

OVERCOMING CLASS IMBALANCE

Oversampling: re-sampling of data from positive class

Under-sampling: randomly eliminate tuples from negative class

Threshold-moving: move the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors

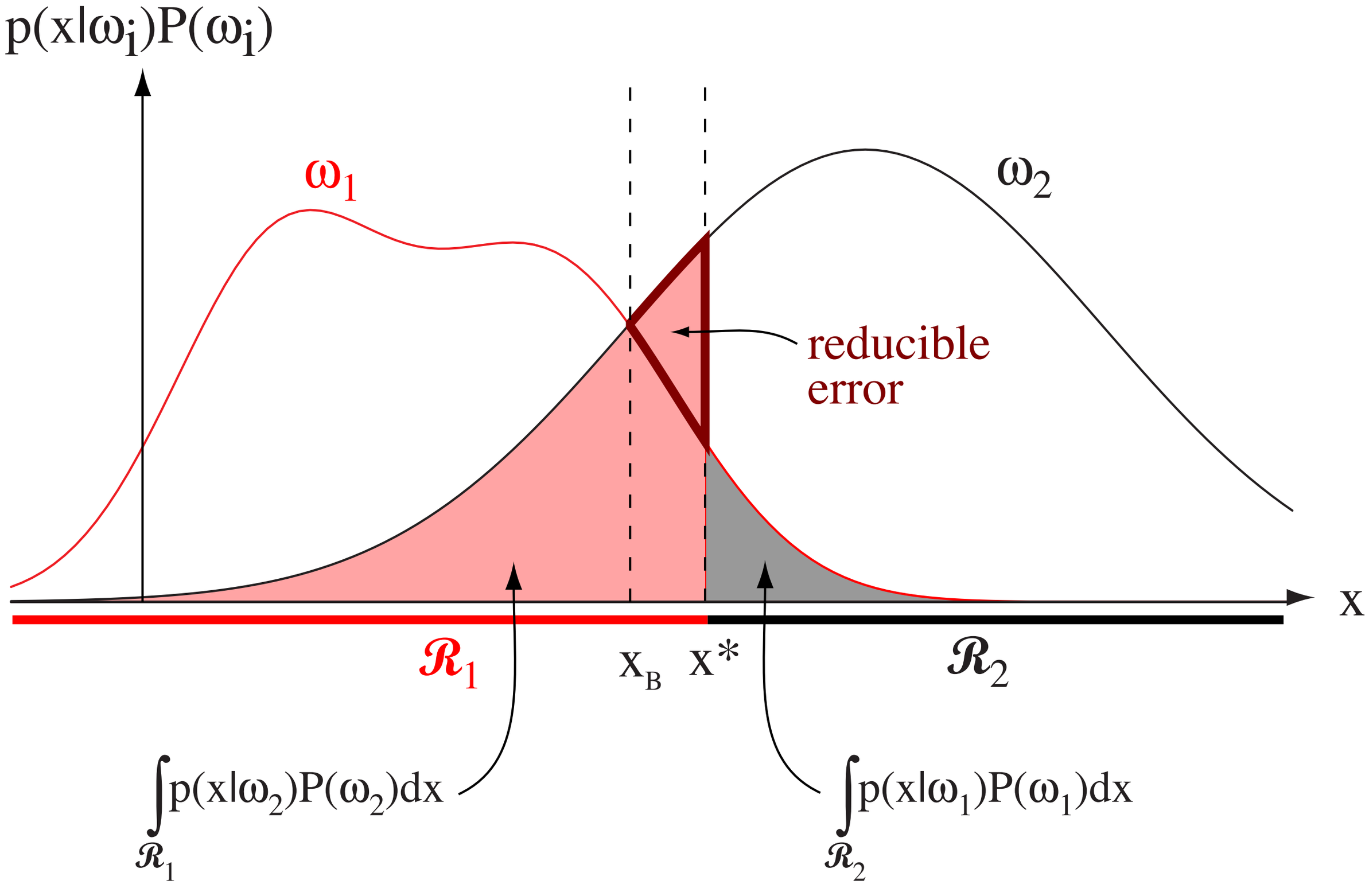
Ensemble techniques: Ensemble multiple classifiers introduced earlier

Exercise!

