Classification

Basic Concepts, Decision Trees, and Model Evaluation

Classification definition

- Given a collection of samples (*training set*)
  - Each sample contains a set of *attributes.*
  - Each sample also has a discrete *class label*.
- Learn a *model* that predicts class label as a function of the values of the attributes.
- Goal: model should assign class labels to previously unseen samples as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

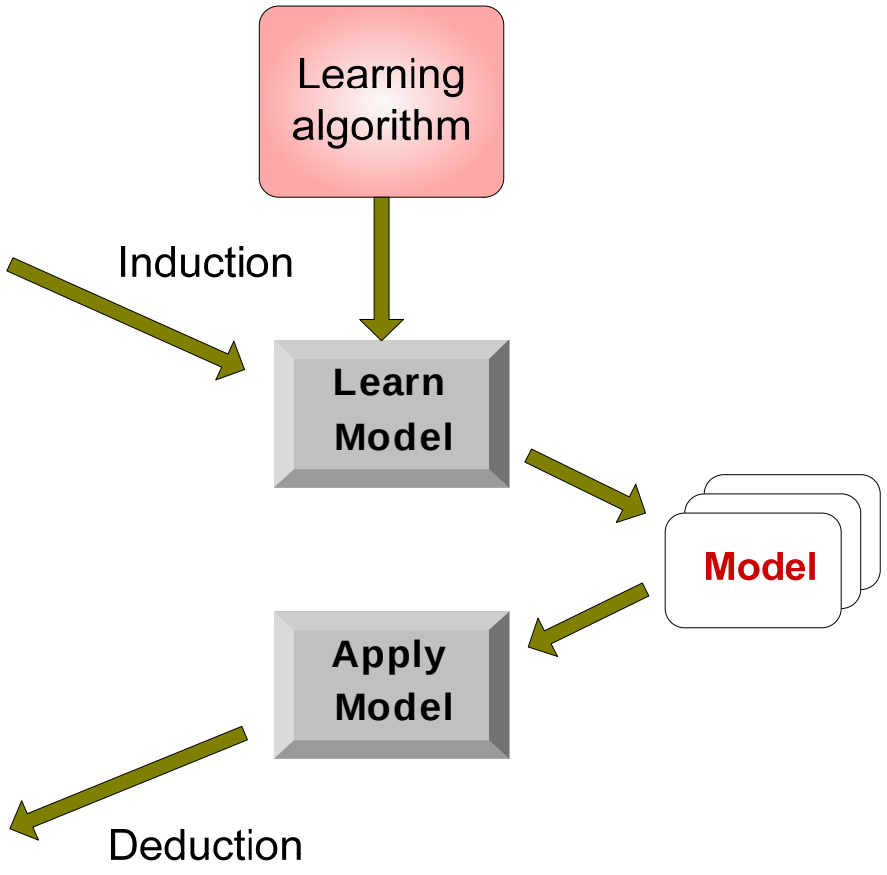
Stages in a classification task

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

Training 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	?

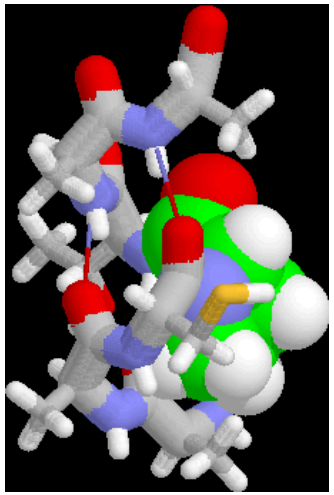
Test Set



Examples of classification tasks

- Two classes
  - Predicting tumor cells as benign or malignant
  - Classifying credit card transactions as legitimate or fraudulent

- Multiple classes
  - Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil
  - Categorizing news stories as finance, weather, entertainment, sports, etc



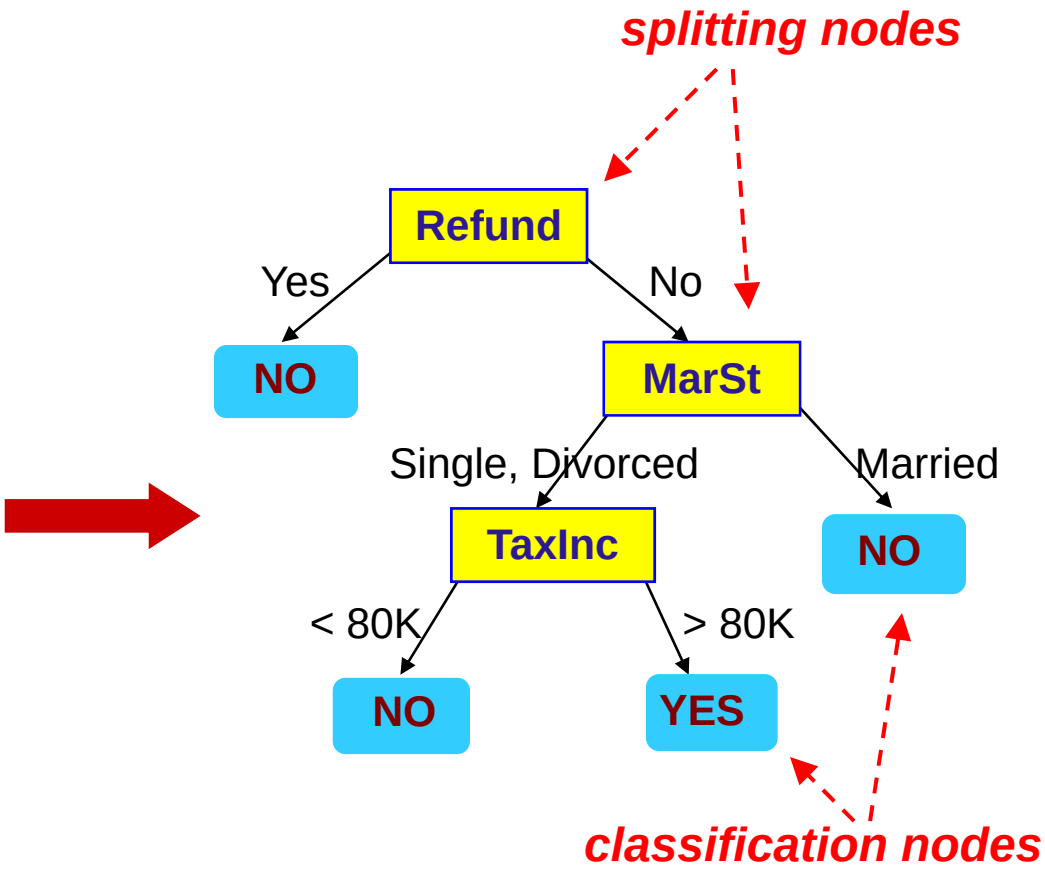
Classification techniques

- Decision trees
- Rule-based methods
- Logistic regression
- Discriminant analysis
- k-Nearest neighbor (instance-based learning)
- Naïve Bayes
- Neural networks
- Support vector machines
- Bayesian belief networks

