Data Mining
Classification: Basic Concepts, Decision Trees, and Model Evaluation

Lecture Notes for Chapter 4

Introduction to Data Mining

by

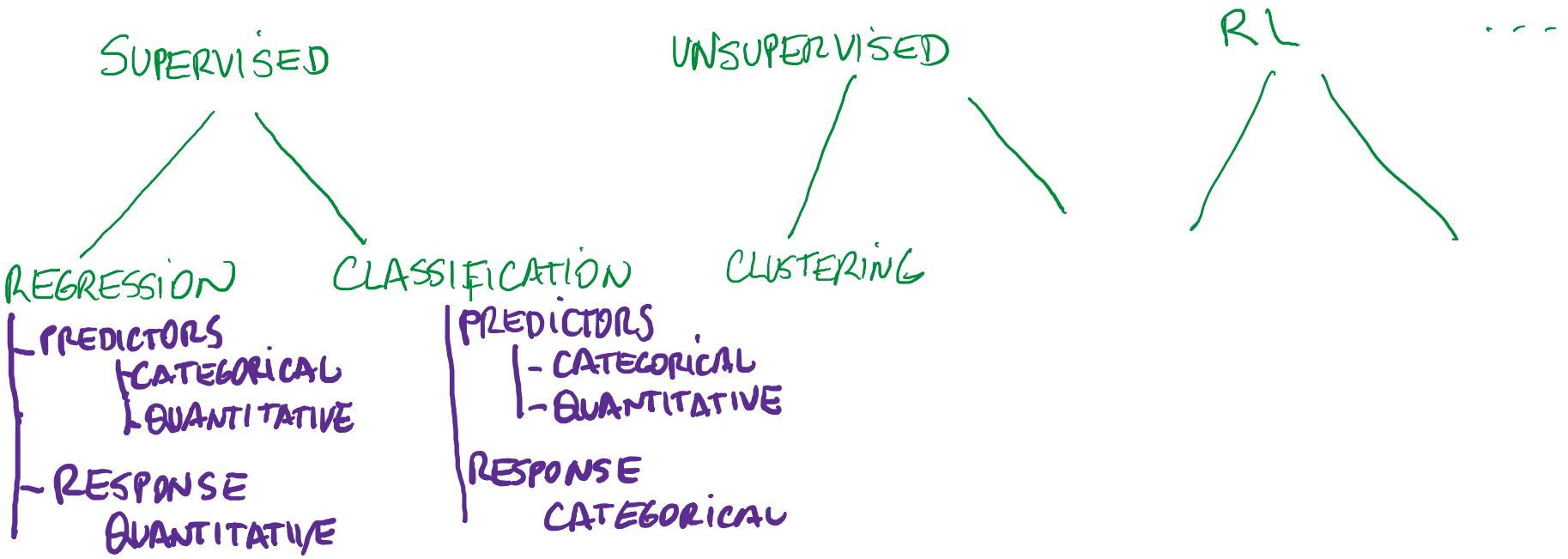
Tan, Steinbach, Kumar

Modified for CS 4232 / 5232

Outline

- The Classification Problem
- Decision Tree Classifiers
- Practical Issues of Classification
  - Overfitting and Underfitting
  - Estimating the Test Error of a Classifier
- Evaluating the Performance of Classifiers

Outline


SUPERVISED
UNSUPERVISED
RL
...
REGRESSION
CLASSIFICATION
CLUSTERING
PREDICTORS
CATEGORICAL
QUANTITATIVE
RESPONSE
QUANTITATIVE
PREDICTORS
- CATEGORICAL
- QUANTITATIVE
RESPONSE
CATEGORICAL

Classification

The Classification Problem

Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class 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.

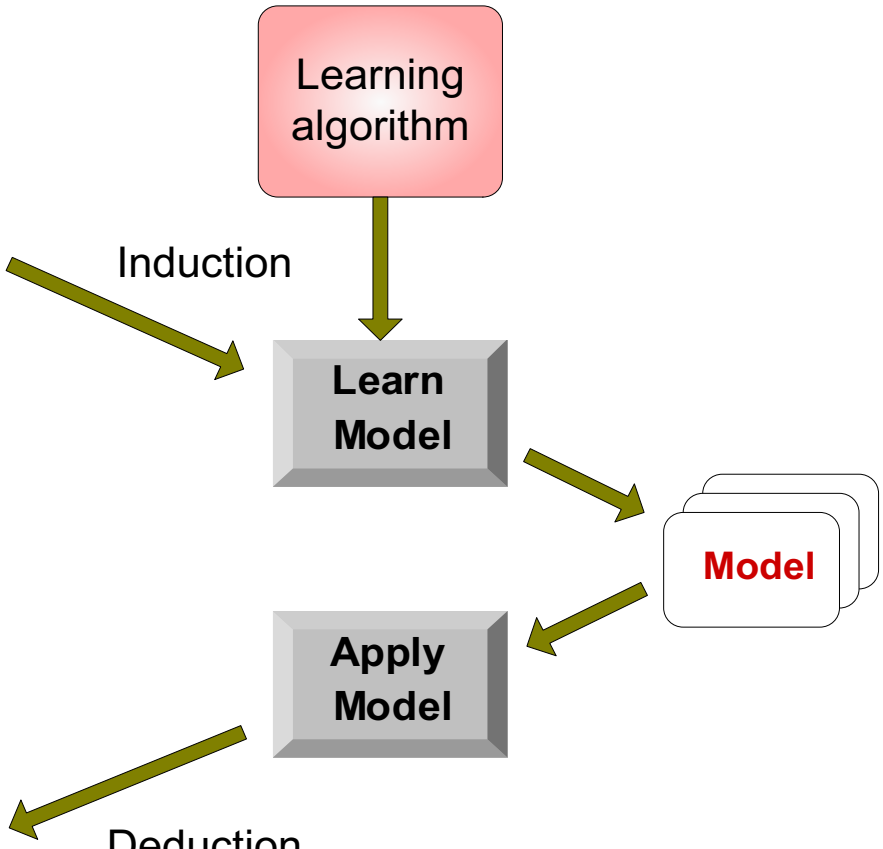
Illustrating Classification Task

RESPONSE

PREDICTORS

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	?

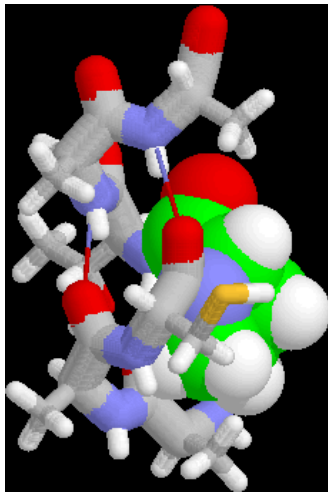
Test Set

Examples of Classification 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
- Logistic Regression
- Neural Networks
- Naïve Bayes and Bayesian Networks
- Support Vector Machines
- Many, many more!!!

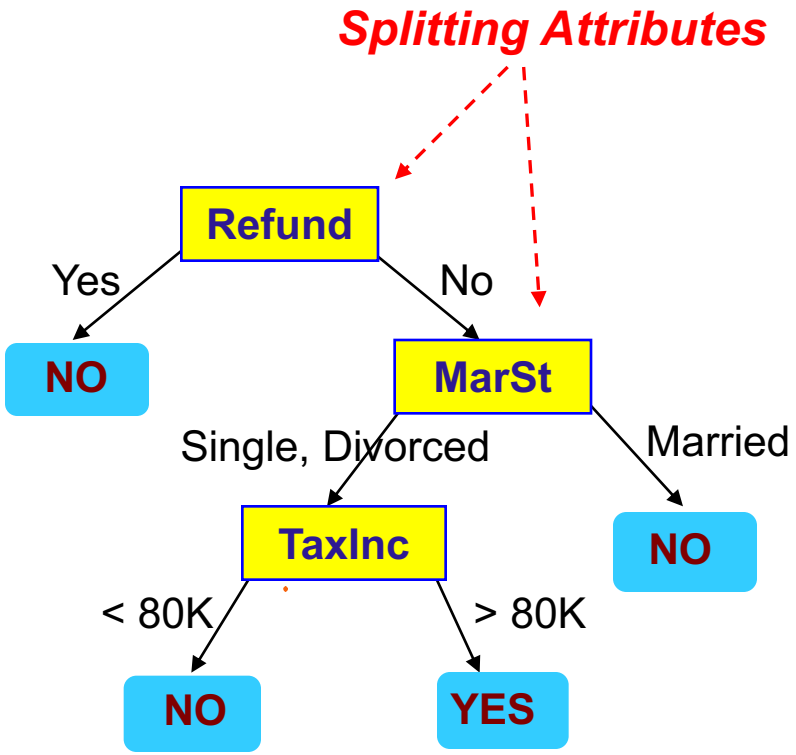
Classification

Decision Tree Classifiers

Example of a Decision Tree

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (class)
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

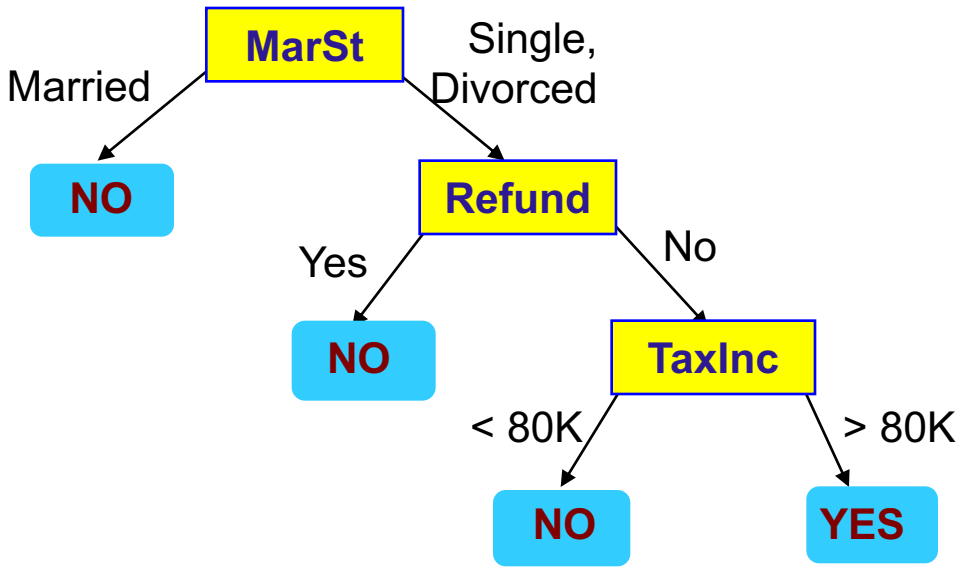
Training Data



Model: Decision Tree

Another Example of Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
	categorical	categorical	continuous	class
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!

