Classification - Basic Concepts

Lecture Notes for Chapter 6

Dr. Greg Chism

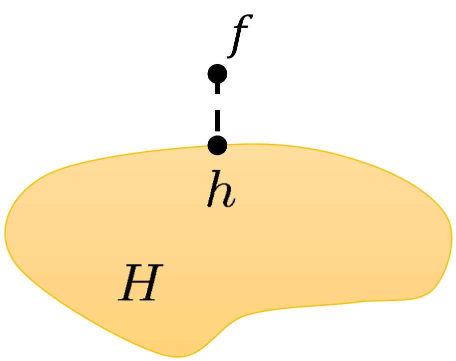
Topics

- **Introduction**
- Decision Trees
  - —Overview
  - —Tree Induction
  - —Overfitting and other Practical Issues
- Model Evaluation
  - —Metrics for Performance Evaluation
  - —Methods to Obtain Reliable Estimates
  - —Model Comparison (Relative Performance)
- Feature Selection
- Class Imbalance

Supervised Learning

- Examples
  - Input-output pairs: $\mathrm{E} = (x_1, y_1), \dots, (x_i, y_i), \dots, (x_N, y_N)$.
  - We assume that the examples are produced iid (with noise and errors) from a target function $y = f(x)$.

- Learning problem
  - Given a hypothesis space H
  - Find a hypothesis $h \in H$ such that $\hat{y}_i = h(x_i) \approx y_i$
  - That is, we want to approximate f by h using E.



- Includes
  - **Regression** (outputs = real numbers). Goal: Predict the number accurately.
    E.g., x is a house and $f(x)$ is its selling price.
  - **Classification** (outputs = class labels). Goal: Assign new records to a class.
    E.g., x is an email and $f(x)$ is spam / ham

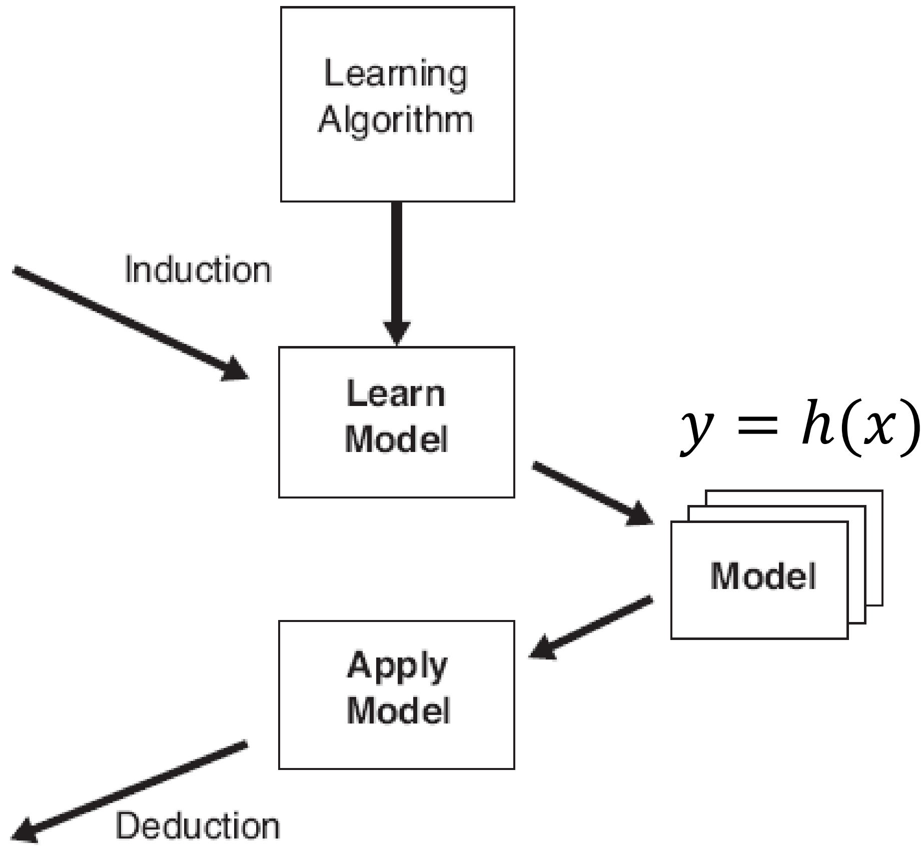
You already know linear regression. We focus on Classification.

Illustrating Classification Task

E x y

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes



Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

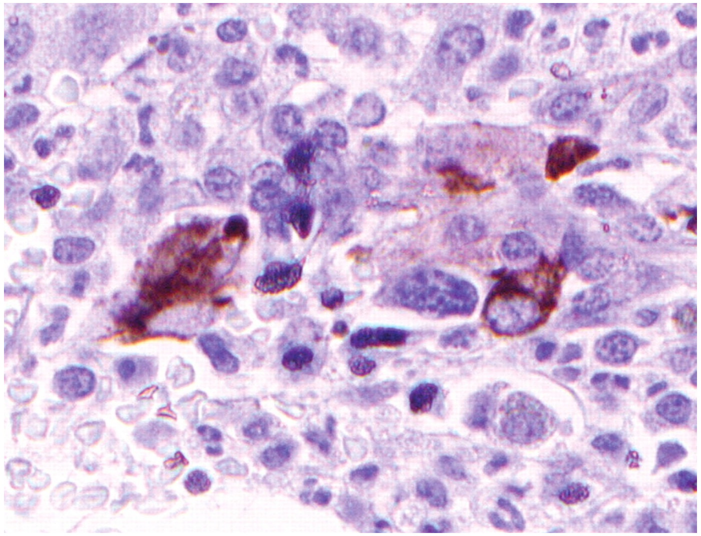
Examples of Classification Task

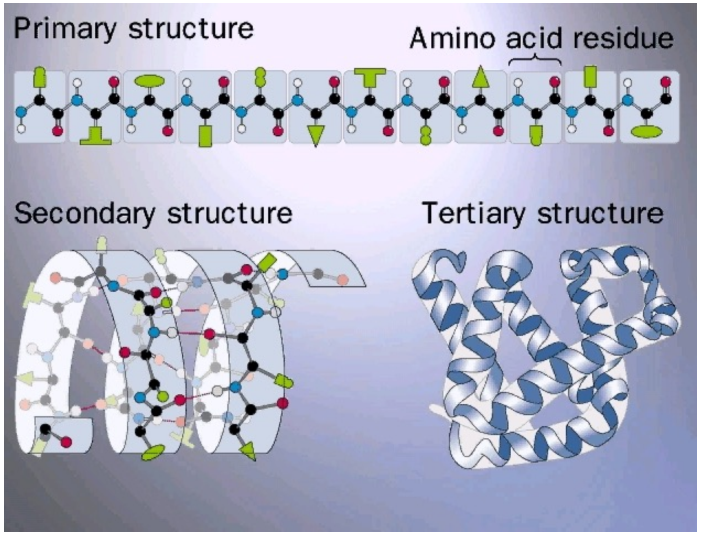
- Predicting tumor cells as benign or malignant

- Classifying credit card transactions as legitimate or fraudulent

- Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil

- Categorizing news stories as finance, weather, entertainment, sports, etc





Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks / Deep Learning
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

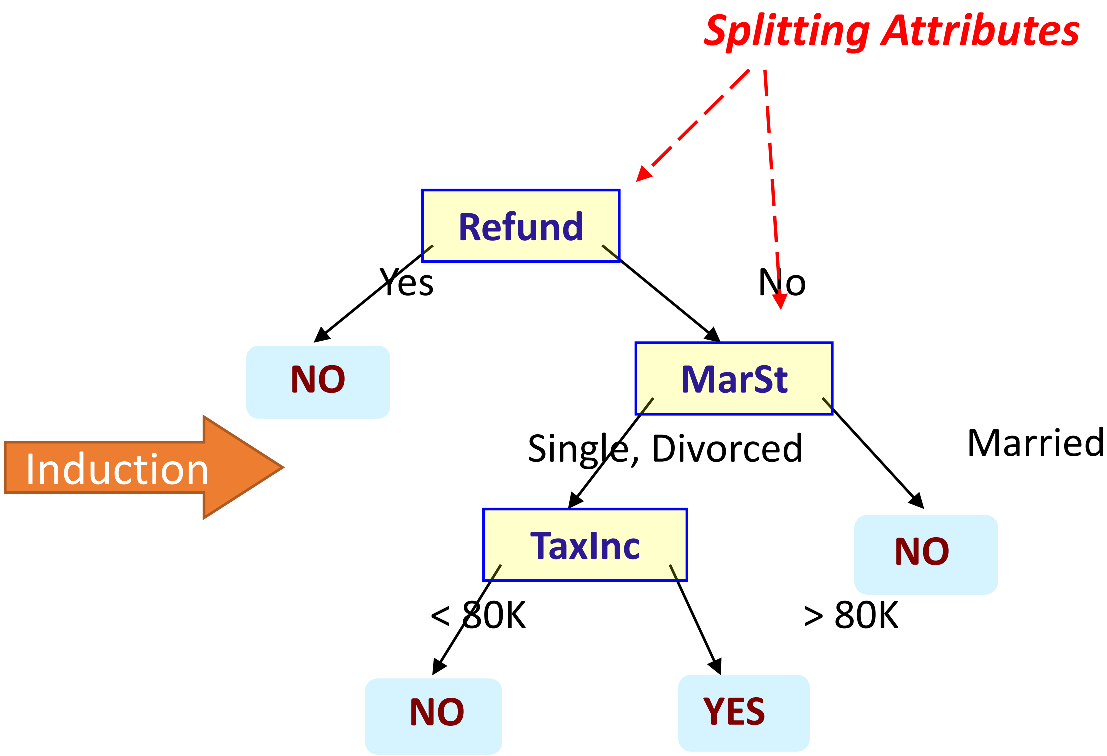
Topics

- Introduction
- **Decision Trees**
  - —**Overview**
  - —Tree Induction
  - —Overfitting and other Practical Issues
- Model Evaluation
  - —Metrics for Performance Evaluation
  - —Methods to Obtain Reliable Estimates
  - —Model Comparison (Relative Performance)
- Feature Selection
- Class Imbalance

Example of a Decision Tree

categorical categorical continuous class

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

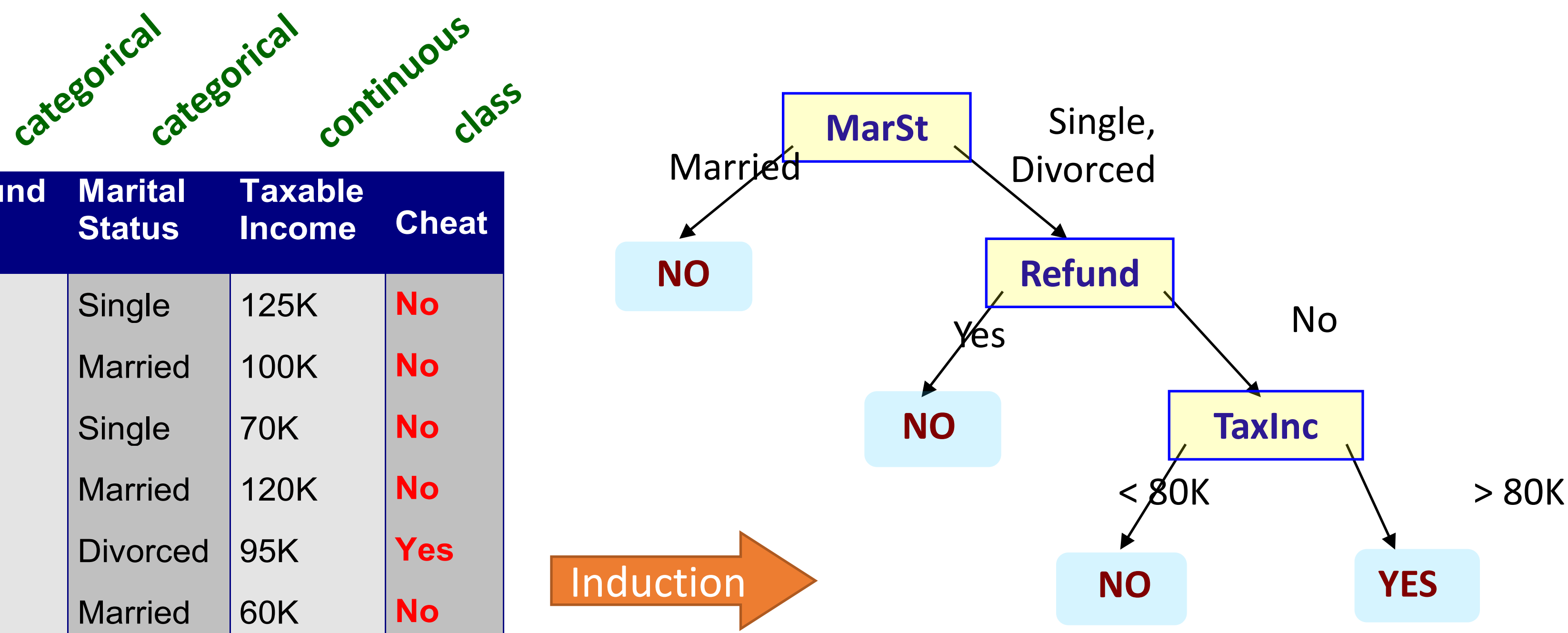


Training Data

Model: Decision Tree

Another Example of Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

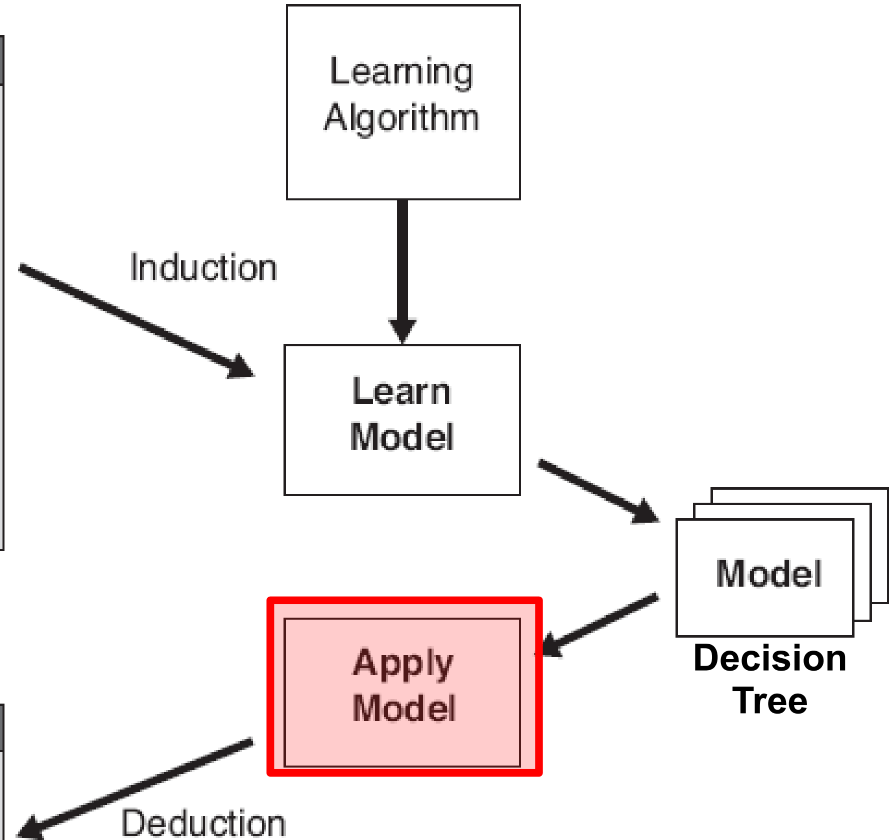


There could be more than one tree that fits the same data!