Example of a decision tree

	nominal	nominal	ratio	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

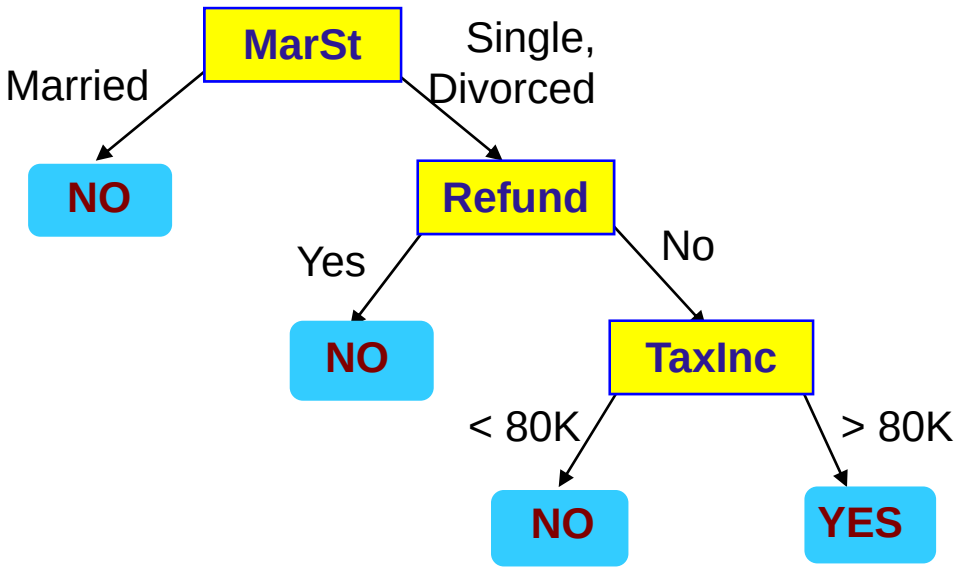
training data



model: decision tree

Another example of decision tree

	nominal	nominal	ratio	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



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

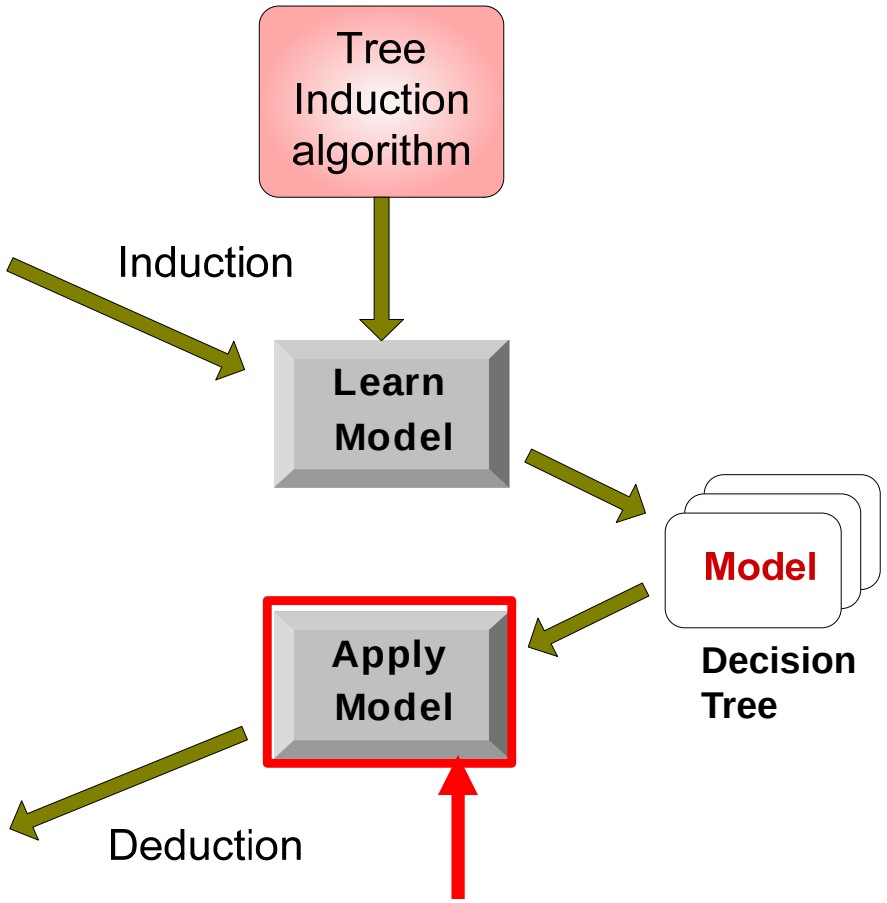
Decision tree classification task

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

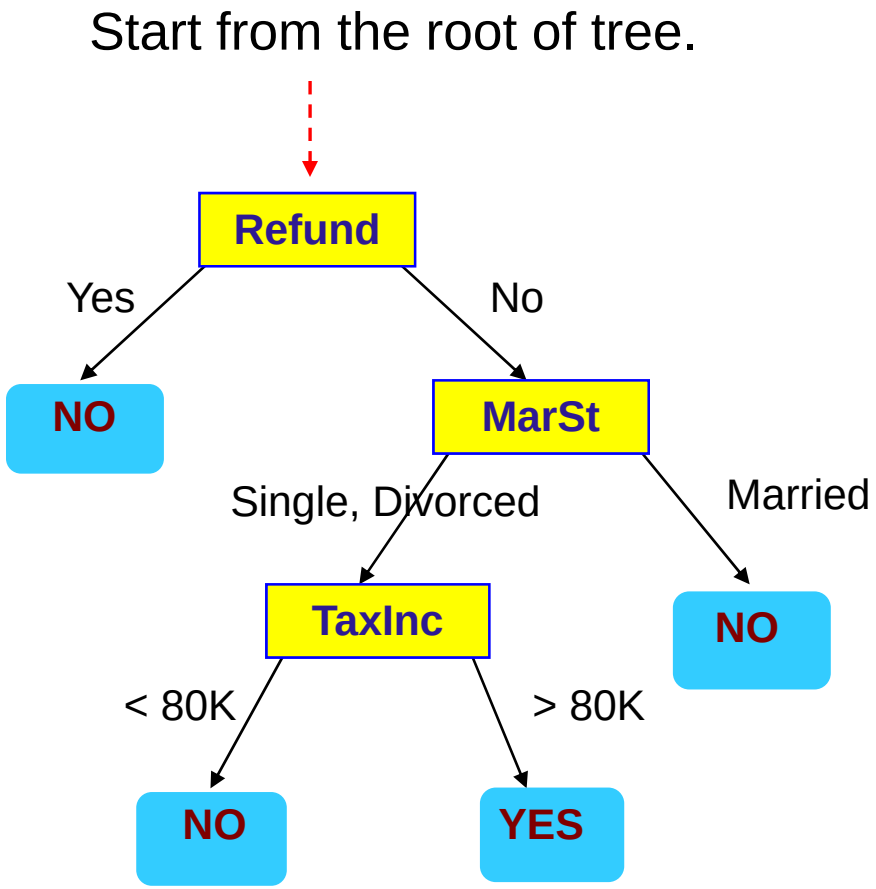
Training Set

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	?

Test Set



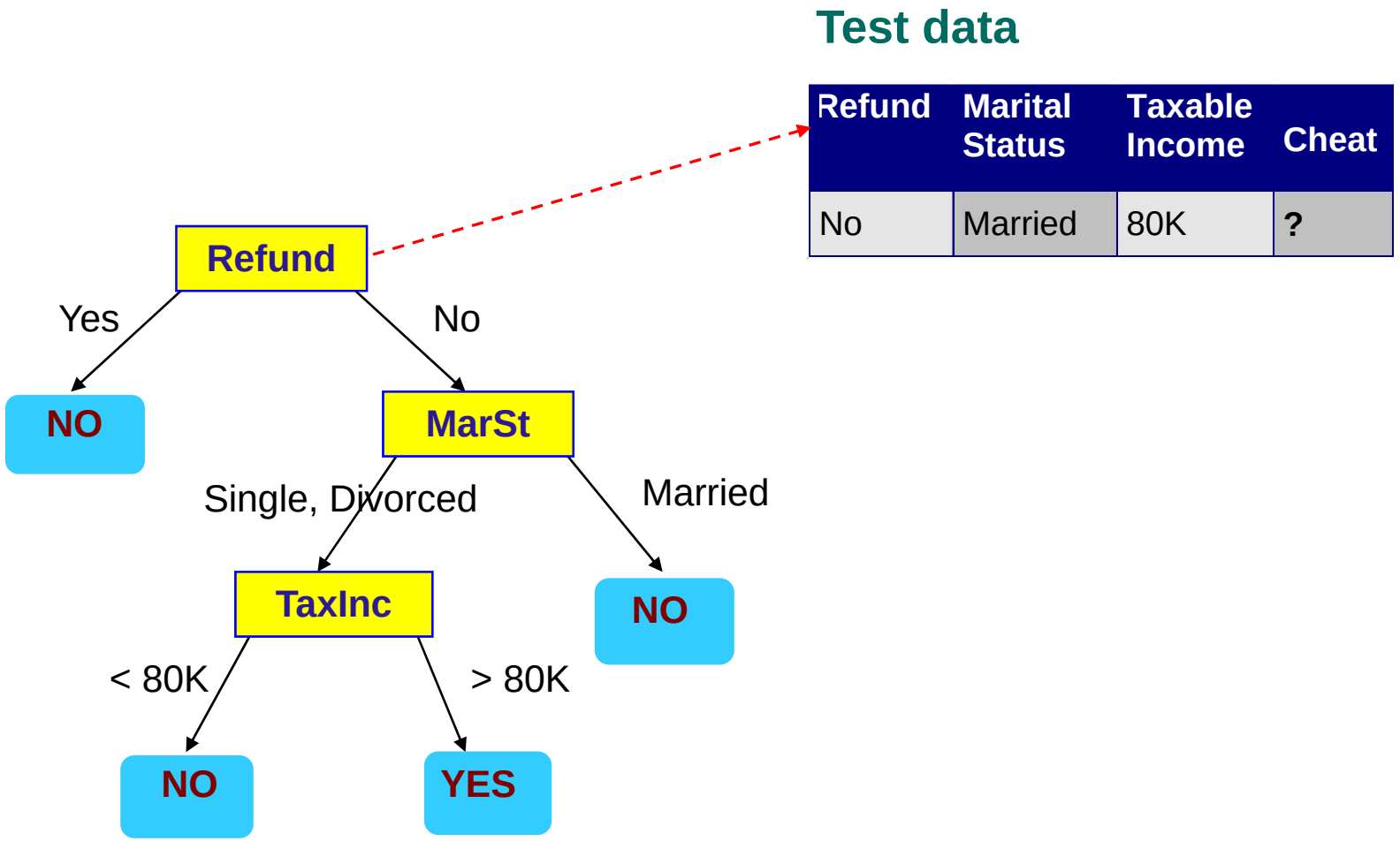
Apply model to test data



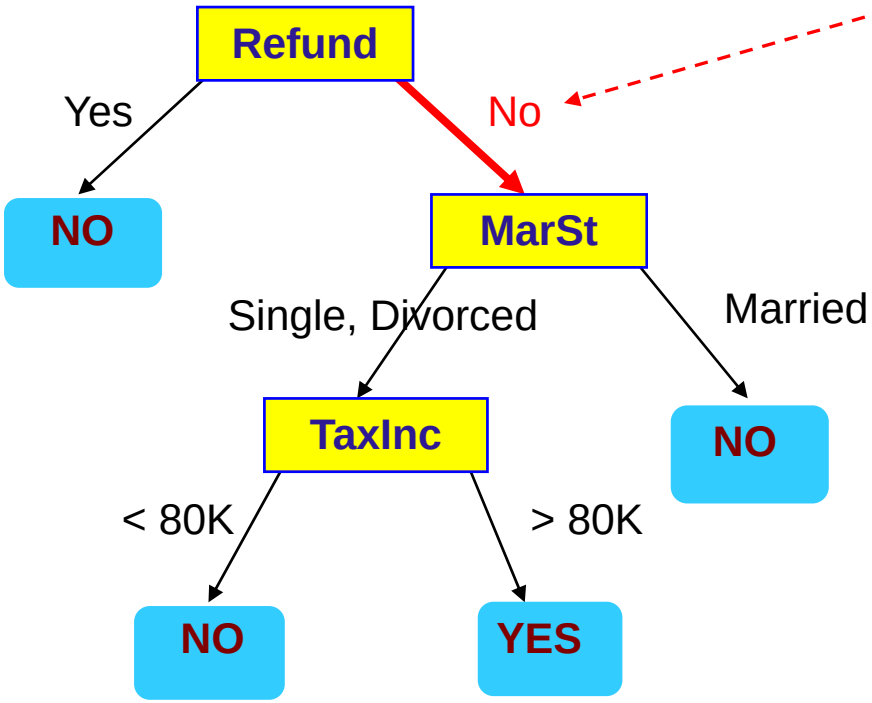
Test data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply model to test data



