Unit 3: Data Mining Classification: Basic Concepts, Decision Trees, and Model Evaluation

Unit 3

Section 1: Basic Classification

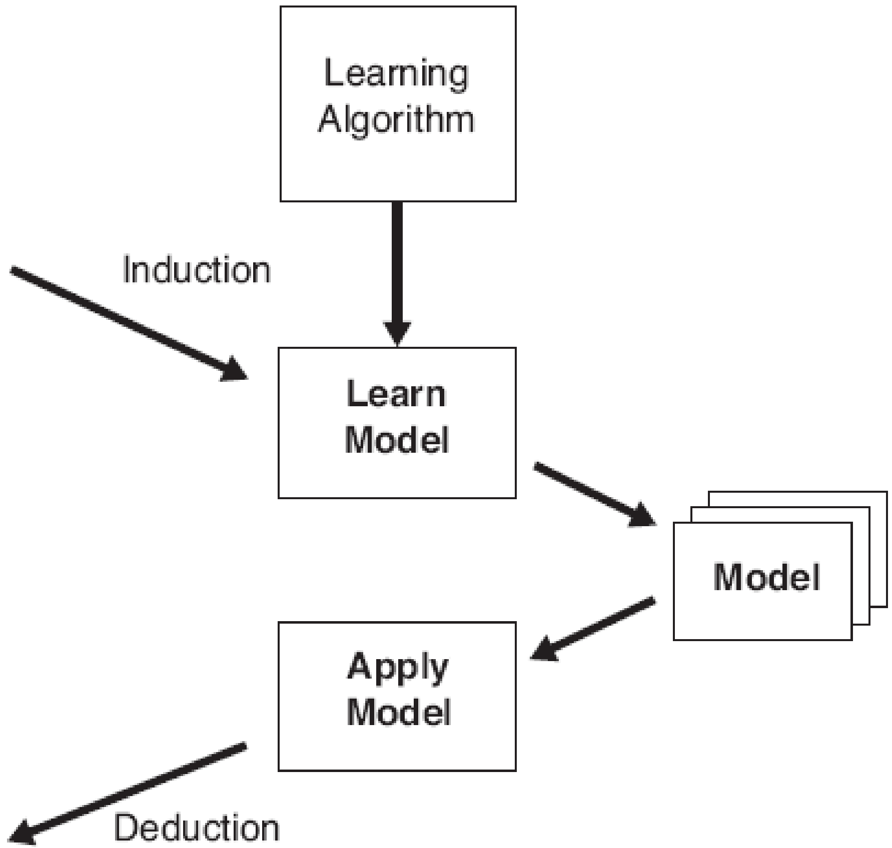
Classification: Definition

- 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.
- Probabilistic vs. non-probabilistic models

Illustrating Classification Task

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



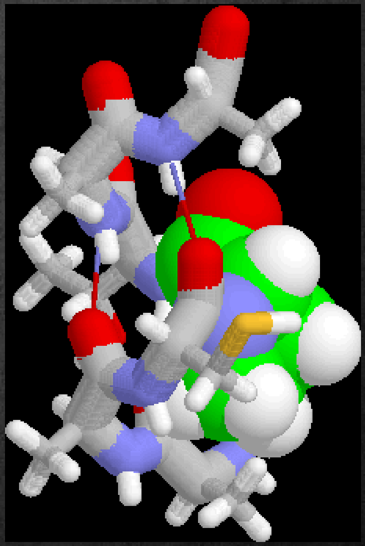
Test Set

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	?

Figure 4.3. General approach for building a classification model.

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



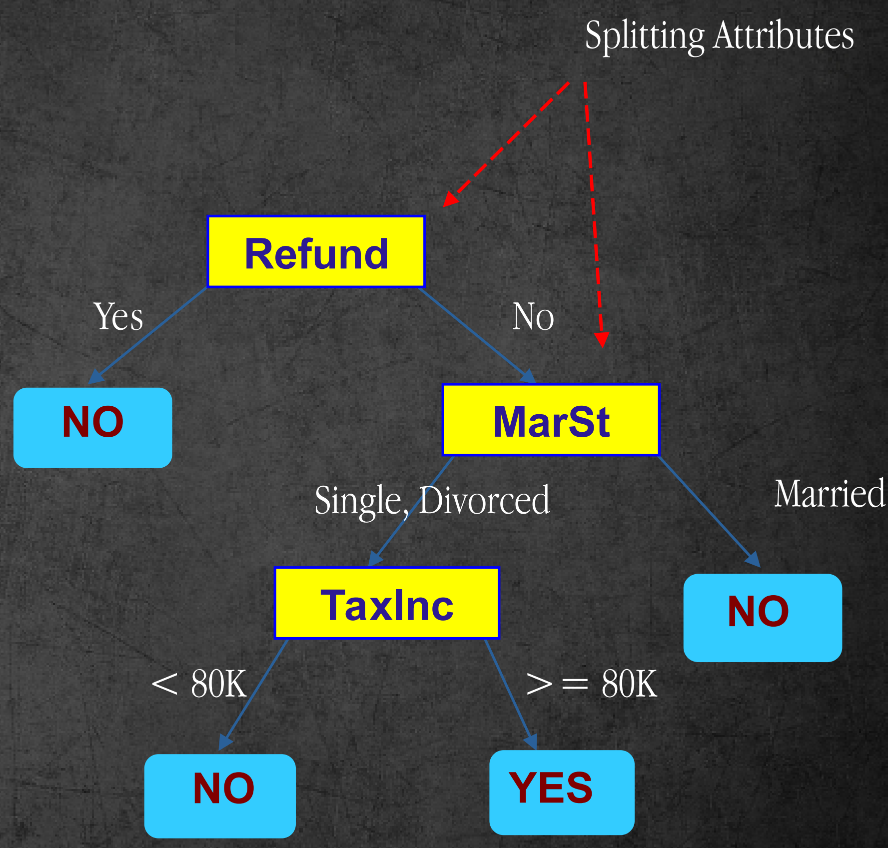
Classification Techniques

- Decision Trees
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
- Instance-based methods
- Many others…

Example of a 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

Training Data

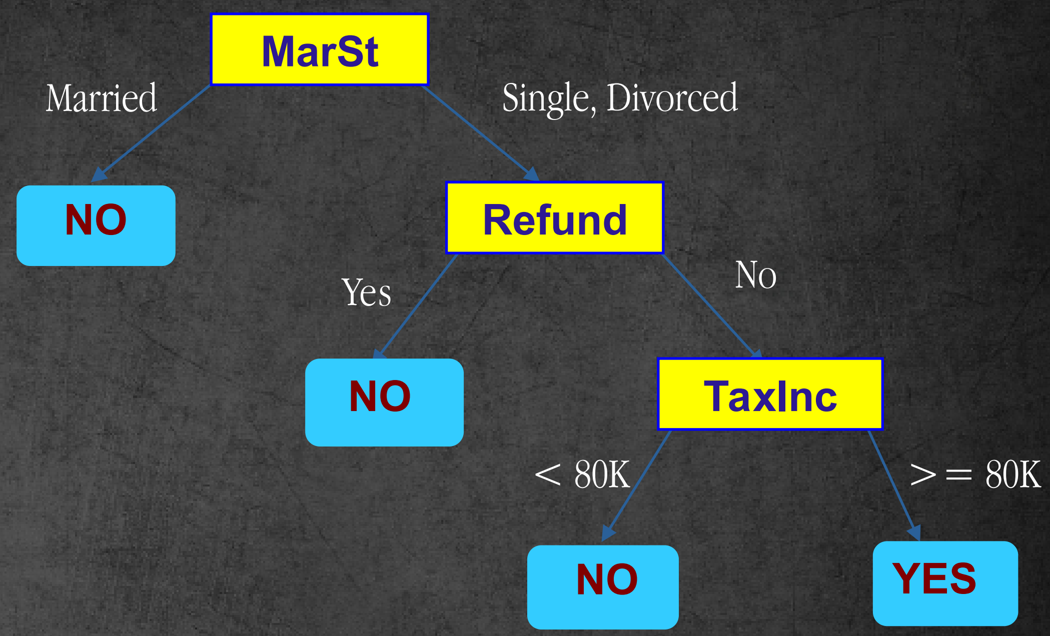


Model: Decision Tree

Another Example of 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



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

Decision Tree Classification Task

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	?

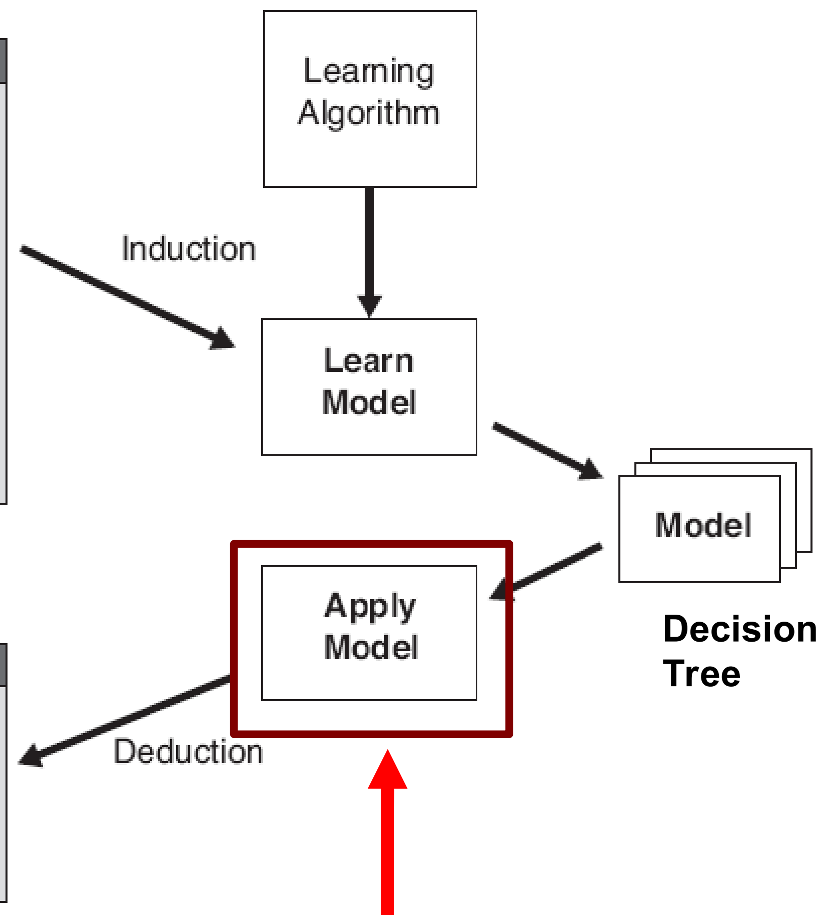
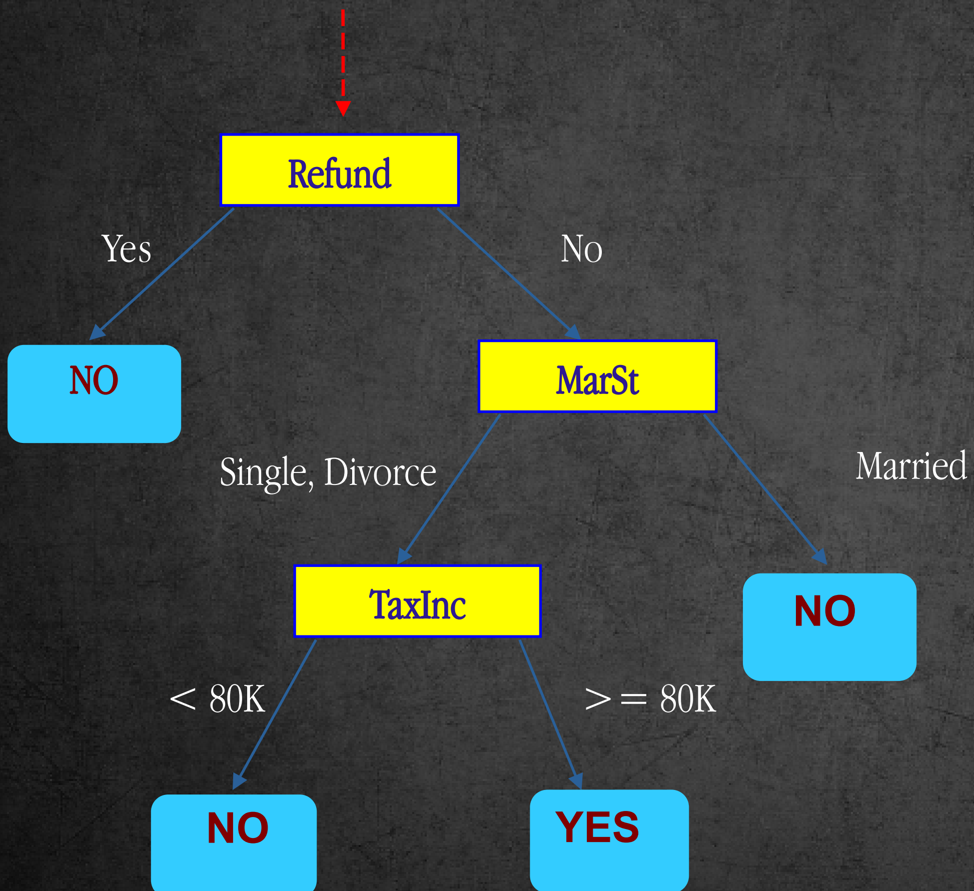