Decision Tree: Deduction

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

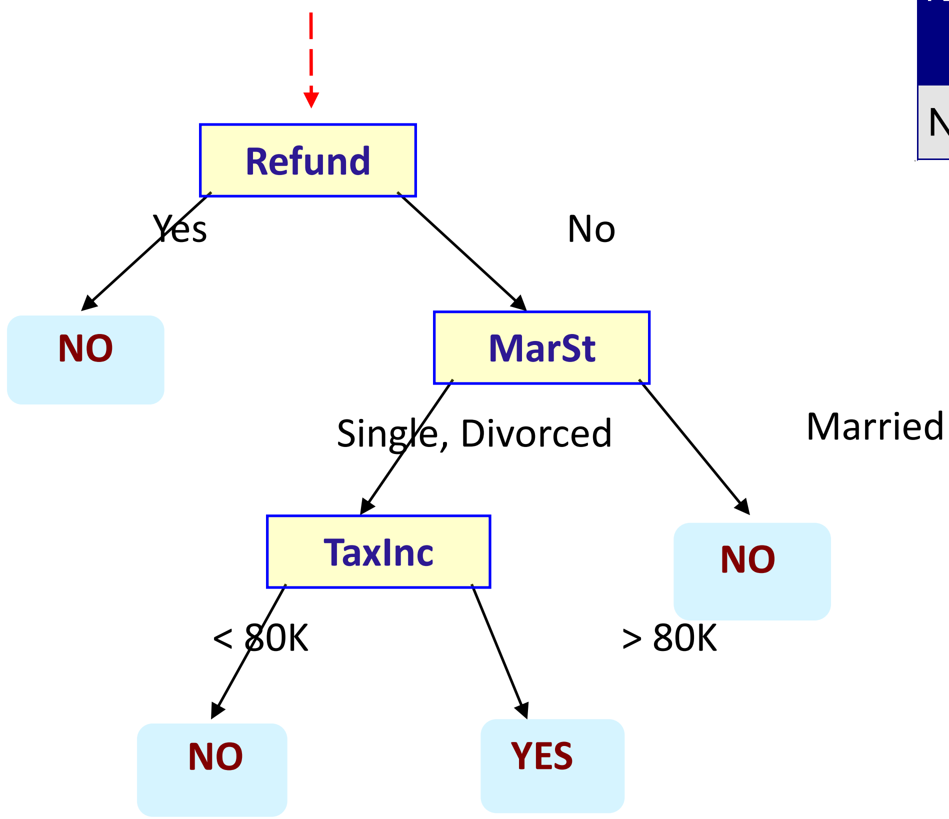


Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Apply Model to Test Data

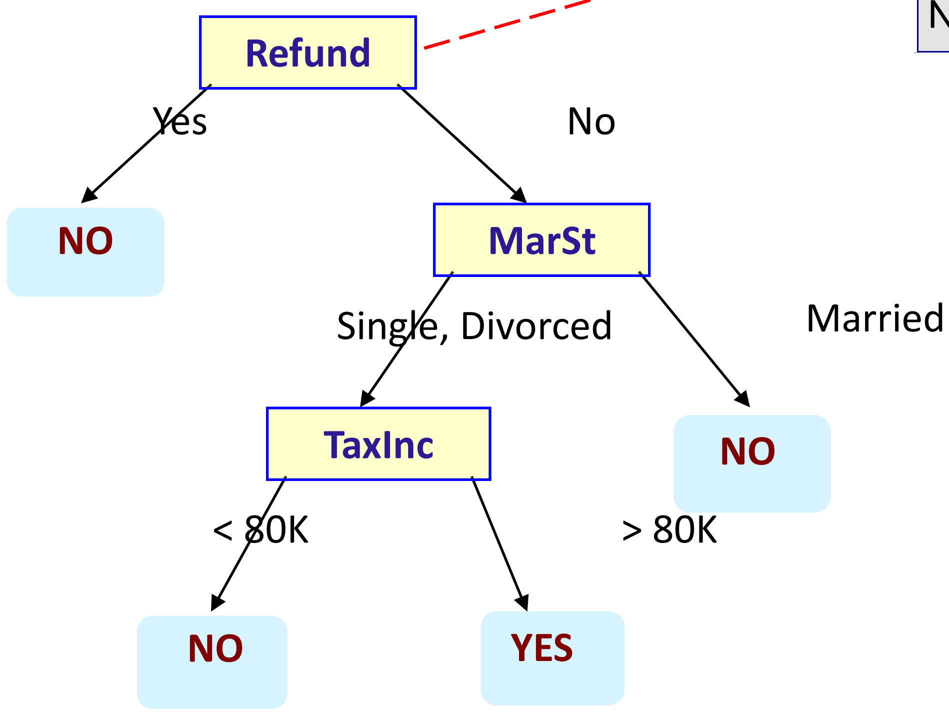
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



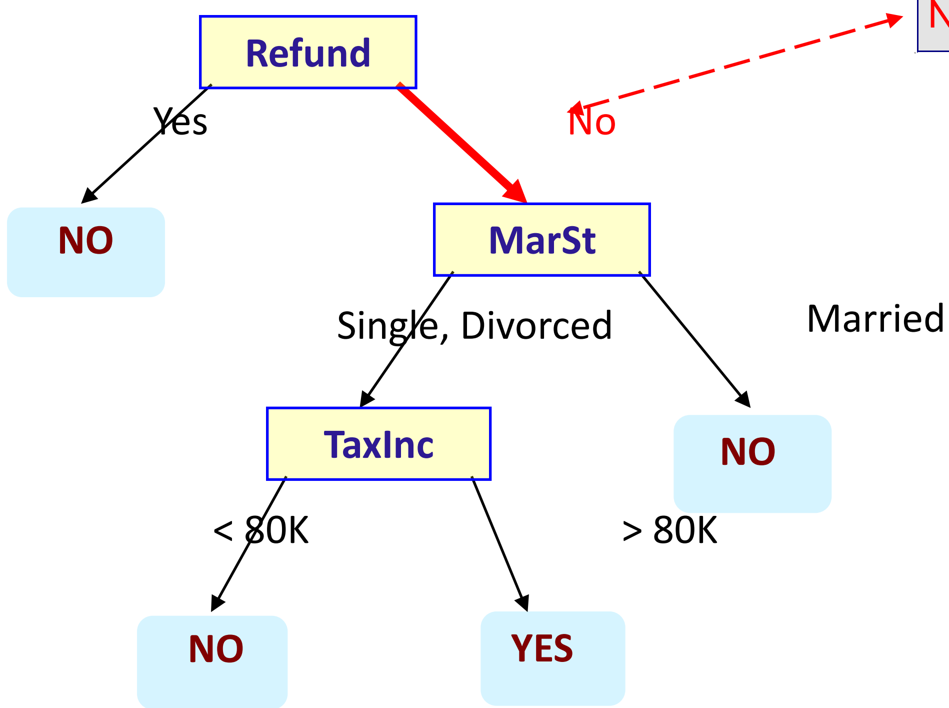
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

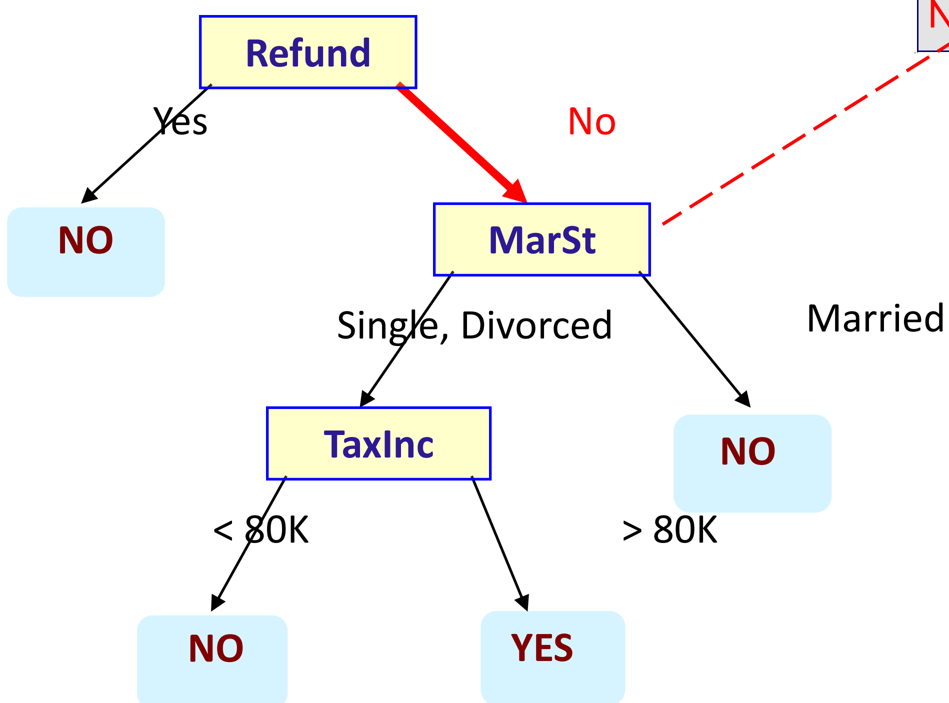
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

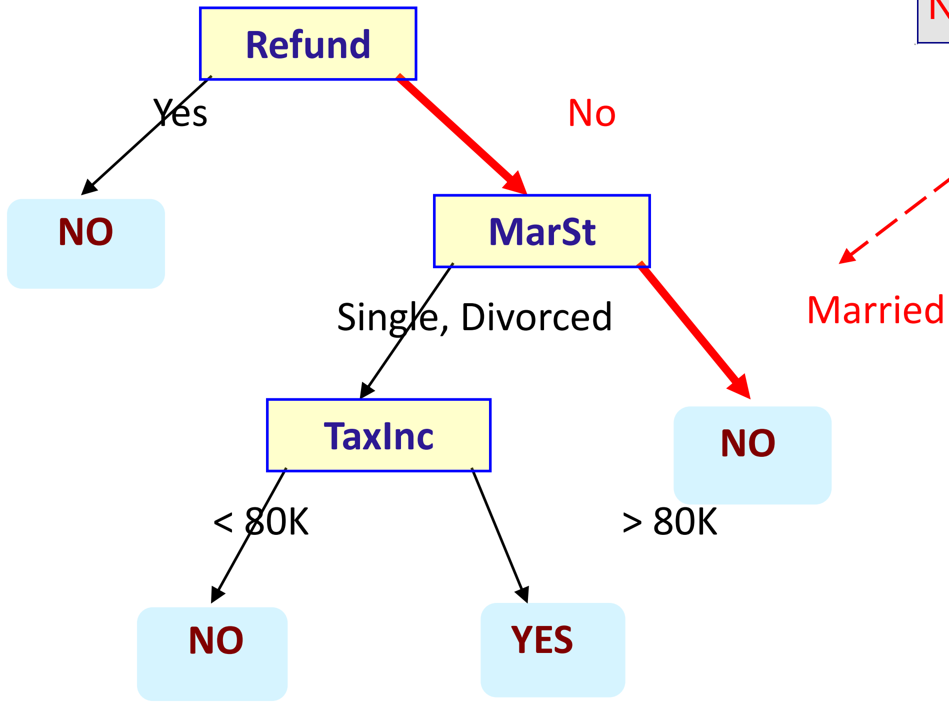
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

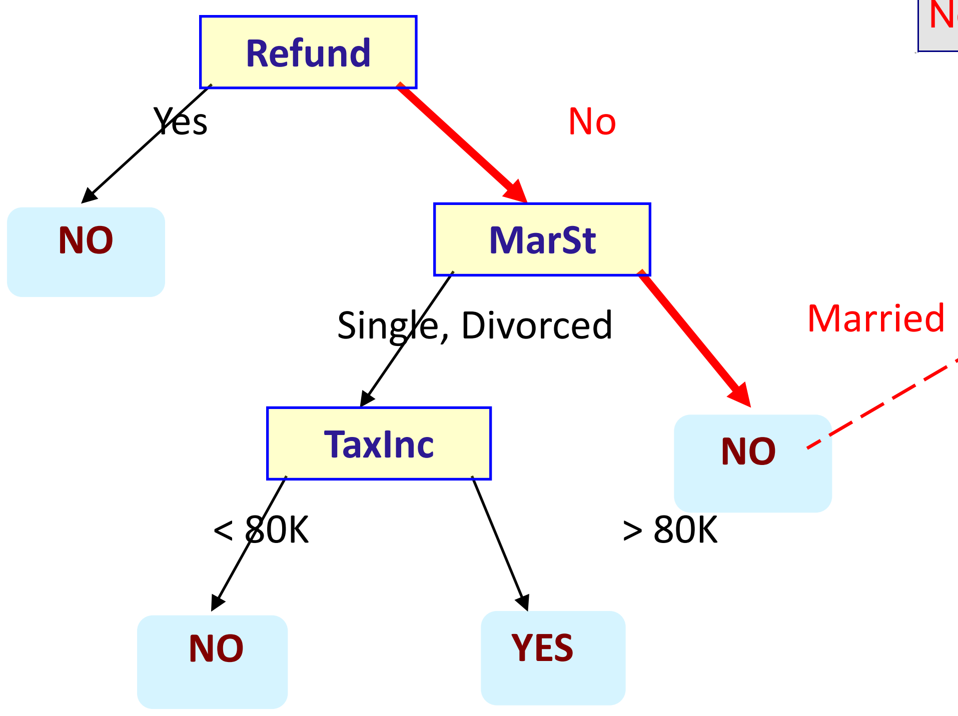
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

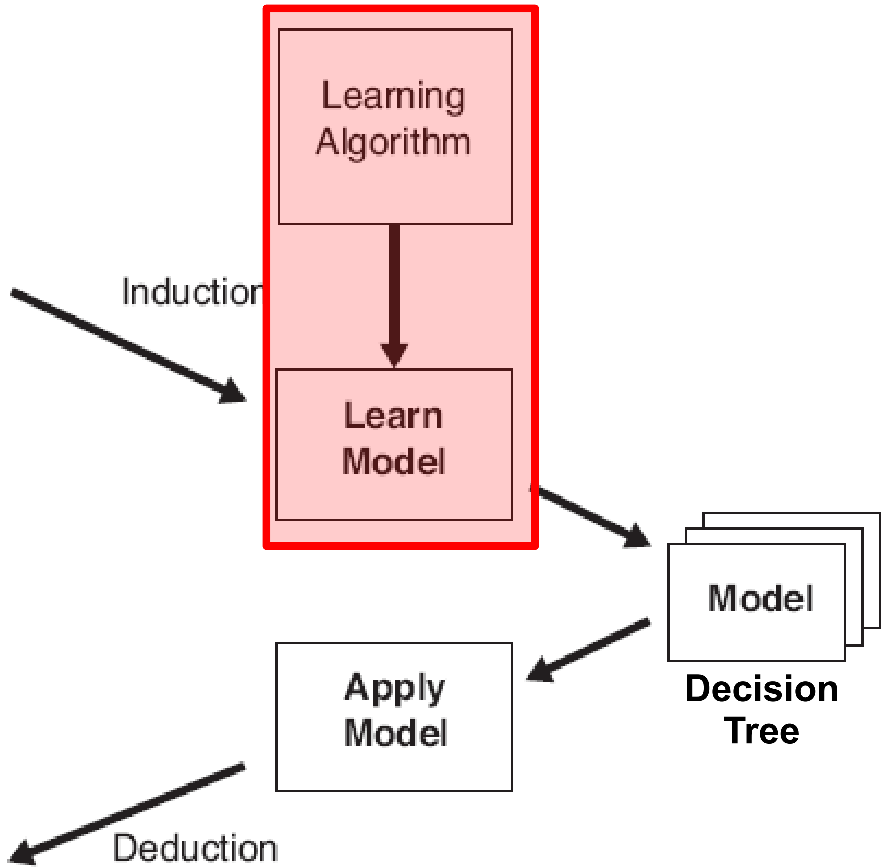
Topics

- Introduction
- **Decision Trees**
  - —Overview
  - —**Tree Induction**
  - —Overfitting and other Practical Issues
- Model Evaluation
  - —Metrics for Performance Evaluation
  - —Methods to Obtain Reliable Estimates
  - —Model Comparison (Relative Performance)
- Feature Selection
- Class Imbalance

Decision Tree: Induction

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes



Test Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Decision Tree Induction

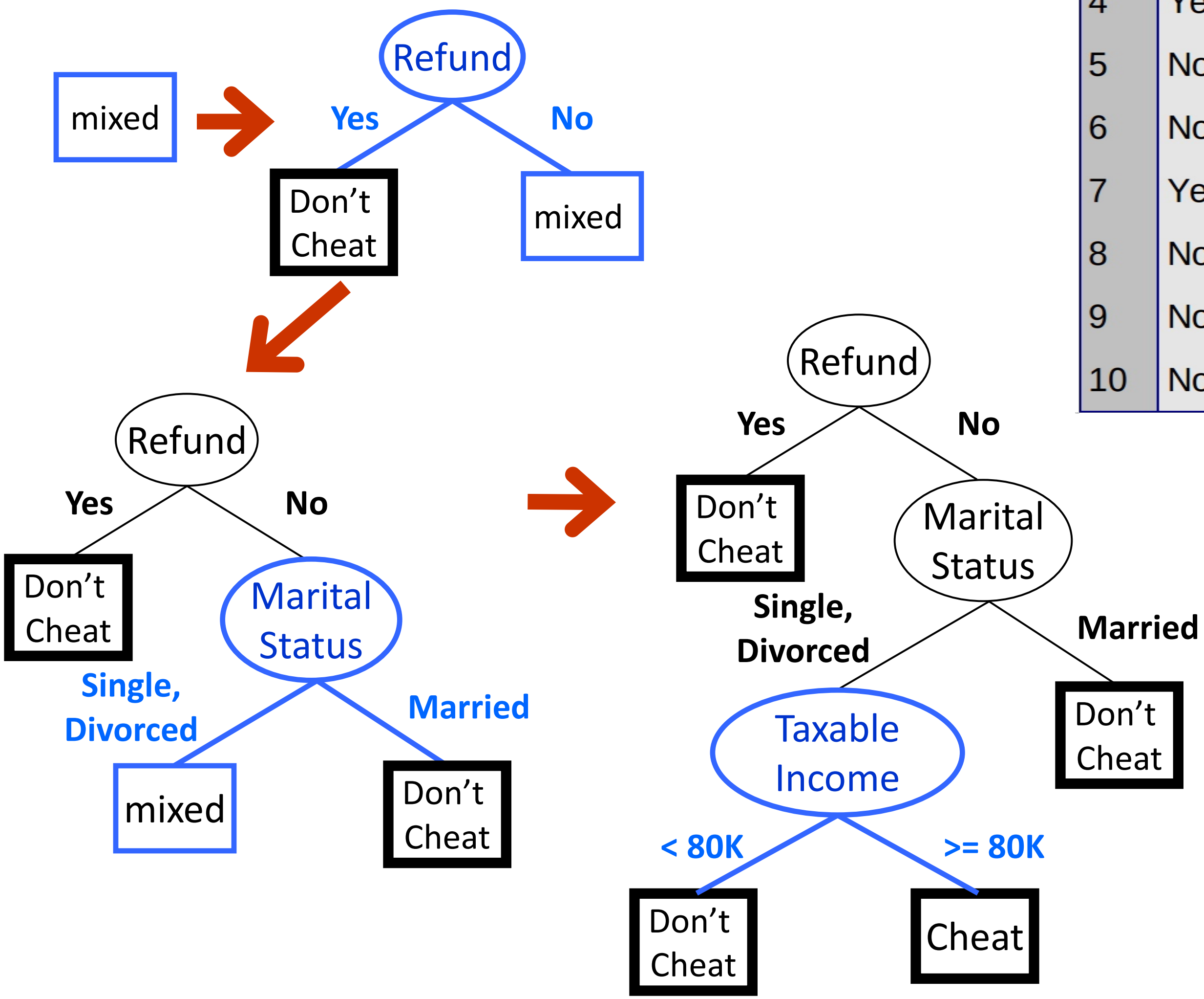
Many Algorithms:

- Hunt's Algorithm (one of the earliest)
- CART (Classification And Regression Tree)
- ID3, C4.5, C5.0 (by Ross Quinlan, information gain)
- CHAID (CHi-squared Automatic Interaction Detection)
- MARS (Improvement for numerical features)
- SLIQ, SPRINT
- Conditional Inference Trees (recursive partitioning using statistical tests)

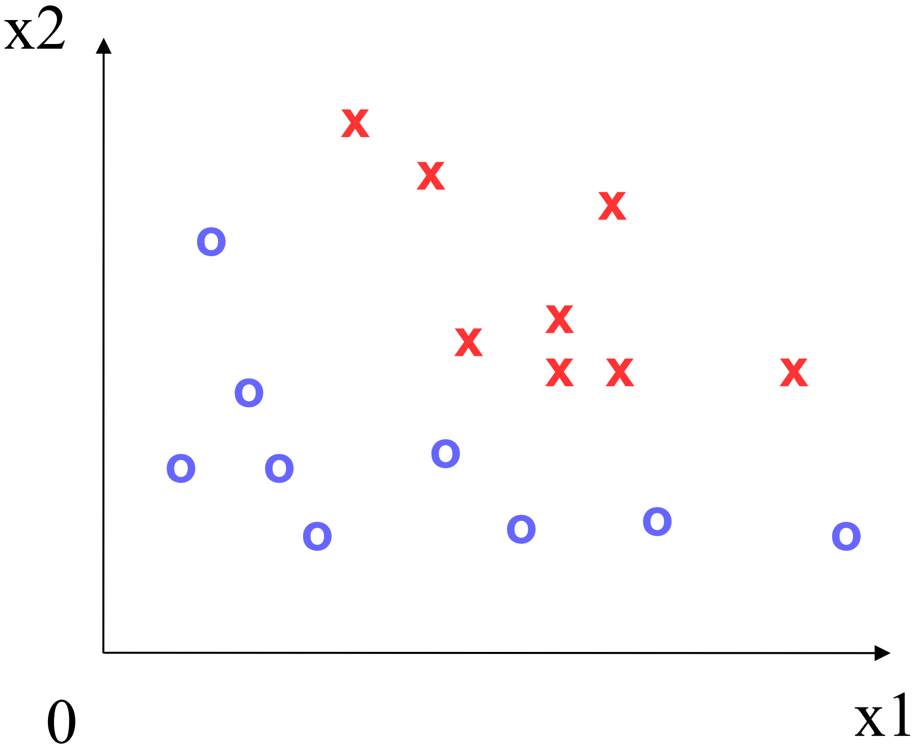
Hunt’s Algorithm

"Use attributes to split the data recursively, till each split contains only a single class."

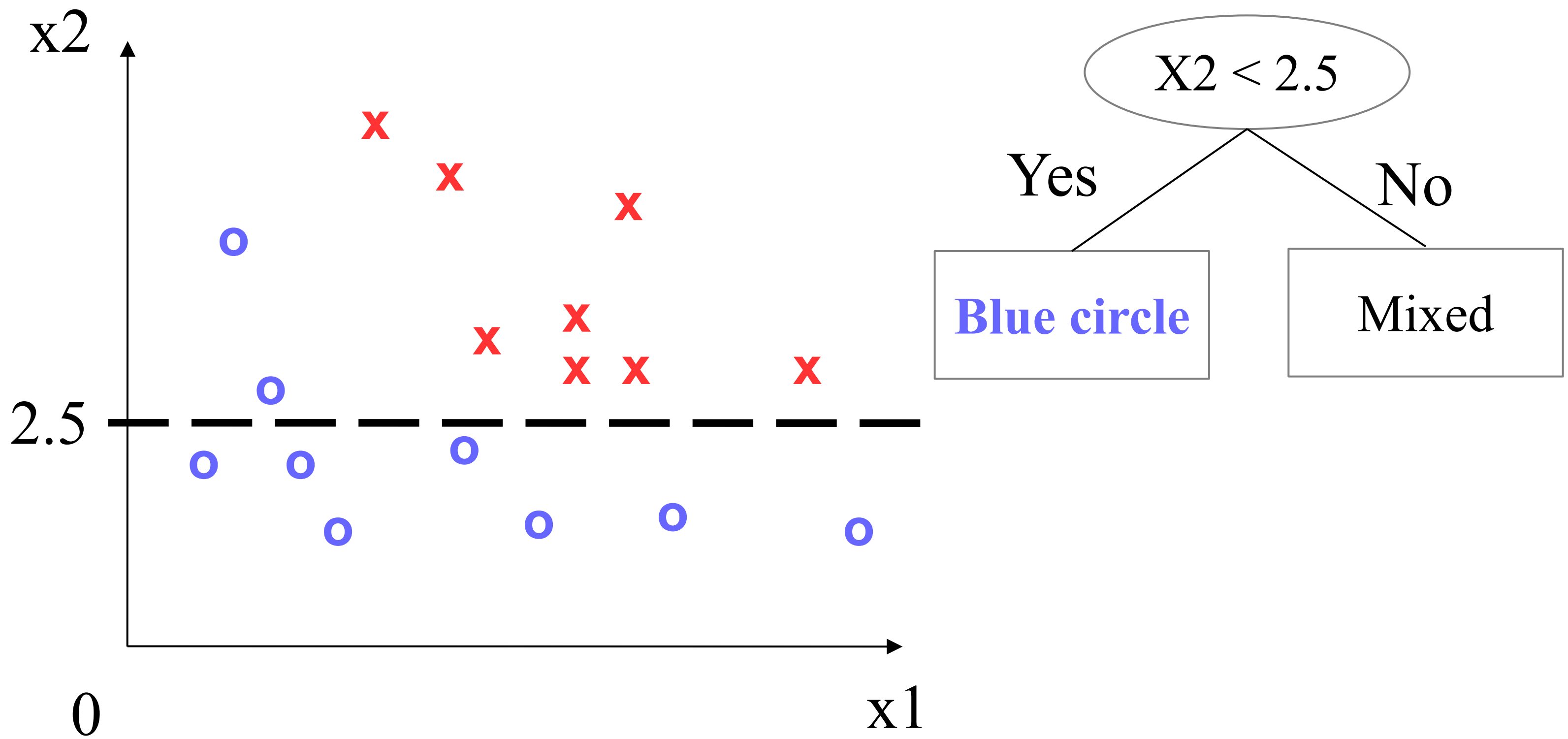
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



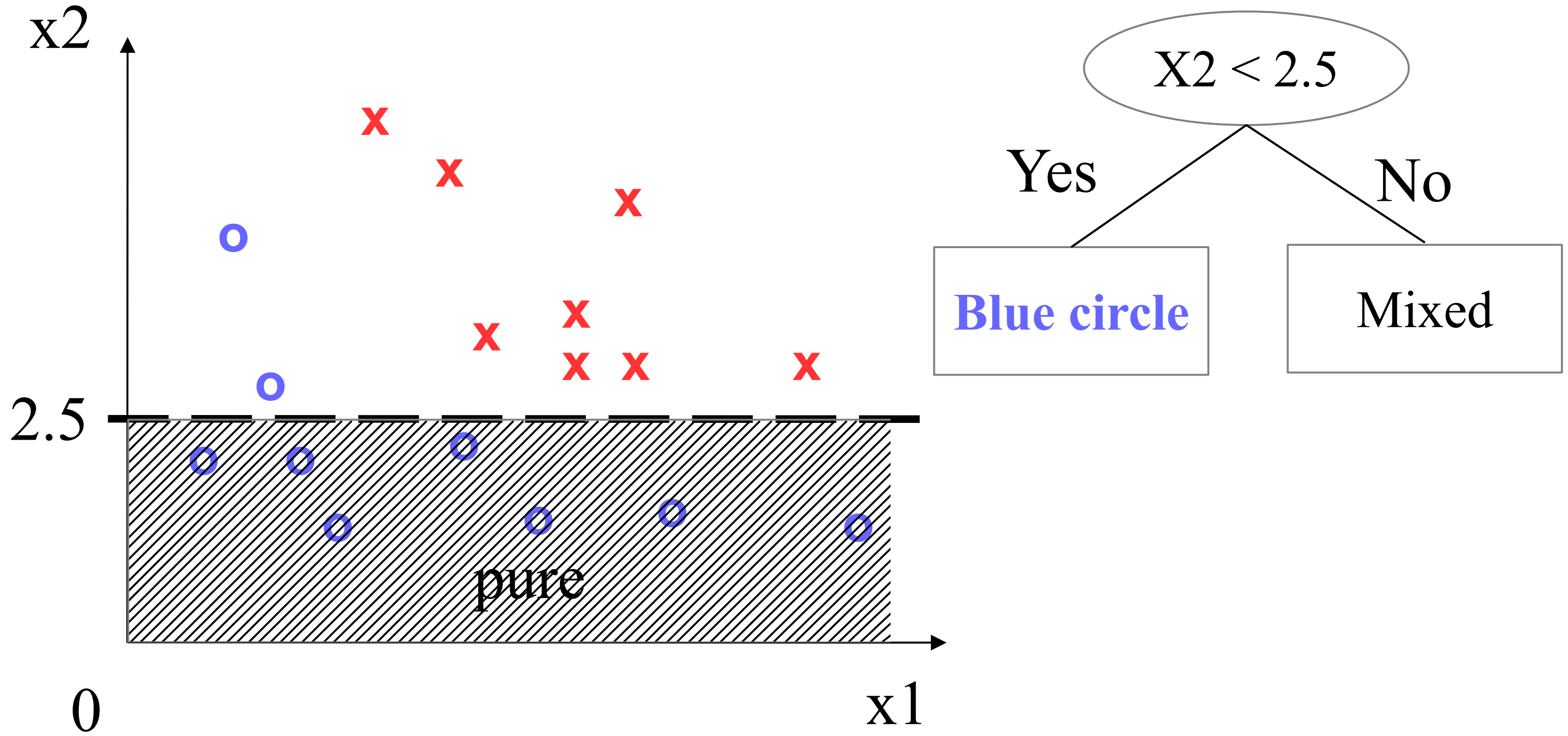
Example 2: Creating a Decision Tree



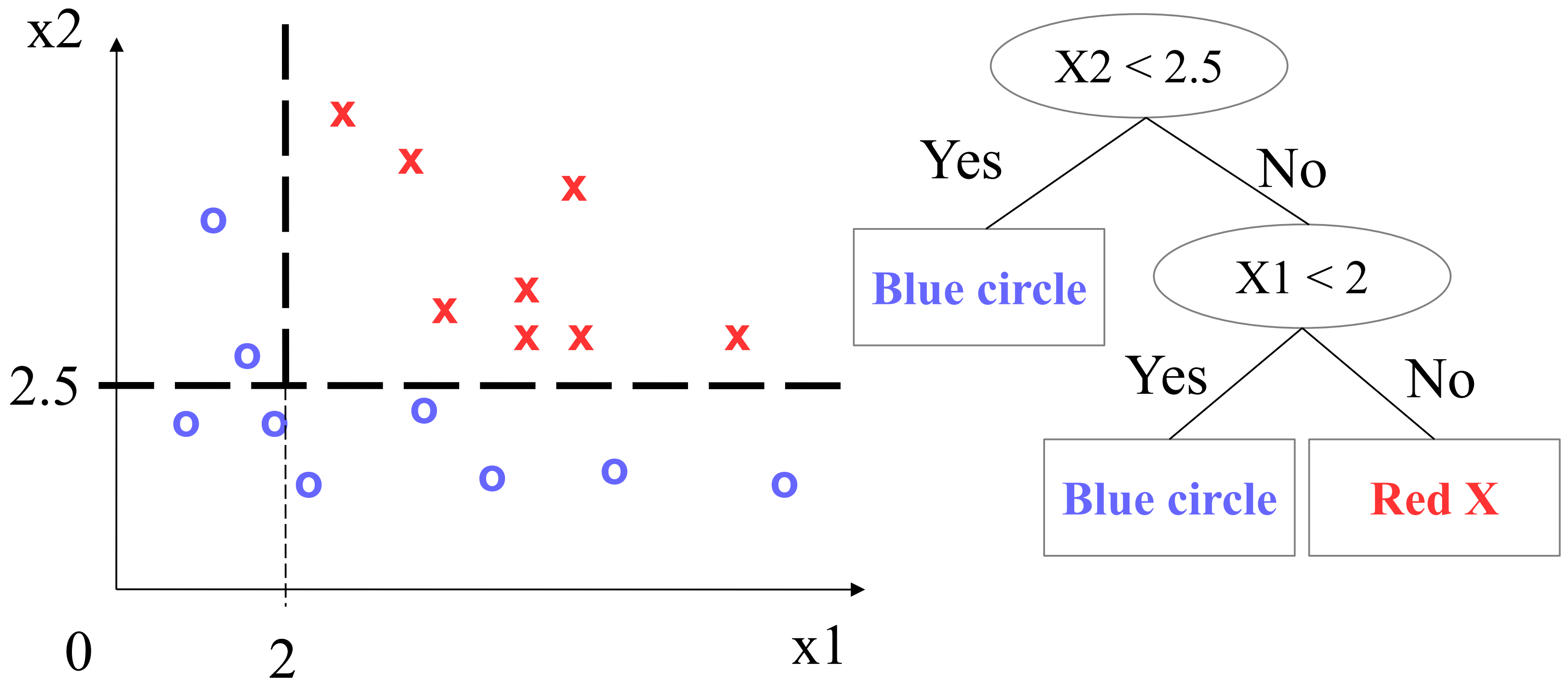
Example 2: Creating a Decision Tree



Example 2: Creating a Decision Tree



Example 2: Creating a Decision Tree



Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.
- Issues
  - Determine how to split the record using different attribute types.
  - How to determine the best split?
  - Determine when to stop splitting

Tree Induction

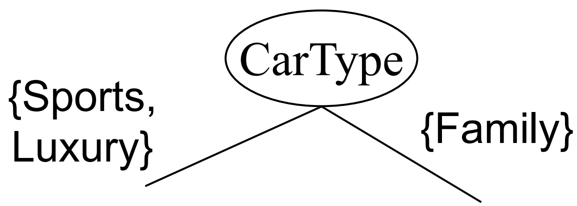
- Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.
- Issues
  - **Determine how to split the record using different attribute types.**
  - How to determine the best split?
  - Determine when to stop splitting

How to Specify Test Condition?

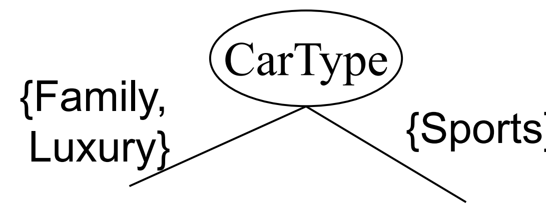
- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous (interval/ratio)

Splitting Based on Nominal Attributes

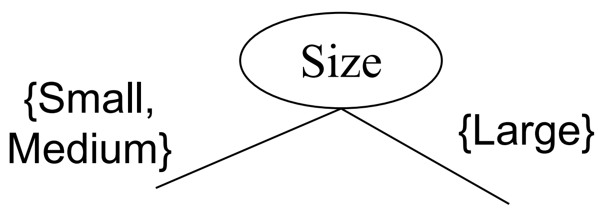
- Nominal Attribute: Divides values into two subsets.
  Need to find optimal partitioning.



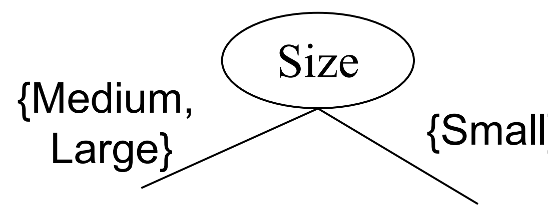
OR



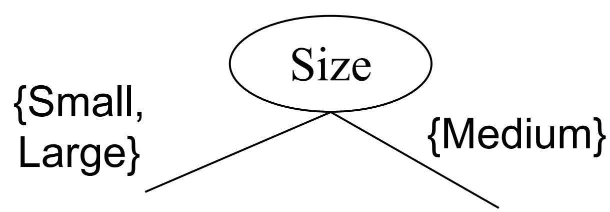
- Ordinal Atribute: Divides values into two subsets.
  Need to find optimal partitioning.



OR

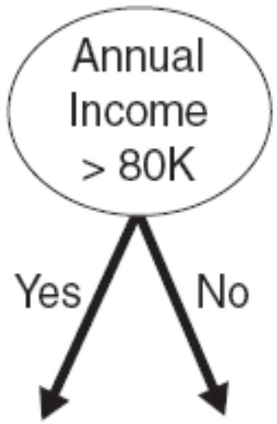


- What about this split?

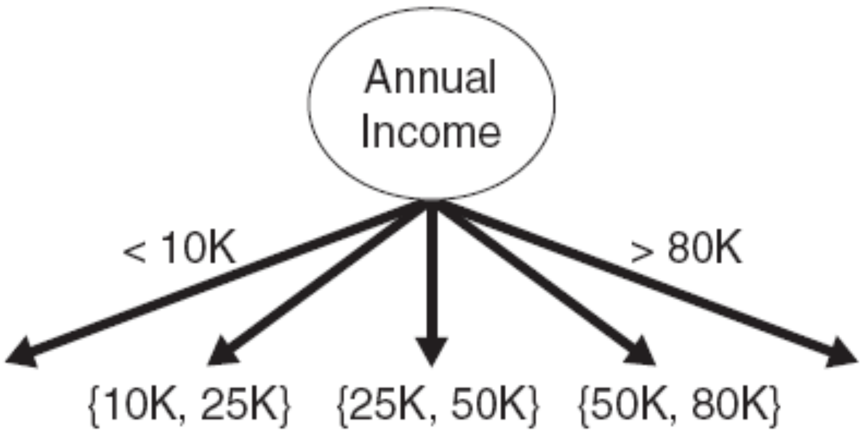


Splitting Based on Continuous Attributes

Binary split



Multi-way split



Discretization to form an ordinal categorical attribute:

- **Static** – discretize the data set once at the beginning (equal interval, equal frequency, etc.).
- **Dynamic** – discretize during the tree construction.
  - Example: For a binary decision ($A < v$) or ($A \geq v$) consider all possible splits and finds the best cut. This can be done efficiently.