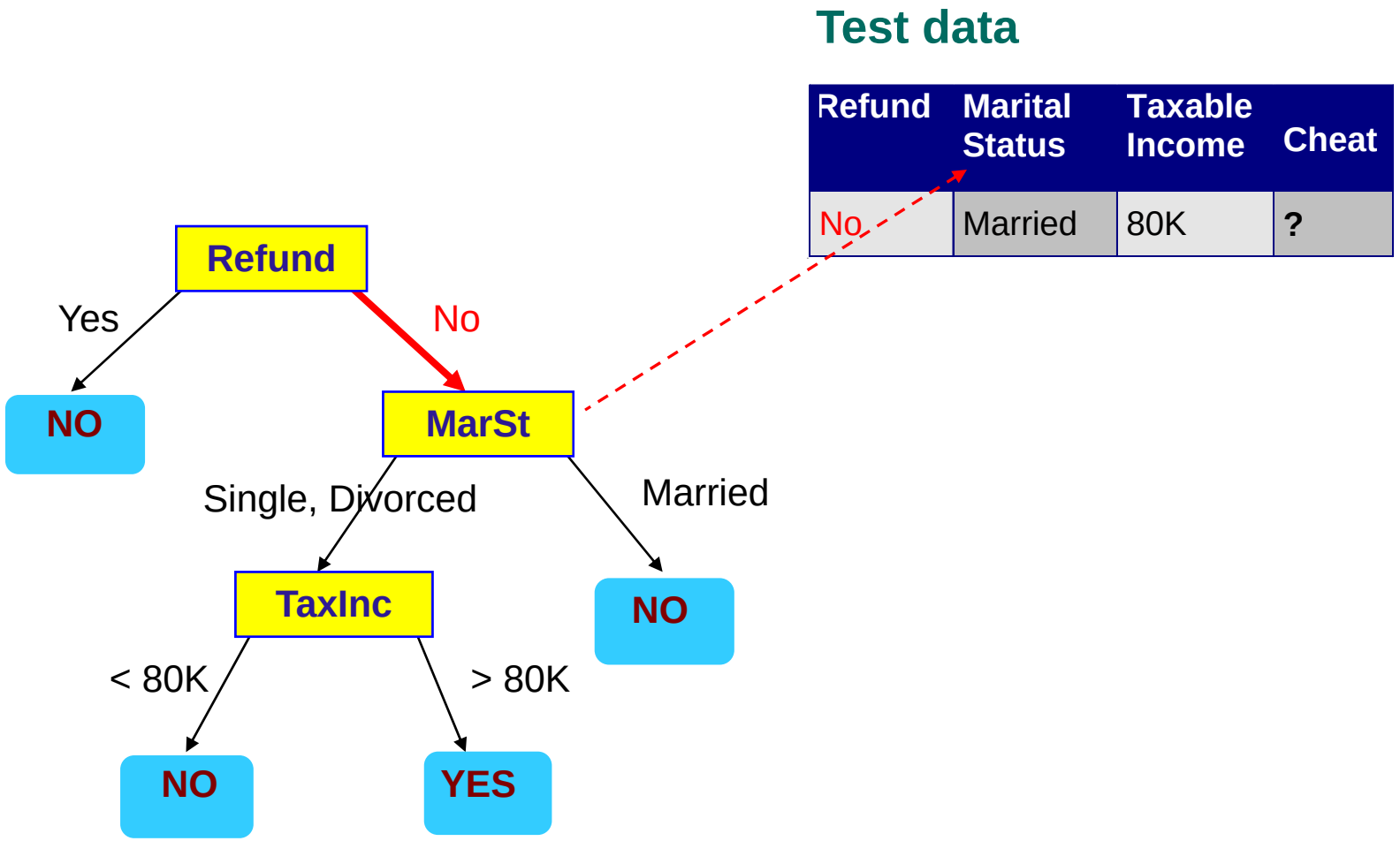
Apply model to test data



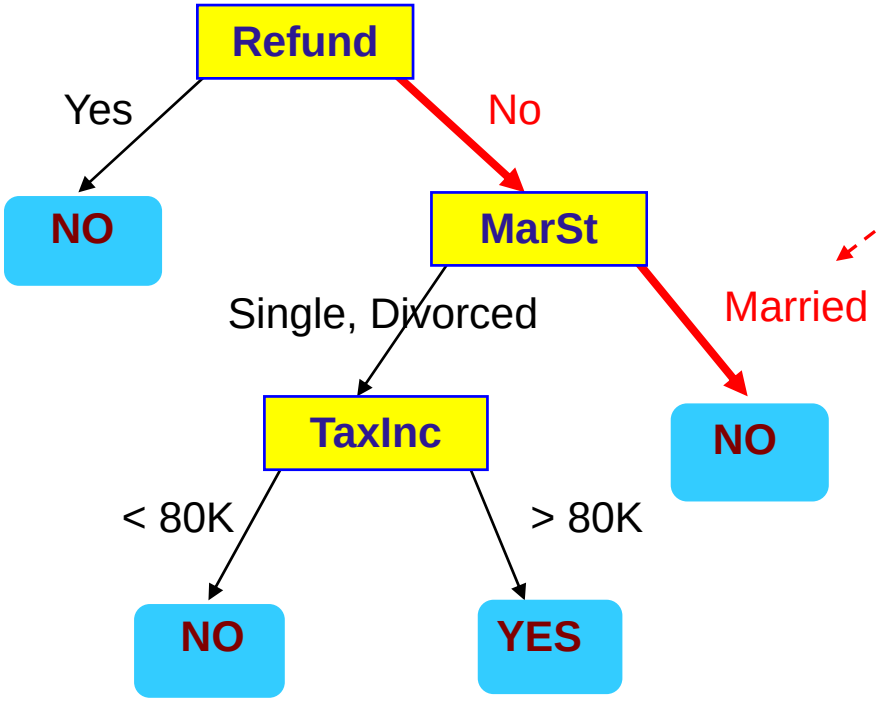
Test data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply model to test data



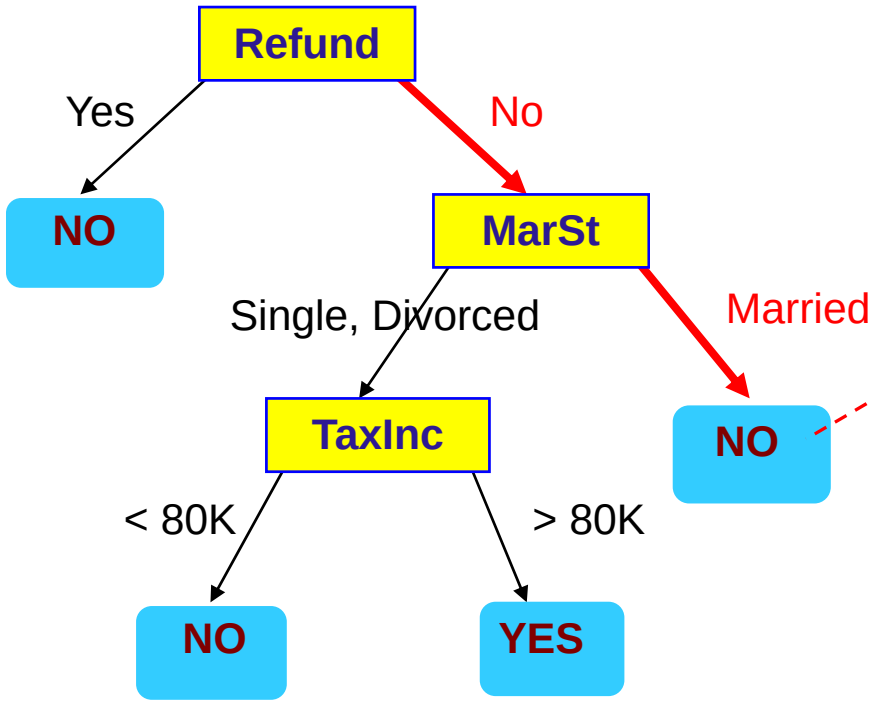
Apply model to test data



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”

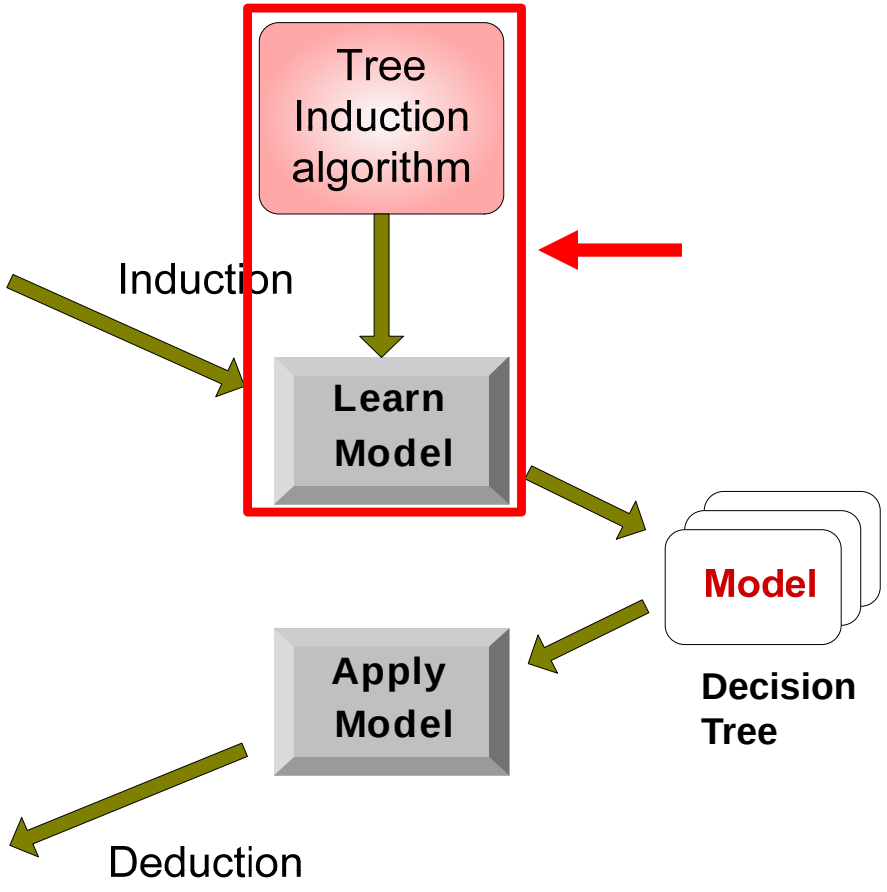
Decision tree classification task

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

Training 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	?