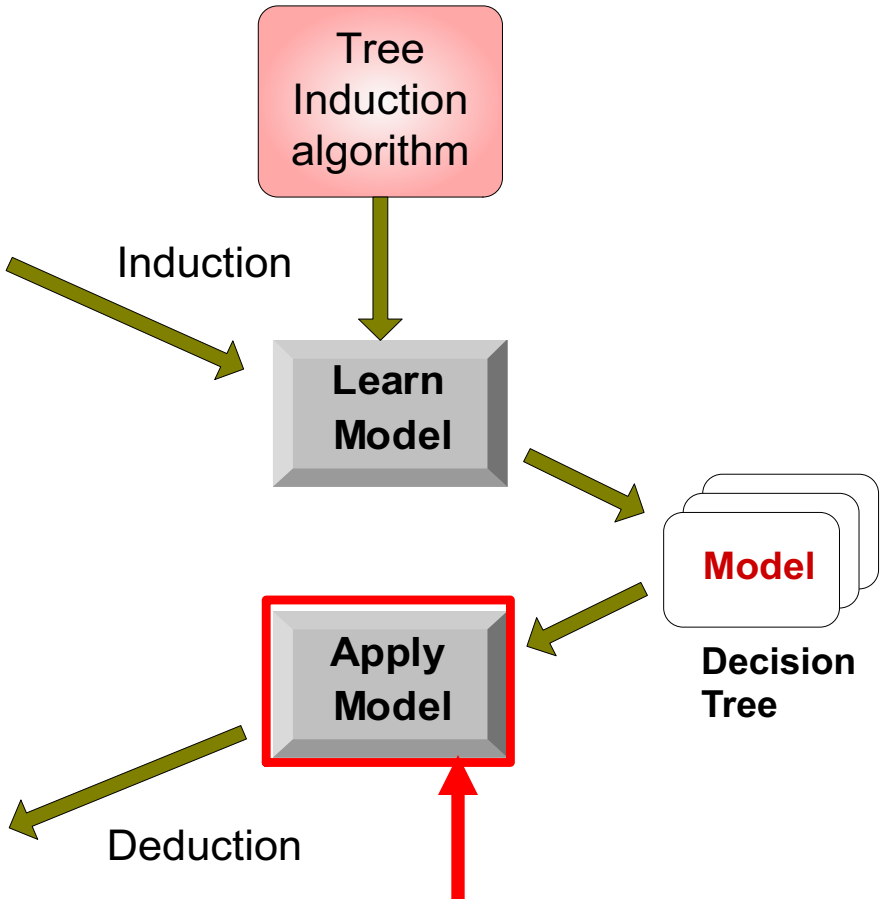
Decision Tree Classification Task

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

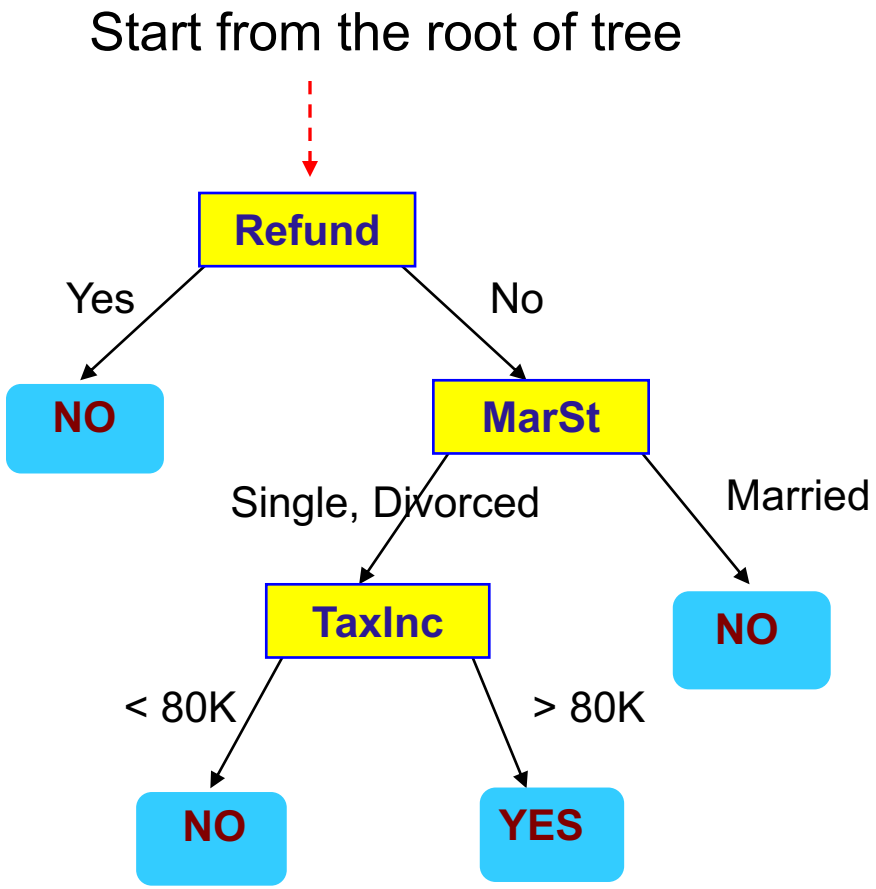
Training Set

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



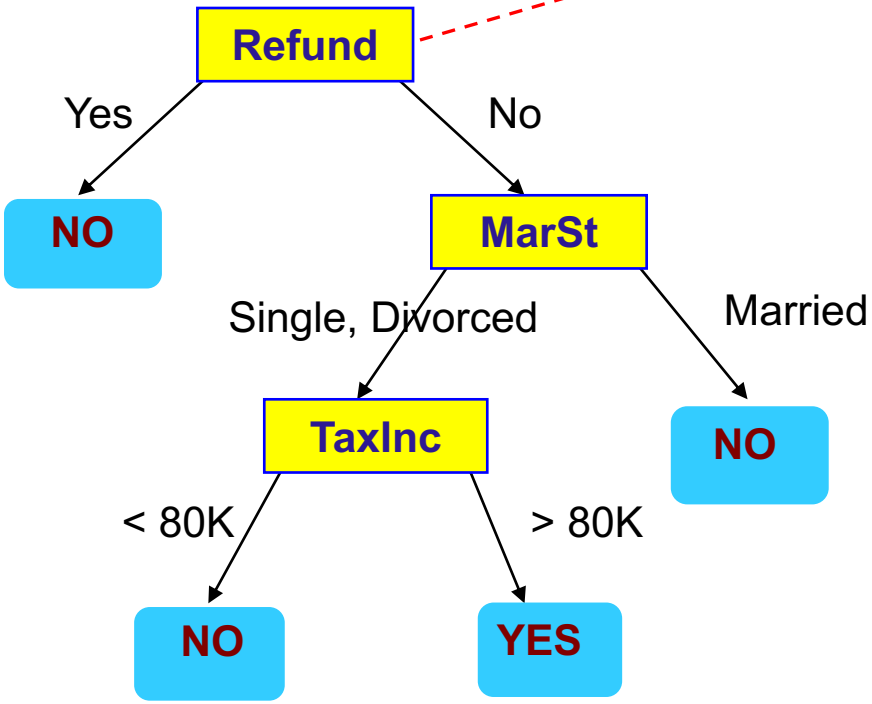
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

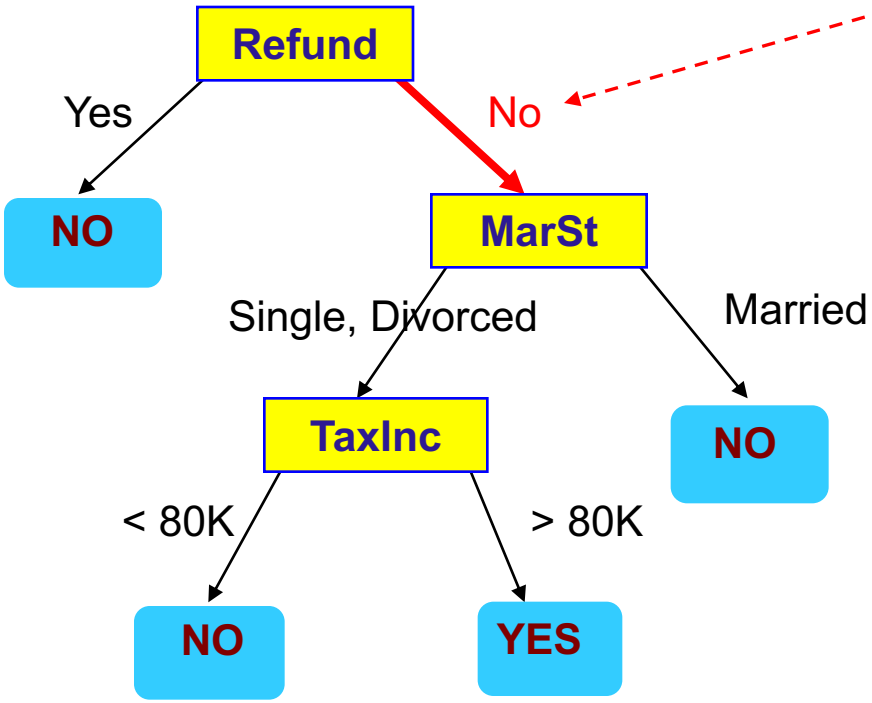
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

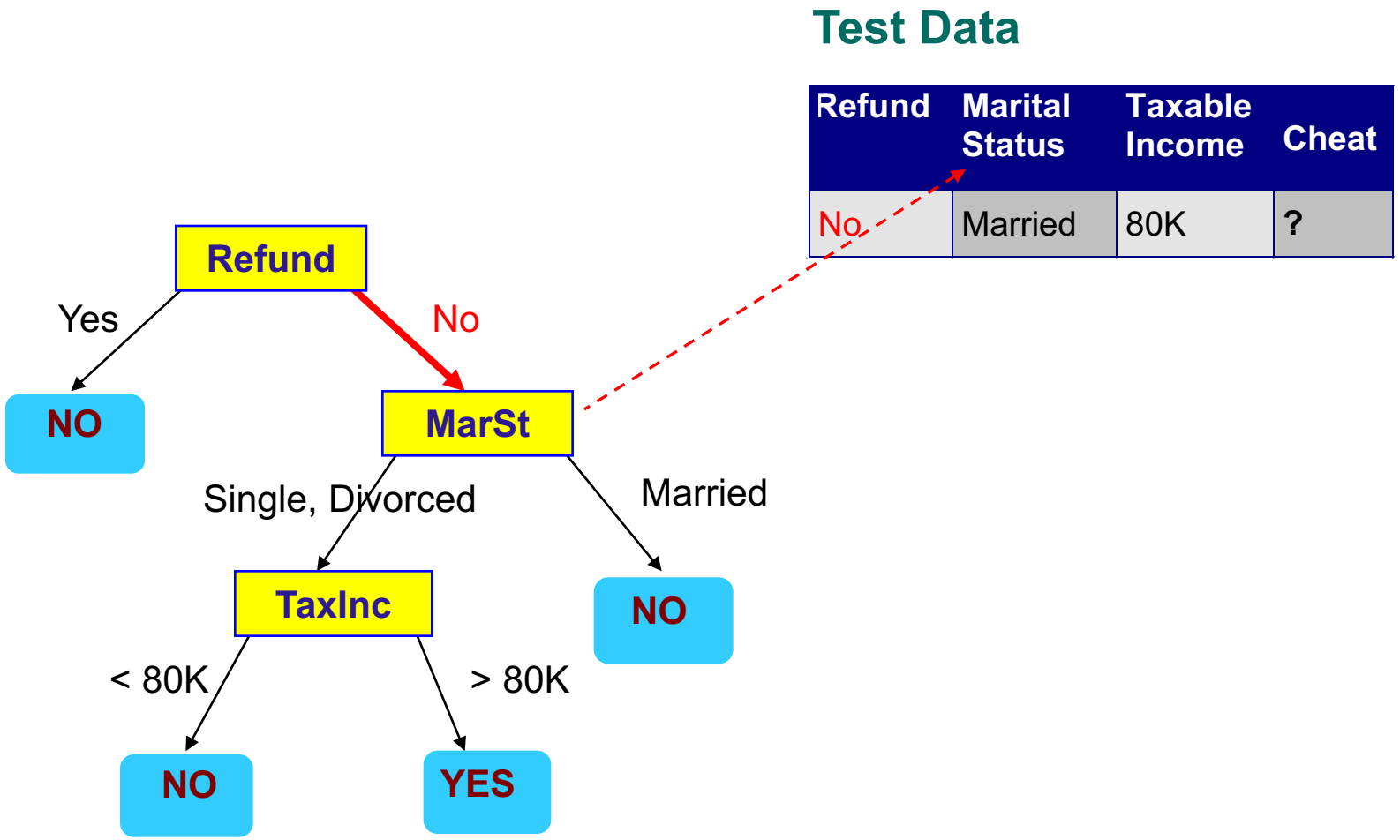
Apply Model to Test Data



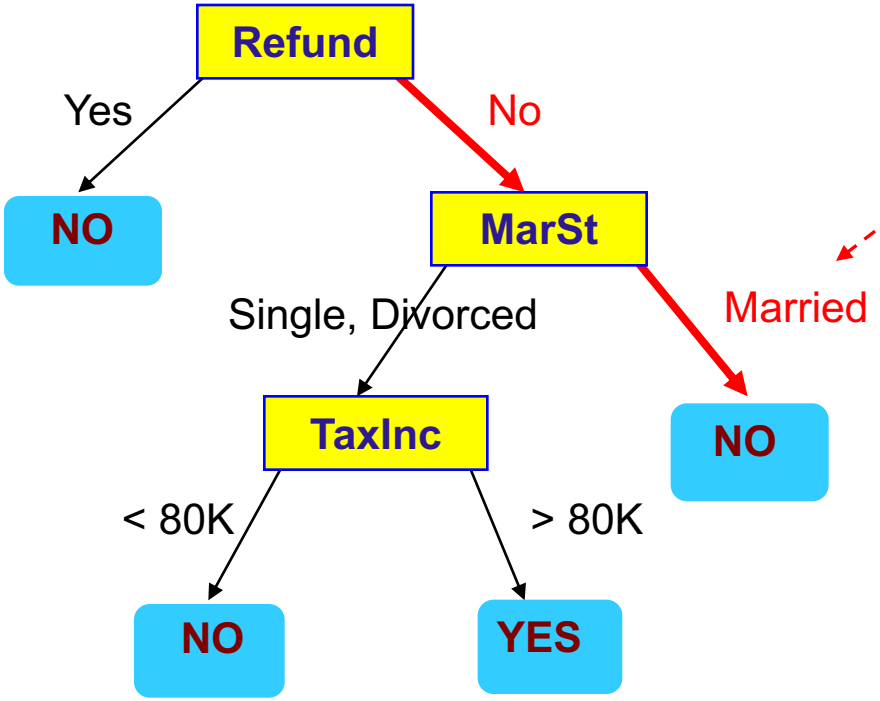
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



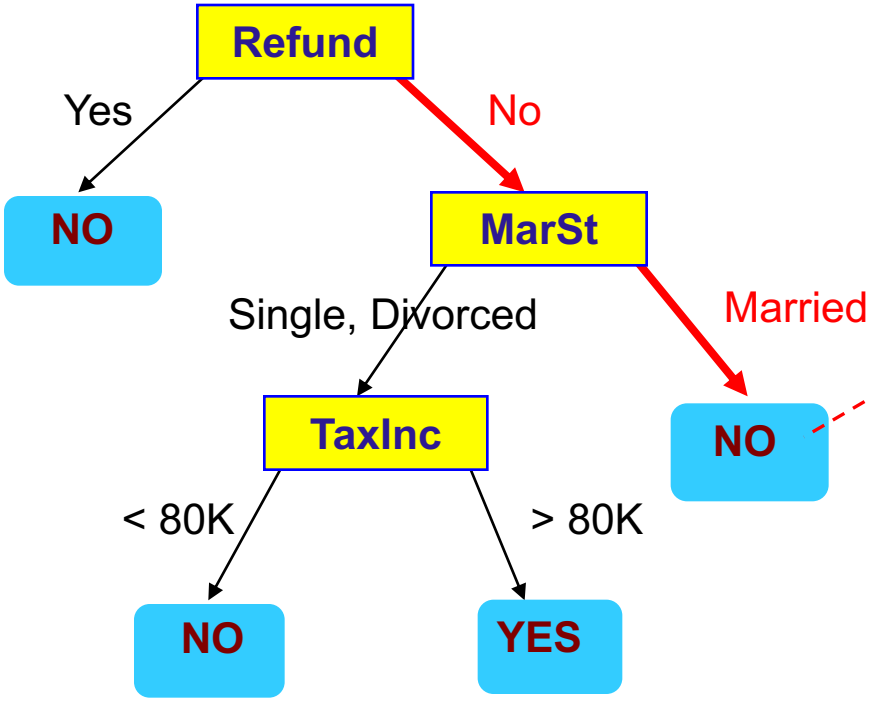
Apply Model to Test Data



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”