Tree Induction

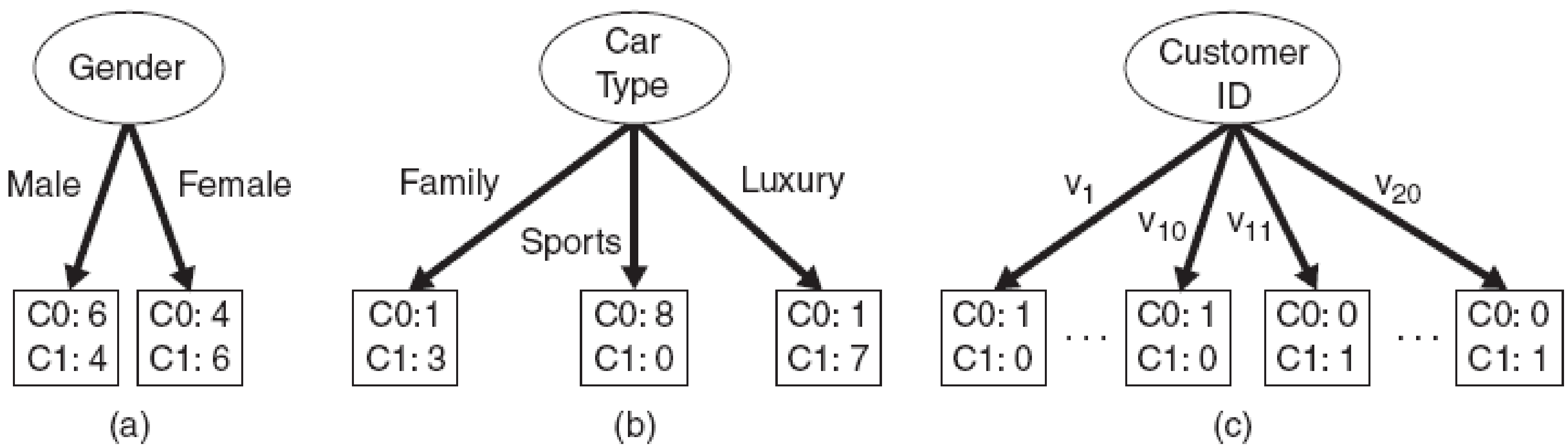
- Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.

- Issues
  - Determine how to split the record using different attribute types.
  - **How to determine the best split?**
  - Determine when to stop splitting

How to determine the Best Split

**Before Splitting: 10 records of class 0,
10 records of class 1**

C0: 10
C1: 10



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

C0:	**5**
C1:	**5**

Non-homogeneous,

High degree of impurity

C0:	**9**
C1:	**1**

Homogeneous,

Low degree of impurity

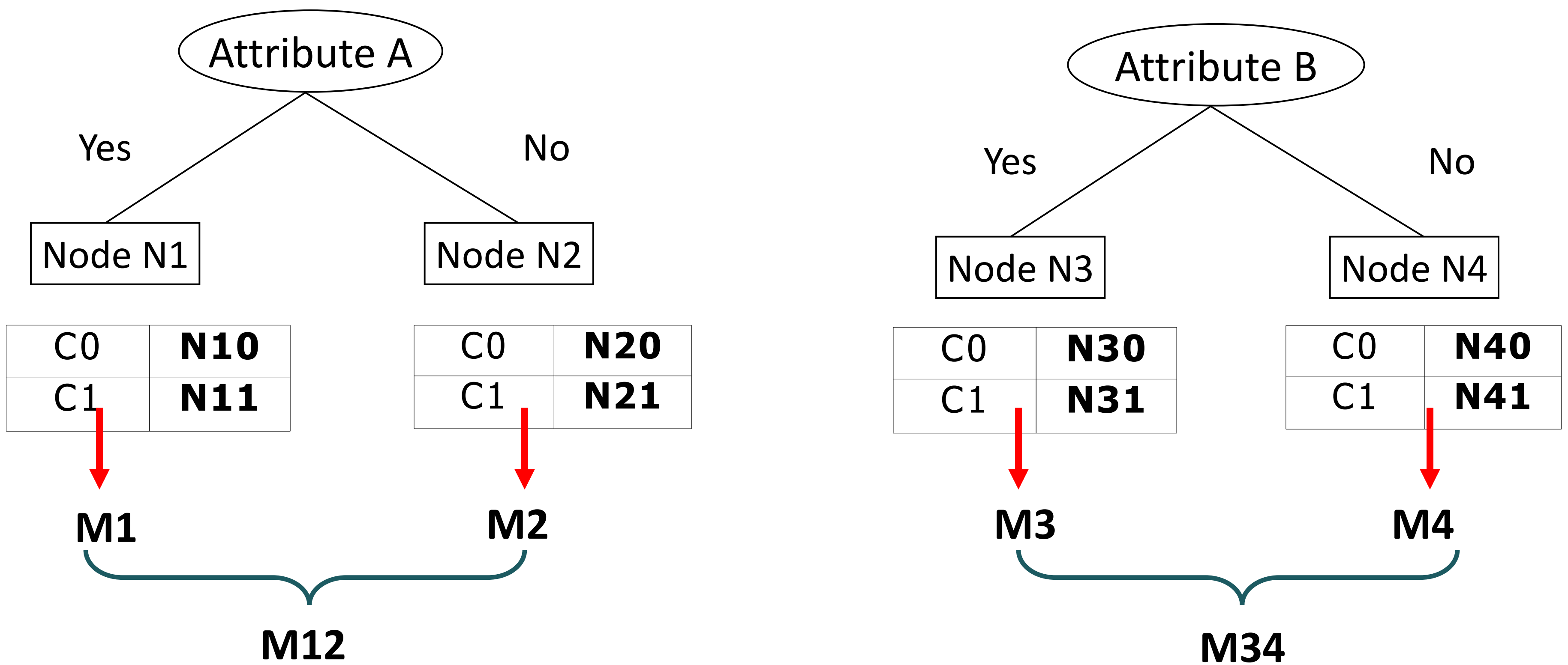
Find the Best Split -General Framework

Assume we have a measure **M** that tells us how "pure" a node is.

Before Splitting:

C0	**N00**
C1	**N01**

→ **M0**



Gain = M0 – M12 vs M0 – M34 → Choose best split

Measures of Node Impurity

Gini Index

Entropy

Classification error

Measures of Node Impurity

Gini Index

Entropy

Classification error

Measure of Impurity: GINI

- Gini Index for a given node t :

$$GINI(t) = \sum_j p(j \mid t)(1 - p(j \mid t)) = 1 - \sum_j p(j \mid t)^2$$

$p(j \mid t)$ is estimated as the relative frequency of class j at node t

- Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.
- Maximum of 1 – $1/n_c$ (number of classes) when records are equally distributed among all classes = maximal impurity.
- Minimum of 0 when all records belong to one class = complete purity.
- Examples:

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j p(j\,|t\,)^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = **0**

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = **0.278**

C1	**2**
C2	**4**

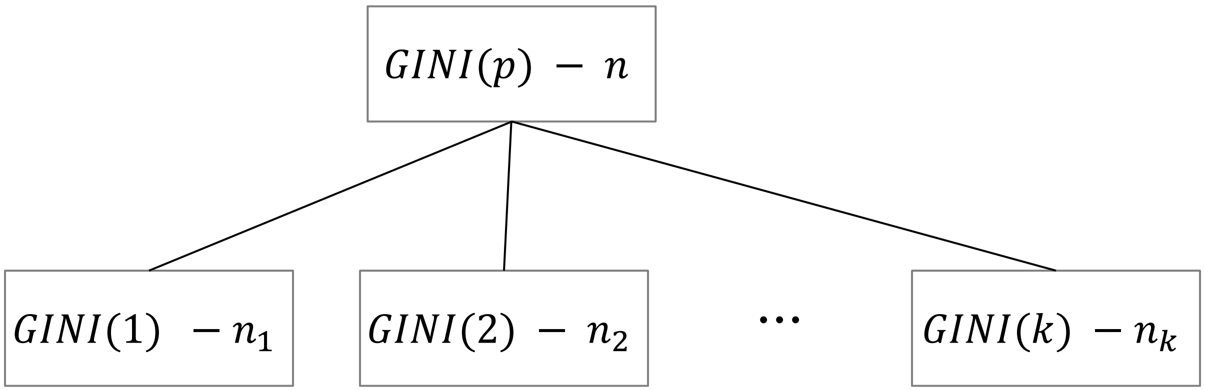
P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = **0.444**

Maximal impurity here is ½ = .5

Splitting Based on GINI

When a node p is split into k partitions (children), the quality of the split is computed as a weighted sum:



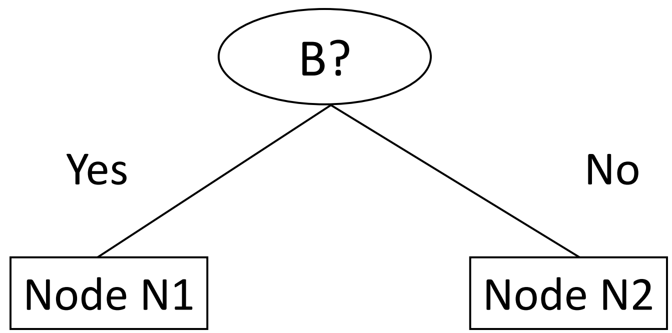
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where n_i = number of records at child i, and n = number of records at node p.

Used in the algorithms CART, SLIQ, SPRINT.

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of weighing partitions: Larger and purer partitions are sought for.



	Parent
C1	**6**
C2	**6**
Gini = 0.500	

Gini(N1)
$= 1 - (5/8)^2 - (3/8)^2$
$= 0.469$

Gini(N2)
$= 1 - (1/4)^2 - (3/4)^2$
$= 0.375$

	N1	N2
C1	**5**	**1**
C2	**3**	**3**
Gini=0.438		

Gini(Children)
$= 8/12 * 0.469 +$
$4/12 * 0.375$
$= 0.438$

GINI improves!

Measures of Node Impurity

Gini Index

Entropy

Classification error

Measure of Impurity: Entropy

- Entropy at a given node t:

$$\text{Entropy(t)} = -\sum_{j} p(j \mid t) \log(p(j \mid t))$$

$p(\, j \mid t)$ is the relative frequency of class j at node t;
$0 \log(0) = 0$ is used!

- Measures homogeneity of a node (originally a measure of uncertainty of a random variable or information content of a message).
- Maximum: $\log(n_c)$ when records are equally distributed among all classes = maximal impurity.
- Minimum: 0 when all records belong to one class = maximal purity.

Examples for computing Entropy

$$\text{Entropy}(t) = -\sum_{j} p(j \mid t) \log(p(j \mid t))$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = $-(1/6)\log_2(1/6) - (5/6)\log_2(1/6) = 0.65$

C1	**3**
C2	**3**

P(C1) = 3/6 P(C2) = 3/6

Entropy = $-(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = 1$

Information Gain

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;
n_i is number of records in partition i

- Measures reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3, C4.5 and C5.0
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Gain Ratio

$$GainRato_{split} = \frac{GAIN_{split}}{SplitInfo}$$

$$SplitInfo = -\sum_{i=1}^{k} \frac{n_i}{n} log\left(\frac{n_i}{n}\right)$$

Parent Node, p is split into k partitions;
n_i is number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitInfo). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain.

Measures of Node Impurity

Gini Index

Entropy

Classification error

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_i p(i \mid t)$$

$p(j \mid t)$ is the relative frequency of class j at node t

- Measures misclassification error made by a node.
- Maximum: $1 - \frac{1}{n_c}$ when records are equally distributed among all classes = maximal impurity (maximal error).
- Minimum: 0 when all records belong to one class = maximal purity (no error)

Examples for Computing Error

$$Error(t) = 1 - \max_i p(i\,|t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

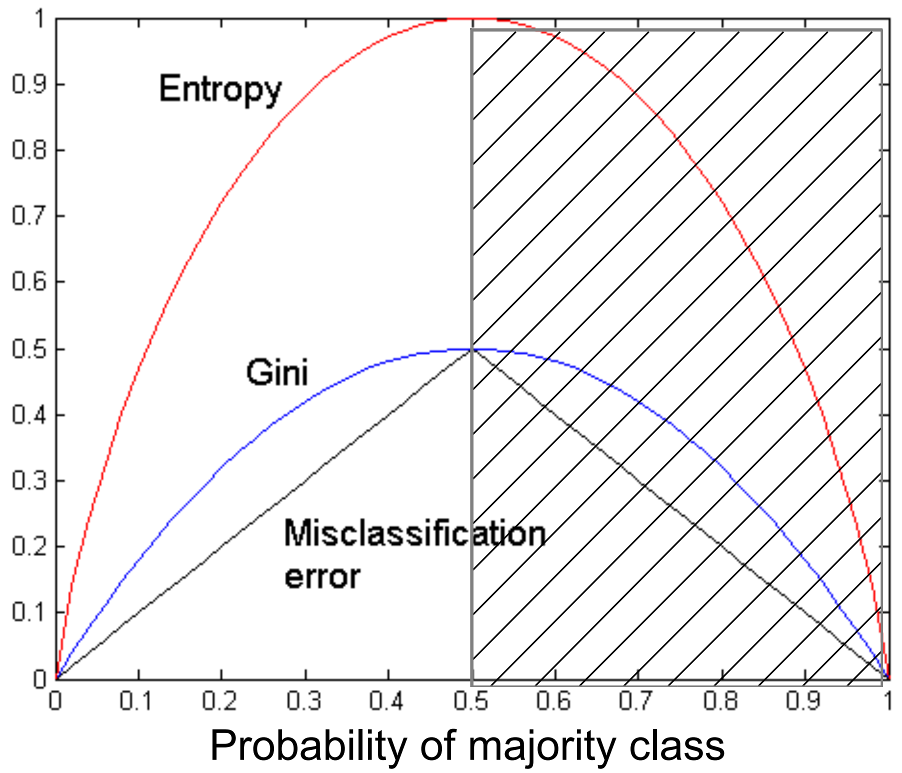
C1	**3**
C2	**3**

P(C1) = 3/6 P(C2) = 3/6

Error = 1 – max (3/6, 3/6) = 1 – 3/6 = .5

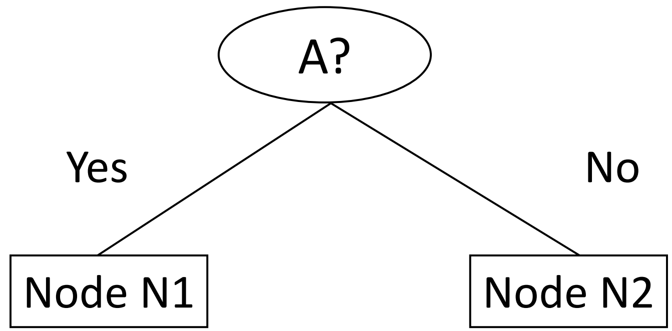
Comparison among Splitting Criteria

For a 2-class problem: Probability of the majority class p is always > .5



Note: The order is the same no matter what splitting criterion is used, however, the gain (differences) are not.

Misclassification Error vs Gini



	Parent
C1	**7**
C2	**3**
Gini = 0.42	
Error = 0.30	

$\text{Gini(N1)} = 1 - (3/3)^2 - (0/3)^2 = 0$

$\text{Gini(N2)} = 1 - (4/7)^2 - (3/7)^2 = 0.489$

$\text{Gini(Split)} = 3/10 * 0 + 7/10 * 0.489 = 0.342$

$\text{Error(N1)} = 1\text{-}3/3=0$

$\text{Error(N2)}=1\text{-}4/7=3/7$

$\text{Error(Split)}= 3/10*0 + 7/10*3/7 = 0.3$

	N1	**N2**
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		
Error = 0.30		

Gini improves!

Error does not improve!!!

Tree Induction

- Greedy strategy
  - Split the records based on an attribute test that optimizes a certain criterion.

- Issues
  - Determine how to split the record using different attribute types.
  - How to determine the best split?
  - **Determine when to stop splitting**

Stopping Criteria for Tree Induction

- Stop expanding a node when **all the records belong to the same class**. Happens guaranteed when there is only one observation left in the node (e.g., Hunt's algorithm).
- Stop expanding a node when all the records in the node have the **same attribute values**. Splitting becomes impossible.
- **Early termination criterion** (to be discussed later with tree pruning)

Advantages of Decision Tree Based Classification

INEXPENSIVE TO CONSTRUCT

EXTREMELY FAST AT CLASSIFYING UNKNOWN RECORDS

EASY TO INTERPRET FOR SMALL-SIZED TREES

ACCURACY IS COMPARABLE TO OTHER CLASSIFICATION TECHNIQUES FOR MANY SIMPLE DATA SETS

Example: C4.5

- Simple depth-first construction.
- Uses Information Gain (improvement in Entropy).
- Handling both continuous and discrete attributes (cont. attributes are split at threshold).
- Needs entire data to fit in memory (unsuitable for large datasets).
- Trees are pruned.