Test Set



Decision tree induction

- Many algorithms:
  - Hunt's algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ, SPRINT

General structure of Hunt's algorithm

- Hunt's algorithm is recursive.
- General procedure:

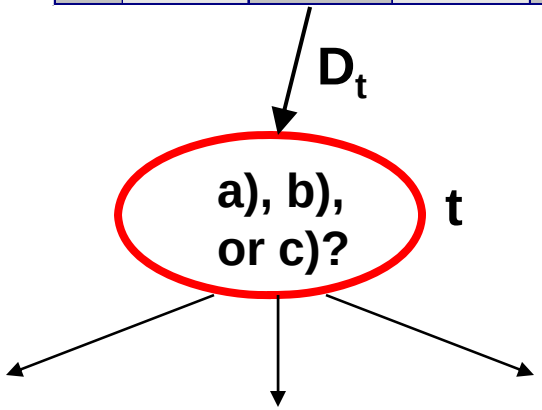
  Let D_t be the set of training records that reach a node t.

  a) If all records in D_t belong to the same class y_t, then t is a leaf node labeled as y_t.

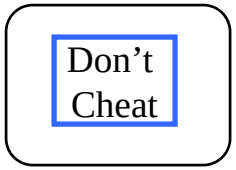
  b) If D_t is an empty set, then t is a leaf node labeled by the default class, y_d.

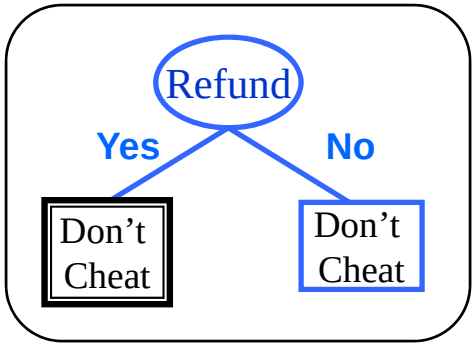
  c) If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets, then apply the procedure to each subset.

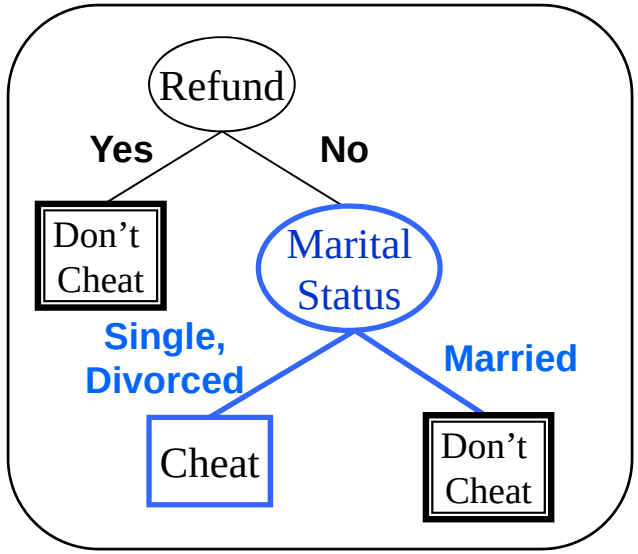
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



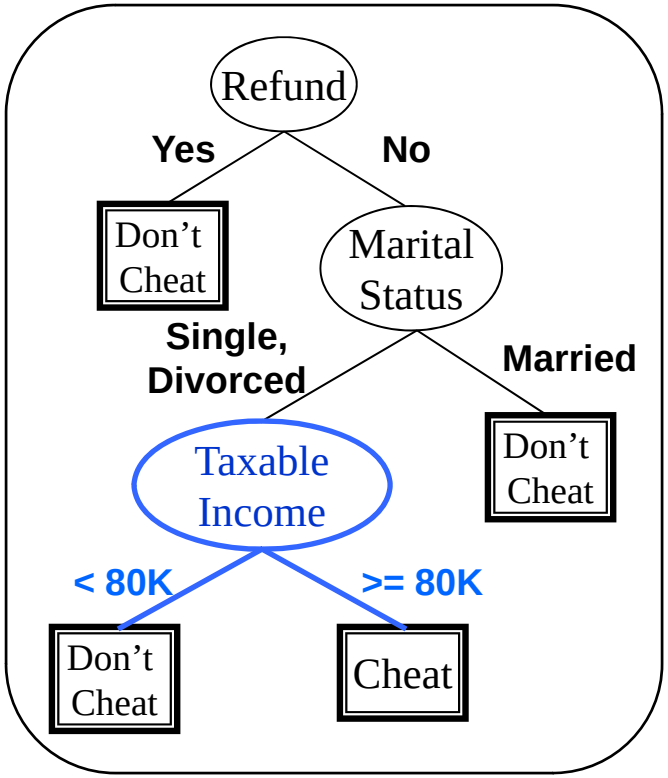
Applying Hunt's algorithm







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



Tree induction

- Greedy strategy
  - Split the records at each node based on an attribute test that optimizes some chosen criterion.

- Issues
  - Determine how to split the records
    - How to specify structure of split?
    - What is best attribute / attribute value for splitting?
  - Determine when to stop splitting

Tree induction

- Greedy strategy
  - Split the records at each node based on an attribute test that optimizes some chosen criterion.

- Issues
  - Determine how to split the records
    - How to specify structure of split?
    - What is best attribute / attribute value for splitting?
  - Determine when to stop splitting

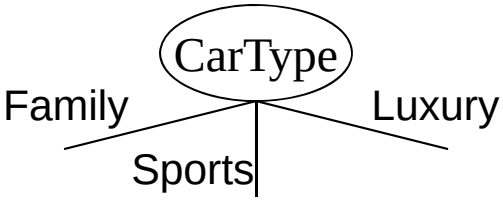
Specifying structure of split

- Depends on attribute type
  - Nominal
  - Ordinal
  - Continuous (interval or ratio)

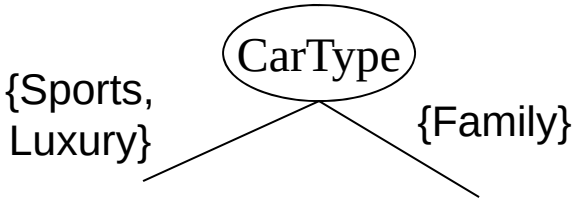
- Depends on number of ways to split
  - Binary (two-way) split
  - Multi-way split

Splitting based on nominal attributes

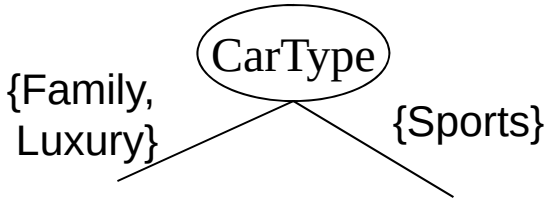
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets. Need to find optimal partitioning.

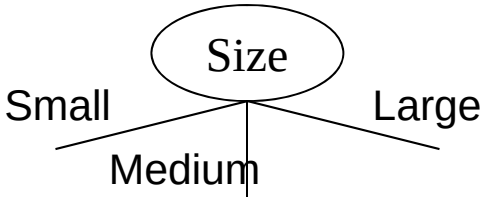


OR

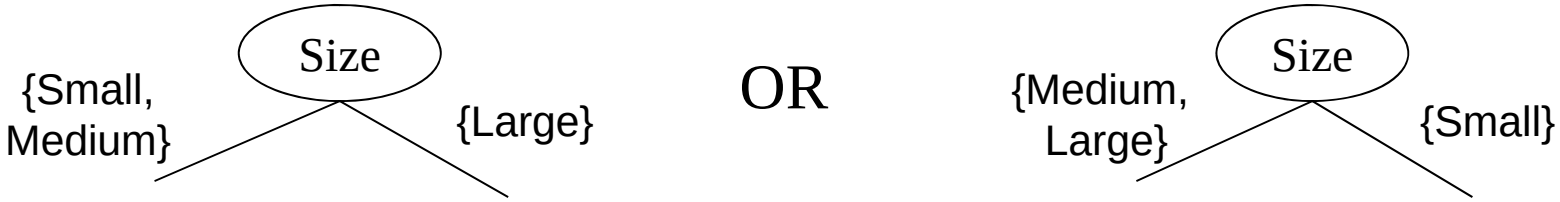


Splitting based on ordinal attributes

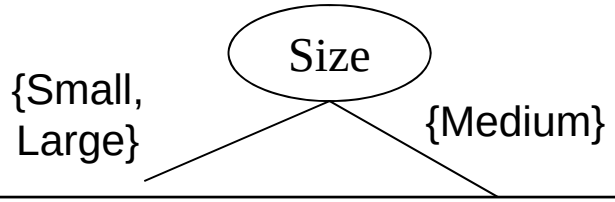
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets. Need to find optimal partitioning.