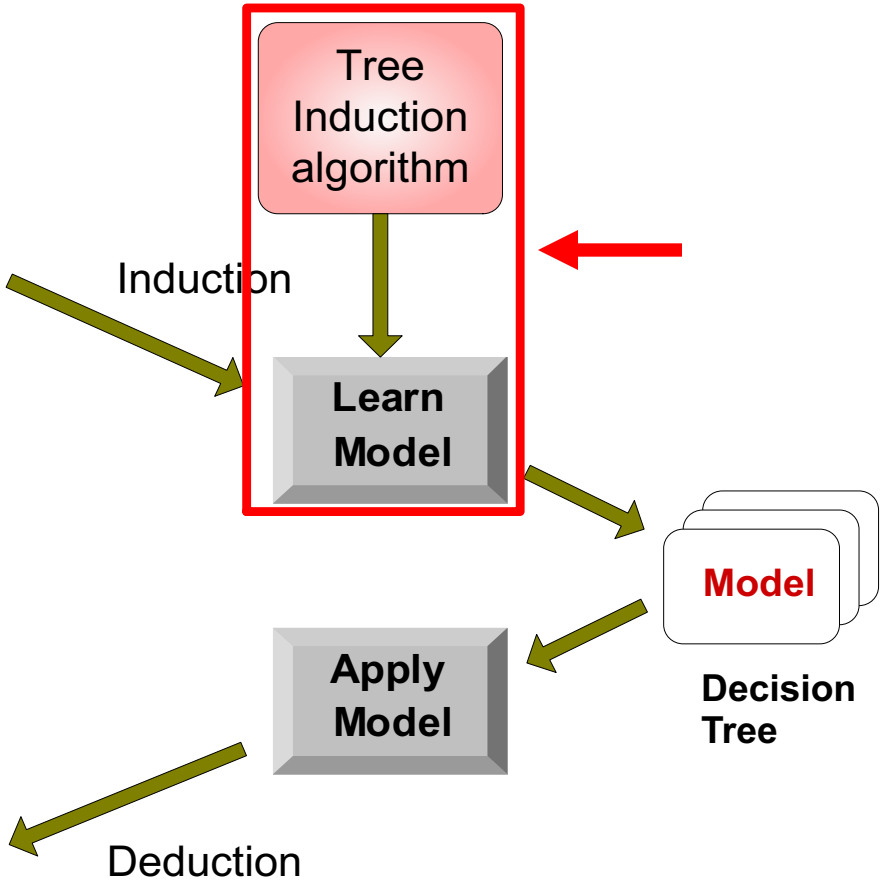
Decision Tree Classification Task

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



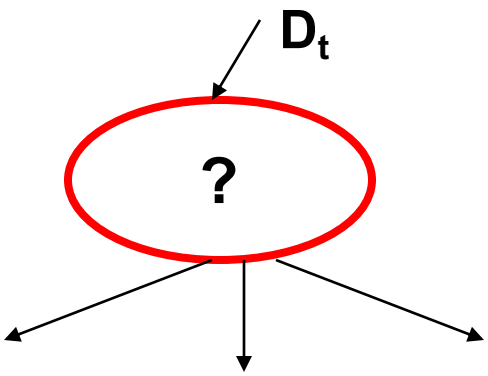
Decision Tree Induction

- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5, C5.0
  - SLIQ, SPRINT
  - CHAID
  - QUEST
  - CRUISE

General Structure of Hunt's Algorithm

- Let D_t be the set of training records and A_t the set of attributes that reach a node t,
- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If either D_t or A_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If D_t contains records that belong to more than one class, use an attribute test to select a splitting attribute S that splits the data into smaller subsets. Recursively apply the procedure to each subset, using $A_t \backslash \{S\}$ as the new set of attributes.

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

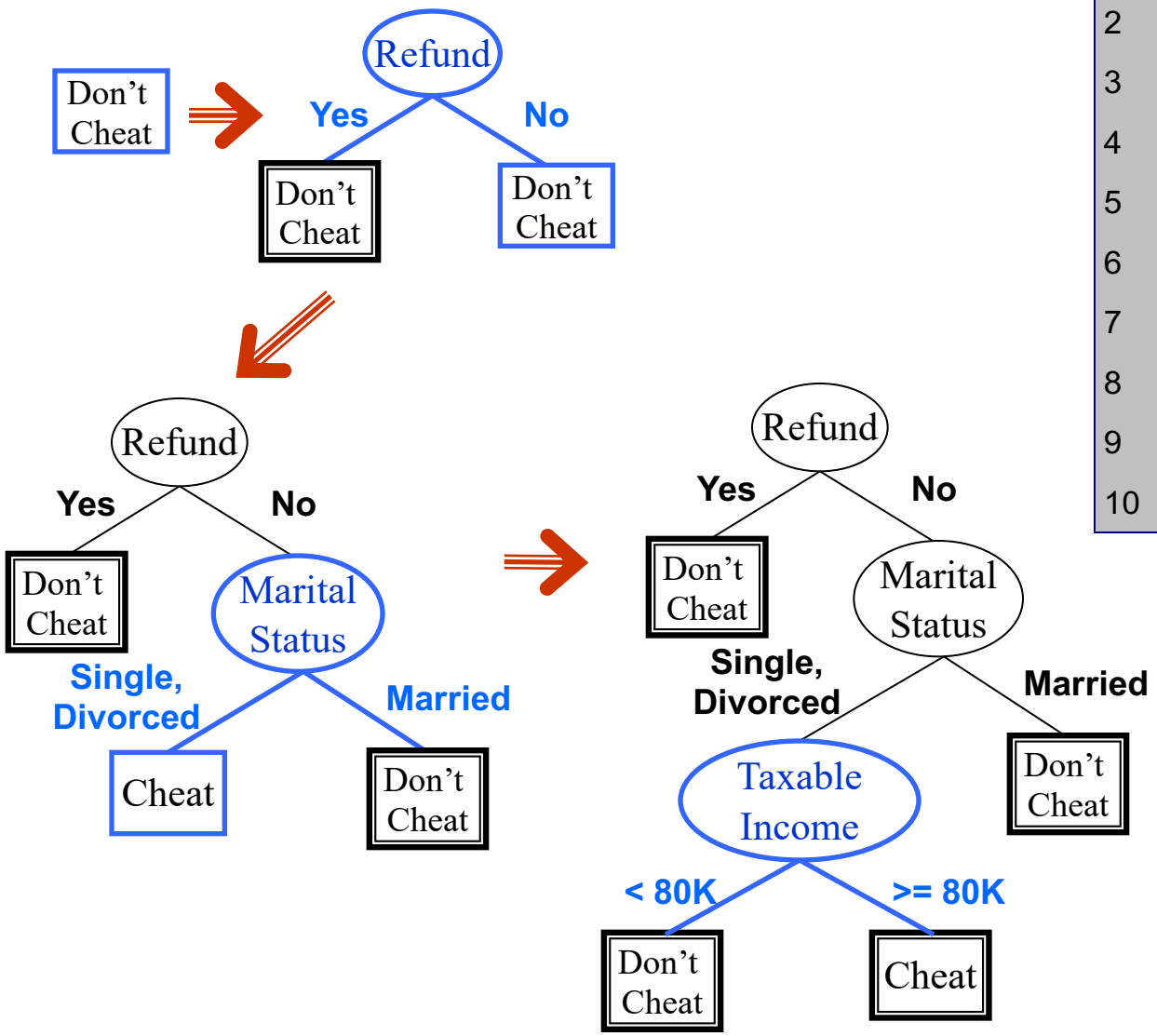


General Structure of Hunt's Algorithm

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

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 based on an attribute test that optimizes certain criterion

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - 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 certain criterion

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

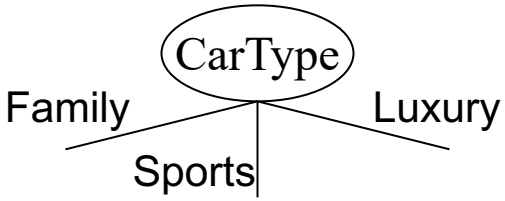
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

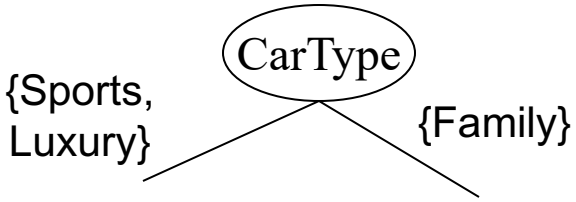
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

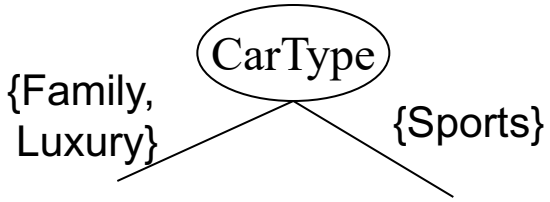
- Multi-way split: Use as many partitions as distinct values.



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

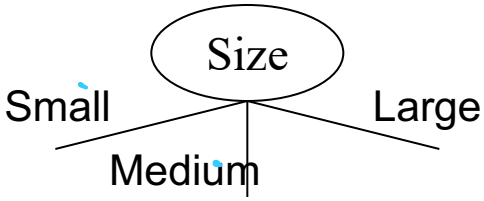


OR

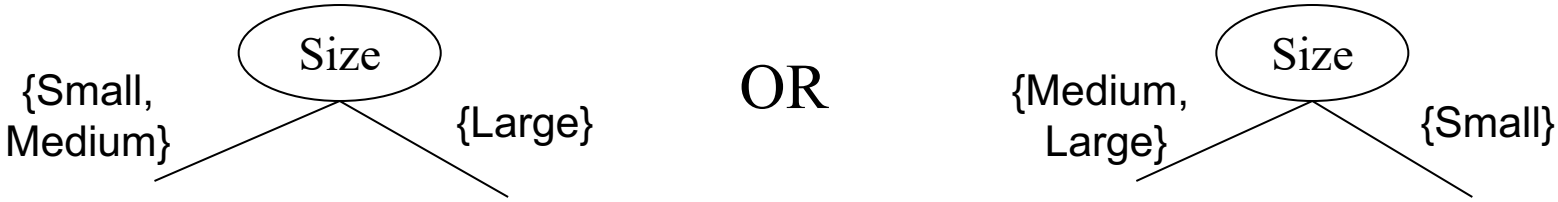


Splitting Based on Ordinal Attributes

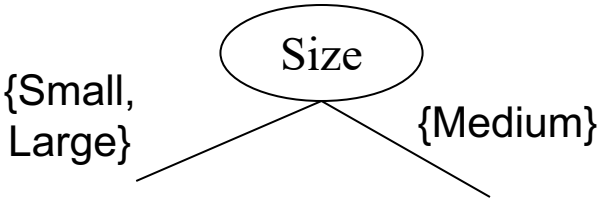
- Multi-way split: Use as many partitions as distinct values.



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



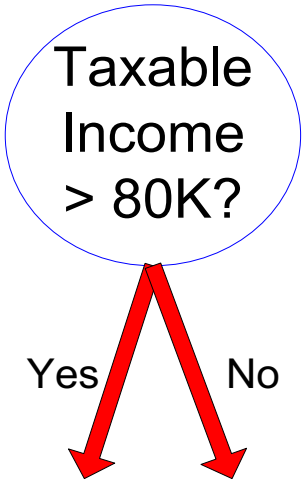
- What about this split?



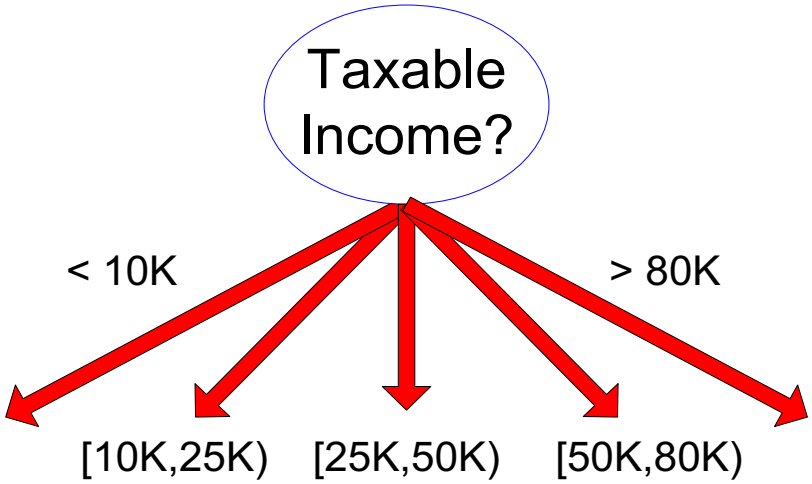
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

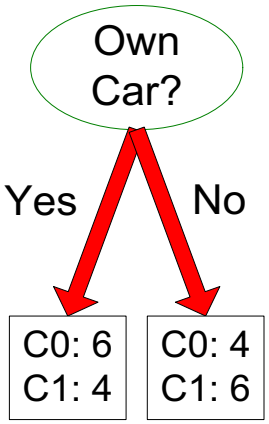
Tree Induction

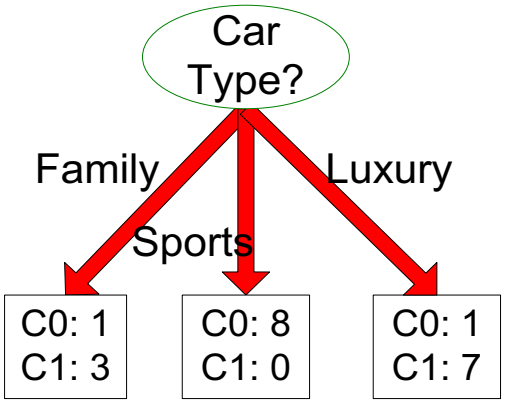
- Greedy strategy
  - Split the records based on an attribute test that optimizes certain criterion

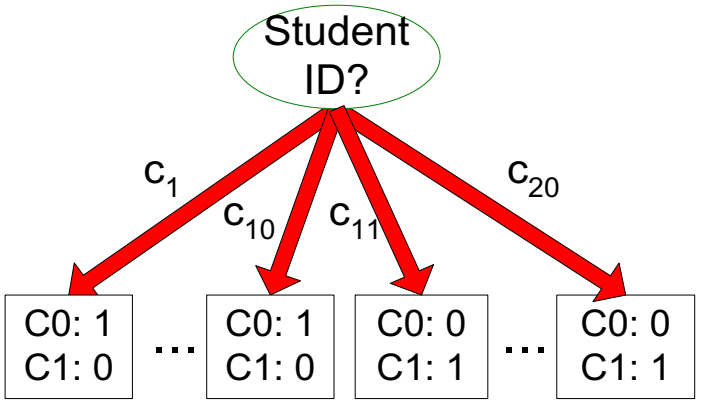
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to determine the Best Split

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







Which test condition is the best?