- Code available at
  - http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz
  - Open Source implementation as J48 in Weka/rWeka

Topics

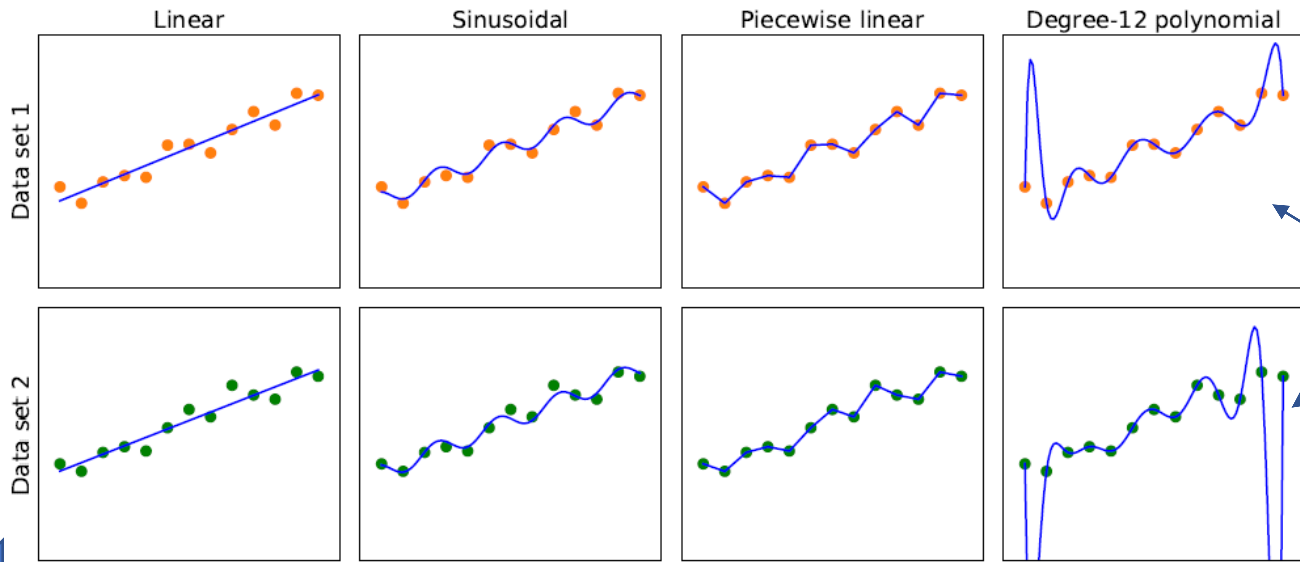
- Introduction
- Decision Trees
  - —Overview
  - —Tree Induction
  - —**Overfitting and other Practical Issues**
- Model Evaluation
  - —Metrics for Performance Evaluation
  - —Methods to Obtain Reliable Estimates
  - —Model Comparison (Relative Performance)
- Feature Selection
- Class Imbalance

Model Selection: Bias vs. Variance

Overfitting to training data

Simplicity

Consistency



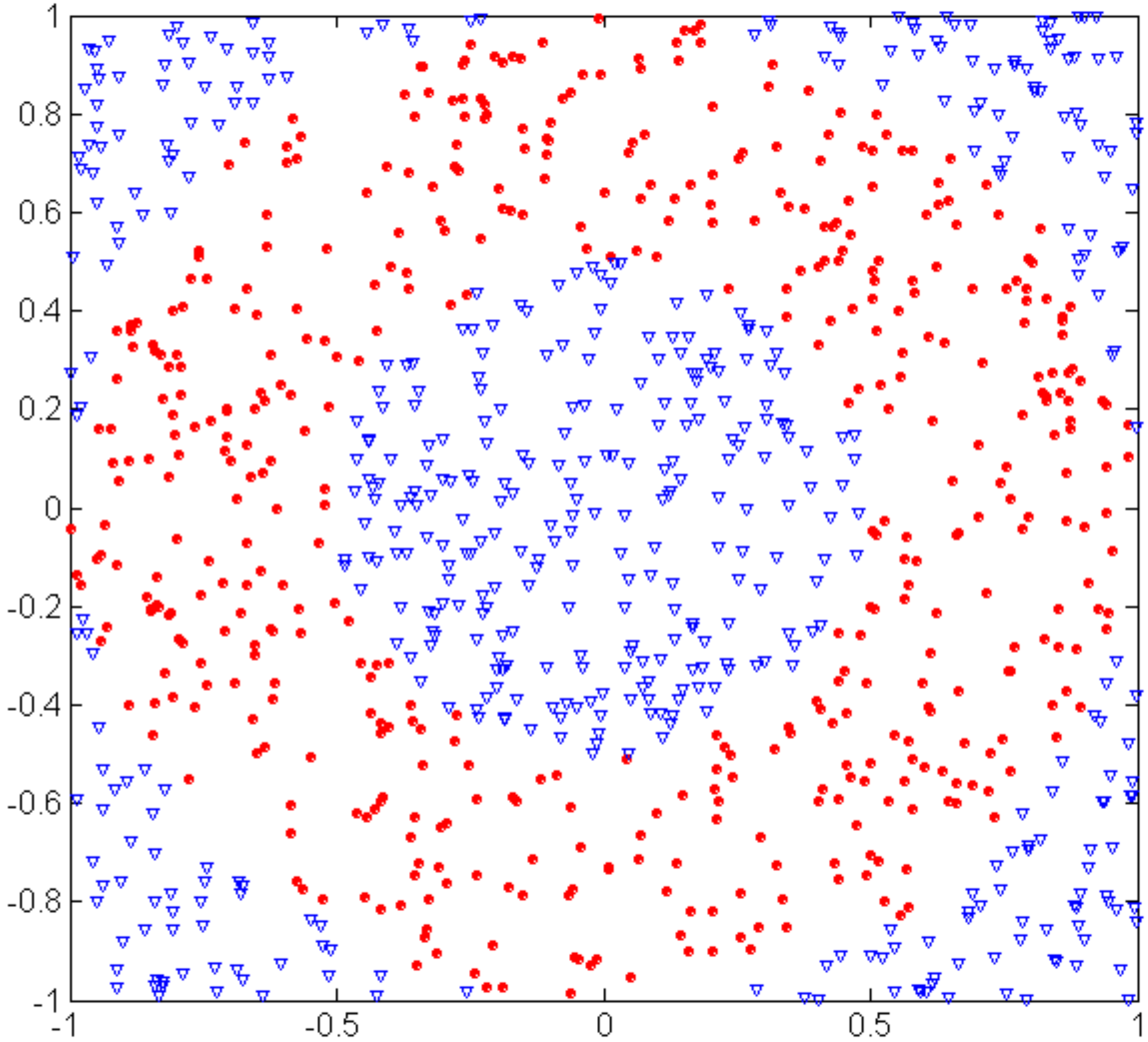
Two samples from the same function f (points) with the learned function h (lines).

High **Bias**: restrictions by the model class Low

Low **Variance**: difference in the model due to slightly different data. high

Note: This trade-off applies to any model.

Example: Underfitting and Overfitting



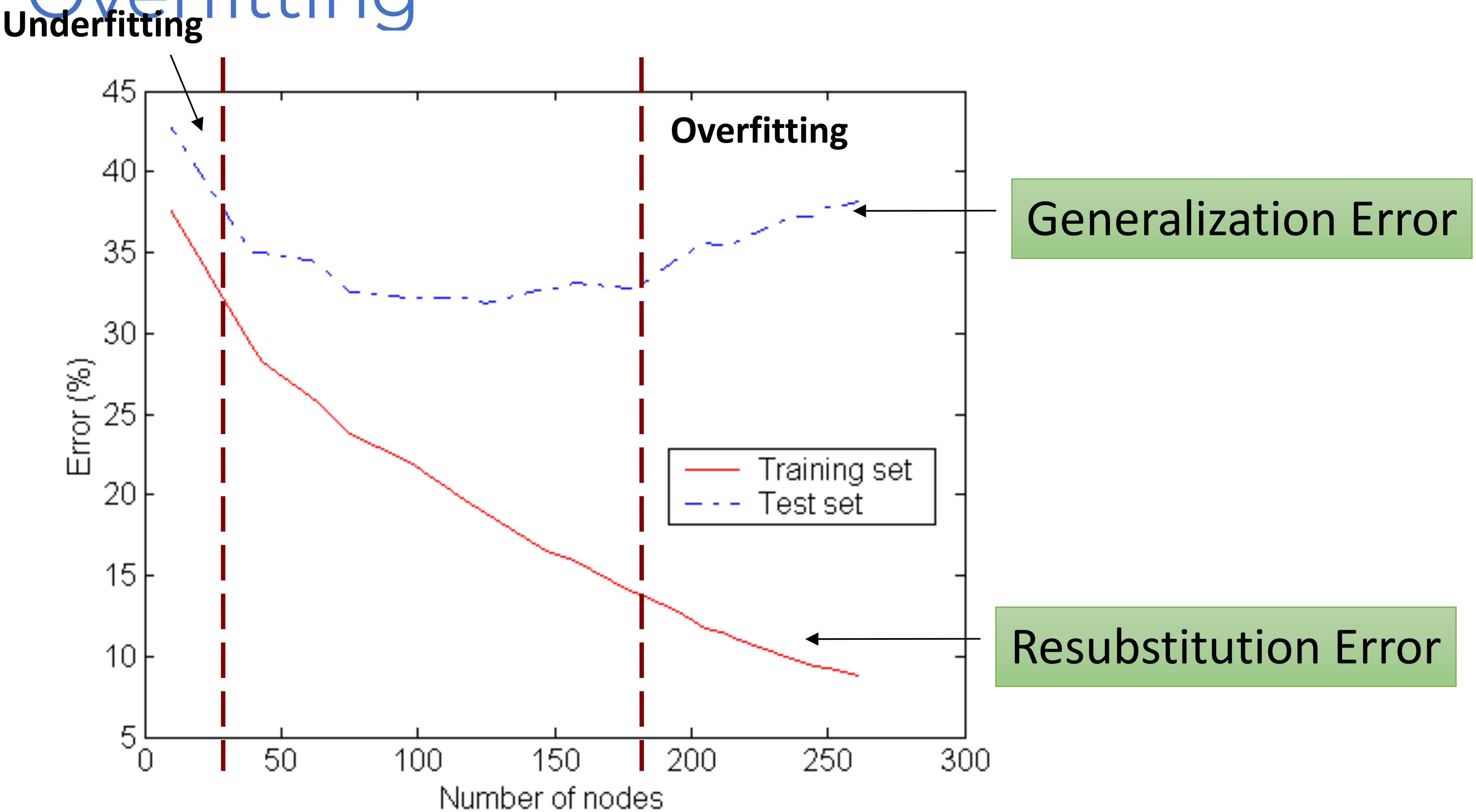
500 circular and 500 triangular data points.

Circular points:
$0.5 \geq sqrt(x_1^2 + x_2^2) \leq 1$

Triangular points:

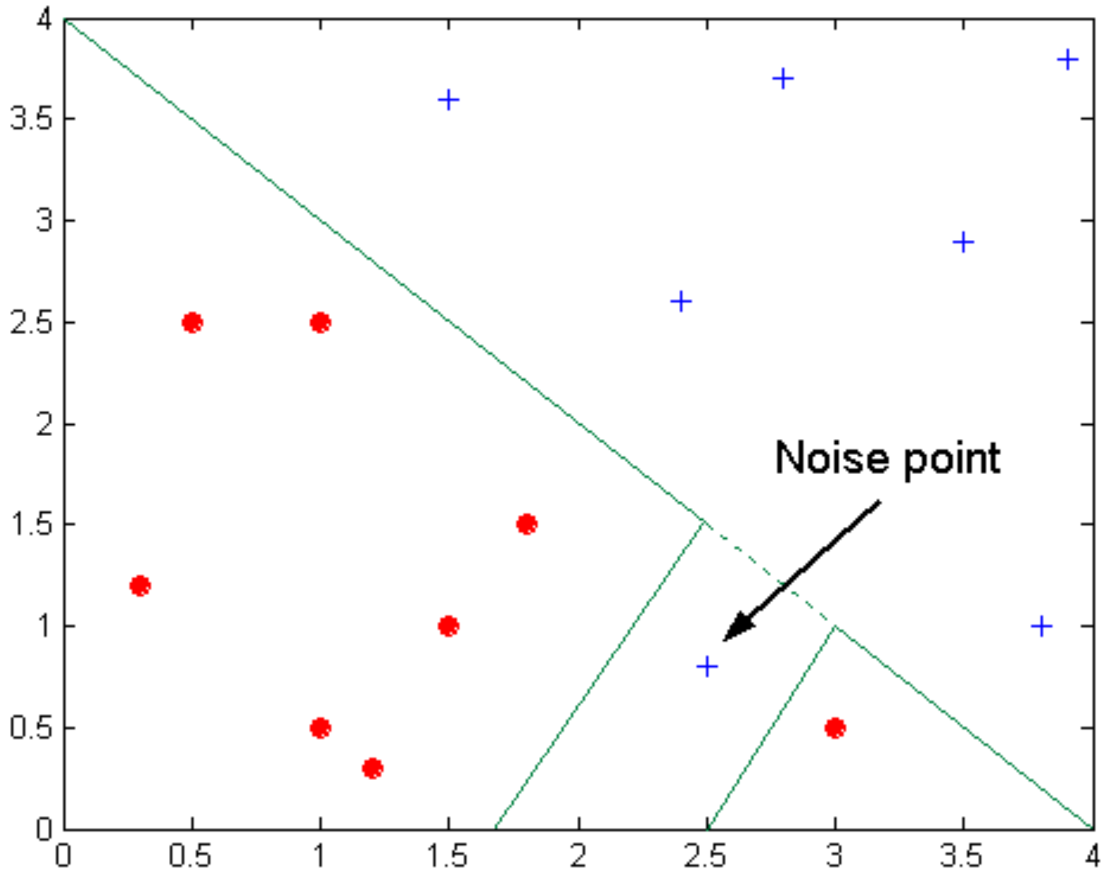
$sqrt(x_1^2 + x_2^2) < 0.5$ or
$sqrt(x_1^2 + x_2^2) > 1$

Example: Underfitting and Overfitting



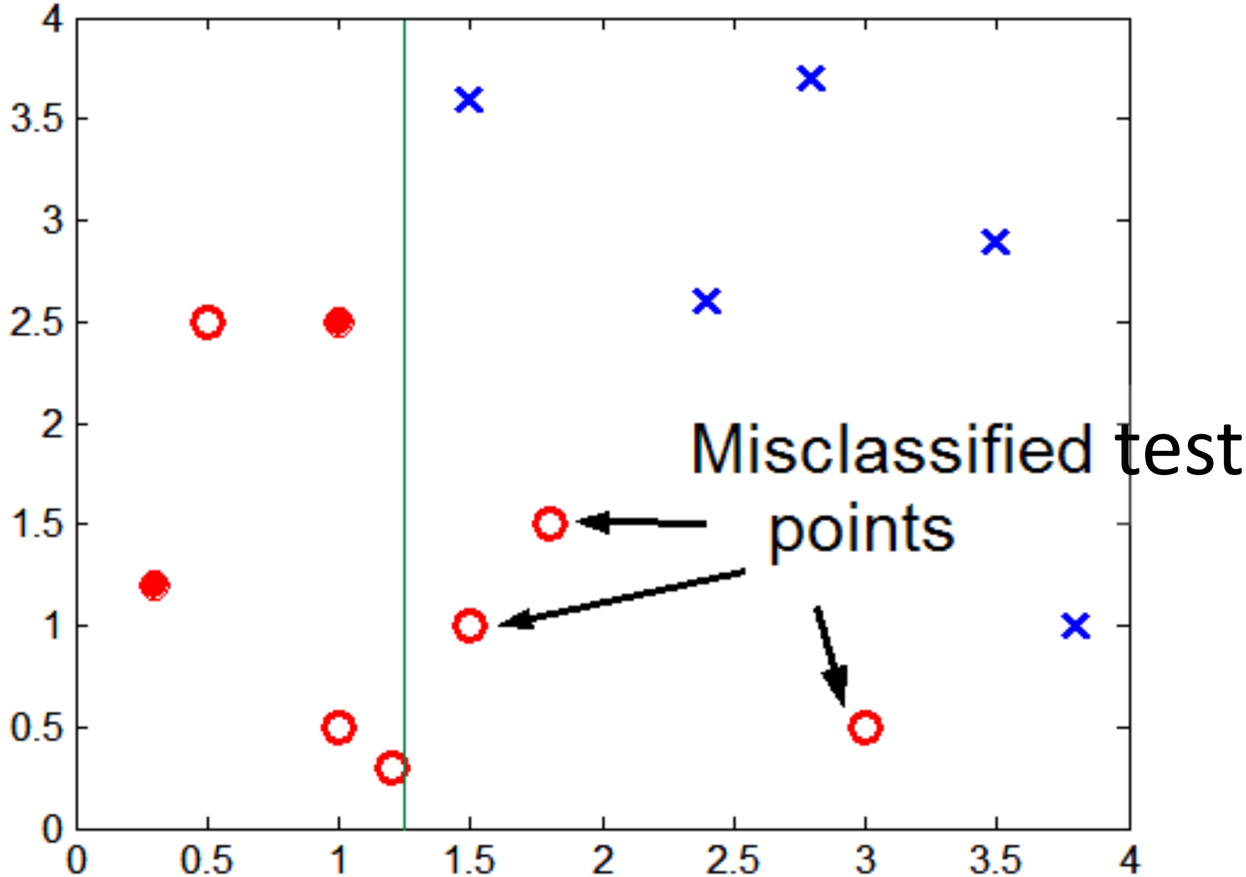
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



Lack of training data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

Generalization Error

- Overfitting results in decision trees that **are more complex than necessary.**

- Training error does not provide a good estimate of how well the tree will perform on previously unseen records (e.g., test data).