- What about this split?

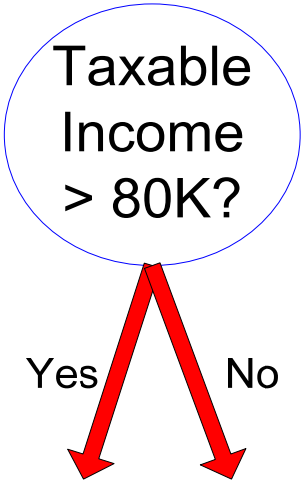


Splitting based on continuous attributes

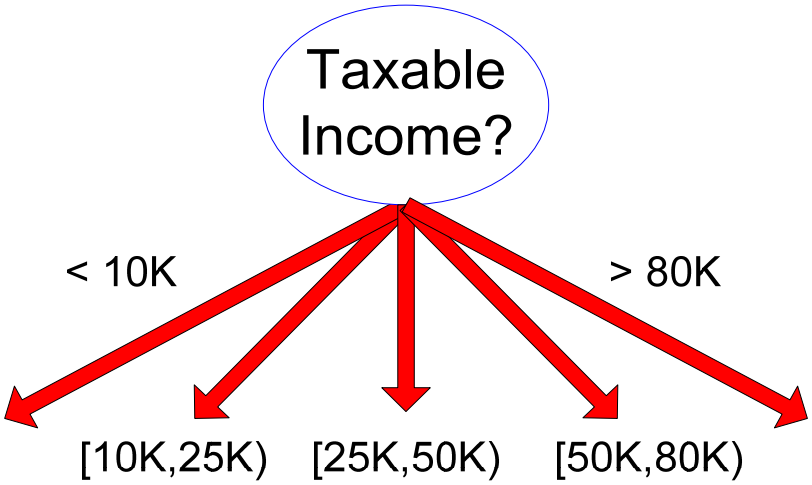
- Different ways of handling
  - Discretization to form an ordinal attribute
    - static – discretize once at the beginning
    - dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Threshold decision: $(A < v)$ or $(A \geq v)$
    - consider all possible split points v and find the one that gives the best split
    - can be more compute intensive

Splitting based on continuous attributes

- Splitting based on threshold decision



(i) Binary split



(ii) Multi-way split

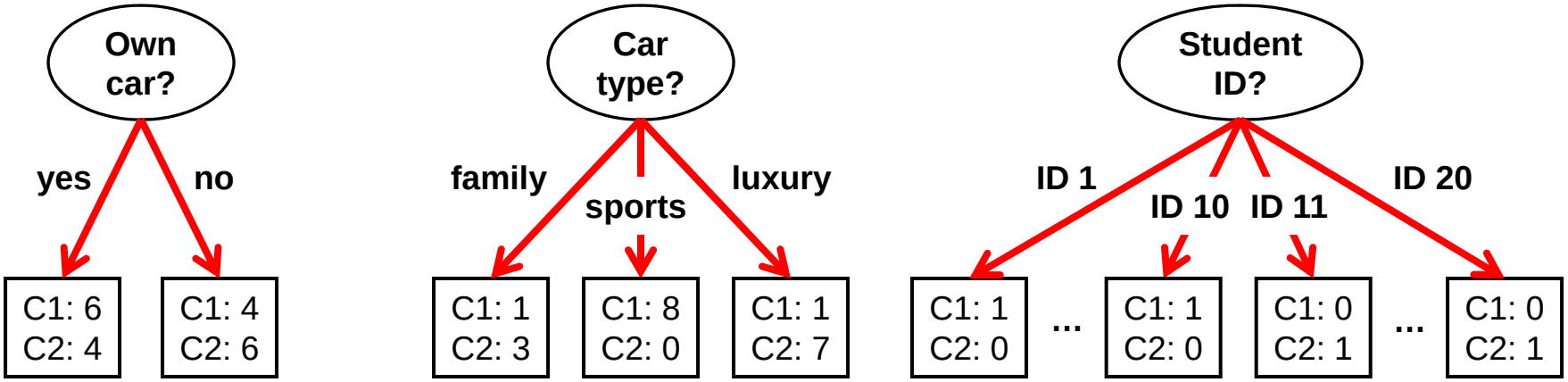
Tree induction

- Greedy strategy
  - Split the records at each node based on an attribute test that optimizes some chosen criterion.

- Issues
  - Determine how to split the records
    - How to specify structure of split?
    - What is best attribute / attribute value for splitting?
  - Determine when to stop splitting

Determining the best split

Before splitting: 10 records of class 1 (C1)
10 records of class 2 (C2)



Which attribute gives the best split?

Determining the best split

- Greedy approach:

  Nodes with homogeneous class distribution are preferred.

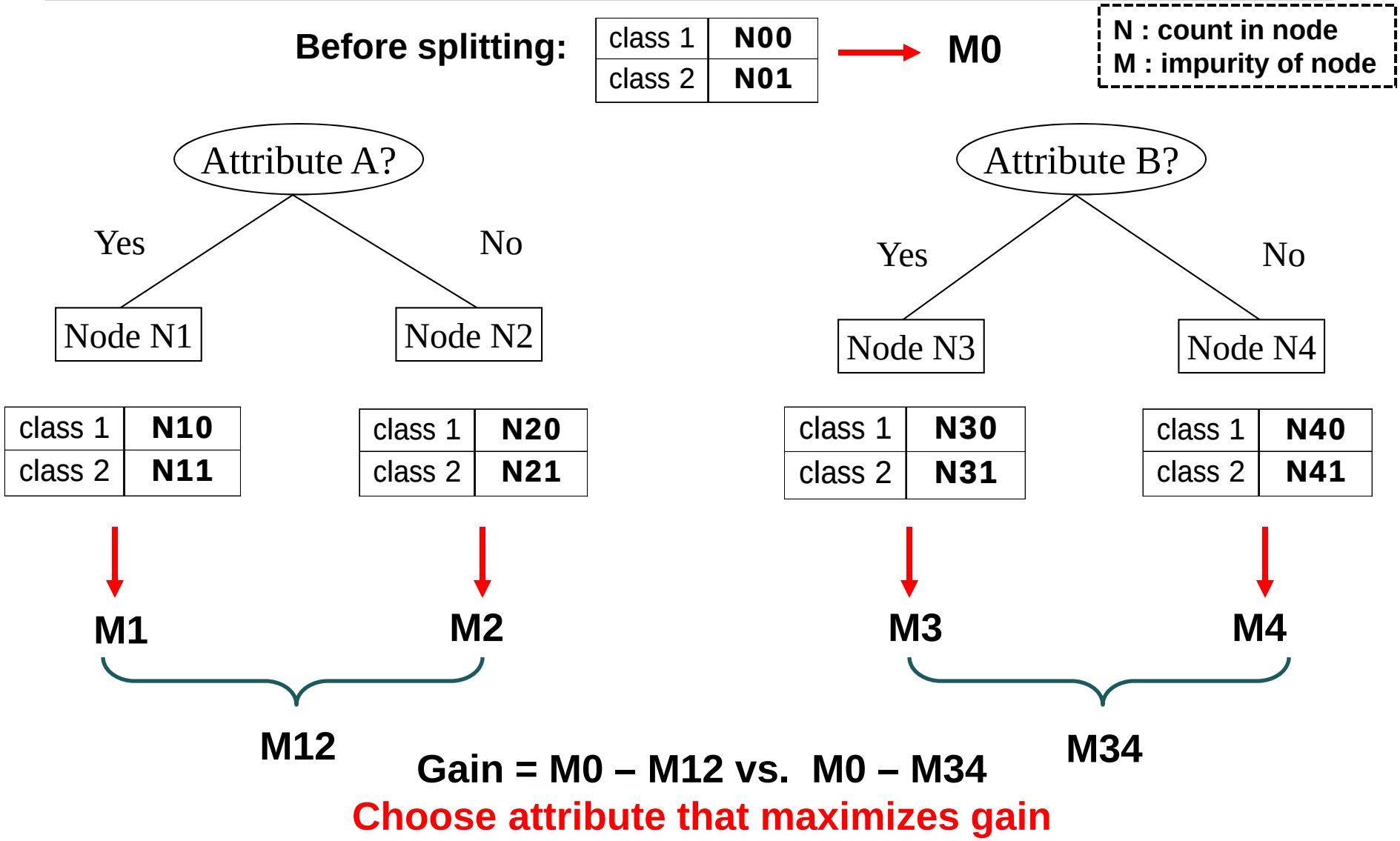
- Need a measure of node impurity:

class 1: 5 class 2: 5	class 1: 9 class 2: 1
Non-homogeneous, **high degree of impurity**	**Homogeneous,** **low degree of impurity**

Measures of node impurity

- Gini index
- Entropy
- Misclassification error

Using a measure of impurity to determine best split



Measure of impurity: Gini index

- Gini index for a given node t :

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

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

  - Maximum $(1 - 1 / n_c)$ when records are equally distributed among all classes, implying least amount of information (n_c = number of classes).
  - Minimum (0.0) when all records belong to one class, implying most amount of information.