Figure 4.3. General approach for building a classification model.

Apply Model to Test Data

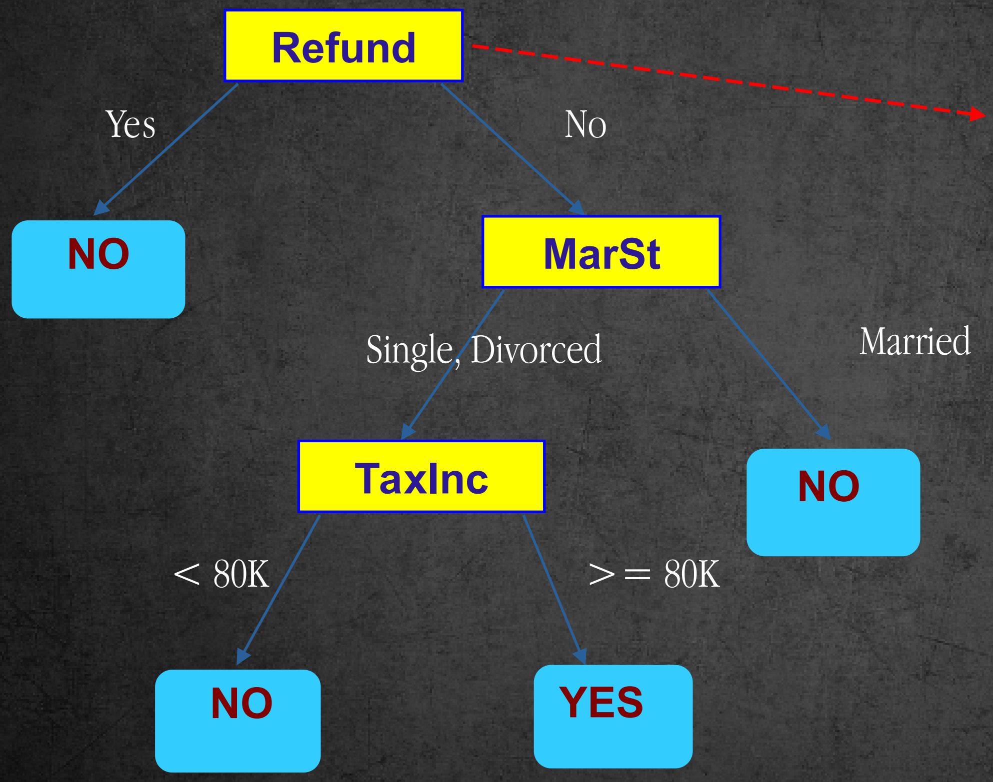
Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

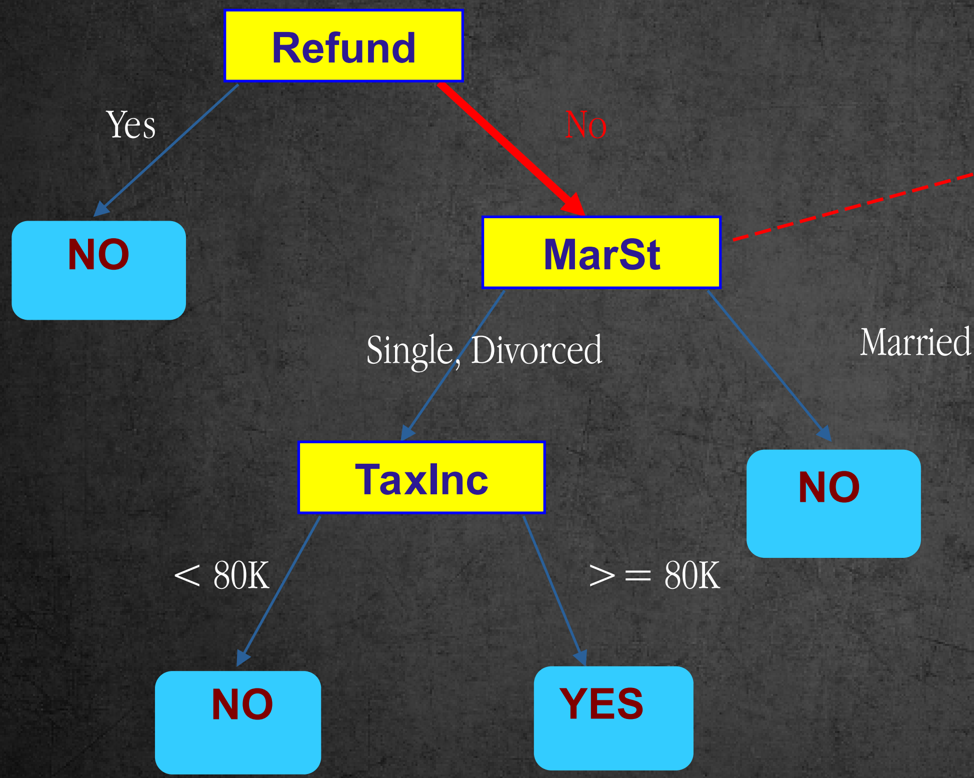
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