How to determine the Best Split

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

C0: 5
C1: 5

Non-homogeneous,

High degree of impurity

Which one is preferable?

C0: 9
C1: 1

Homogeneous,

Low degree of impurity

The RHS: the most pure

Measures of Node Impurity

- Gini Index
- Entropy

Measure of Impurity: GINI

- Gini Index for a given node t :

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

(NOTE: $p(j \mid t)$ 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 interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting 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 for computing GINI

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

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

- Used in CART, SLIQ, SPRINT
- When a node p is split into k partitions (children), the quality of split is computed as,

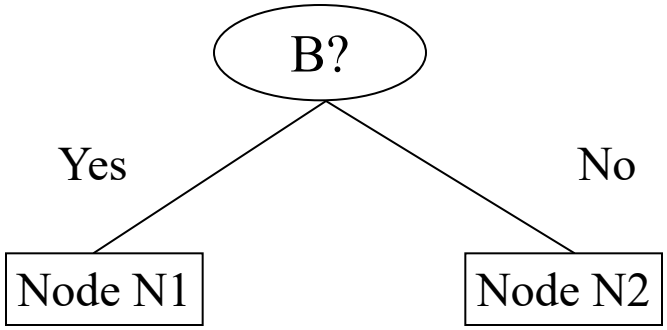
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.

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 – $(5/7)^2$ – $(2/7)^2$
= 0.408

Gini(N2)
= 1 – $(1/5)^2$ – $(4/5)^2$
= 0.32

	N1	N2
C1	**5**	**1**
C2	**2**	**4**
Gini=0.371		

Gini(Children)
= 7/12 * 0.408 +
5/12 * 0.32
= 0.371

Categorical Attributes: Computing Gini Index

- 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 values)

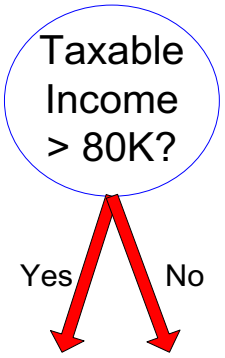
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

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

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values
- Each splitting value 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 database to gather count matrix and 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



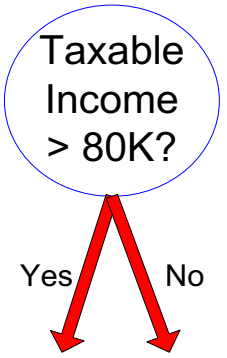
Continuous Attributes: Computing Gini Index

- Let's compute the best split on the continuous attribute Taxable Income:
  1. Sort the values of the attribute

     60, 70, 75, 85, 90, 95, 100, 120, 125, 220

  2. Consider splits in between each pair of consecutive values
  3. Pick the split with the smallest GINI index (most pure split, i.e., the most homogeneous split).

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



Continuous Attributes: Computing Gini Index

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

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

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	

We want the lowest GINI because the GINI coefficient is larger with greater impurity. We want low impurity.

How to Find the Best Split

Before Splitting:

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

→ **Measure0**

Option 1

A?

Yes — Node N1

C0	**N10**
C1	**N11**

→ **Measure1**

No — Node N2

C0	**N20**
C1	**N21**

→ **Measure2**

Measure12

The larger the gain, the better

Option 2

B?

Yes — Node N3

C0	**N30**
C1	**N31**

→ **Measure3**

No — Node N4

C0	**N40**
C1	**N41**

→ **Measure4**

Measure34

Gain = M0 – M12 vs M0 – M34

Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

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

  - Measures the homogeneity of a node
    - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
    - Minimum (0.0) when all records belong to one class, implying most information (high purity, high homogeneity)
  - Entropy based computations are similar to the GINI index computations

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

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) $\log_2$ (5/6) = 0.65

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

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

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Remember that log(0) is undefined. However, for entropy calculations, we define 0*log(0) to be 0.

Splitting Based on INFO...

- 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 and C4.5
  - Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

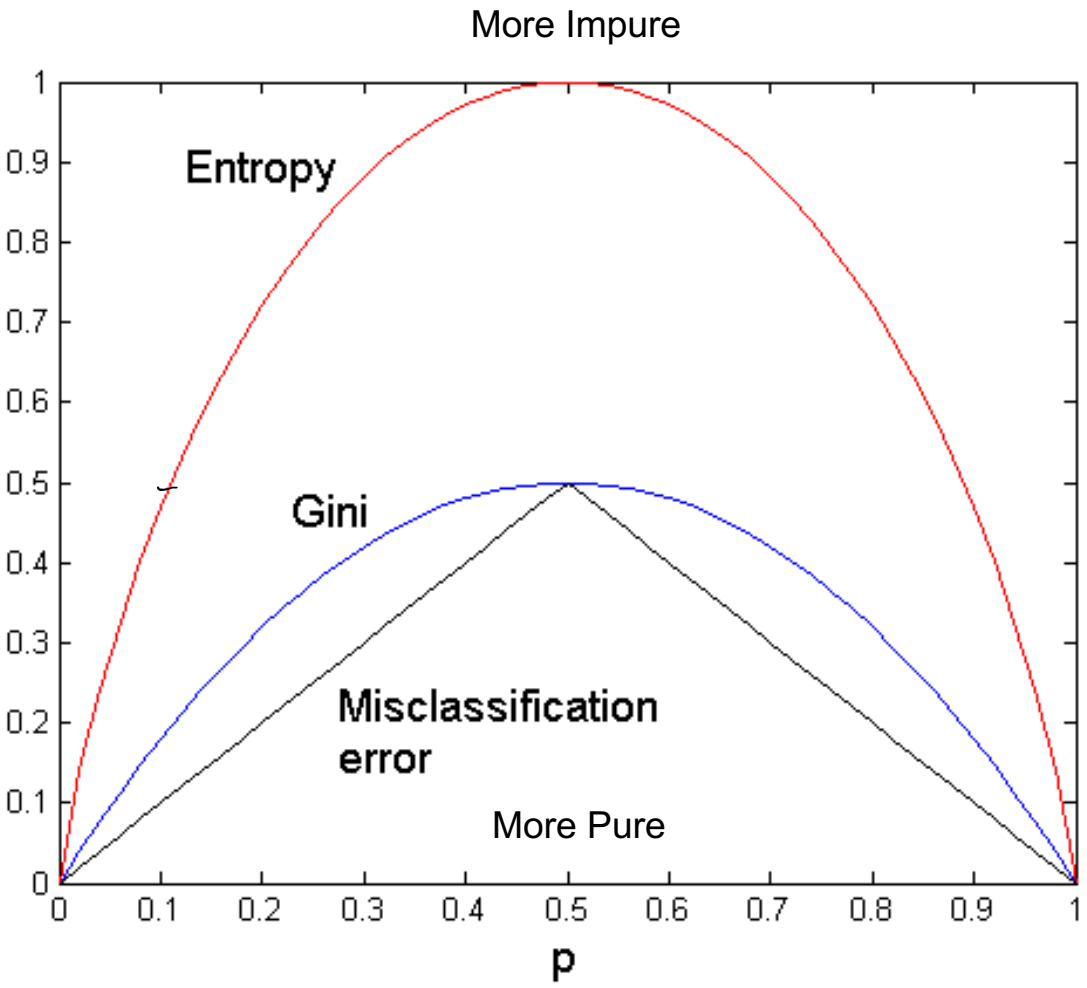
Comparison among Splitting Criteria

For a 2-class problem:

Class Name	Num. Elements
C1	N1
C2	N2

p = N1 / (N1 + N2)

In practice, using either Gini or Entropy does not make a big impact. Computing the entropy is perhaps more expensive.



Tree Induction

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

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - 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
- Stop expanding a node when all the records have similar attribute values
- Early termination (to be discussed later)

Decision Tree Based Classification

- Advantages:
  - 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

Decision Trees in R

- To fit a decision tree with CART in R,

```
library(rpart)
library(rattle)
```

Here we predict mpg as a function of weight, transmission type of the car, and engine displacement

```
```{r}
tree.model <- rpart(mpg ~ wt + am + disp, data = mtcars)
summary(tree.model)
```
```

```
Call:
rpart(formula = mpg ~ wt + am + disp, data = mtcars)
  n= 32

        CP nsplit rel error    xerror      xstd
1 0.635663      0 1.0000000 1.0862165 0.2621433
2 0.174912      1 0.3643370 0.6105166 0.1224439
3 0.010000      2 0.1894251 0.5615926 0.1196085

Variable importance
  wt disp   am
  48   45    8

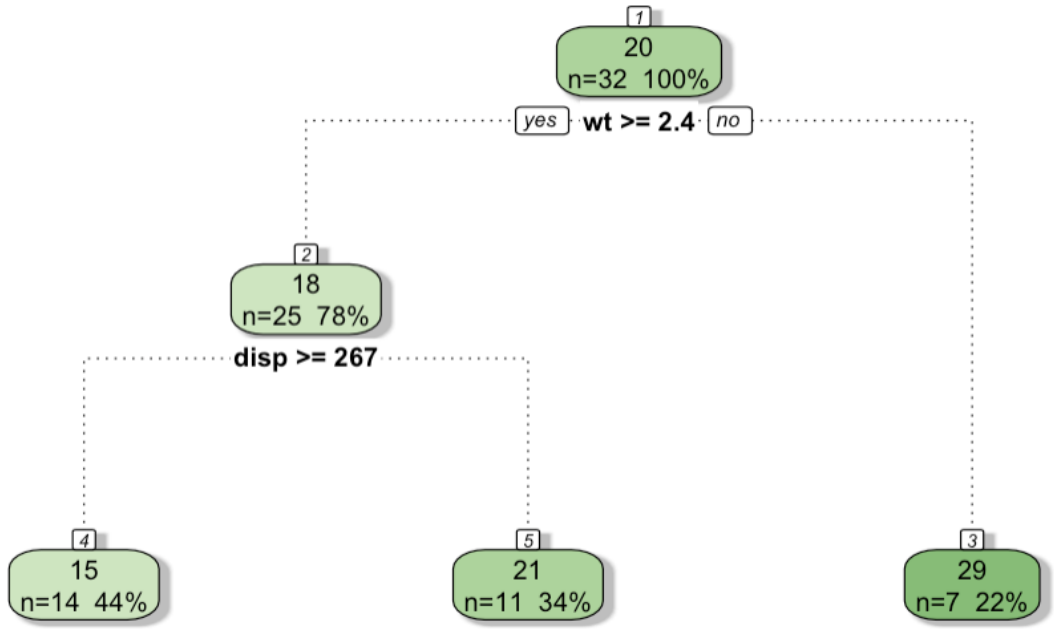
Node number 1: 32 observations,    complexity param=0.635663
  mean=20.09062, MSE=35.18897
  left son=2 (25 obs) right son=3 (7 obs)
  Primary splits:
      wt   < 2.3925 to the right, improve=0.6356630, (0 missing)
      disp < 163.8  to the right, improve=0.6130502, (0 missing)
      am   < 0.5    to the left,  improve=0.3597989, (0 missing)
```

Decision Trees in R

- To plot a decision tree in R,

```
library(rpart)
library(rattle)
tree.model <- rpart(mpg ~ wt + am +disp , data = mtcar)
fancyRpartPlot(tree.model)
```