- Need new ways for estimating errors $\rightarrow$ **Generalization Error**

Estimating Generalization Errors

- **Re-substitution errors**: error on training set e
- **Generalization errors:** error on testing set e'

Methods for estimating generalization errors:

1. **Optimistic approach**: e' = e
2. **Pessimistic approach:**
   - e' = e + N x 0.5 (N: number of leaf nodes)
   - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
     Training error (rate) = 10/1000 = 1%
     Estimated generalization error (rate)= (10 + 30 x 0.5)/1000 = 2.5%
3. **Validation approach:**
   - uses a validation (test) data set (or cross-validation) to estimate generalization error.

Penalty for model complexity!
0.5 per leave node is often used for binary splits.

Occam's Razor (Principle of parsimony)

"Simpler is better"

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.
- For complex models, there is a greater chance of overfitting (i.e., it fitted accidentally errors in the training data).

Therefore, one should include model complexity when evaluating a model.

How to Address Overfitting in Decision Trees

Pre-Pruning (Early Stopping Rule): Stop the algorithm before it becomes a fully-grown tree.

- Typical stopping conditions for a node:
  - Stop if all instances belong to the **same class**
  - Stop if all the **attribute values are the same**
- More restrictive conditions:
  - Stop if **number of instances** is less than some user-specified threshold (estimates become bad for small sets of instances)
  - Stop if class distribution of instances are **independent** of the available features (e.g., using a χ^2 test)
  - Stop if expanding the current node **does not improve impurity** measures (e.g., Gini or information gain).

How to Address Overfitting in Decision Trees

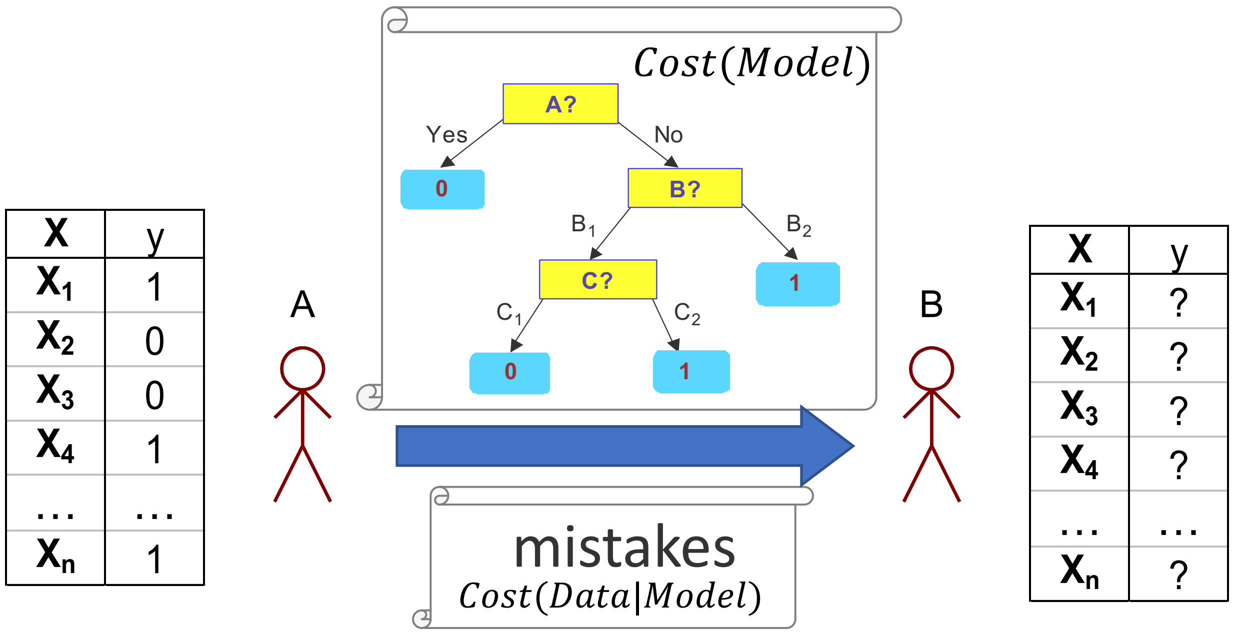
Post-pruning

1. Grow decision tree to its entirety
2. Try trimming sub-trees of the decision tree in a bottom-up fashion

- If generalization error improves after trimming a sub-tree, replace the sub-tree by a leaf node (class label of leaf node is determined from majority class of instances in the sub-tree)

- You can use MDL instead of error for post-pruning

Refresher: Minimum Description Length (MDL)



- $Cost(Model, Data) = \ Cost(Data|Model) + \ Cost(Model) \rightarrow min$
  - Cost is the number of bits needed for encoding.
- $Cost(Model)$ encodes each node (splitting condition and children).
- $Cost(Data|Model)$ encodes information to correct misclassification errors.

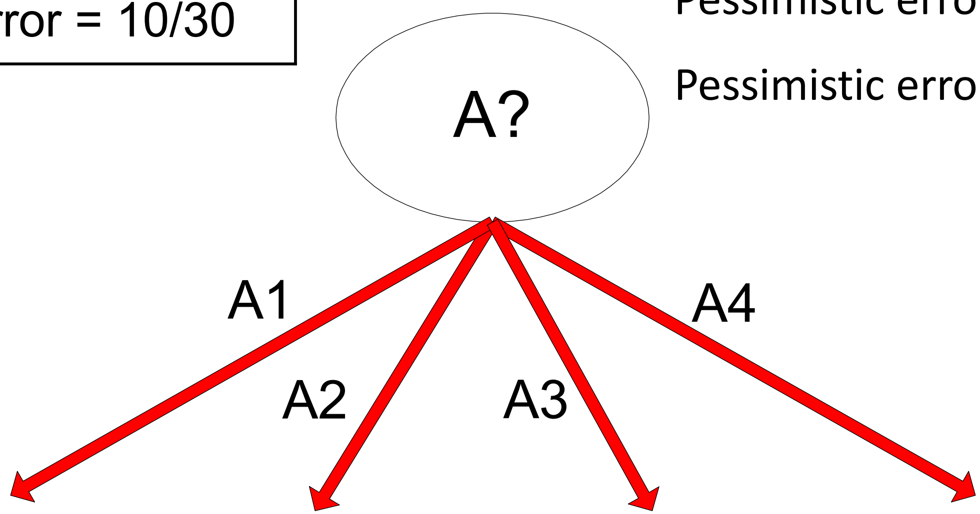
Example of Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Before split:
Training Error = 10/30
Pessimistic error = (10 + 1 x 0.5)/30 = 10.5/30

After split:
Training Error = 9/30
Pessimistic error = (9 + 4 x 0.5)/30 = 11/30

Pessimistic error increases! **PRUNE!**



	A1	A2	A3	A4
Class = Yes	8	3	4	5
Class = No	4	4	1	1

Error = 9

Other issues: Data Fragmentation and Search Strategy

Data Fragmentation

- Number of instances gets smaller as you traverse down the tree and can become too small to make a statistically significant decision (splitting or determining the class in a leaf node)

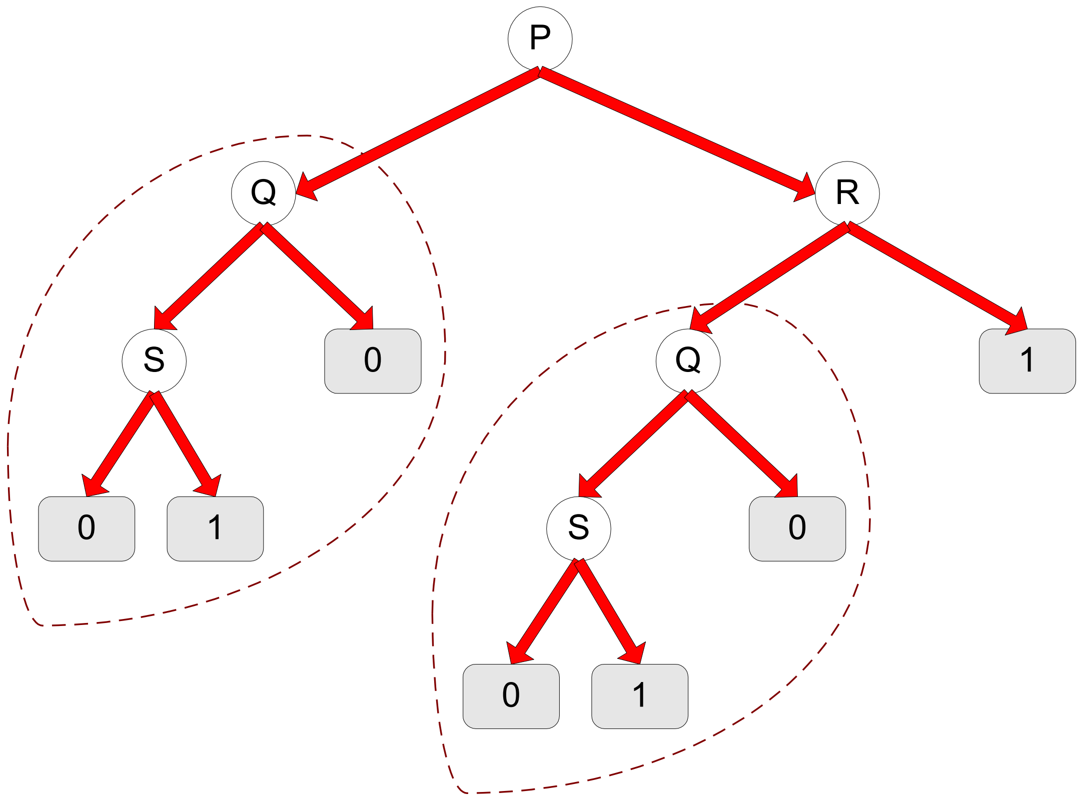
→ Many algorithms **stop when a node has not enough instances**.

Search Strategy

- Finding an optimal decision tree is NP-hard

→ Most algorithm use a **greedy, top-down, recursive partitioning strategy** to induce a reasonable solution.

Other issues: Tree Replication



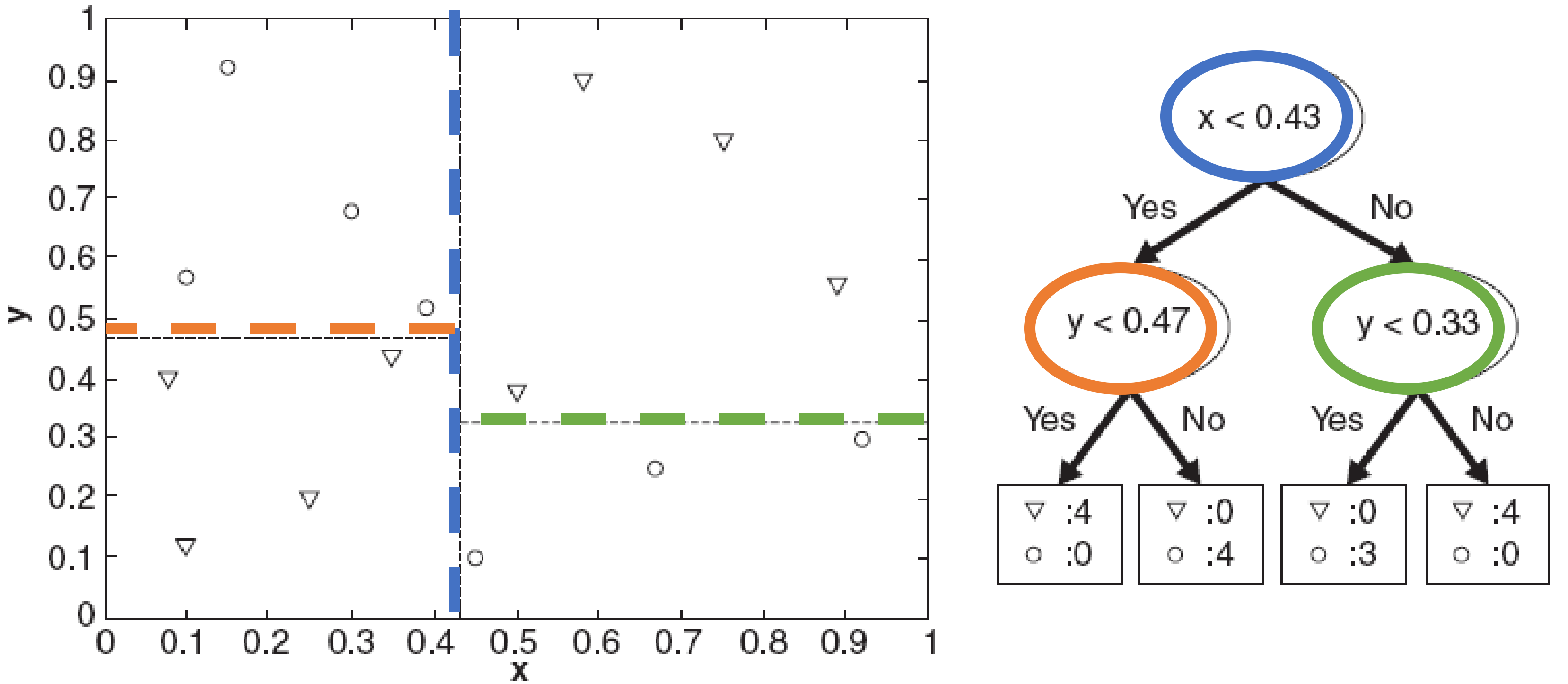
- Same subtree appears in multiple branches
- Makes the model more complicated and harder to interpret

Expressiveness of Decision Trees

- Decision tree can learn discrete-valued functions to separate classes.
- This function represents the **decision boundary.**

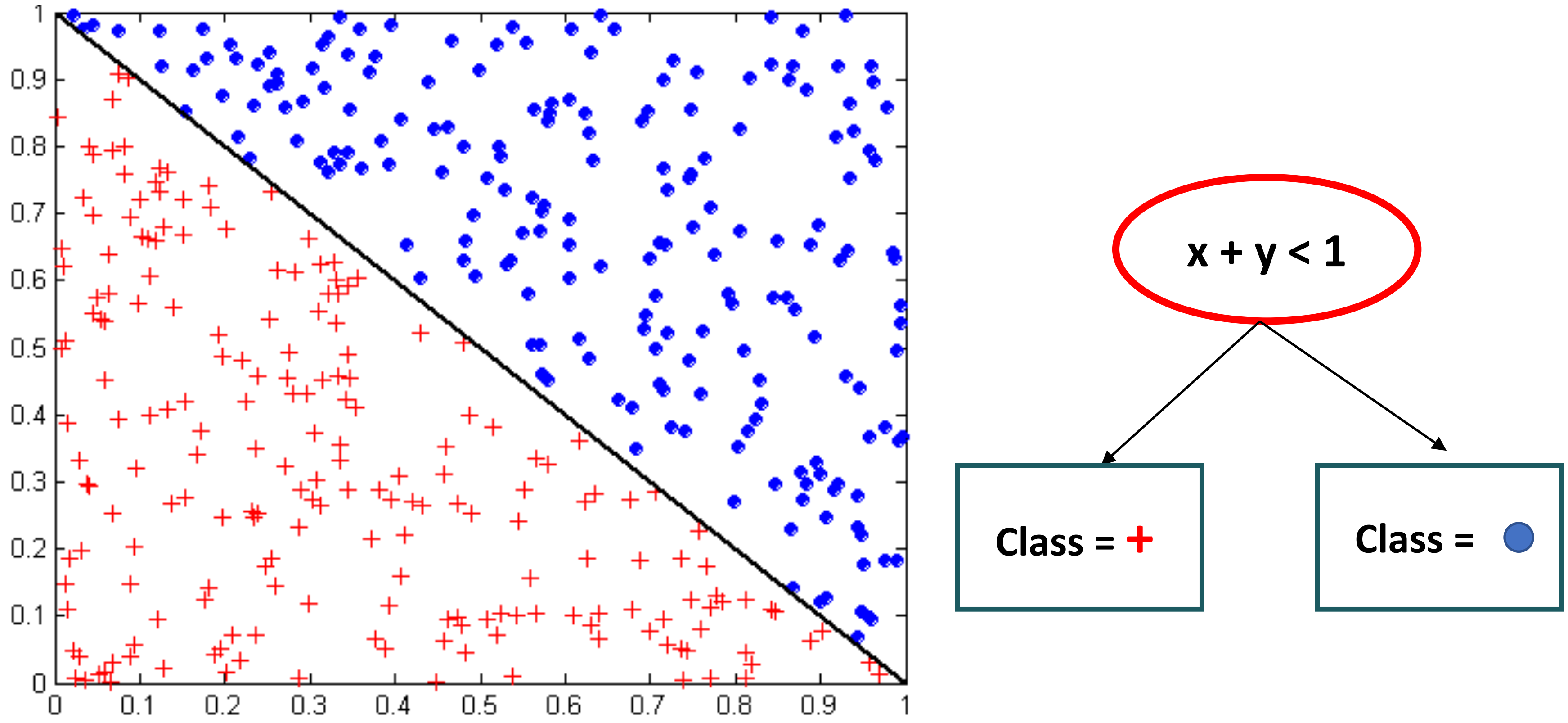
- Issues
  - —Not expressive enough for modeling continuous variables directly (need to be discretized for the split).
  - —Do not generalize well to certain types of Boolean functions like the parity function (Class = 1 if there is an even number of Boolean attributes with truth value = True and 0 otherwise). These functions lead to excessive tree replication.

Decision Boundary



- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

Oblique Decision Trees



- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive -> Not used in practice.

Topics

- Introduction
- Decision Trees
  - —Overview
  - —Tree Induction
  - —Overfitting and other Practical Issues
- **Model Evaluation**
  - —**Metrics for Performance Evaluation**
  - —Methods to Obtain Reliable Estimates
  - —Model Comparison (Relative Performance)
- Feature Selection
- Class Imbalance

Metrics for Performance Evaluation: Confusion Matrix

- Focus on the predictive capability of a model (not speed, scalability, etc.)
- Here we will focus on binary classification problems!