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 of computing Gini index

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

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

Splitting based on Gini index

- Used in CART, SLIQ, SPRINT.
- When a node t is split into k partitions (child nodes), the quality of split is computed as,

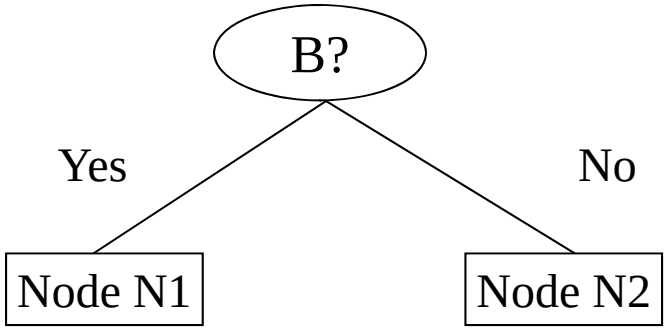
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where n_i = number of records at child node i

n = number of records at parent node t

Computing Gini index: binary attributes

- Splits into two partitions
- Effect of weighting partitions: favors larger and purer partitions



	Parent
C1	6
C2	6
Gini = 0.500	

Gini(N1)
= 1 – (5/7)^2 – (2/7)^2
= 0.408

Gini(N2)
= 1 – (1/5)^2 – (4/5)^2
= 0.320

	N1	N2
C1	5	1
C2	2	4
Gini = 0.371		

Gini(children)
= 7/12 * 0.408 +
5/12 * 0.320
= 0.371

Computing Gini index: categorical attributes

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of attribute values)

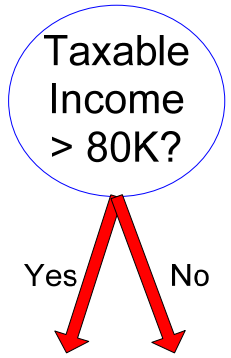
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Computing Gini index: continuous attributes

- Make binary split based on a threshold (splitting) value of attribute
- Number of possible splitting values = (number of distinct values attribute has at that node) - 1
- Each splitting value *v* has a count matrix associated with it
  - Class counts in each of the partitions, A < *v* and A ≥ *v*
- Simple method to choose best *v*
  - For each *v*, scan the attribute values at the node to gather count matrix, then compute its Gini index.
  - Computationally inefficient! Repetition of work.

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



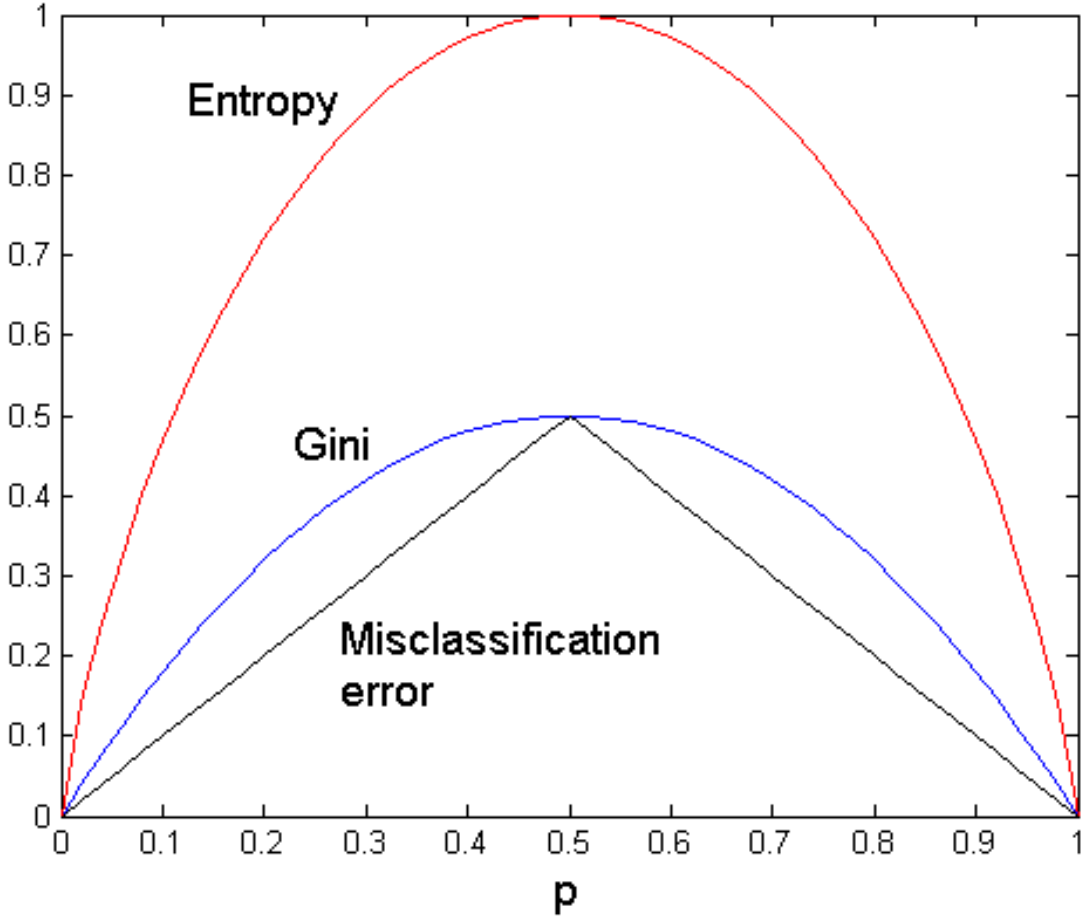
Computing Gini index: continuous attributes

- For efficient computation, do following for each (continuous) attribute:
  - Sort attribute values.
  - Linearly scan these values, each time updating the count matrix and computing Gini index.
  - Choose split position that has minimum Gini index.

Cheat	No		No		No		Yes		Yes		Yes		No		No		No		No	
Taxable Income																				
sorted values →	60		70		75		85		90		95		100		120		125		220	
split positions →	55		65		72		80		87		92		97		110		122		172		230	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

Comparison among splitting criteria

For a two-class problem:



Tree induction

- Greedy strategy
  - Split the records at each node based on an attribute test that optimizes some chosen criterion.

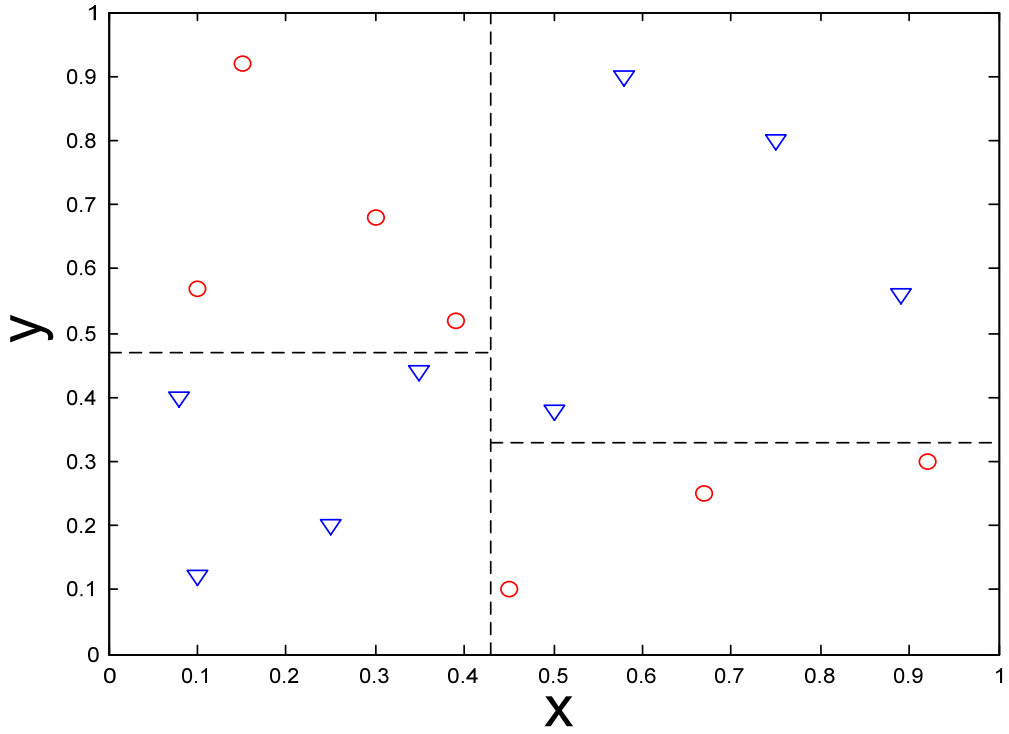
- Issues
  - Determine how to split the records
    - How to specify structure of split?
    - What is best attribute / attribute value for splitting?
  - Determine when to stop splitting

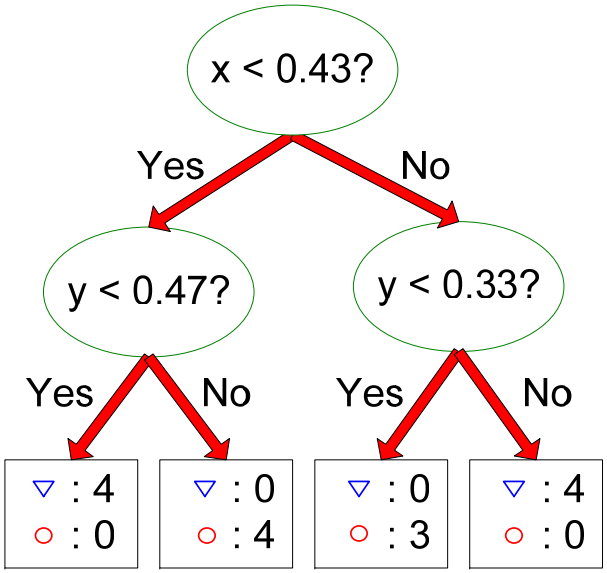
Stopping criteria for tree induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have identical (or very similar) attribute values
  - No remaining basis for splitting
- Early termination

Can also prune tree post-induction

Decision trees: decision boundary





- Border between two neighboring regions of different classes is known as decision boundary.
- In decision trees, decision boundary segments are always parallel to attribute axes, because test condition involves one attribute at a time.