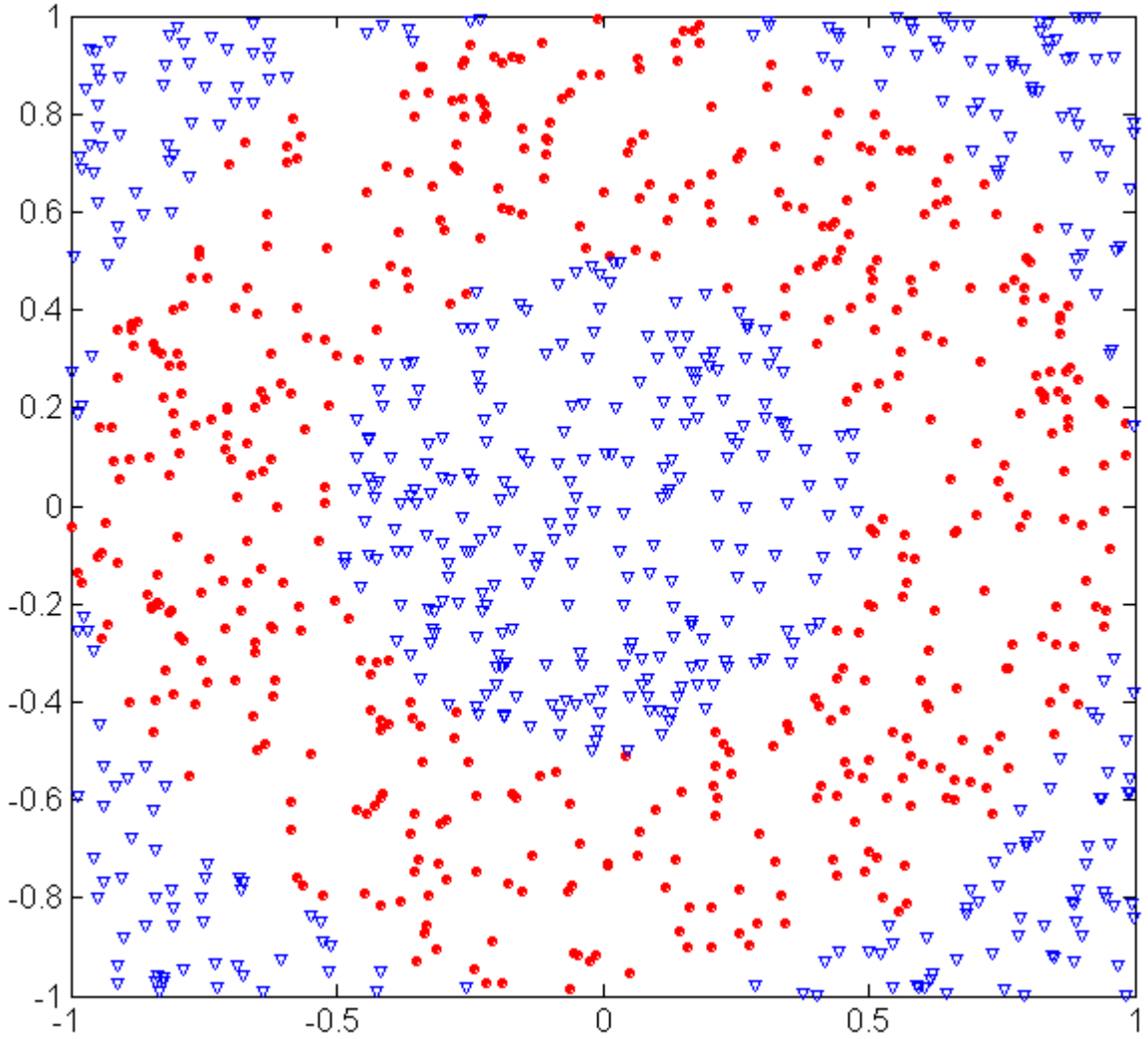
```{r}
fancyRpartPlot(tree.model)
```



Practical Issues of Classification

Overfitting and Underfitting

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

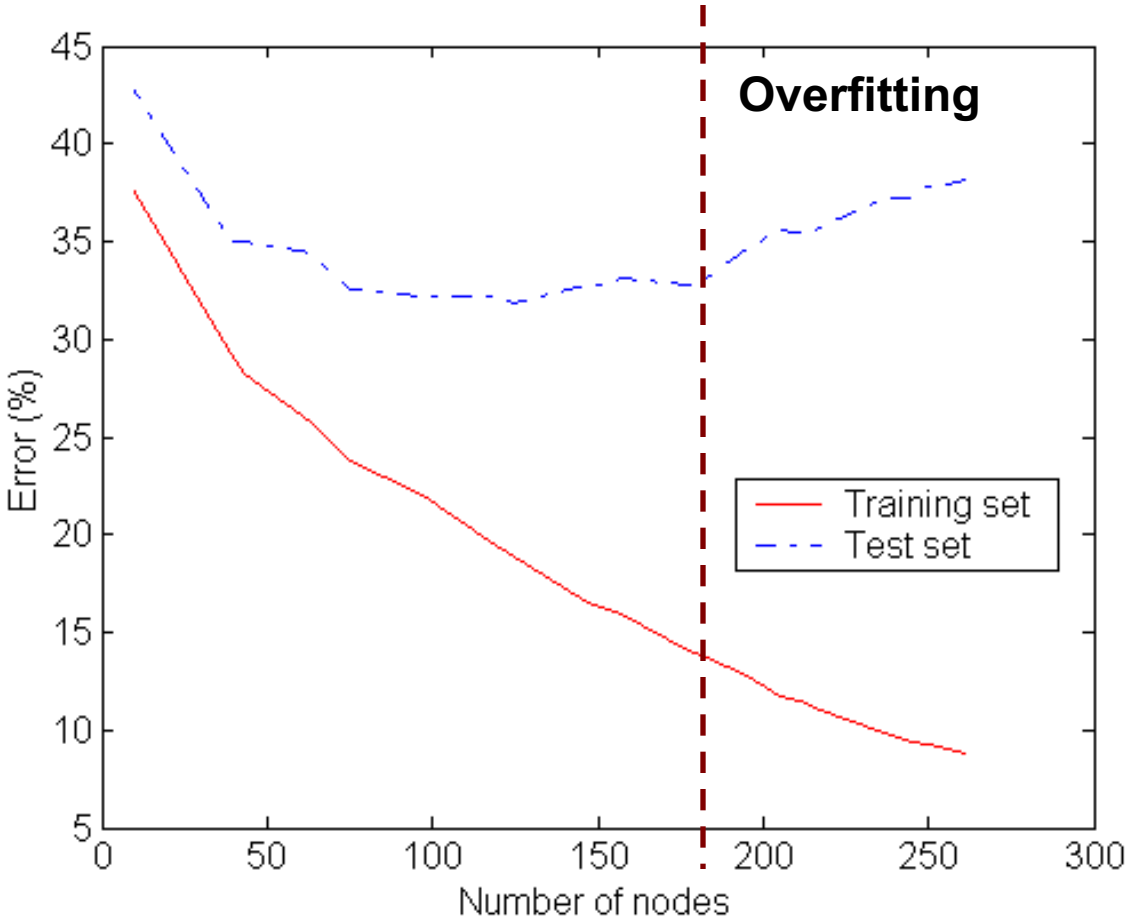
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 0.5$ or

$\text{sqrt}(x_1^2+x_2^2) < 1$

Underfitting and Overfitting



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

Underfitting and Overfitting

Plot that shows under and overfitting of polynomials of degree M (shown in red) with respect to a dataset of points in blue. The true function that generates the points is shown in green, and in unknown in practice.

Which models underfit and which models overfit? Why?

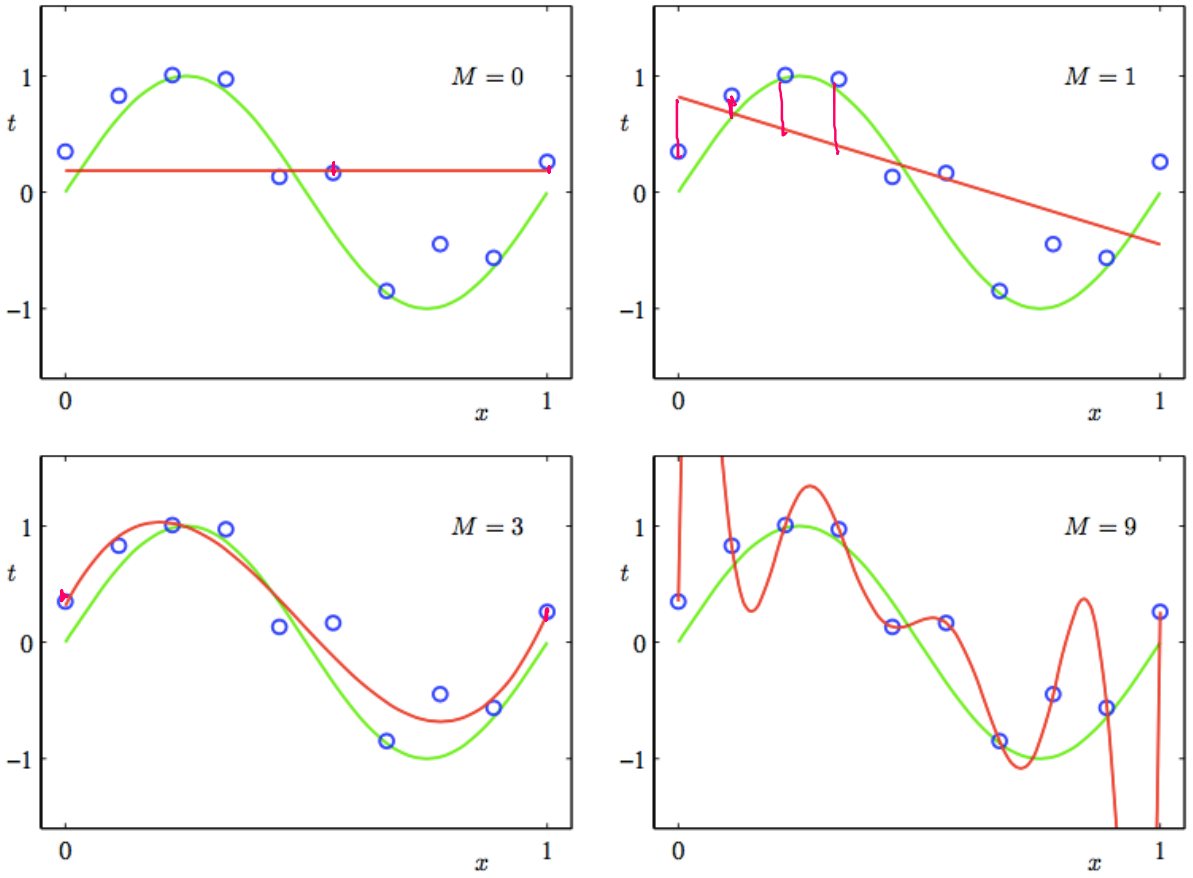
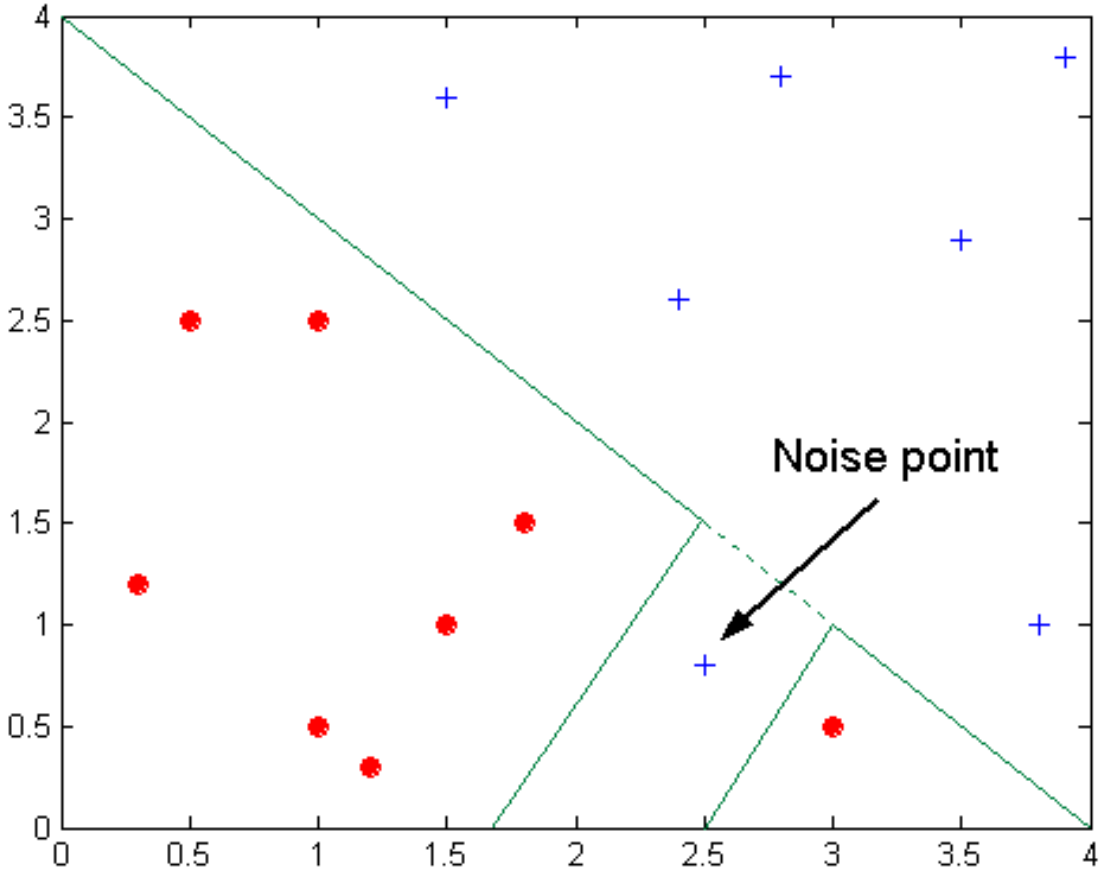


Figure from Bishop, C. Pattern Recognition and Machine Learning. Springer, 2006

Overfitting due to Noise



Decision boundary is distorted by noise point

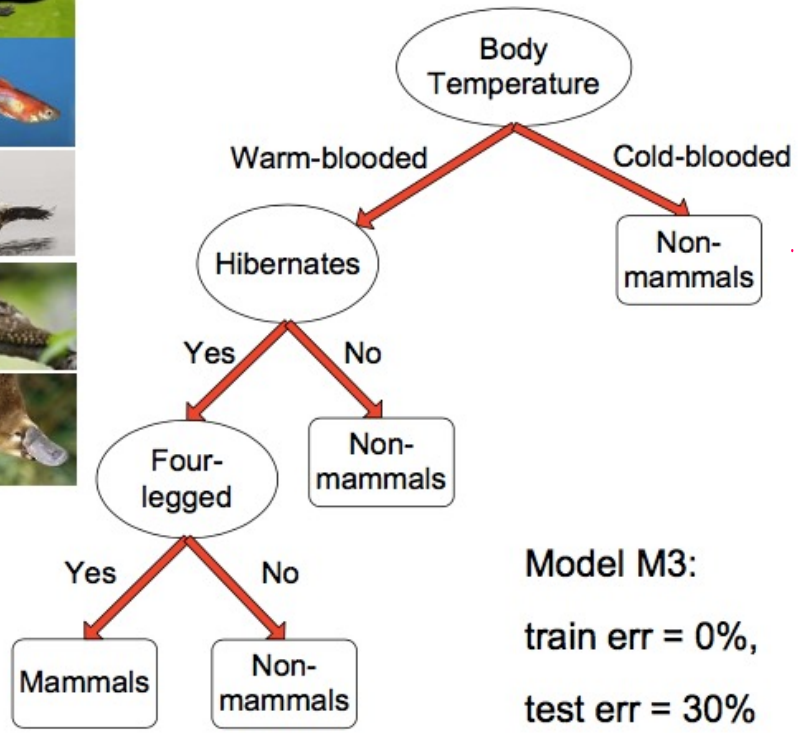
Overfitting: Lack of Representative Samples

Training Set:

Name	Body Temperature	Four-legged	Hibernates	Class Label
salamander	cold-blooded	yes	yes	no
guppy	cold-blooded	no	no	no
eagle	warm-blooded	no	no	no
poorwill	warm-blooded	no	yes	no
platypus	warm-blooded	yes	yes	yes

Test Set:

Name	Body Temperature	Four-legged	Hibernates	Class Label
human	warm-blooded	no	no	yes
pigeon	warm-blooded	no	no	no
elephant	warm-blooded	yes	no	yes
leopard shark	cold-blooded	no	no	no
turtle	cold-blooded	yes	no	no
penguin	cold-blooded	no	no	no
eel	cold-blooded	no	no	no
dolphin	warm-blooded	no	no	yes
spiny anteater	warm-blooded	yes	yes	yes
gila monster	cold-blooded	yes	yes	no



Model M3:

train err = 0%,

test err = 30%

Lack of training records at the leaf nodes for making reliable classification

Humans, elephants, and dolphins are misclassified:

Warm-blooded + not hibernate => non-mammals

Only one training record (eagle) with such characteristics

Notes on Overfitting

- Overfitting results in decision trees that are **more complex** than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors

Practical Issues of Classification

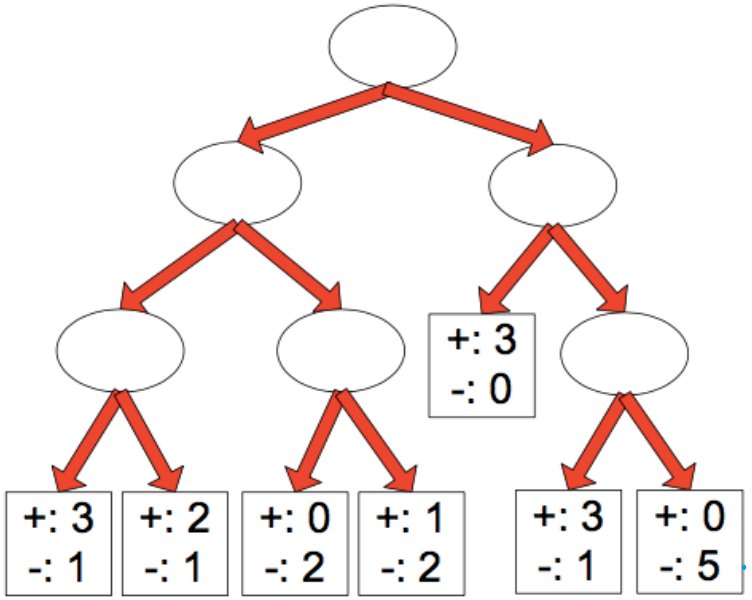
Estimating the Test Error of a Classifier

Methods for Estimating Generalization Errors

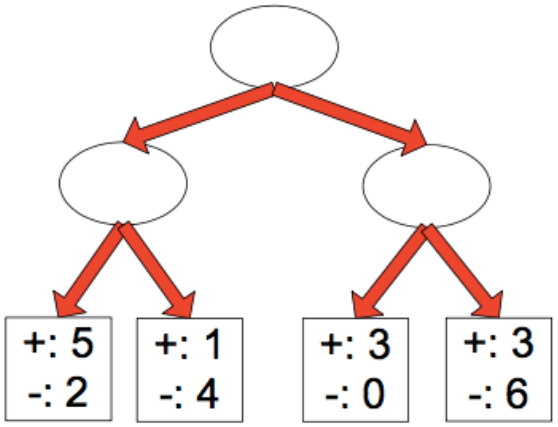
- Using the Resubstitution Estimate
  - Assume training set is a good representation of the true population (i.e., assume that the training error is a good estimator of the test error)
- Incorporating Model Complexity
  - Occam's razor
  - Pessimistic Estimate
  - Minimum Description Length (MDL)

Resubstitution Estimate

- Assume training set is a good representation of overall data
- Use training error (re-substitution error) as an optimistic estimate of generalization error
- Select the model that produces the lowest training error rate as its final model.



Decision Tree, T_L



Decision Tree, T_R

Which tree do you choose?

Occam's Razor

- 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 that it was fitted accidentally by errors in data

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

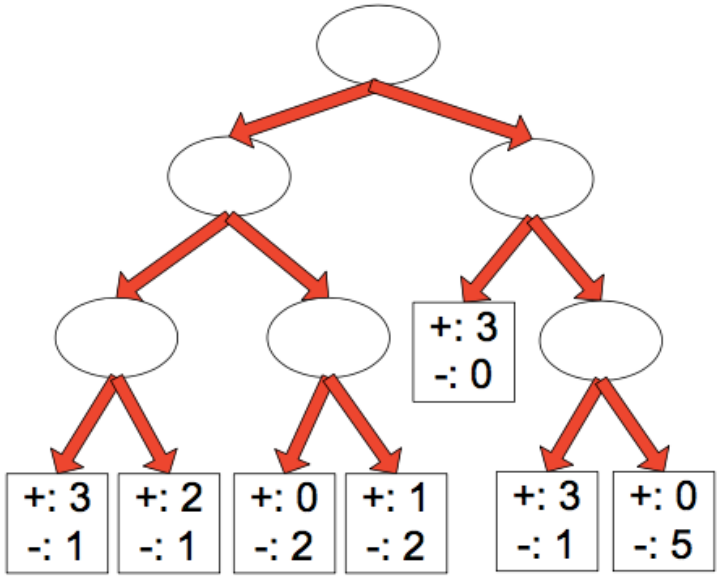
Pessimistic Estimate

- Compute generalization error as the sum of training error and a penalty term for model complexity
- Given a decision tree node t
  - $n(t)$: number of training records classified by t
  - $e(t)$: misclassification error of node t
  - Training error of tree T:

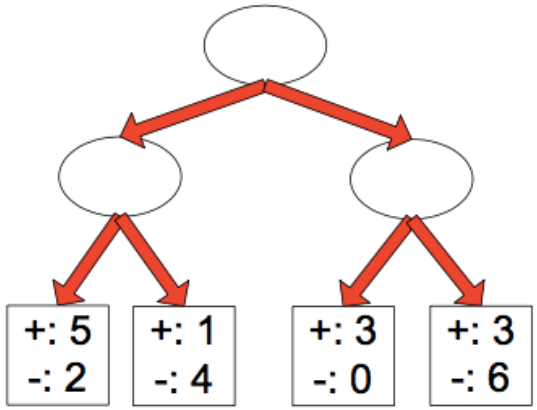
$$e'(T) = \frac{\sum_i [e(t_i) + \Omega(t_i)]}{\sum_i n(t_i)} = \frac{e(T) + \Omega(T)}{N}$$

  - Ω: is the cost of adding a node
  - N: total number of training records

Pessimistic Estimate



Decision Tree, T_L



Decision Tree, T_R

$e(T_L) = 4/24$

$e(T_R) = 6/24$

Cost of adding a node

$\Omega = 1$

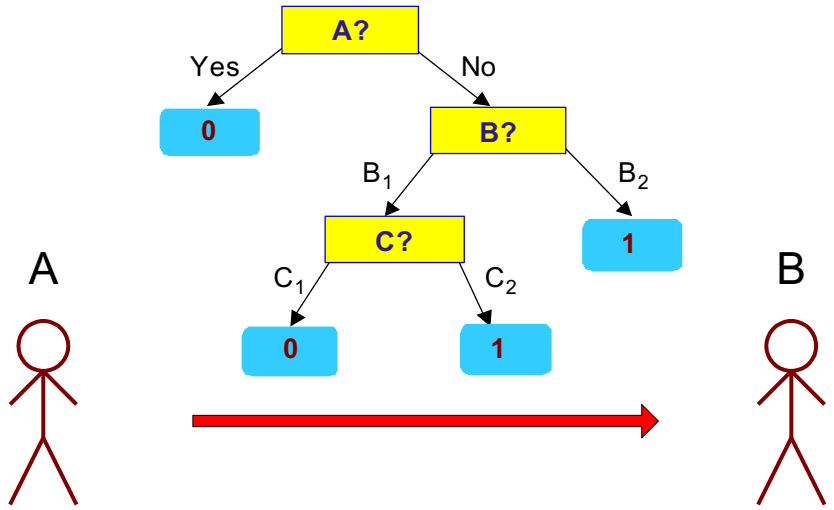
$e'(T_L) = (4 + 7 \times 1)/24 = 0.458$

$e'(T_R) = (6 + 4 \times 1)/24 = 0.417$

$$e'(T) = \frac{\sum_i [e(t_i) + \Omega(t_i)]}{\sum_i n(t_i)} = \frac{e(T) + \Omega(T)}{N}$$

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1

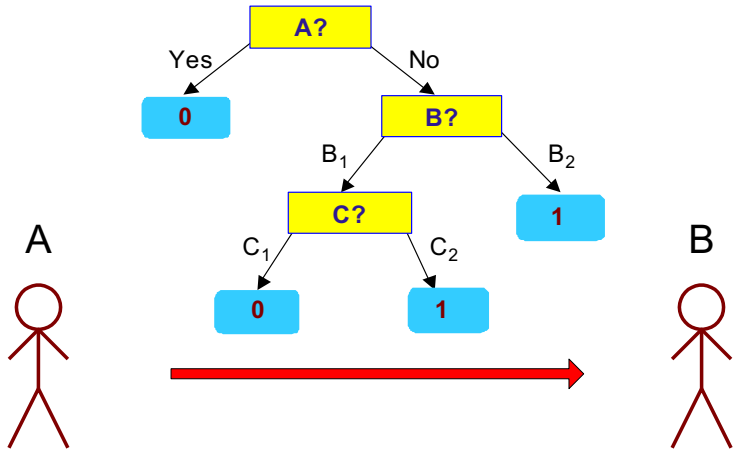


X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- C(Model,Data) := Cost(Data|Model) + Cost(Model)
  - Cost is the number of bits needed for encoding
  - Search for the least costly model
- Cost(Data|Model) encodes the misclassification errors
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding

Minimum Description Length (MDL)

- C(Model,Data) := Cost(Data|Model) + Cost(Model)
- Example with decision trees:
  - There are 8 attributes, and 2 classes: 1 and 0.
  - Internal tree nodes are encoded by the attribute number, and leaves are encoded by the class number.
  - Cost(tree) is the sum of the cost to encode all nodes.
  - Cost(data|tree) uses the classification errors on the training set.
  - Assume that there are 3 classification errors in the data.



Cost(tree) = Cost(Internal nodes) + Cost(leaves)
= 3 x ceil(log2(8 attributes))+ 4 x ceil(log2(2 classes))
= 3 x 3 + 4 x 1 = 9 + 4 = 13 bits

Cost(data|tree) = 3 x ceil(log2(n)), where n is the number of training instances.
Assume that there are 16 instances
Cost(data|tree) = 3 x 4 bits = 12 bits
So Ctree,data) = 13 bits + 12 bits = 25 bits.

This is just one way of computing the cost of a model. Notice that it barely incorporates knowledge about the structure of the tree.

Practical Issues of Classification

Dealing with Overfitting in Decision Tree Classifiers

How to Address Overfitting

- 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

How to Address Overfitting

- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes of the decision 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 MDL for post-pruning

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

Pessimistic error = (10 + 1*0.5)/30 = 10.5/30

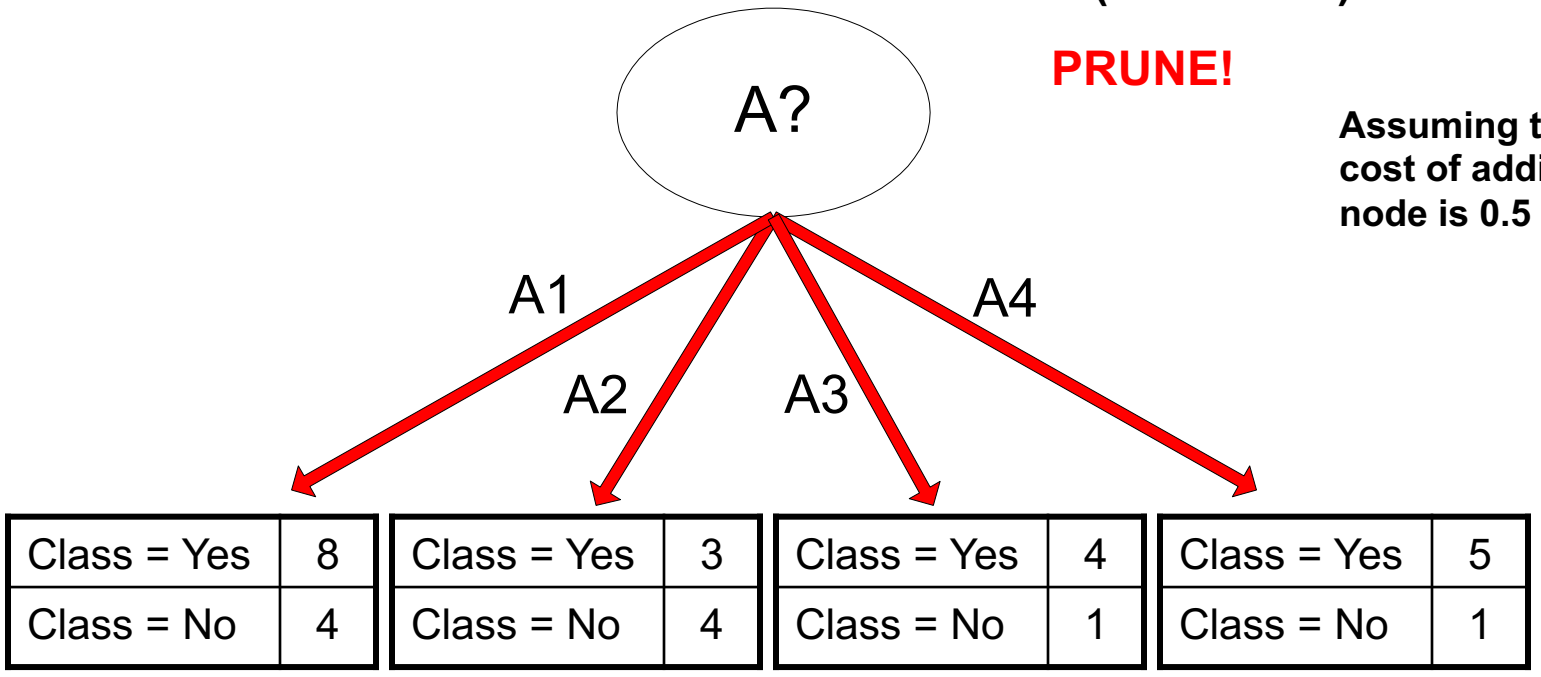
Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!