Confusion Matrix

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	**a** **(TP)**	*b* *(FN)*
	Class=No	*c* *(FP)*	**d** **(TN)**

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation: Statistical Test

From Statistics: Null Hypotheses H0 is that the actual class is yes

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes		Type I error
	Class=No	Type II error	

Type I error: $P(NO \mid H0\ is\ true)$ → Significance α

Type II error: $P(Yes \mid H0\ is\ false)$ → Power 1-β

Metrics for Performance Evaluation: Accuracy

Most widely-used metric: How many do we predict correct (in percent)?

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

$$Accuracy = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{N}$$

Limitation of Accuracy

Consider a 2-class problem

- Number of Class 0 examples = 9990
- Number of Class 1 examples = 10

If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %

- Accuracy is misleading because the model does not detect any class 1 example

→ **Class imbalance problem!**

Cost Matrix

Different types of error can have different cost!

	PREDICTED CLASS		
	C(i|j)	**Class=Yes**	**Class=No**
ACTUAL CLASS	**Class=Yes**	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): Cost of misclassifying class j example as class i

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	-1	100
	-	1	0

Missing a + case is really bad!

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Accuracy = 80%

Cost = -1*150+100*40+
1*60+0*250 = 3910

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 90%

Cost = 4255

Cost vs Accuracy

Count	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	p	q
	Class=No	q	p

Accuracy is only proportional to cost if

1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

$\text{Accuracy} = (a + d)/N$

$$\begin{aligned}
\text{Cost} &= p\,(a + d) + q\,(b + c) \\
&= p\,(a + d) + q\,(N - a - d) \\
&= q\,N - (q - p)(a + d) \\
&= N\,[q - (q-p) \times \text{Accuracy}]
\end{aligned}$$

Cost-Biased Measures

	PREDICTED CLASS		
ACTUAL CLASS		Class Yes	Class No
	Class Yes	a (TP)	b (FN)
	Class No	c (FP)	d (TN)

$$Precision\ (p) = \frac{a}{a+c}$$

$$Recall\ (r) = \frac{a}{a+b}$$

$$F - measure\ (F) = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

Kappa Statistic

Idea: Compare the accuracy of the classifier with a random classifier. The classifier should be better than random!

	PREDICTED CLASS		
ACTUAL CLASS		Class Yes	Class No
	Class Yes	a (TP)	b (FN)
	Class No	c (FP)	d (TN)

$$\kappa = \frac{total\ accuracy\ - random\ accuracy}{1\ - random\ accuracy}$$

$$total\ accuracy = \frac{TP + TN}{N}$$

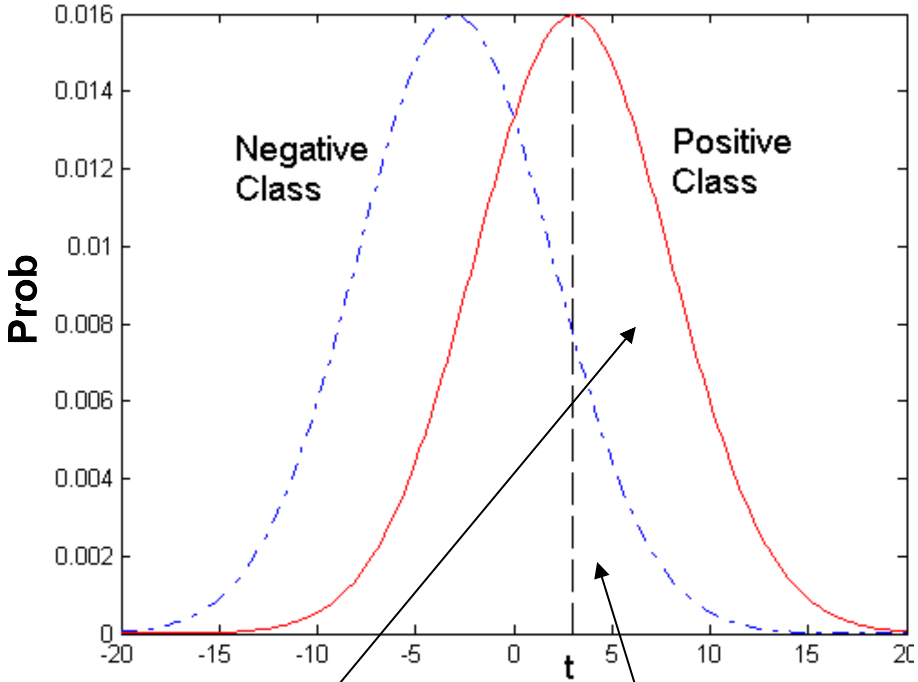
$$random\ accuracy = \frac{TP + FP \times TN + FN + FN + TN \times FP + TP}{N^2}$$

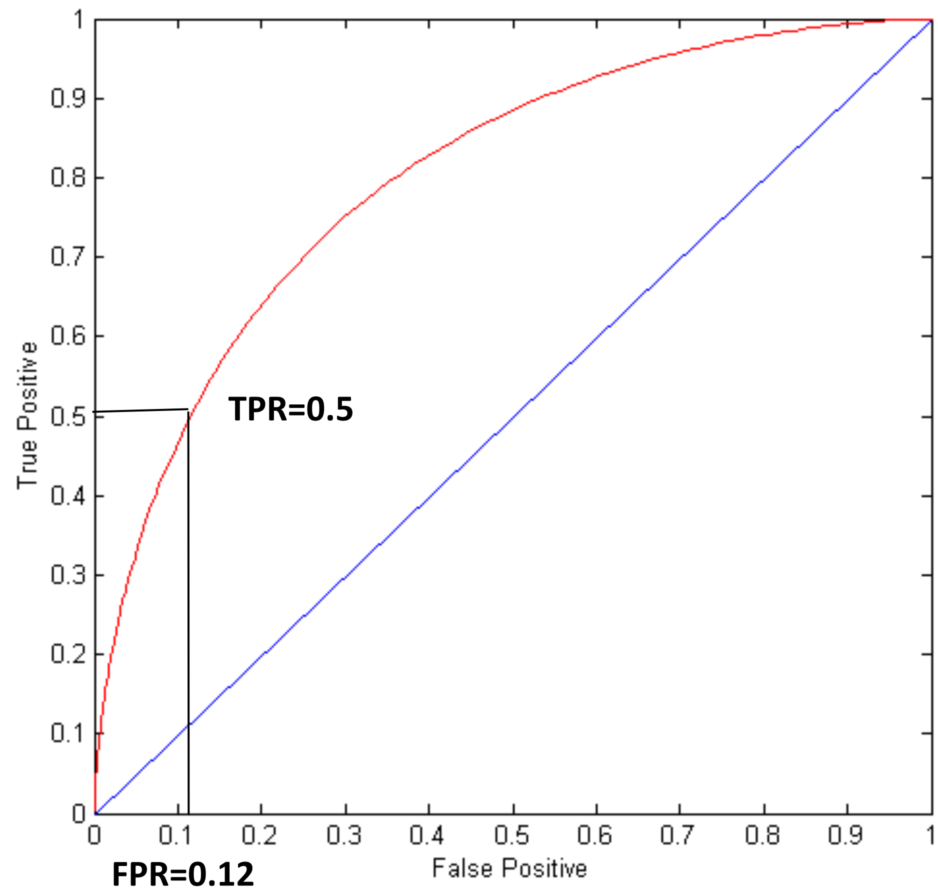
ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals to characterize the trade-off between positive hits and false alarms.
- Works only for binary classification (two-class problems). The classes are called the positive and the other is the negative class.
- ROC curve plots TPR (true positive rate) on the y-axis against FPR (false positive rate) on the x-axis.
- Performance of each classifier represented as a point. Changing the threshold of the algorithm, sample distribution or cost matrix changes the location of the point and forms a curve.

ROC Curve

- Example with 1-dimensional data set containing 2 classes (positive and negative)
- Any points located at x > t is classified as positive





At threshold t:

TPR=0.5, FNR=0.5, **FPR=0.12,** FNR=0.88

- Move t to get the other points on the ROC curve.

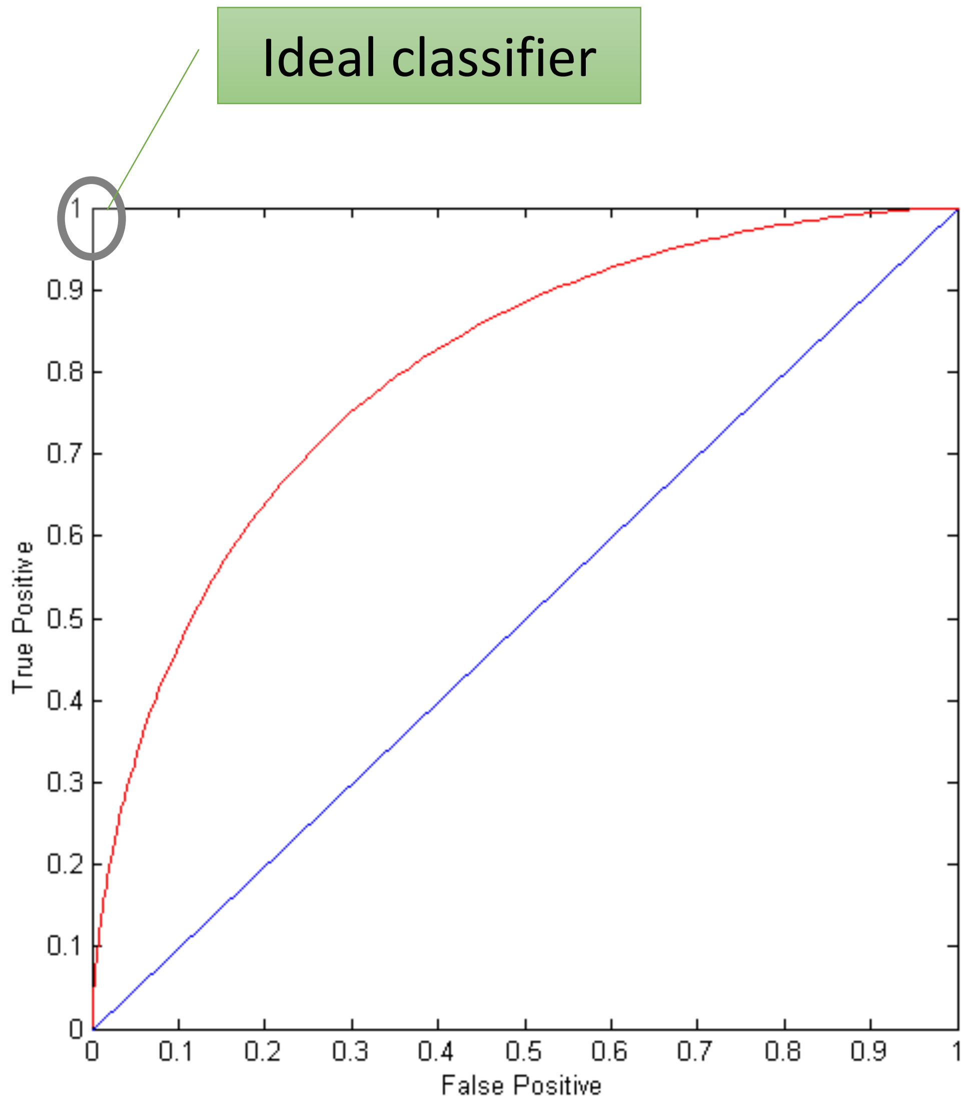
ROC Curve

(TPR,FPR):

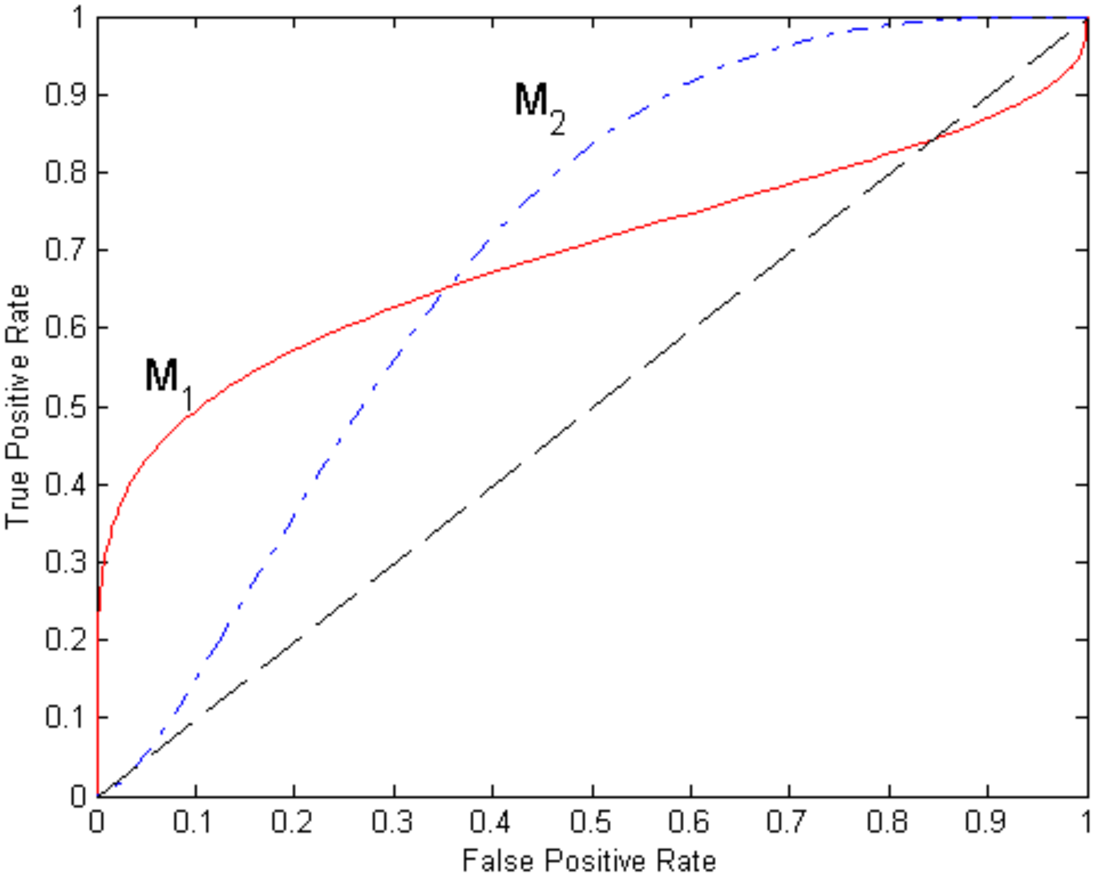
- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

Diagonal line:

- Random guessing
- Below diagonal line: prediction is opposite of the true class



Using ROC for Model Comparison



No model consistently outperform the other

- M1 is better for small FPR
- M2 is better for large FPR

Area Under the ROC curve (AUC)

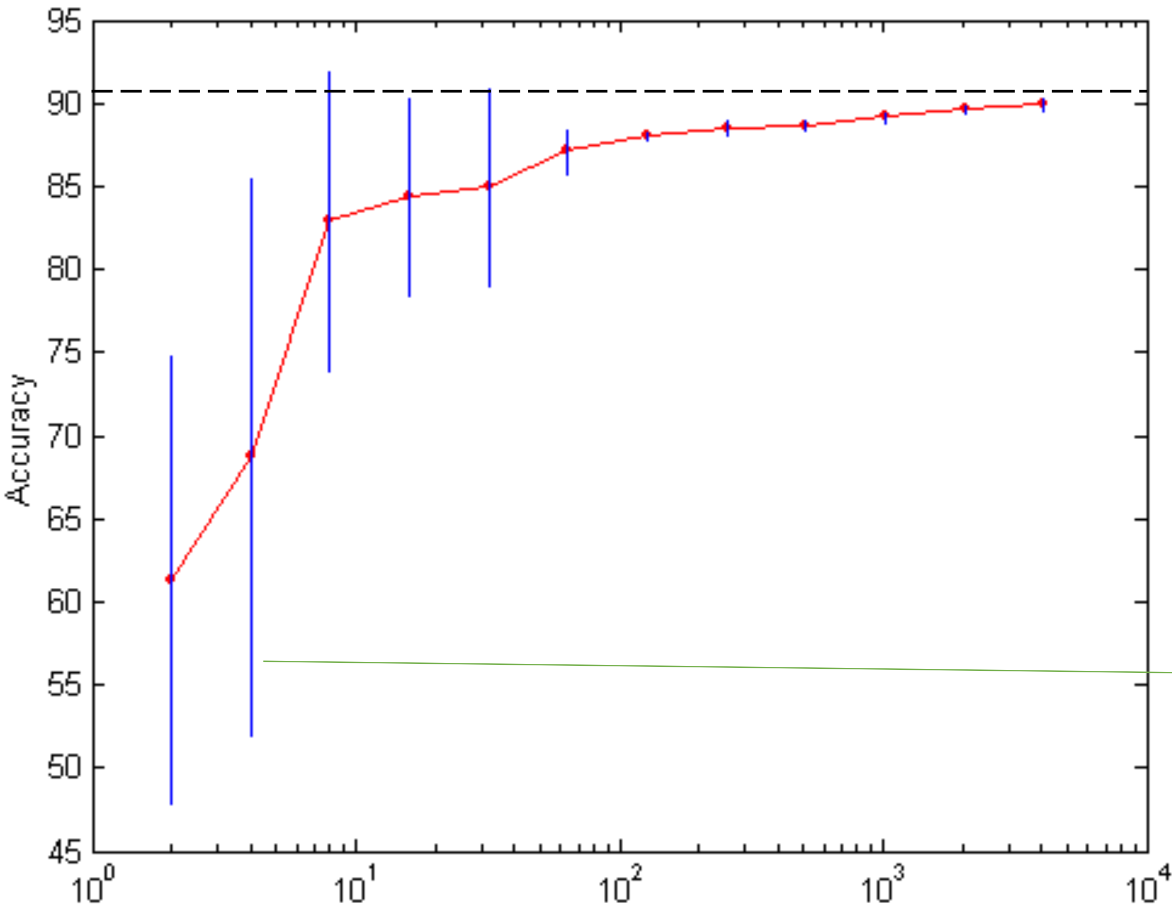
- Ideal:
  - AUC = 1
- Random guess:
  - AUC = 0.5

Topics

- Introduction
- Decision Trees
  - —Overview
  - —Tree Induction
  - —Overfitting and other Practical Issues
- **Model Evaluation**
  - —Metrics for Performance Evaluation
  - —**Methods to Obtain Reliable Estimates**
  - —Model Comparison (Relative Performance)
- Feature Selection
- Class Imbalance

Learning Curve

Accuracy and variance between runs depend on the size of the training data.