Classification with decision trees

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy comparable to other classification techniques for many simple data sets
- Disadvantages:
  - Easy to overfit
  - Decision boundary restricted to being parallel to attribute axes

MATLAB interlude

matlab_demo_04.m

Part A

Producing useful models: topics

- Generalization
- Measuring classifier performance
- Overfitting, underfitting
- Validation

Generalization

- Definition: model does a good job of correctly predicting class labels of previously unseen samples.
- Generalization is typically evaluated using a *test* set of data that was not involved in the training process.
- Evaluating generalization requires:
  - Correct labels for test set are known.
  - A quantitative measure (*metric*) of tendency for model to predict correct labels.

NOTE: Generalization is separate from other performance issues around models, e.g. computational efficiency, scalability.

Generalization of decision trees

- If you make a decision tree deep enough, it can usually do a perfect job of predicting class labels on training set.

Is this a good thing?

NO!

- Leaf nodes do not have to be pure for a tree to generalize well. In fact, it's often better if they aren't.
- Class prediction of an impure leaf node is simply the majority class of the records in the node.
- An impure node can also be interpreted as making a probabilistic prediction.
  - Example: 7 / 10 class 1 means $p(1) = 0.7$

Metrics for classifier performance

- Accuracy

  a = number of test samples with label correctly predicted

  b = number of test samples with label incorrectly predicted

$$\text{accuracy} = \frac{a}{a+b}$$

  example

  - 75 samples in test set
  - correct class label predicted for 62 samples
  - wrong class label predicted for 13 samples
  - accuracy = 62 / 75 = 0.827

Metrics for classifier performance

- Limitations of accuracy as a metric
  - Consider a two-class problem
    - number of class 1 test samples = 9990
    - number of class 2 test samples = 10
  - What if model predicts everything to be class 1?
    - accuracy is extremely high: 9990 / 10000 = 99.9 %
    - but model will never correctly predict any sample in class 2
    - in this case accuracy is misleading and does not give a good picture of model quality

Metrics for classifier performance

- Confusion matrix

example
(continued from
two slides back)

		actual class	
		class 1	**class 2**
predicted class	**class 1**	**21**	**6**
	class 2	**7**	**41**

$$\text{accuracy} = \frac{21+41}{21+6+7+41} = \frac{62}{75}$$

Metrics for classifier performance

- Confusion matrix
  - derived metrics (for two classes)

		actual class	
		class 1 (*negative*)	**class 2** (*positive*)
predicted class	**class 1** (*negative*)	**21** (*TN*)	**6** (*FN*)
	class 2 (*positive*)	**7** (*FP*)	**41** (*TP*)

TN: true negatives

FN: false negatives

FP: false positives

TP: true positives

Metrics for classifier performance

- Confusion matrix
  - derived metrics (for two classes)

		actual class	
		class 1 (*negative*)	**class 2** (*positive*)
predicted class	**class 1** (*negative*)	**21** (*TN*)	**6** (*FN*)
	class 2 (*positive*)	**7** (*FP*)	**41** (*TP*)

$$\text{sensitivity} = \frac{TP}{TP + FN} \qquad \text{specificity} = \frac{TN}{TN + FP}$$

MATLAB interlude

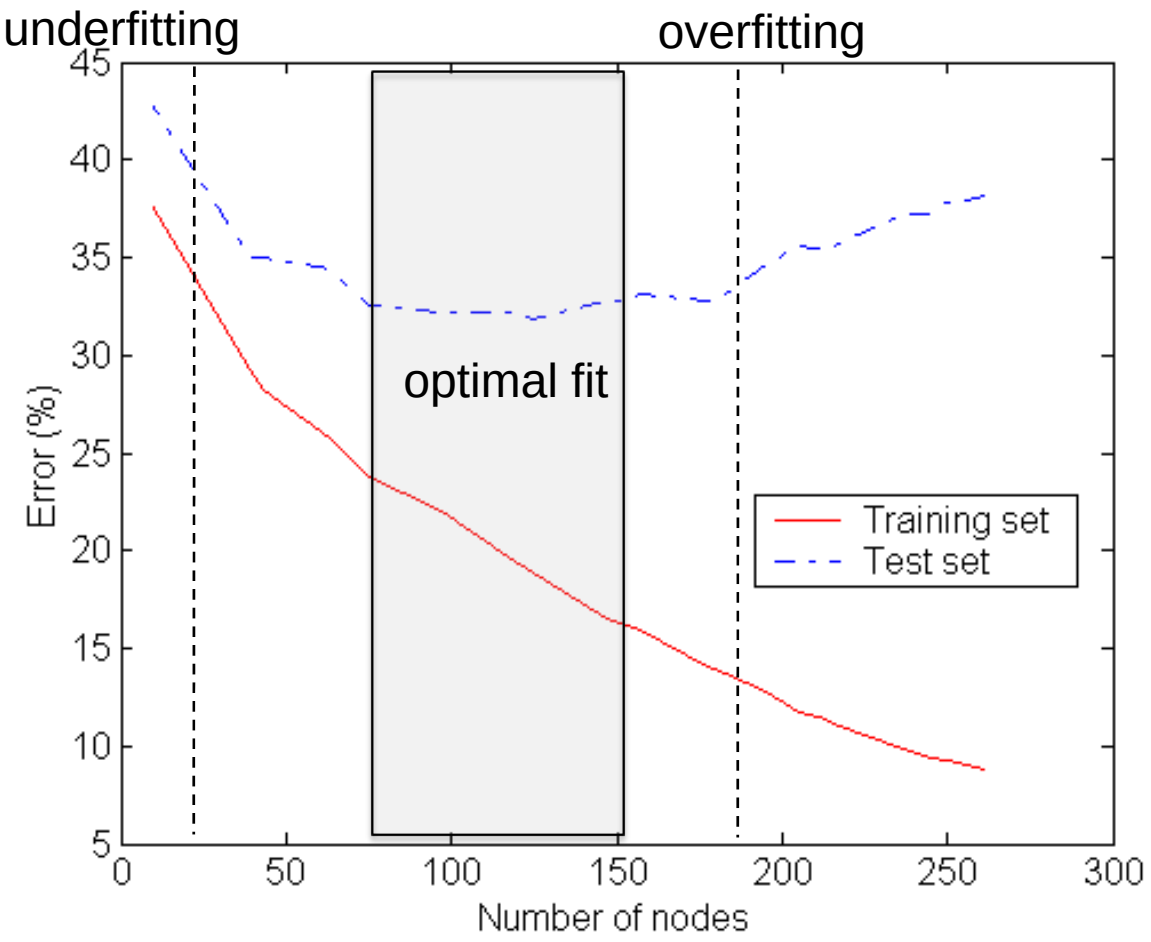
matlab_demo_04.m

Part B

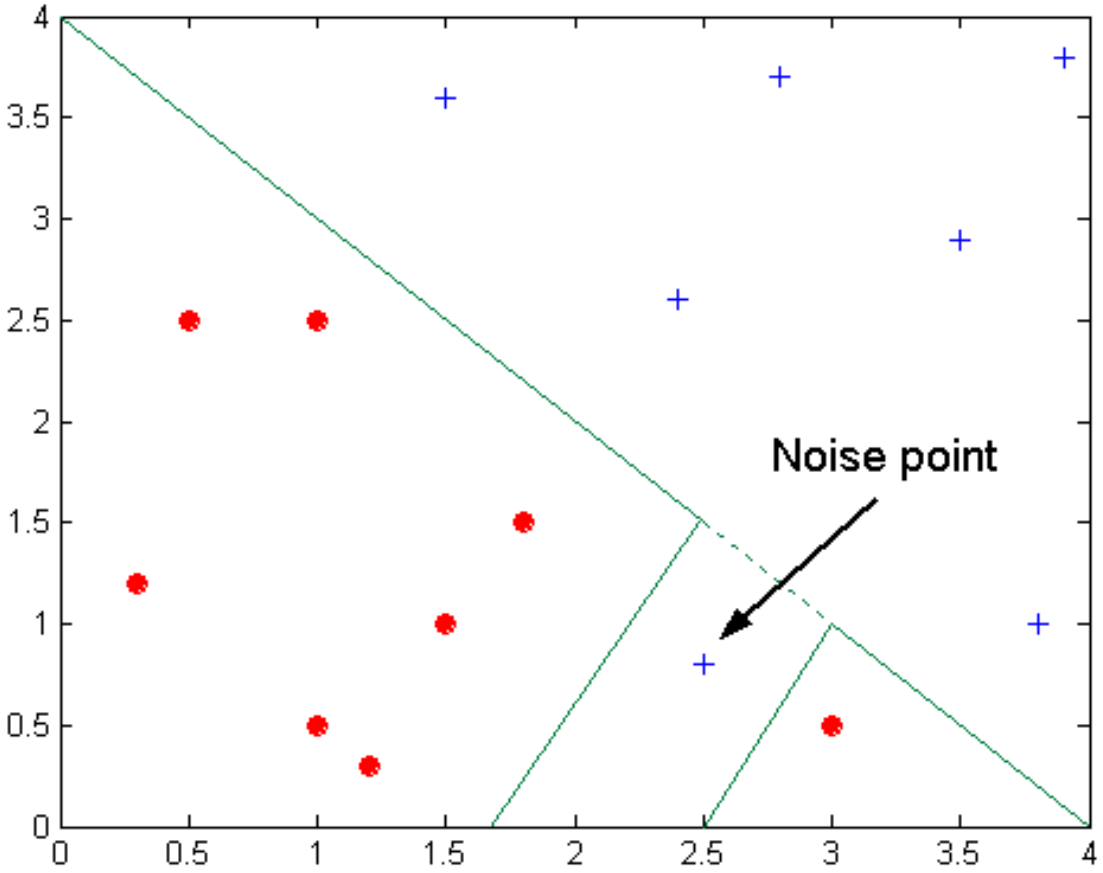
Underfitting and overfitting

- Fit of model to training and test sets is controlled by:
  - model capacity (≈ number of parameters)
    - example: number of nodes in decision tree
  - stage of optimization
    - example: number of iterations in a gradient descent optimization