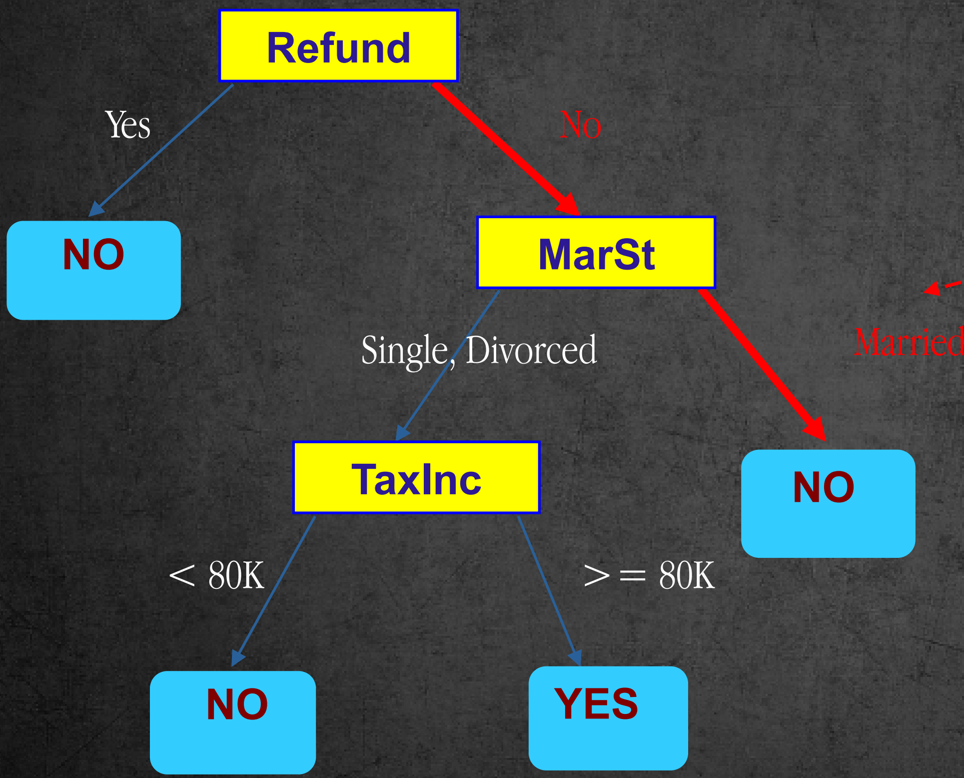
Apply Model to Test Data



TEST DATA

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

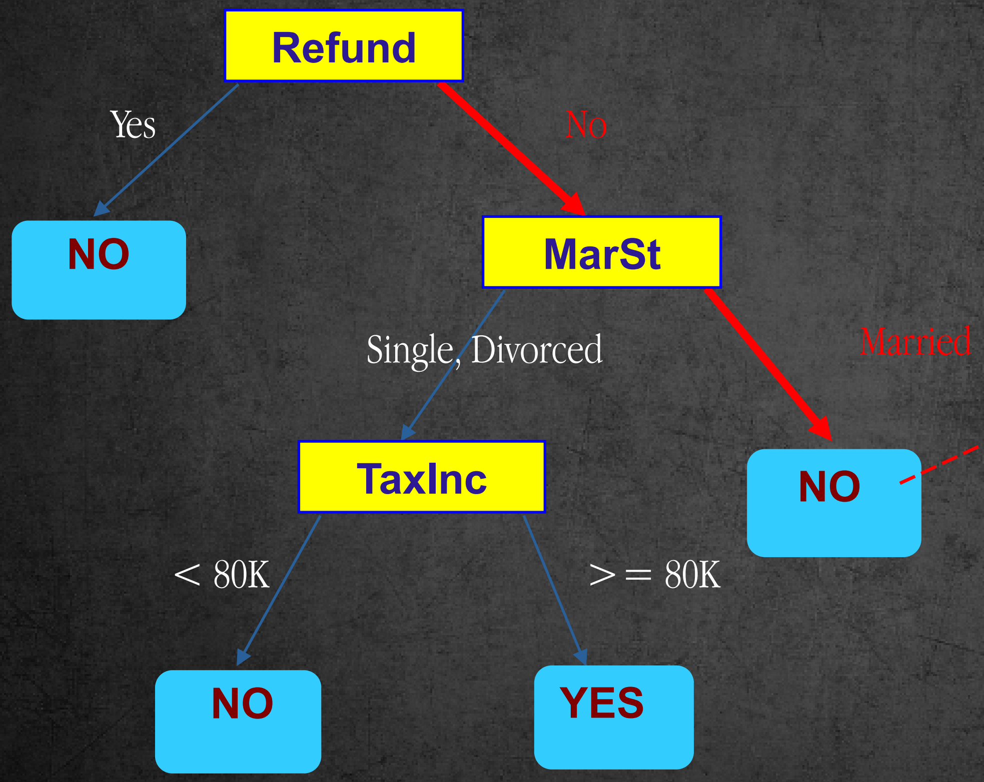
Apply Model to Test Data



Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



Test Data

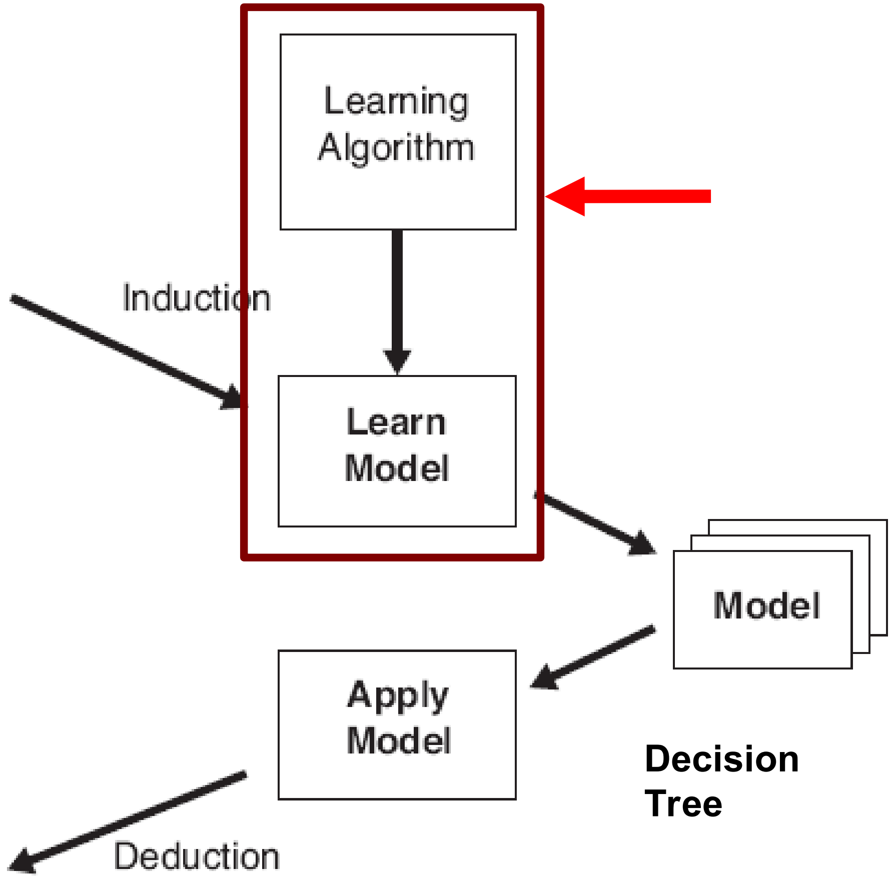
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Assign Cheat to "No"

Decision Tree Classification Task

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	?

Figure 4.3. General approach for building a classification model.

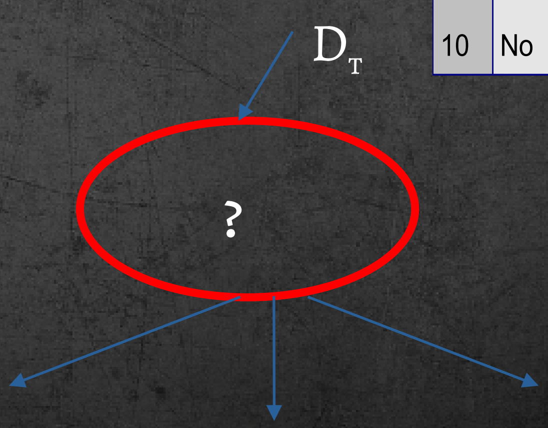
Decision Tree Induction

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

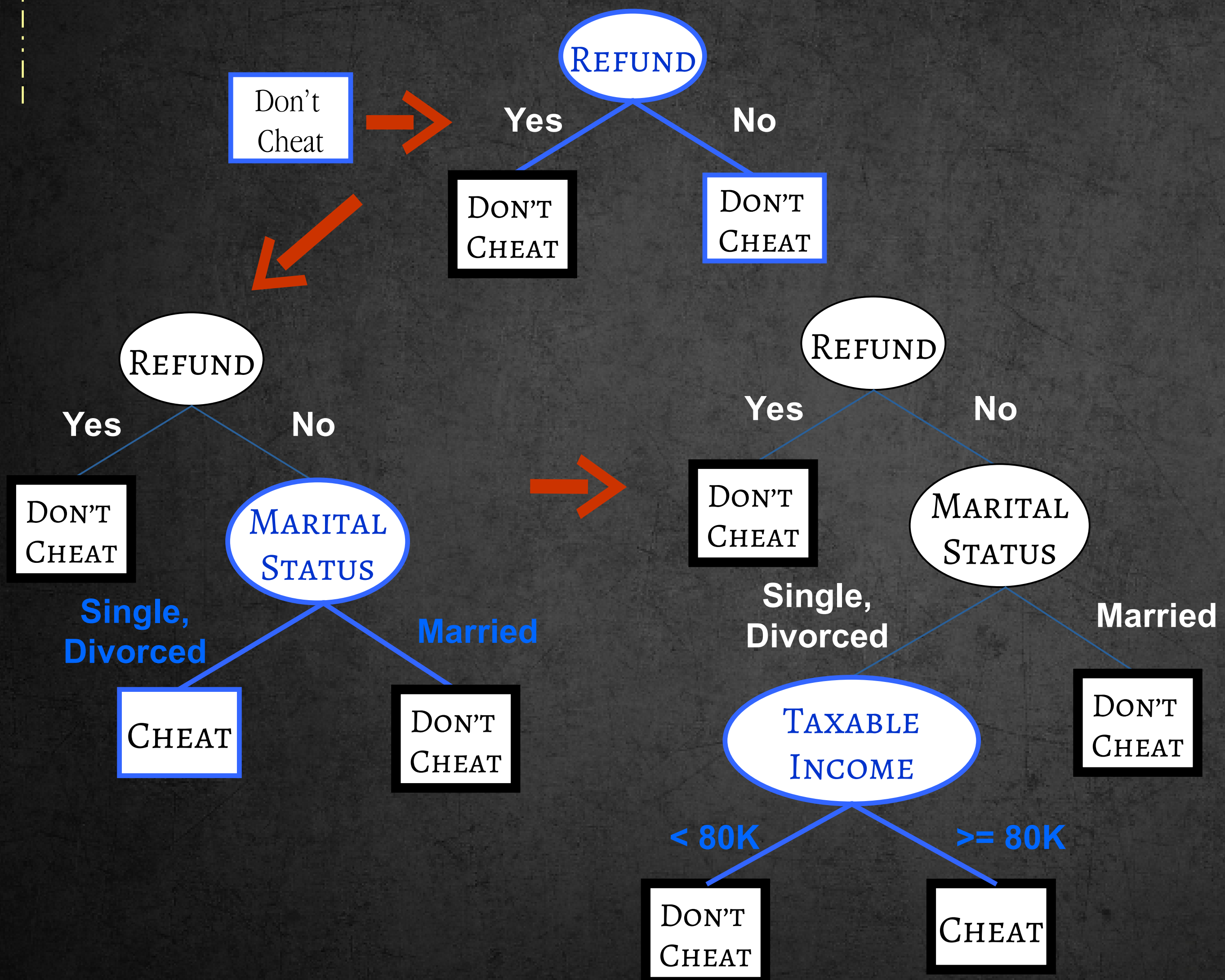
General Structure of Hunt's Algorithm

- Let D_t be the set of training records 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 D_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If Dt contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively 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



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

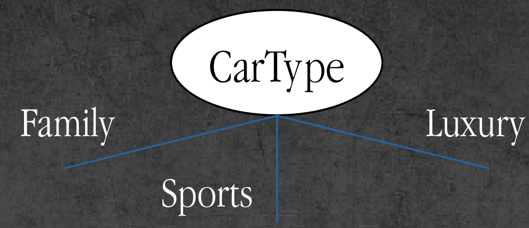
How to Specify Test Condition?

- 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

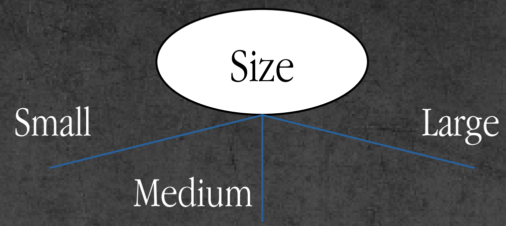
- Multi-way split: Use as many partitions as distinct values.



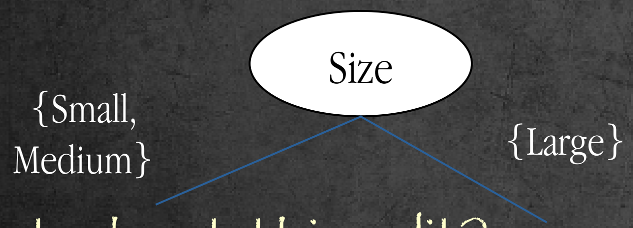
- Binary split: Divides values into two subsets. Need to find optimal partitioning.

Splitting Based on Ordinal Attributes

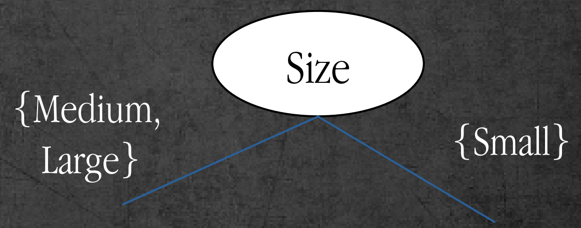
- Multi-way split: Use as many partitions as distinct values.



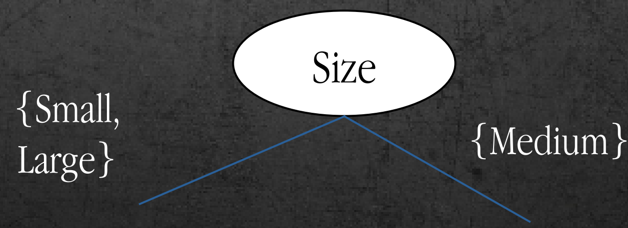
- Binary split: Divides values into two subsets. Need to find optimal partitioning.