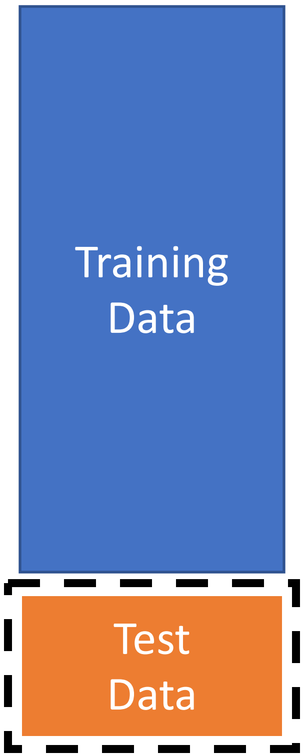
Training data (log scale)

Learning curve shows how accuracy on unseen examples changes with varying training sample size

Variance for several runs

Training and Test Data

- Separate data into a set to train and a set to test.
- **Holdout testing/Random splits**: Split the data randomly into, e.g., 80% training and 20% testing.
- **k-fold cross validation**: Use training & validation data better
  - split the training & validation data randomly into k folds.
  - For k rounds hold 1 fold back for testing and use the remaining k-1 folds for training.
  - Use the average the error/accuracy as a better estimate.
  - Some algorithms/tools do that internally.
- **LOOCV** (leave-one-out cross validation): k = n used if very little data is available.



Very important: the algorithm can never look at the test set during learning!

Training and Testing with Hyperparameters

Hyperparameters: Many algorithms allow choices for learning. E.g.,

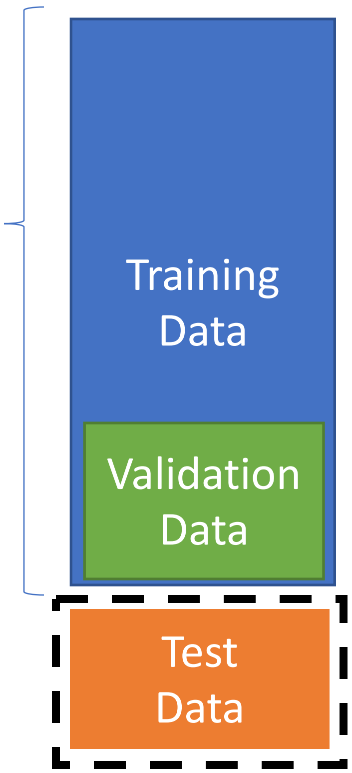
- maximal decision tree depth
- selected features

1. **Train:** Learn models on the **training data** (without the validation data) using different hyperparameters.
   - A grid of possible hyperparameter combinations
   - greedy search
2. **Model Selection:** Evaluate the models using the **validation data** and choose the hyperparameters with the best accuracy. Rebuild the model using all the training data.
3. **Test** the final model using the **test data**.

How to Split the Dataset

- **Random splits**: Split the data randomly in 60% training, 20% validation, and 20% testing.

- **k-fold cross validation**: Use training & validation data better
  - split the training & validation data randomly into k folds.
  - For k rounds hold 1 fold back for testing and use the remaining k-1 folds for training.
  - Use the average the error/accuracy as a better estimate.
  - Some algorithms/tools do that internally.



Confidence Interval for Accuracy

- Each prediction can be regarded as a **Bernoulli trial**: A Bernoulli trial (a biased coin toss) has 2 possible outcomes: heads (correct) or tails (wrong)

  We use p for the true chance that prediction is correct (= true accuracy).

- Predictions for a test set of size N are a collection of N Bernoulli trials. The number of correct predictions x has a **Binomial distribution**:

$$X \sim Binomial(N, p)$$

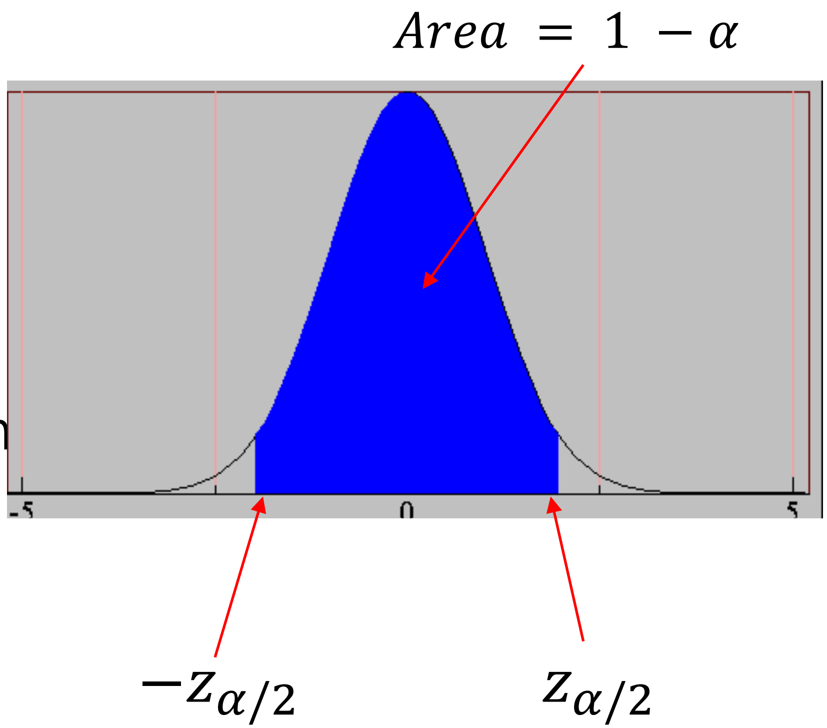
  Example: Toss a fair coin 50 times, how many heads would turn up?
  Expected number of heads $E[X] = Np = 50 \times 0.5 = 25$

- Given we observe x correct predictions (an observed accuracy of $\hat{p} = x/N$):

  Can we give bounds for the true accuracy of model p?

Confidence Interval for Accuracy

For large test sets ($N > 30$) we can approximate the Binomial distribution by a Normal distribution



$$X \sim Normal(Np, Np(1-p))$$

Confidence Interval for $p = X/N$ (Wald Method):

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$$

Confidence Interval for Accuracy

Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:

- N = 100, acc = 0.8
- Let $1-\alpha = 0.95$ (95% confidence)
- From probability table, $z_{\alpha/2} = 1.96$

$1-\alpha/2$	$z_{\alpha/2}$
0.99	2.58
0.98	2.33
0.95	1.96
0.90	1.65

Table or R $qnorm(1-\alpha/2)$

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$$

N	50	100	500	1000	5000
p(lower)	0.689	0.722	0.765	0.775	0.789
p(upper)	0.911	0.878	0.835	0.825	0.811

Topics

- Introduction
- Decision Trees
  - —Overview
  - —Tree Induction
  - —Overfitting and other Practical Issues
- **Model Evaluation**
  - —Metrics for Performance Evaluation
  - —Methods to Obtain Reliable Estimates
  - —**Model Comparison (Relative Performance)**
- Feature Selection
- Class Imbalance

Comparing Performance between 2 Models

Given two models, say M_1 and $M2$, which is better?

For large test sets ($N > 30$) we have approximately:
$acc_1 \sim Normal(p_1, Np_1(1-p_1))$
$acc_2 \sim Normal(p_2, Np_2(1-p_2))$

Perform a paired t-test with:

H0: There is no difference in accuracy between the models.

H1: There is a difference.

Comparing multiple models: You need to correct for multiple comparisons! For example using Bonferroni correction.

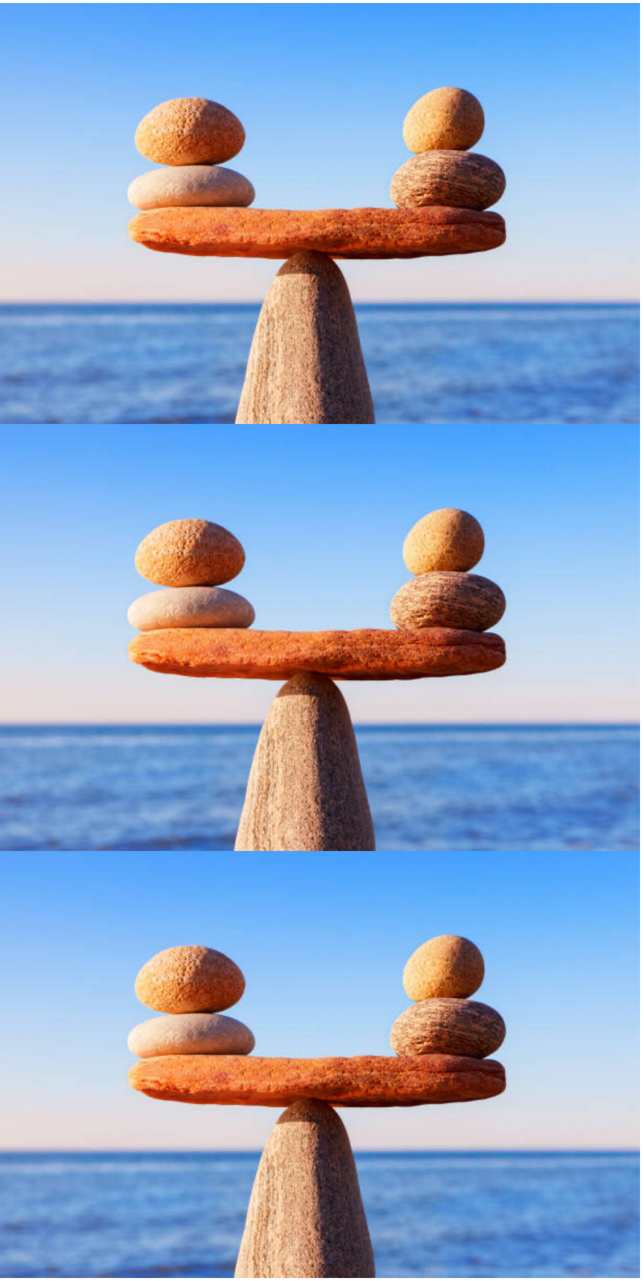
Topics

- Introduction
- Decision Trees
  - Overview
  - Tree Induction
  - Overfitting and other Practical Issues
- Model Evaluation
  - Metrics for Performance Evaluation
  - Methods to Obtain Reliable Estimates
  - Model Comparison (Relative Performance)
- **Feature Selection**
- Class Imbalance

Feature Selection

What features should be used in the model?

Univariate feature importance score	Feature subset selection
• measures how related each feature is to the class variable. • E.g., chi-squared statistic, information gain.	• tries to find the best set of features. • Often uses a black box approach where different subsets are evaluated using a greedy search strategy.



Topics

- Introduction
- Decision Trees
  - —Overview
  - —Tree Induction
  - —Overfitting and other Practical Issues
- Model Evaluation
  - —Metrics for Performance Evaluation
  - —Methods to Obtain Reliable Estimates
  - —Model Comparison (Relative Performance)
- Feature Selection
- **Class Imbalance**

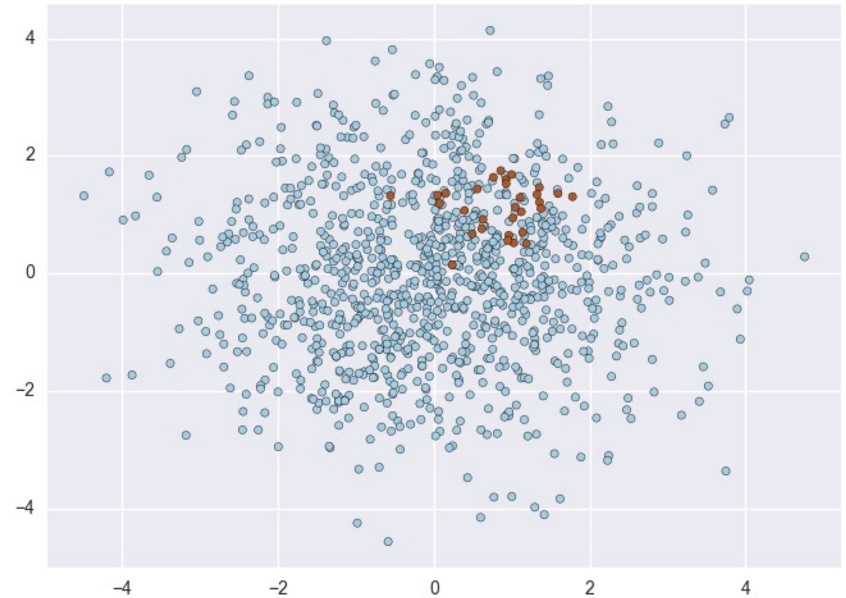
Class Imbalance Problem

Consider a 2-class problem

- Number of Class 0 examples = 9990
- Number of Class 1 examples = 10

A simple model:

- Always predict Class 0
- accuracy = 9990/10000 = 99.9 %
- error = 0.1%



Issues:

1. Evaluation: accuracy is misleading.
2. Learning: Most classifiers try to optimize accuracy/error. **These classifiers will not learn how to find examples of Class 1!**

Class Imbalance Problem: Evaluation

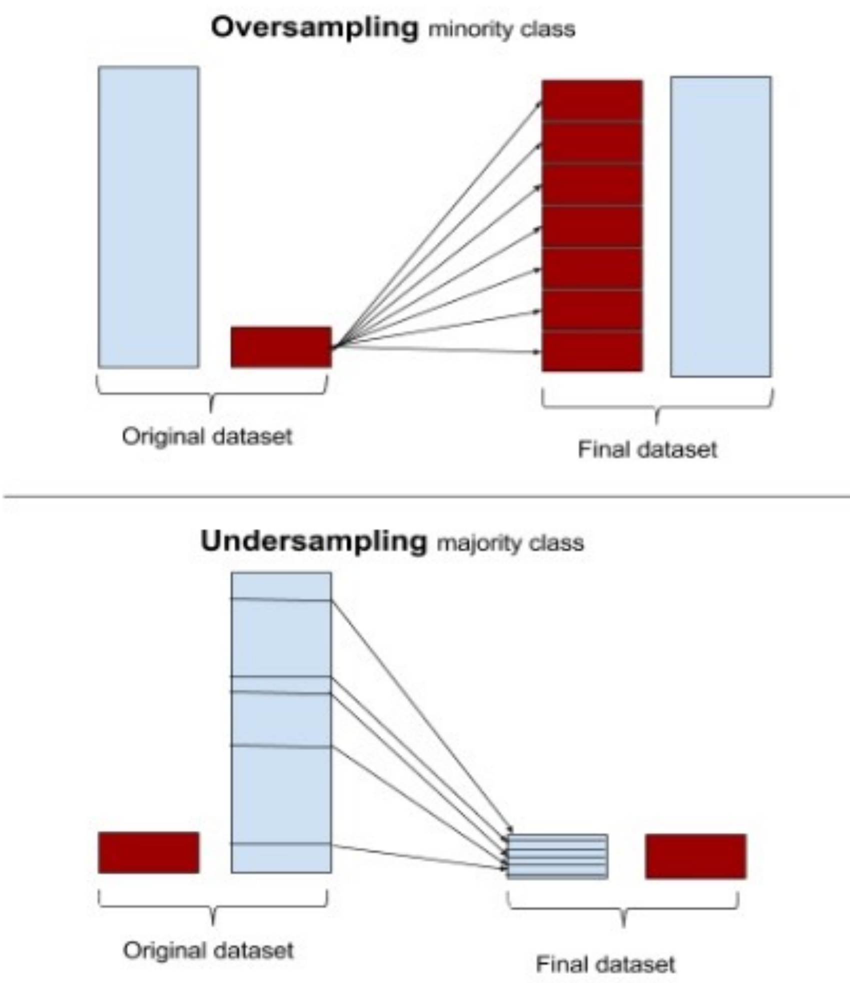
Do not use accuracy to evaluate for problems with strong class imbalance!

Use instead:

- ROC curves and AUC (area under the curve)
- Precision/Recall plots or the F1 Score
- Cohen's Kappa
- Misclassification cost

Class Imbalance Problem: Learning

- **Do nothing**. Sometimes you get lucky!
- **Balance the data set**: Down-sample the majority class and/or up-sample the minority class (use sampling with replacement). Synthesize new examples with SMOTE.
  This will artificially increase the error for a mistake in the minority class.
- Use **algorithms** that can deal with class imbalance (see next slide).
- Throw away minority examples and switch to an **anomaly detection** framework.



Class Imbalance Problem: Learning

Algorithms that can deal with class imbalance:

- Use a classifier that **predict a probability** and lower the decision threshold (from the default of .5). We can estimate probabilities for decision trees using the positive and negative training examples in each leaf node.
- Use a **cost-sensitive classifier** that considers a cost matrix (not too many are available).
- Use boosting techniques like **AdaBoost**.

Conclusion

- Classification is **supervised learning** with the goal to find a model that generalizes well.
- **Generalization error** can be estimated using test sets/cross-validation.
- Model evaluation and comparison needs to take **model complexity** into account.
- Accuracy is problematic for **imbalanced data sets**.