Underfitting and overfitting

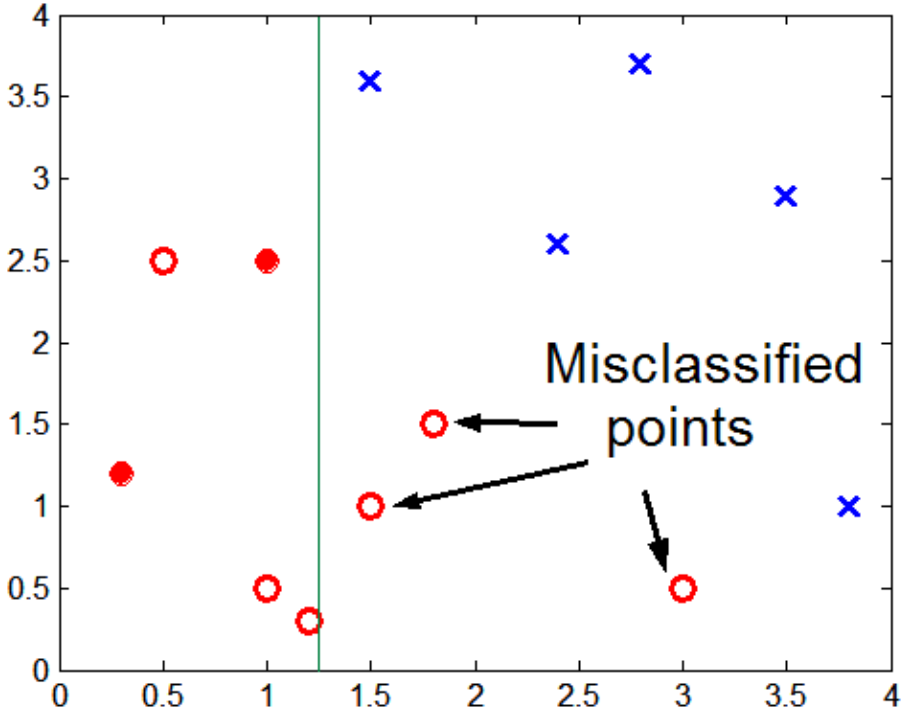


Sources of overfitting: noise



Decision boundary distorted by noise point

Sources of overfitting: insufficient examples



- Lack of data points in lower half of diagram makes it difficult to correctly predict class labels in that region.
  - Insufficient training records in the region causes decision tree to predict the test examples using other training records that are irrelevant to the classification task.

Occam's Razor

- Given two models with similar generalization errors, one should prefer the simpler model over the more complex model.
- For complex models, there is a greater chance it was fitted accidentally by errors in data.
- Model complexity should therefore be considered when evaluating a model.

Decision trees: addressing overfitting

- Pre-pruning (early stopping rules)
  - 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
  - Early stopping conditions (more restrictive):
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

Decision trees: addressing overfitting

- Post-pruning
  - Grow full decision tree
  - Trim nodes of full tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree
  - Can use various measures of generalization error for post-pruning (see textbook)

Example of post-pruning

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

Training error (before splitting) = 10/30

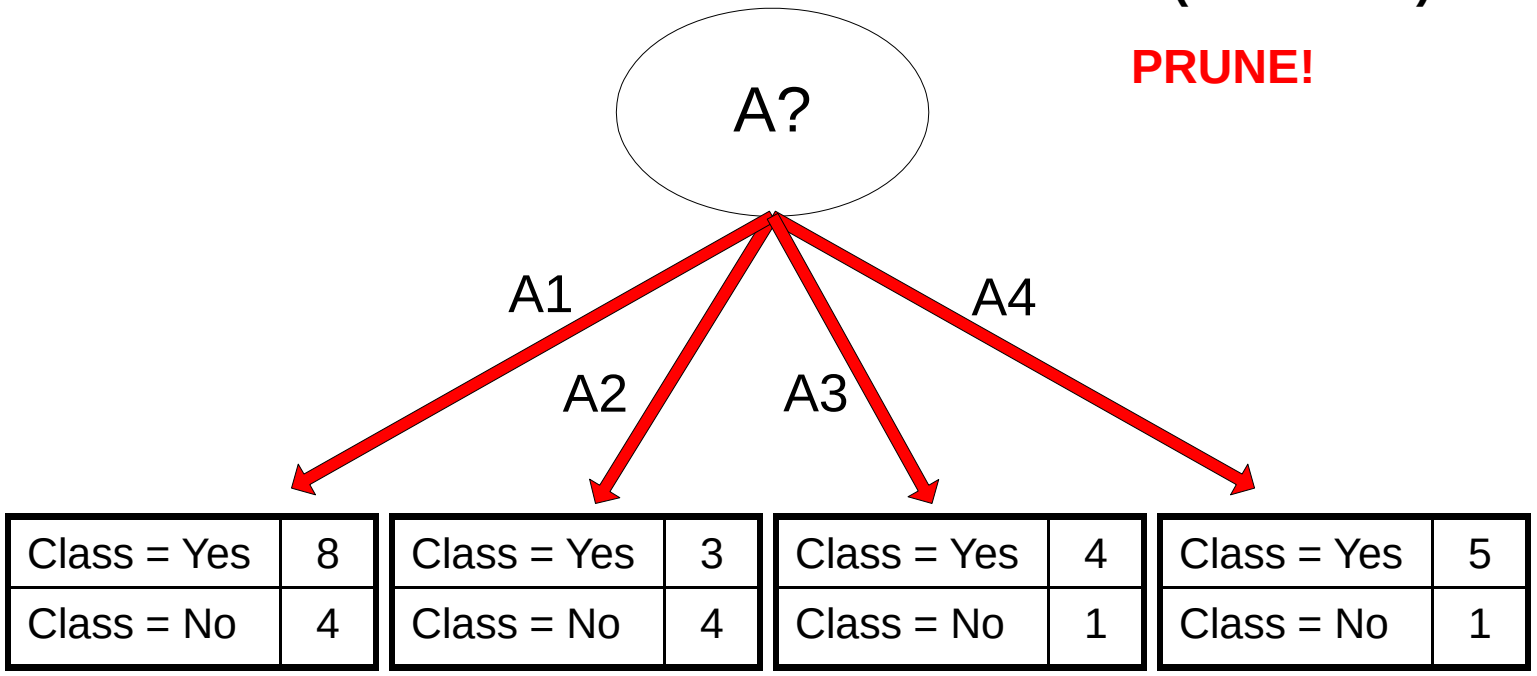
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training error (after splitting) = 9/30

Pessimistic error (after splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!



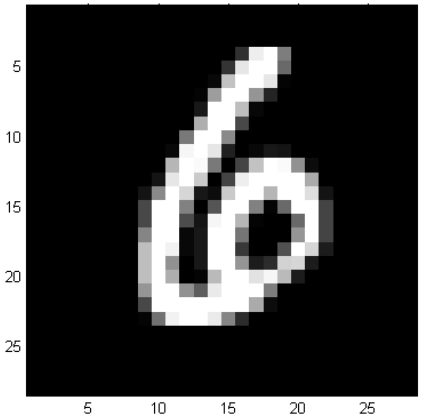
Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

MNIST database of handwritten digits



- Gray-scale images, 28 x 28 pixels.
- 10 classes, labels 0 through 9.
- Training set of 60,000 samples.
- Test set of 10,000 samples.
- Subset of a larger set available from NIST.
- Each digit size-normalized and centered in a fixed-size image.
- Good database for people who want to try machine learning techniques on real-world data while spending minimal effort on preprocessing and formatting.
- http://yann.lecun.com/exdb/mnist/
- We will use a subset of MNIST with 5000 training and 1000 test samples and formatted for MATLAB (`mnistabridged.mat`).

MATLAB interlude

matlab_demo_04.m

Part C

Model validation

- Every (useful) model offers choices in one or more of:
  - model structure
    - e.g. number of nodes and connections
  - types and numbers of parameters
    - e.g. coefficients, weights, etc.
- Furthermore, the values of most of these parameters will be modified (optimized) during the model training process.
- Suppose the test data somehow influences the choice of model structure, or the optimization of parameters …

Model validation

The one commandment of machine learning

Model validation

Divide available labeled data into three sets:

- Training set:
  - Used to drive model building and parameter optimization
- Validation set
  - Used to gauge status of generalization error
  - Results can be used to guide decisions during training process
    - typically used mostly to optimize small number of high-level *meta* parameters, e.g. regularization constants; number of gradient descent iterations
- Test set
  - Used only for final assessment of model quality, after training + validation completely finished

Validation strategies

- Holdout
- Cross-validation
- Leave-one-out (LOO)

- Random vs. block folds
  - Use random folds if data are independent samples from an underlying population
  - Must use block folds if any there is any spatial or temporal correlation between samples

Validation strategies

- Holdout
  - Pro: results in single model that can be used directly in production
  - Con: can be wasteful of data
  - Con: a single static holdout partition has the potential to be unrepresentative and statistically misleading
- Cross-validation and leave-one-out (LOO)
  - Con: do not lead directly to a single production model
  - Pro: use all available data for evaulation
  - Pro: many partitions of data, helps average out statistical variability

Validation: example of block folds