OR



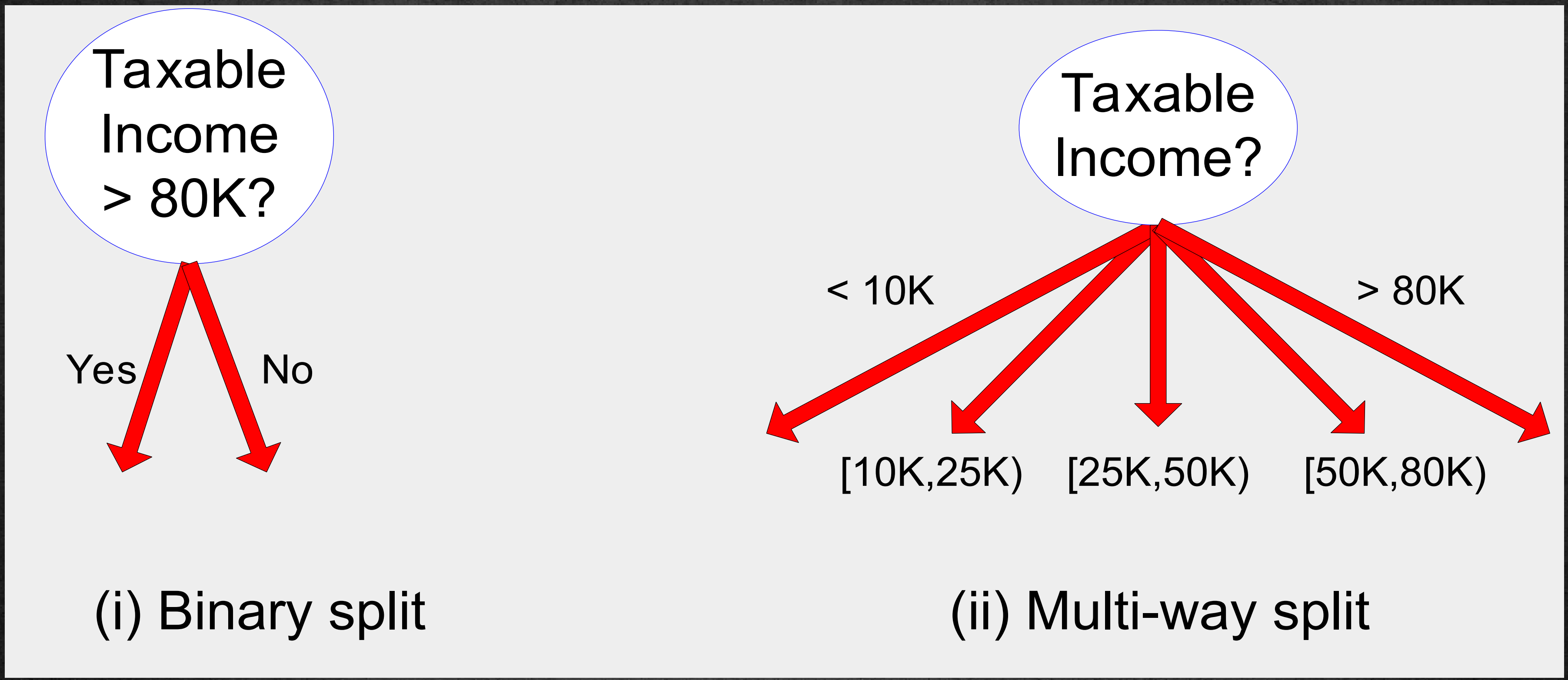
- What about this split?



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 (consider all vallues): (A < v) or (A >= v)
    - consider all possible splits and finds the best cut
    - can be more compute intensive
    - In some case too many splits

Splitting Based on Continuous Attributes

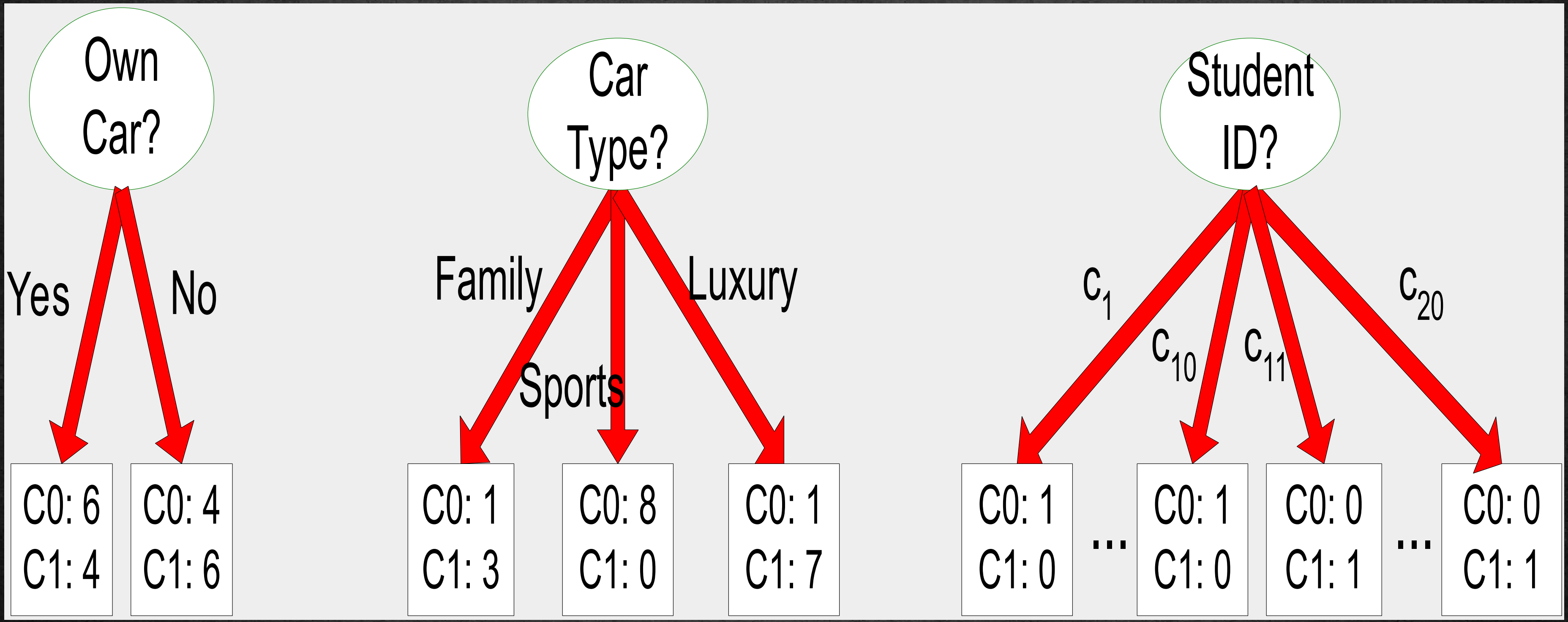


Tree Induction

- 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 distribution are preferred
- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	Homogeneous, Low degree of impurity

Measures of Node Impurity

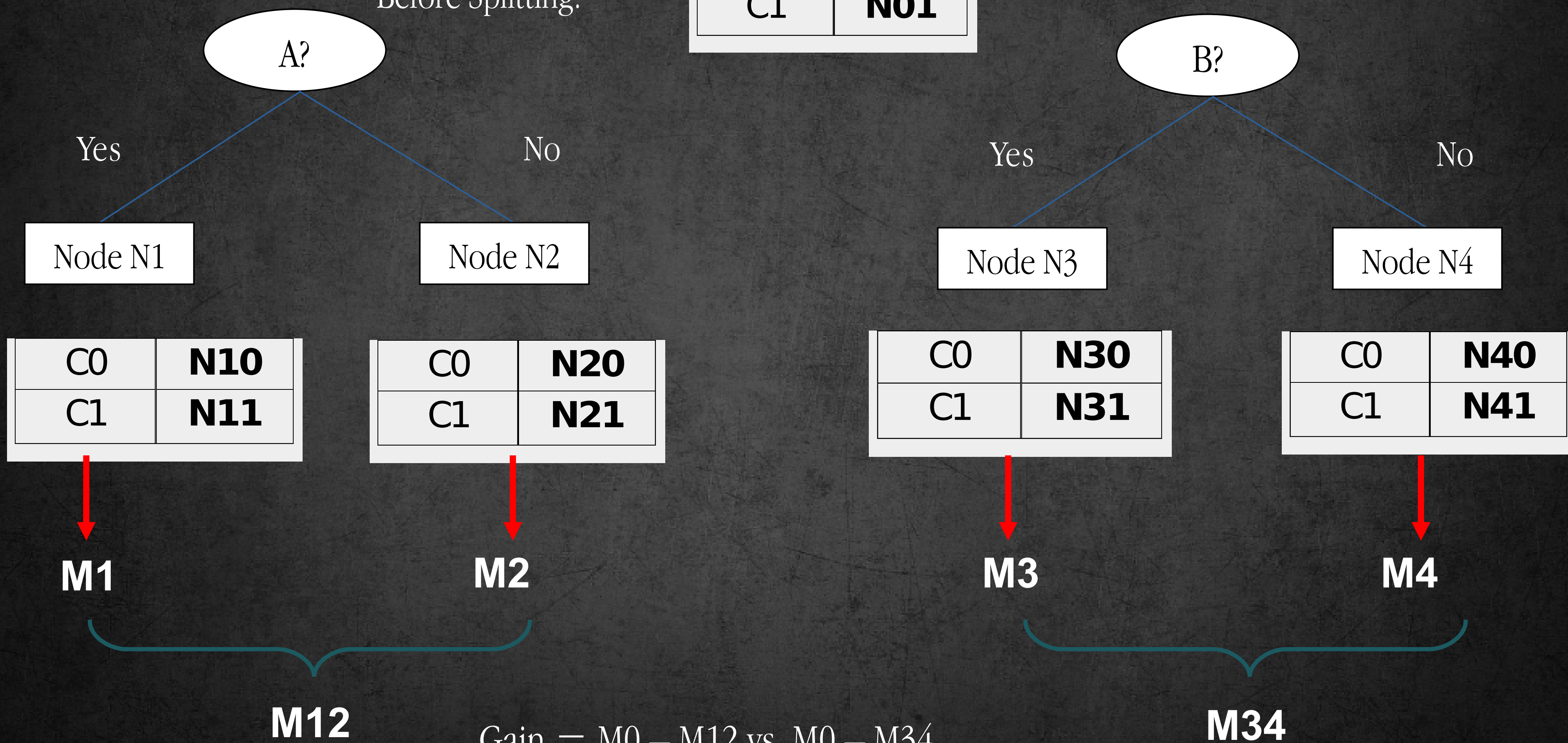
- Gini Index
- Entropy
- Misclassification error

How to Find the Best Split

Before Splitting:

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

→ **M0**



$$\text{Gain} = M0 - M12 \text{ vs } M0 - M34$$

Measure of Impurity: GINI

- Gini Index for a given node t:

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

(NOTE: $p(j|t)$ 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

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

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:

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

n_i = number of records at child i

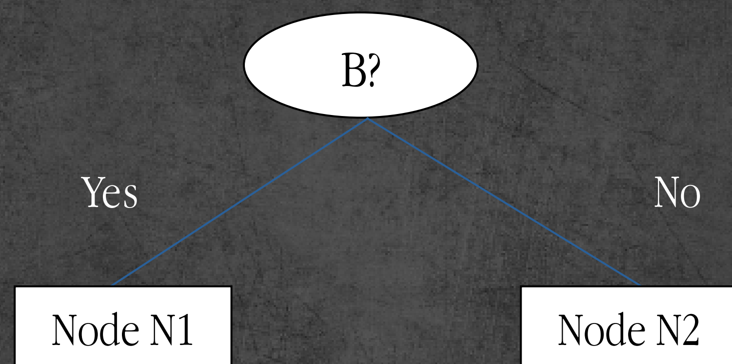
n = number of records at node p

- Best: **Minimum** value

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)
$= 1 – (5/7)^2 – (2/7)^2$
$= 0.4081$

Gini(N2)
$= 1 – (1/5)^2 – (4/5)^2$
$= 0.3200$

	N1	N2
C1	5	1
C2	2	4
Gini = 0.3714		

Gini(Children)
$= 7/12 * 0.4081 +$
$5/12 * 0.3200$
$= 0.3714$

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		

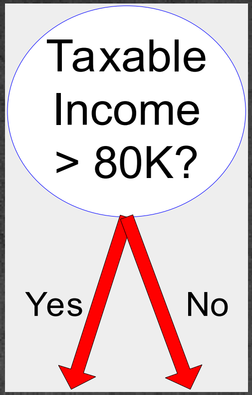
Two-way split
(find best partition of values)

	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...

- 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

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	

Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

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

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

- Measures 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
- Entropy based computations are similar to the GINI index computations
- Best: **Minimum** value

Examples for computing Entropy

$$\text{Entropy}(t) = -\sum_j p(j|t)\log_2 p(j|t)$$

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	**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

Splitting Based on information gain

- 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.

Splitting Based on gain ratio

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

n_i is the 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

Splitting Criteria based on Classification Error

- Classification error at a node t:

$$Error(t) = 1 - max_i P(i|t)$$

- Measures misclassification error made by a node.
  - 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

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

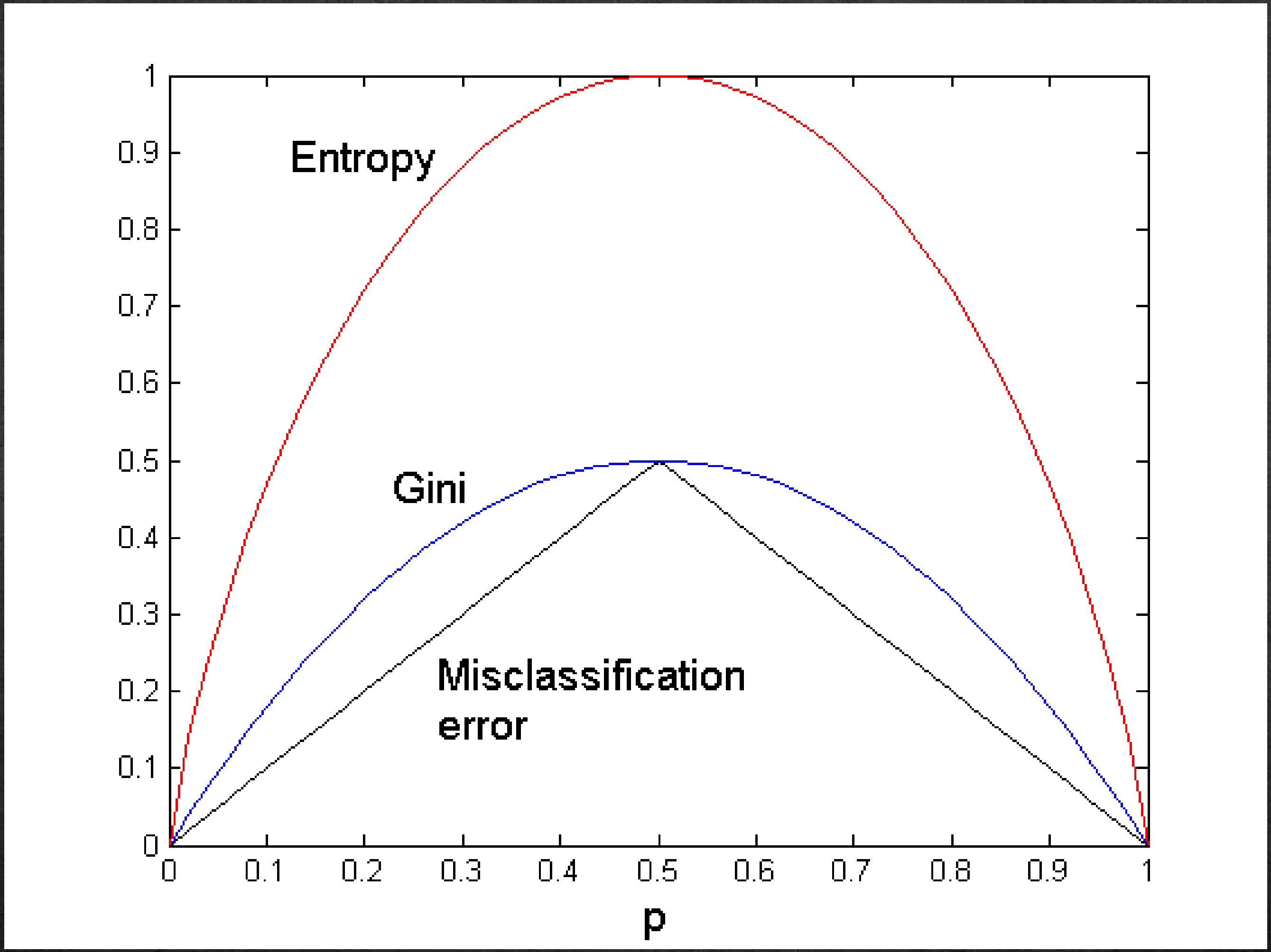
C1	**2**
C2	**4**

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

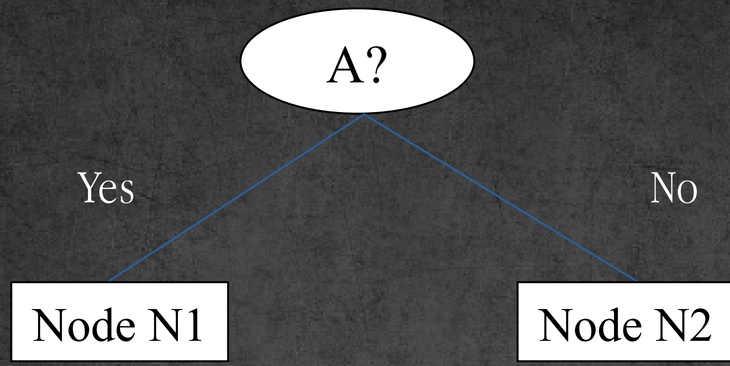
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

- For a two-class problem



Misclassification Error vs. Gini



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

Gini(N1)
$= 1 – (3/3)^2 – (0/3)^2$
$= 0.0000$

Gini(N2)
$= 1 – (4/7)^2 – (3/7)^2$
$= 0.4898$

	N1	**N2**
C1	3	4
C2	0	3
Gini = 0.3427		

Gini(Children)
$= 3/10 * 0.000$
$+ 7/10 * 0.4898$
$= 0.3427$

Gini improves !!

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
  - Unstable

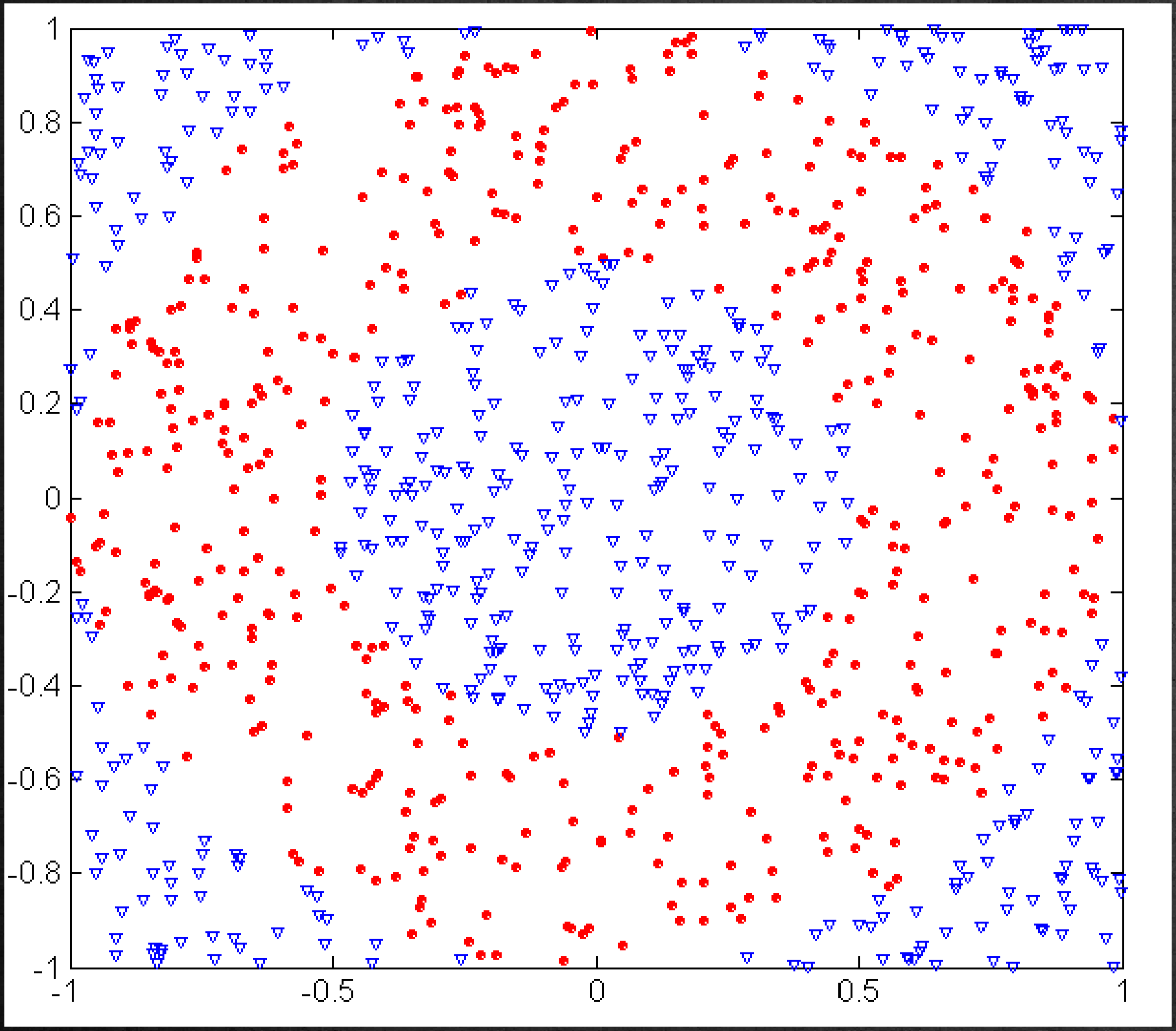
Example: C4.5

- Simple depth-first construction.
- Uses Information Gain
- Sorts Continuous Attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
  - Needs out-of-core sorting.
- You can download the software from: http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz

Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

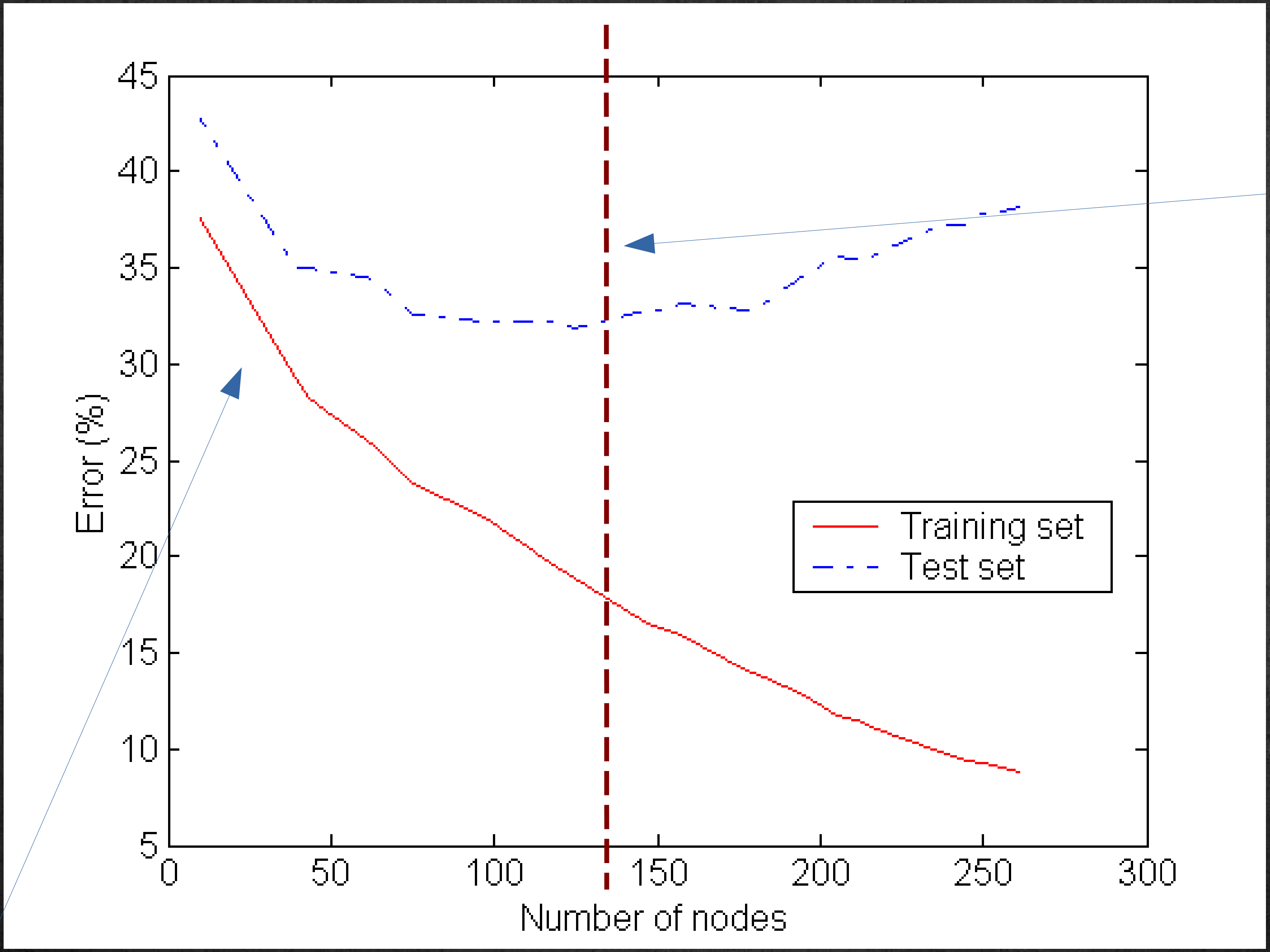
$0.5 <= \text{sqrt}(x_1^2+x_2^2) <= 1$

Triangular points:

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

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

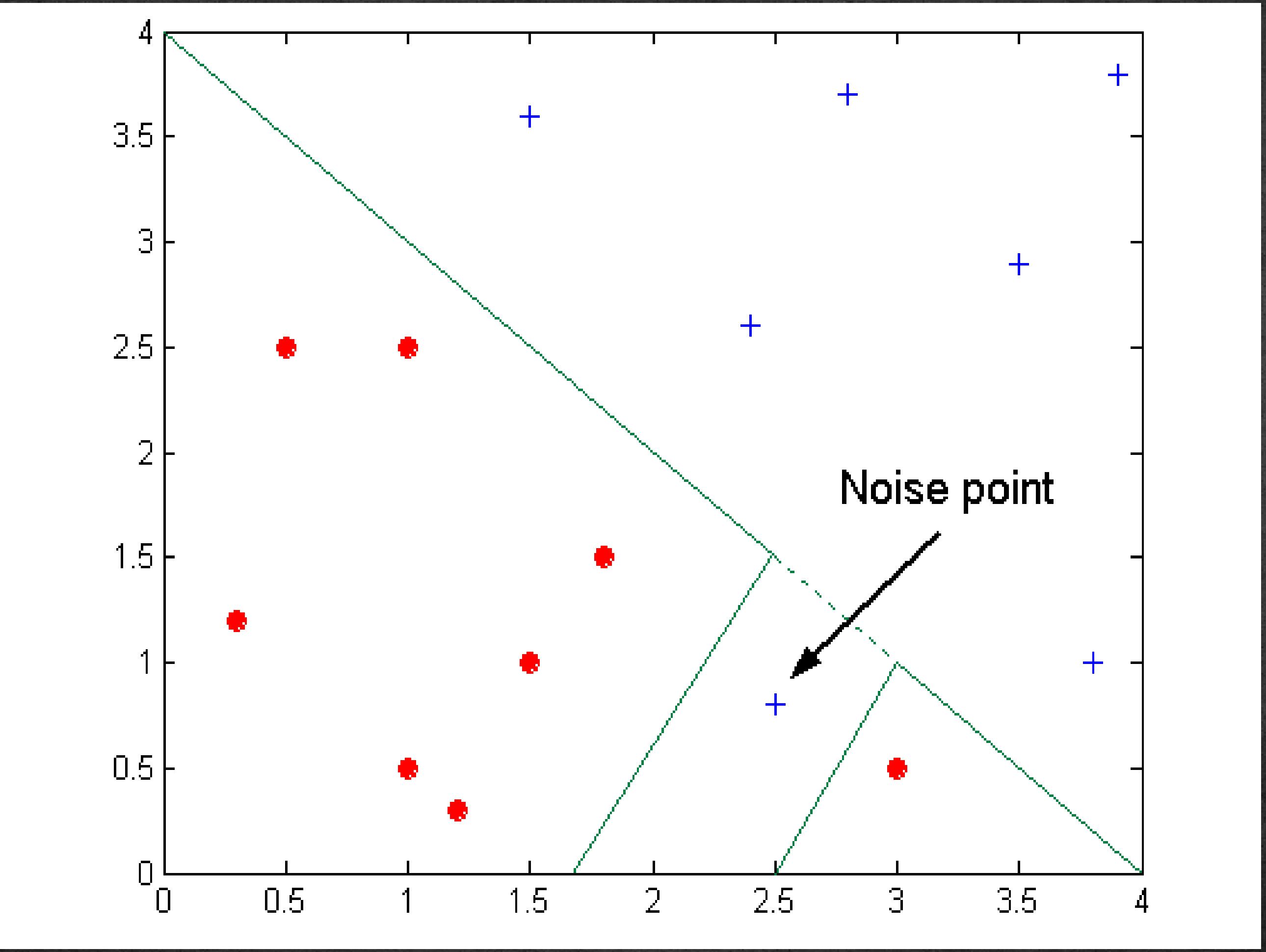
Underfitting and Overfitting



Overfitting

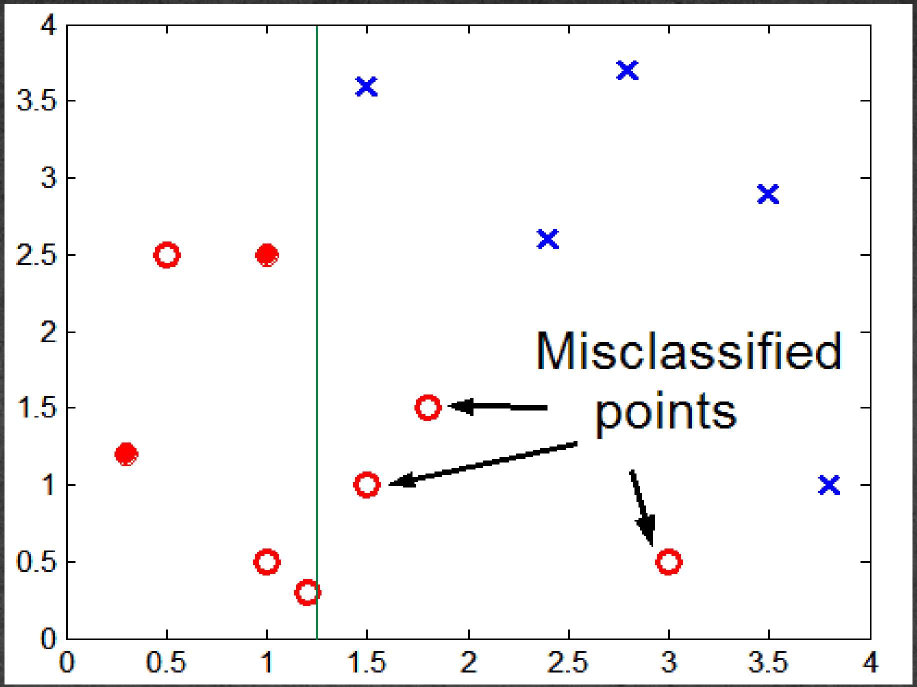
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



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

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

Overfitting

- Curse of overfitting:
  - Related to learning
  - Worse when you learning algorithm is better
  - No matter who hard you try, it's worse

Estimating Generalization Errors

- Re-substitution errors: error on training ($\Sigma e(t)$)
- Generalization errors: error on testing ($\Sigma e'(t)$)
- Methods for estimating generalization errors:
  - Optimistic approach: $e'(t) = e(t)$
  - Pessimistic approach (adds model complexity):
    - For each leaf node: $e'(t) = (e(t)+0.5)$
    - Total errors: $e'(T) = e(T) + N \times 0.5$ (N: number of leaf nodes)
    - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):