Assuming that the cost of adding a node is 0.5



Classification

Evaluating the Performance of Classifiers

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

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

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation…

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

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Height Example Data

Name	Gender	Height	Output1	Output2
Kristina	F	1.6m	Short	Medium
Jim	M	2m	Tall	Medium
Maggie	F	1.9m	Medium	Tall
Martha	F	1.88m	Medium	Tall
Stephanie	F	1.7m	Short	Medium
Bob	M	1.85m	Medium	Medium
Kathy	F	1.6m	Short	Medium
Dave	M	1.7m	Short	Medium
Worth	M	2.2m	Tall	Tall
Steven	M	2.1m	Tall	Tall
Debbie	F	1.8m	Medium	Medium
Todd	M	1.95m	Medium	Medium
Kim	F	1.9m	Medium	Tall
Amy	F	1.8m	Medium	Medium
Wynette	F	1.75m	Medium	Medium

Confusion Matrix Example for Non-Binary Classification

Using height data example with Output1 as actual class and Output2 as predicted class

ACTUAL CLASS	PREDICTED CLASS Short	Medium	Tall
Short	0	4	0
Medium	0	5	3
Tall	0	1	2

Accuracy = (5+2) / (4+5+3+1+2) = 47%

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?

- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

Evaluating the Performance of a Classifier

- Model Selection
  - Performed during model building
  - Purpose is to ensure that the model is not overly complex (to avoid overfitting)
  - Need to estimate the generalization error

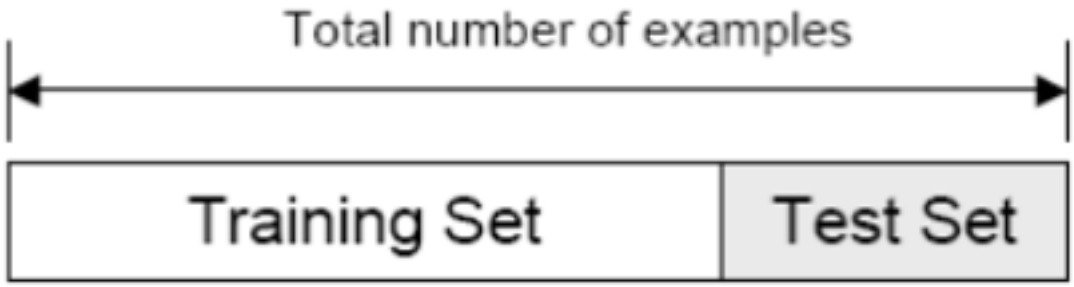
- Model Evaluation
  - Performed after the model has been constructed
  - Purpose is to estimate the performance of the classifier **on previously unseen data** (e.g. test set).

Methods for Classifier Evaluation

- There are several ways of evaluating the performance of a classifier on unseen data:
  - Holdout
  - Bootstrap
  - K-fold cross-validation (k-fold CV)
  - Leave-one-out cross validation (LOOCV)

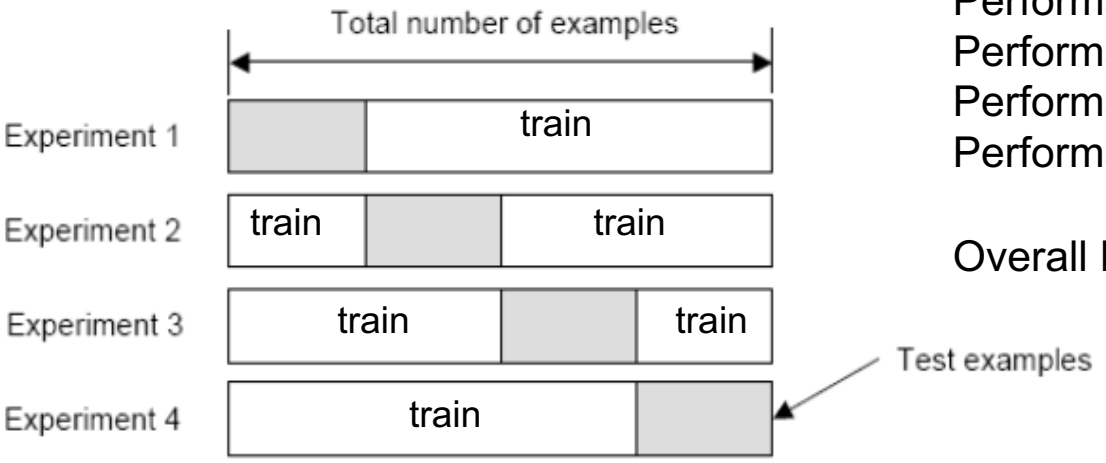
Methods for Classifier Evaluation

- **Holdout Method**
  - Reserve x% for training and (100-x)% for testing
  - In problems with small datasets, we may not be able to afford the “luxury” of setting aside a portion of the dataset for testing.
  - Since it is a single train-and-test experiment, the holdout estimate for the classifier error rate will be misleading if we happen to get an “unfortunate” split



Methods for Classifier Evaluation

- **k-fold Cross Validation (k-fold CV)**
  - Partition the data set into k disjoint subsets or folds
  - Train in the k-1 fold, **test on the remaining fold**
  - The final estimate of k-fold CV is the average performance obtained in the test sets of all k rounds
  - In general, k-fold CV is run multiple times (repeats) and the estimates of these repeats are also averaged.



4-fold cross-validation

Performance on Fold 1 = e1
Performance on Fold 2 = e2
Performance on Fold 3 = e3
Performance on Fold 4 = e4

Overall Performance = mean(e1, e2, e3, e4)

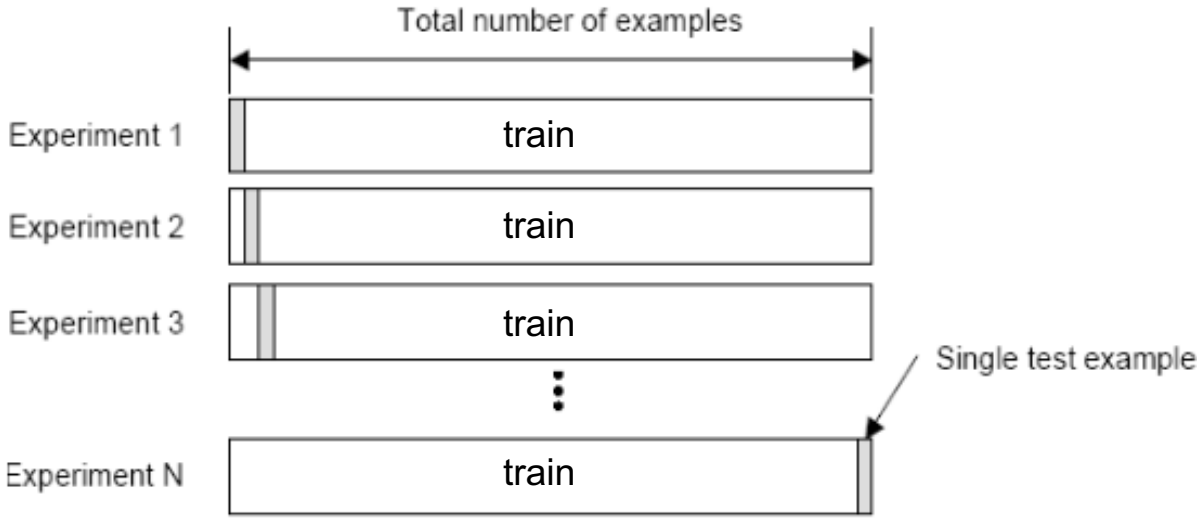
Methods for Classifier Evaluation

- **Pseudo-code of k-fold Cross Validation (k-fold CV):**
  - Partition the data set into k disjoint folds $F = \{F_1, F_2, \ldots F_k\}$
  - For i in [1, 2, …, k]:
    - $Model_i$ = Train your model using the data contained in the folds $F - F_i$
    - e_i = Evaluate the performance of $Model_i$ on F_i
  - Return $(e_1 + e_2 + \ldots + e_k) / k$

- **Important Observations:**
  - **The final model will be trained on the complete dataset!!**
  - **Once you have obtained a k-fold CV estimate, you cannot go back and use that estimate to tune your model. If you do this, then running k-fold CV will not be a good estimator of the performance of the model on unseen data.**

Methods for Classifier Evaluation

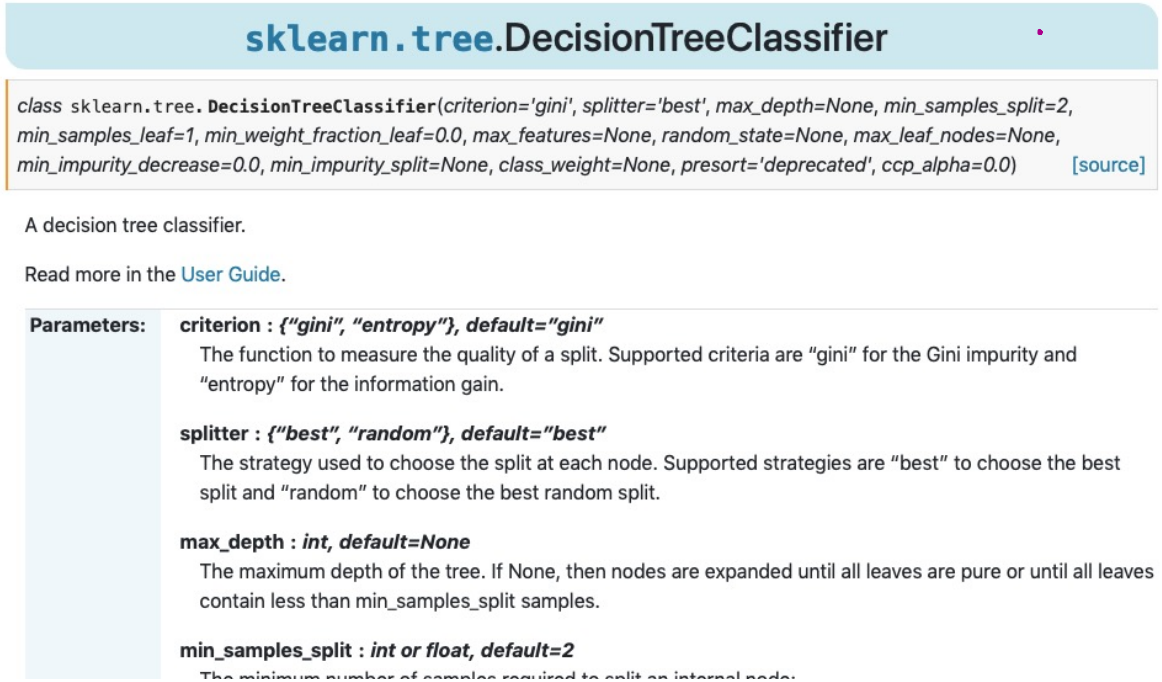
- **Leave-one-out cross validation (LOOCV):**
  - Like k-fold cross validation with k = n
  - Each test set contains only one record
  - Utilizes as much data as possible for training
  - Test sets are mutually exclusive and effectively cover the entire data set



Leave-one-out

Model Hyperparameters

- **Hyperparameters** are parameters of the model that you set before learning takes place.
- Examples of hyperparameters for decision trees are:
  - The maximum depth of the tree
  - The minimum number of samples of a node
- How do we choose the values of the hyperparameters for a model?



sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.` **`DecisionTreeClassifier`**(*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort='deprecated', ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*

We can use the data itself to find out the "best" hyperparameter values!!

This figure shows the hyperparameters of the DecisionTreeClassifier of the scikit-learn library in Python

Methods for Hyperparameter Tuning

- A model's hyperparameters can be tuned from the data set itself. We can use k-fold CV for this.
- **Grid Search Cross-Validation (Grid Search CV) Procedure**:
  1. Split your data set into 2 disjoint training and test subsets.
  2. Define a set *C* of hyperparameter values over which you want to see which one is the best.
  3. For each value v in *C*:

     perf[v] = Run normal k-fold cross validation **on the training set only** to obtain an estimate of the performance using hyperparameter value v
  4. Find the v such that perf[v] is maximum. This is the best hyperparameter value in C.
  5. $Model_v$ = Train your model using *v* as a hyperparameter value on the complete training set.
  6. Evaluate the performance of $Model_v$ on the test set.
  7. Train the final model on the whole dataset

Methods for Hyperparameter Tuning

- Grid Search CV for hyperparameter tuning:

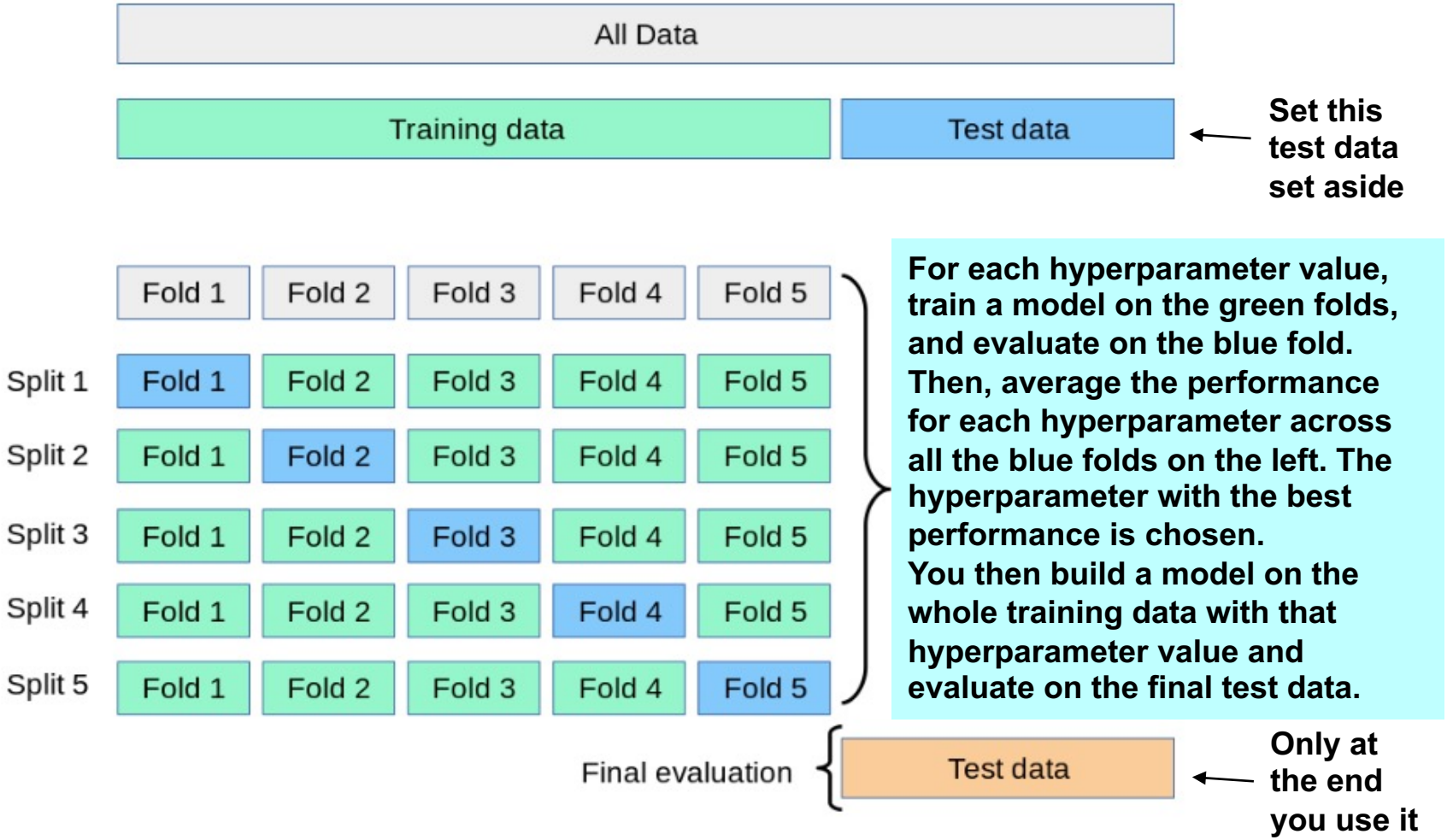


Figure modified from https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation. Accessed on Feb 11, 2020

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Confidence Interval for Accuracy

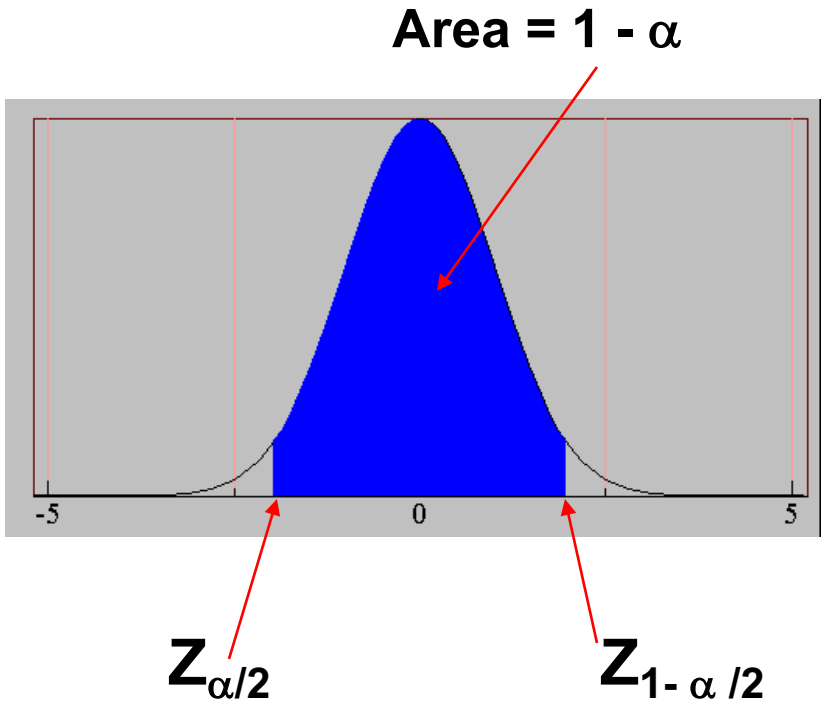
- Prediction can be regarded as a Bernoulli trial
  - A Bernoulli trial has 2 possible outcomes
  - Possible outcomes for prediction: correct or wrong
  - Collection of Bernoulli trials has a Binomial distribution:
    - $x \sim Bin(N, p)$ x: number of correct predictions
    - e.g: Toss a fair coin 50 times, how many heads would turn up?
      Expected number of heads = $N \times p = 50 \times 0.5 = 25$

- Given x (# of correct predictions) or equivalently, acc=x/N, and N (# of test instances),

  Can we predict p (true accuracy of model)?

Confidence Interval for Accuracy

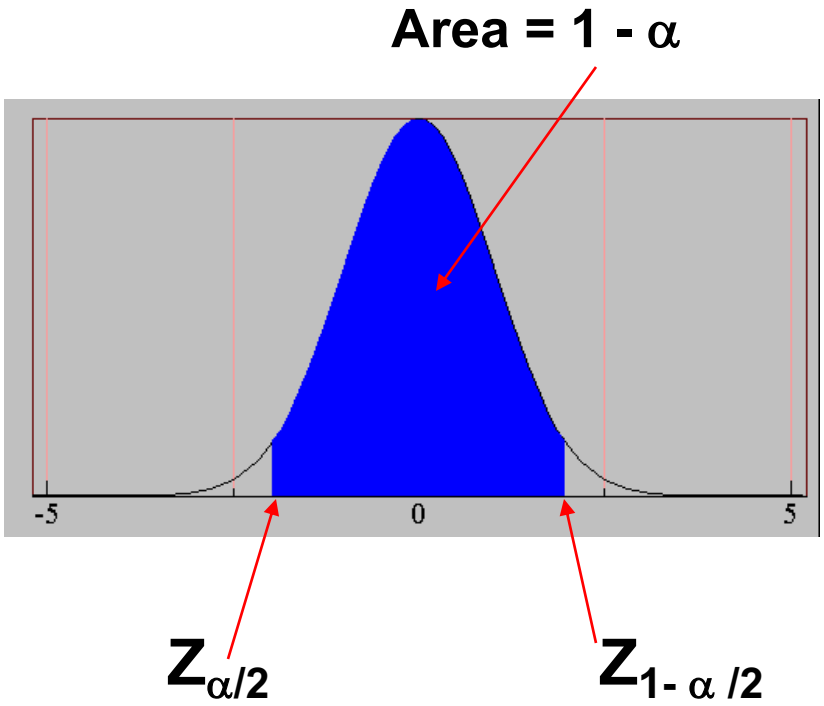
- For large test sets (N > 30), we can approximate the
  - The true accuracy *p* of a predictor has a **normal** distribution with mean p and variance p(1-p)/N



$$P(Z_{\alpha/2} < \frac{acc - p}{\sqrt{p(1-p)/N}} < Z_{1-\alpha/2}) = 1 - \alpha$$

Confidence Interval for Accuracy

- For large test sets (N > 30),
  - acc has a **normal** distribution with mean p and variance p(1-p)/N



- 1-alpha confidence interval for the true accuracy:

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

Where $\hat{p}$ is the observed accuracy, N is the number of rows, and $Z_{\alpha/2}$ is the value in the x axis such that the cumulative normal distribution is equal to alpha/2.

Confidence Interval for Accuracy

- (1-alpha) confidence interval for the true accuracy:

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

- We can compute this directly in R:

```r
prop.test(x=80, n = 100, conf.level = 0.95, correct = FALSE)
```

```
        1-sample proportions test without continuity correction

data:  80 out of 100, null probability 0.5
X-squared = 36, df = 1, p-value = 1.973e-09
alternative hypothesis: true p is not equal to 0.5
95 percent confidence interval:
 0.7111708 0.8666331
sample estimates:
  p
0.8
```

Confidence Interval for Accuracy

- Consider a model that produces an accuracy of 80% when evaluated on 100 test instances:
  - N=100, p_hat = 0.8
  - Let 1-α = 0.95 (95% confidence)
  - From probability table, $Z_{\alpha/2}$=1.96

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

N	50	100	500	1000	5000
p(lower)	0.670	0.711	0.763	0.774	0.789
p(upper)	0.888	0.866	0.833	0.824	0.811

RESULT: The confidence interval for the true accuracy p is [71.1%, 86.6%]

Confidence Interval for Accuracy

- Another way to compute a confidence Interval for p is using **Wilson's score interval**:

$$p = \frac{2 \times N \times acc + Z^2_{\alpha/2} \pm \sqrt{Z^2_{\alpha/2} + 4 \times N \times acc - 4 \times N \times acc^2}}{2(N + Z^2_{\alpha/2})}$$

Confidence Interval for Accuracy

- Given models with the following observed accuracy values:
  - Model M1: accuracy = 80%, tested on 100 instances
  - Model M2: accuracy = 80%, tested on 500 instances
  - Model M3: accuracy = 80%, tested on 1000 instances

- Can we find confidence intervals for their true accuracy?

Test of Significance

- Given two models:
  - Model M1: accuracy = 85%, tested on 30 instances
  - Model M2: accuracy = 75%, tested on 5,000 instances

- Can we say M1 is better than M2?
  - How much confidence can we place on accuracy of M1 and M2?
  - Can the difference in performance measure be explained as a result of random fluctuations in the test set?

Test of Significance

- Assume that you have two brands of refrigerators A and B, which are each guaranteed for 1 year. In a random sample of 50 fridges of brand A, 12 failed before the warranty expired. An independent sample of 60 brand B fridges also had 12 failures within 1 year. Estimate the true difference (p_1-p_2) between proportions of failures with a 98% confidence level.

1-α	$Z_{alpha/2}$
0.99	2.58
0.98	**2.33**
0.95	1.96
0.90	1.65

$$(\hat{p}_1 - \hat{p}_2) \pm z_{\alpha/2}\sqrt{\frac{p_1 q_1}{n_1} + \frac{p_2 q_2}{n_2}} = (0.24 - 0.2) \pm 2.33\sqrt{\frac{(0.24)(0.76)}{50} + \frac{(0.2)(0.8)}{60}}$$

$$= 0.04 \pm 0.1851$$

$$= [-0.1451, 0.2251]$$

Test of Significance

- Assume that you have two brands of refrigerators A and B, which are each guaranteed for 1 year. In a random sample of 50 fridges of brand A, 12 failed before the warranty expired. An independent sample of 60 brand B fridges also had 12 failures within 1 year. Estimate the true difference (p_1-p_2) between proportions of failures with a 98% confidence coefficient.

```r
prop.test(x = c(12, 12), n = c(50, 60), conf.level = 0.98, correct = FALSE)
```

```
        2-sample test for equality of proportions without continuity correction

data:  c(12, 12) out of c(50, 60)
X-squared = 0.25581, df = 1, p-value = 0.613
alternative hypothesis: two.sided
98 percent confidence interval:
 -0.1448629  0.2248629
sample estimates:
prop 1 prop 2
  0.24   0.20
```

Comparing Performance of 2 Models

- Given two models, say M1 and M2, evaluated in 2 independent test sets which is better?
  - M1 is tested on D1 (size=n_1), found error rate = e_1
  - M2 is tested on D2 (size=n_2), found error rate = e_2
  - Test whether the observed difference between e_1 and e_2 is statistically significant
  - Assume D1 and D2 are independent
  - If n_1 and n_2 are sufficiently large, then they can be approximated with normal distributions:

$$e_1 \sim N(\mu_1, \sigma_1)$$

$$e_2 \sim N(\mu_2, \sigma_2)$$

  - Variance of the error rate (approximate):

$$\hat{\sigma}_i = \frac{e_i(1-e_i)}{n_i}$$

Comparing Performance of 2 Models

- To test if performance difference is statistically significant: d = e1 – e2
  - d ~ $N(d_t,\sigma_t)$ where d_t is the true difference
  - Since D1 and D2 are **independent**, their **variance** adds up:

$$\sigma_t^2 = \sigma_1^2 + \sigma_2^2 \cong \hat{\sigma}_1^2 + \hat{\sigma}_2^2$$

$$= \frac{e1(1-e1)}{n1} + \frac{e2(1-e2)}{n2}$$

  - At (1-α) confidence level, $d_t = d \pm Z_{\alpha/2}\hat{\sigma}_t$

An Illustrative Example

$$\sigma_t^2 = \sigma_1^2 + \sigma_2^2 \approx \hat{\sigma_1}^2 + \hat{\sigma_2}^2 = \frac{e_1(1-e_1)}{n_1} + \frac{e_2(1-e_2)}{n_2}$$

- Given: M1: n1 = 30, e1 = 0.15
  M2: n2 = 5000, e2 = 0.25
- d = |e2 – e1| = 0.1 (2-sided test)

$$\hat{\sigma}_d^2 = \frac{0.15(1-0.15)}{30} + \frac{0.25(1-0.25)}{5000} = 0.0043$$

- At 95% confidence level, $Z_{\alpha/2}$=1.96

$$d_t = 0.100 \pm 1.96 \times \sqrt{0.0043} = 0.100 \pm 0.128$$

=> Interval contains 0 => difference may not be statistically significant
(otherwise you could reject the hypothesis that the true error rates are equal)

Summary

- Decision Tree Induction
  - How a decision tree works
  - Building a decision tree
  - Measures for selecting the best split
  - Algorithm for decision tree induction
- Model Overfitting
  - Overfitting due to Presence of Noise
  - Overfitting due to Lack of Representative examples
  - Overfitting and the multiple comparison procedure
  - Estimation of Generalization errors
  - Handling overfitting in decision tree induction
- Evaluating the performance of a classifier
  - Holdout method
  - K-fold Cross-validation
  - LOOCV
- Methods for comparing classifiers
  - Estimating a Confidence Interval for accuracy
  - Comparing the performance of two models