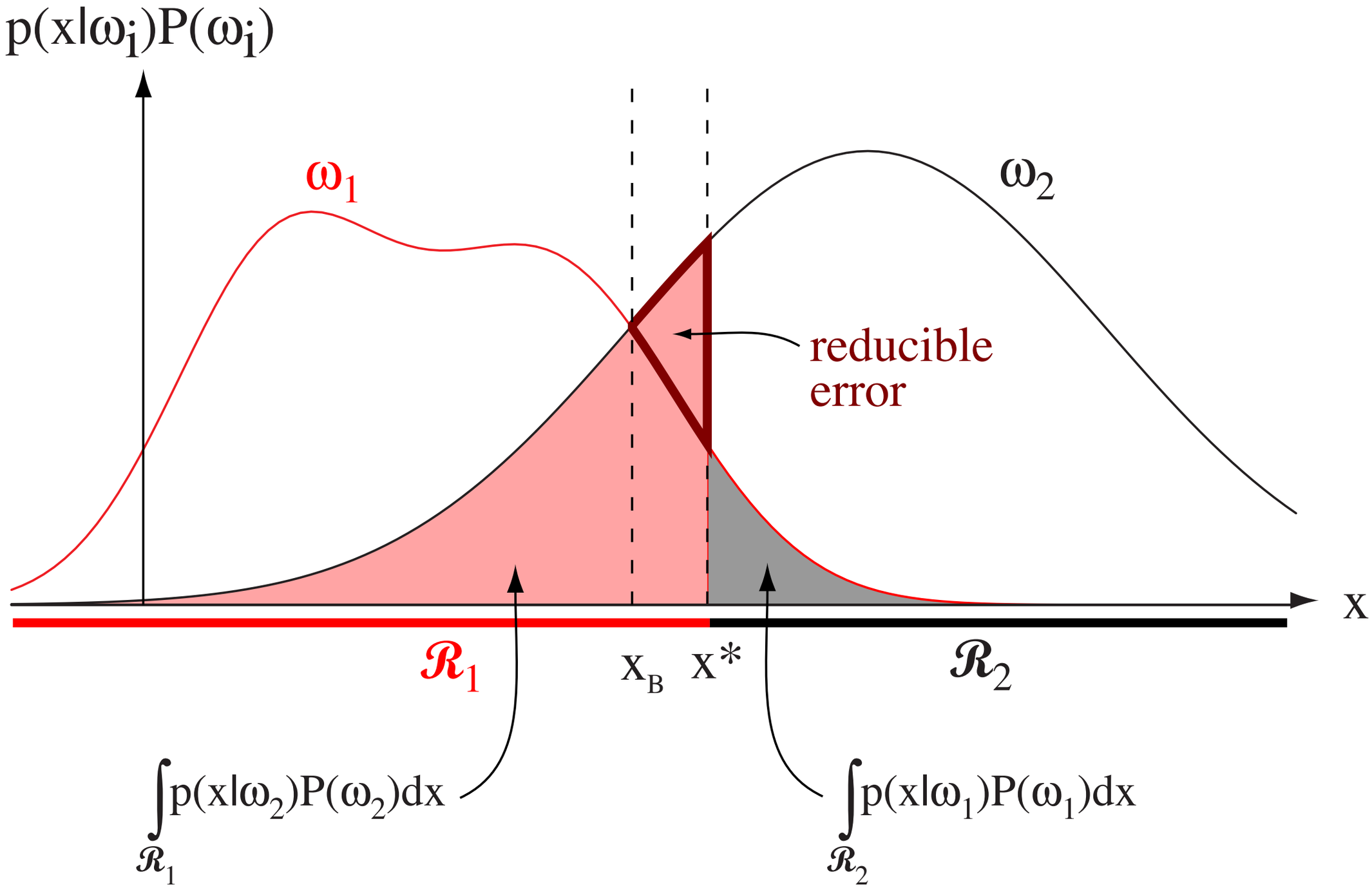
Why is a Binary classifier with a **10%** classification accuracy considered good?

Exercise!

Can classification accuracy ever be **100%**?


p(x|ωi)P(ωi)
ω1
ω2
reducible
error
x
R1
R2
xB
x*
∫R1 p(x|ω2)P(ω2)dx
∫R2 p(x|ω1)P(ω1)dx

How do you **beat** this problem?



Methods for estimating a classifier's accuracy:

Holdout method, random subsampling
Cross-validation
Bootstrap

Predicted class

Actual class		*yes*	*no*	Total
	yes	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

no single classifier is best in all cases

Classification is a form of data analysis that extracts models describing important data classes.

Decision tree induction, Naive Bayesian classification, rule-based classification

ensemble techniques: bagging, boosting, random forests

SUMMARY

accuracy, sensitivity, specificity, precision, recall, F measure, and F_β measure.