      Training error = 10/1000 = 1%

      Generalization error = $(10 + 30 \times 0.5)/1000 = 2.5\%$
  - Reduced error pruning (REP):
    - Uses validation data set to estimate generalization error

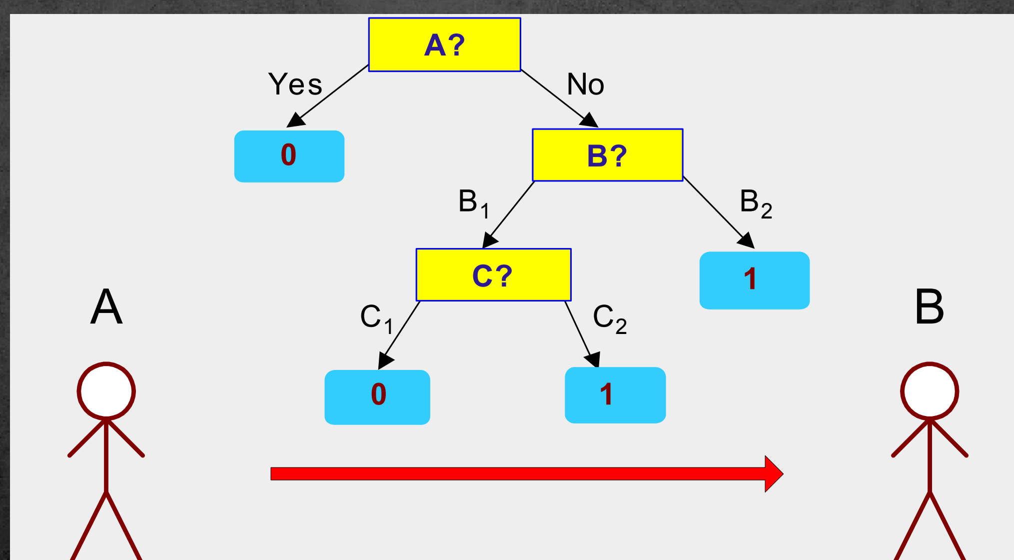
Occam's Razor

- 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
- Problem: Not easy to know which is the simpler model

Minimum Description Length (MDL)

- Cost(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.

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	?

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
    - 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).

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 validation 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	

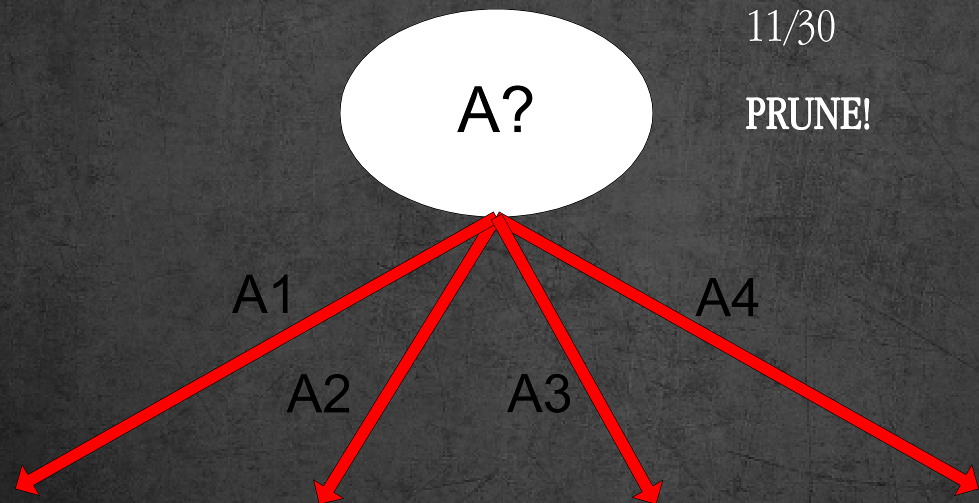
Training Error (Before splitting) = 10/30

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

Training Error (After splitting) = 9/30

Pessimistic error (After splitting) = (9 + 4 x 0.5)/30 = 11/30

PRUNE!



Class = Yes	8
Class = No	4

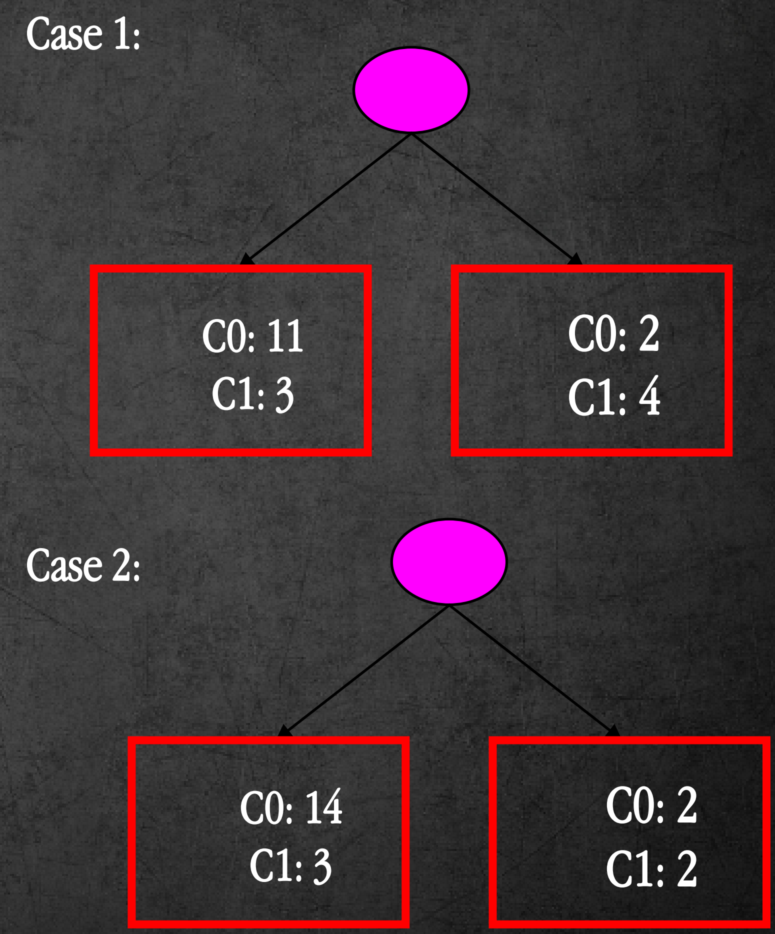
Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Examples of Post-pruning

- Optimistic error?
  - Don't prune for both cases
- Pessimistic error?
  - Don't prune case 1, prune case 2
- Reduced error pruning?
  - Depends on validation set



Handling Missing Attribute Values

- Missing values affect decision tree construction in three different ways:
  - Affects how impurity measures are computed
  - Affects how to distribute instance with missing value to child nodes
  - Affects how a test instance with missing value is classified

Computing Impurity Measure

Tid	Refund	Marital Status	Taxable Income	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	?	Single	90K	Yes

Missing value

Before Splitting:

Entropy(Parent)
$= -0.3 \log(0.3)-(0.7)\log(0.7) = 0.8813$

	Class = Yes	Class = No
Refund=Yes	0	3
Refund=No	2	4
Refund=?	1	0

Split on Refund:

Entropy(Refund=Yes) = 0

Entropy(Refund=No)
$= -(2/6)\log(2/6) - (4/6)\log(4/6) = 0.9183$

Entropy(Children)
$= 0.3\ (0) + 0.6\ (0.9183) = 0.551$

Gain $= 0.9 \times (0.8813 - 0.551) = 0.3303$

Distribute Instances

Tid	Refund	Marital Status	Taxable Income	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

Refund

Yes

Class=Yes	0
Class=No	3

No

Cheat=Yes	2
Cheat=No	4

Tid	Refund	Marital Status	Taxable Income	Class
10	?	Single	90K	Yes

Refund

Yes

Class=Yes	0 + 3/ 9
Class=No	3

No

Class=Yes	2 + 6/ 9
Class=No	4

Probability that Refund=Yes is 3/9

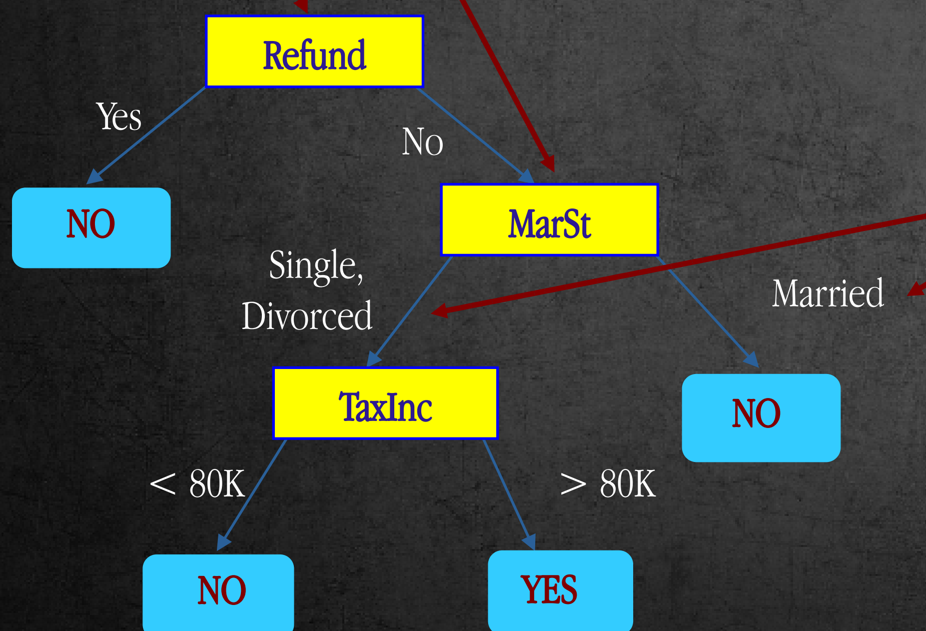
Probability that Refund=No is 6/9

Assign record to the left child with weight = 3/9 and to the right child with weight = 6/9

Classify Instances

New record:

Tid	Refund	Marital Status	Taxable Income	Class
11	No	?	85K	?



	Married	Single	Divorced	Total
Class=No	3	1	0	4
Class=Yes	6/9	1	1	2.67
Total	3.67	2	1	6.67

Probability that Marital Status = Married is 3.67/6.67

Probability that Marital Status ={Single,Divorced} is 3/6.67

Other Issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

Data Fragmentation

- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

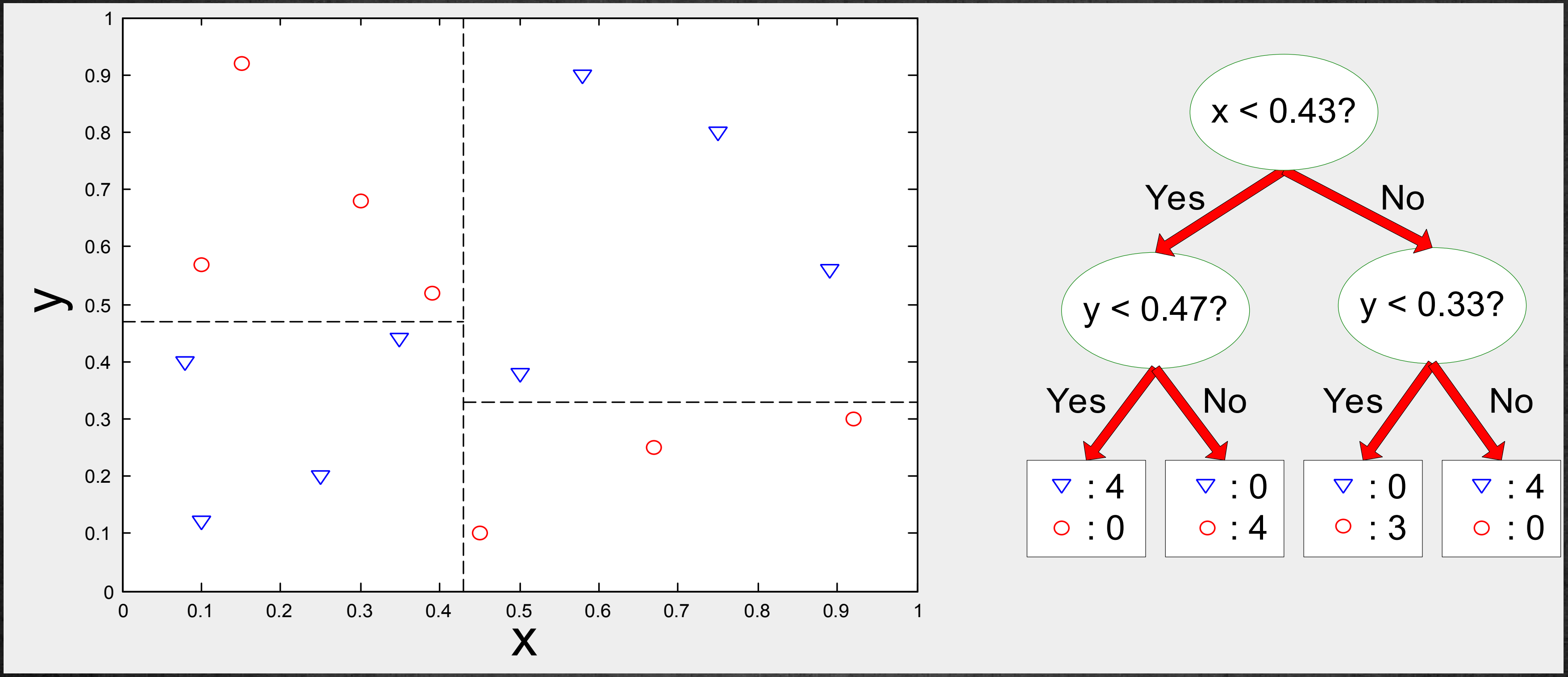
Search Strategy

- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

Expressiveness

- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
    - Example: parity function:
      - Class = 1 if there is an even number of Boolean attributes with truth value = True
      - Class = 0 if there is an odd number of Boolean attributes with truth value = True
    - For accurate modeling, must have a complete tree
- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

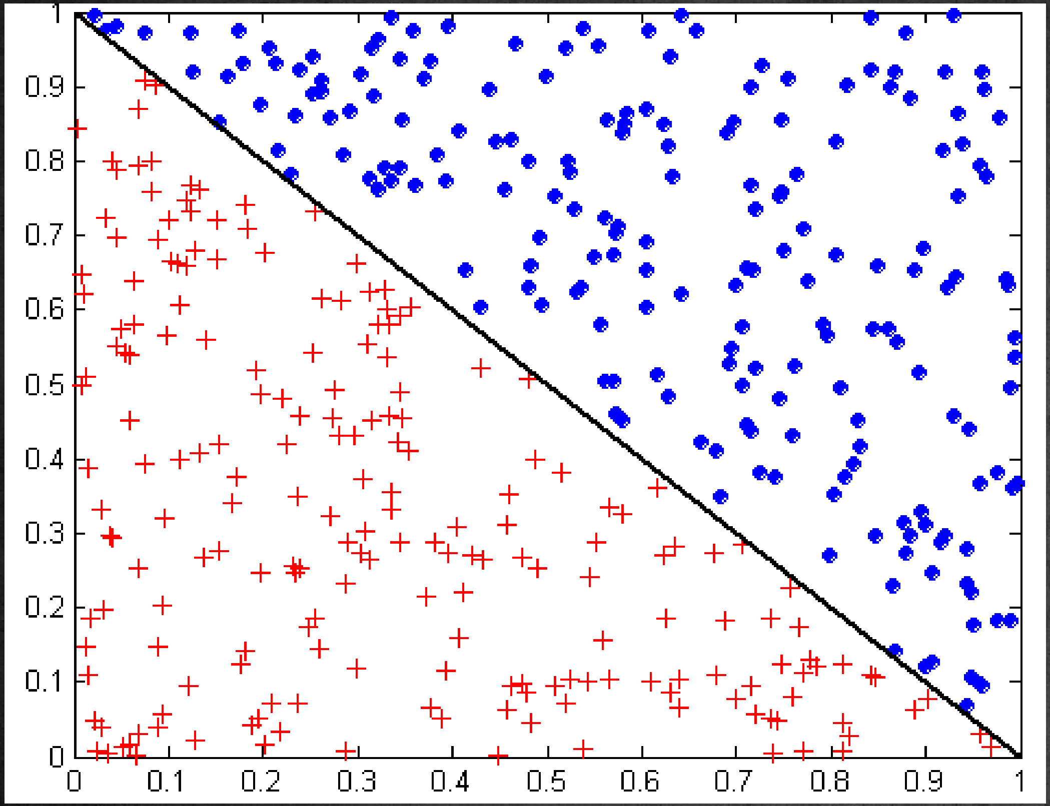
Decision Boundary

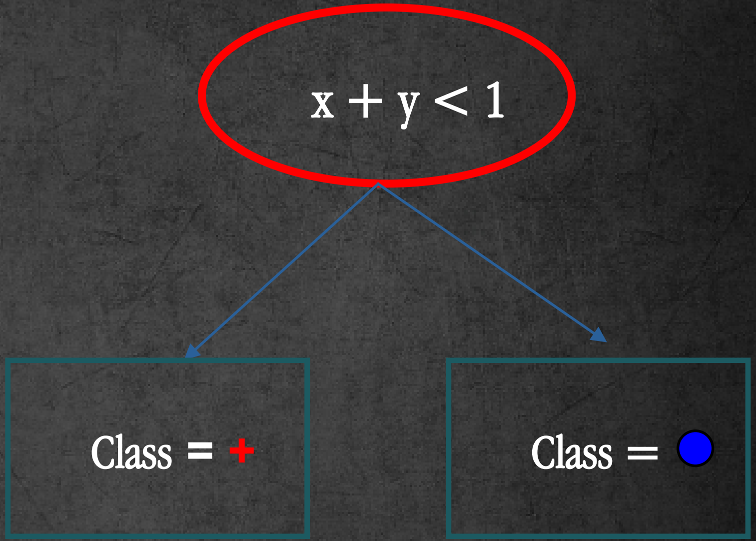


Border line between two neighboring regions of different classes is known as decision boundary

Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

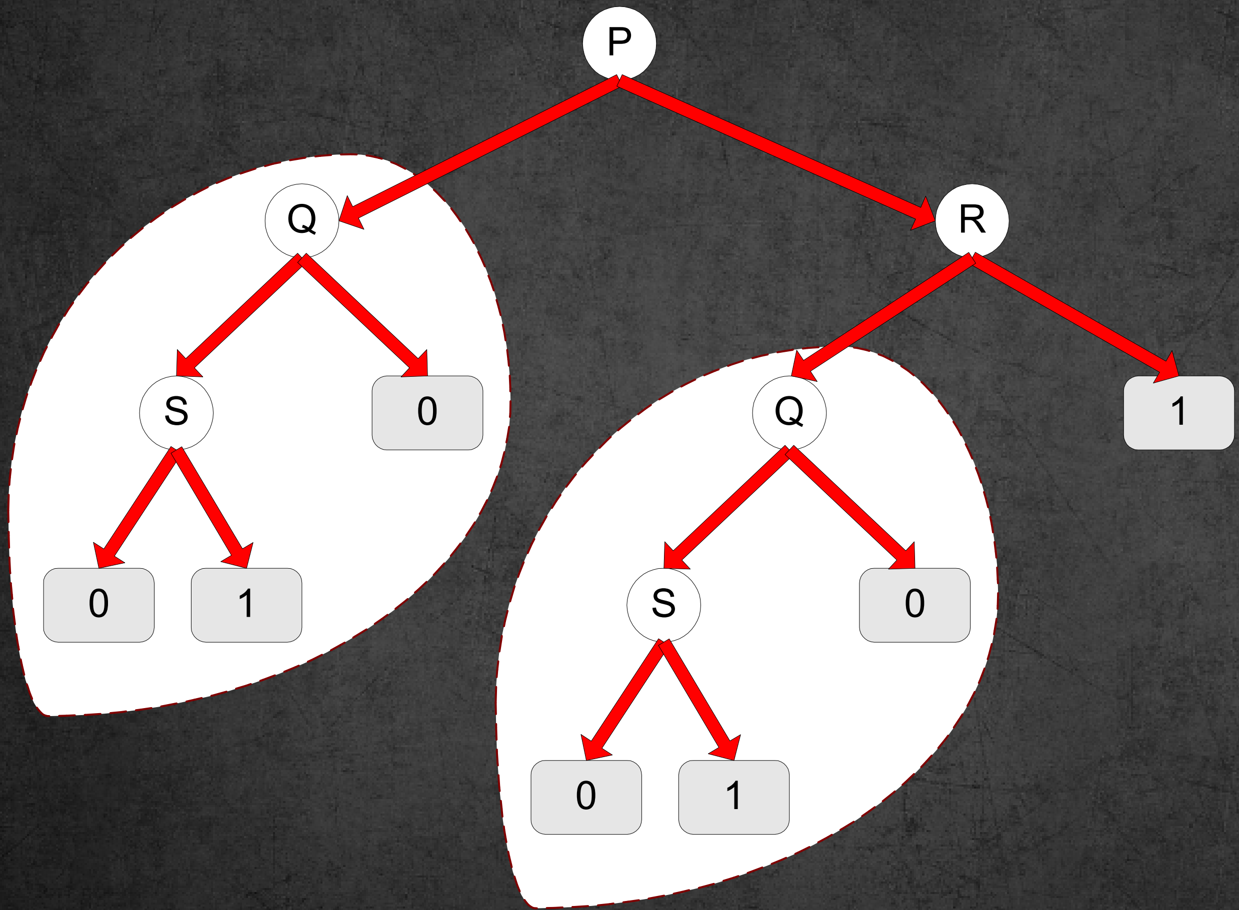
Oblique Decision Trees





- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

Tree Replication



- Same subtree appears in multiple branches

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	TP	FN
	Class=No	FP	TN

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:

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

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 model does not detect any class 1 example

Cost Matrix

	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	**Class=Yes**	**Class=No**
	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		
	C(i|j)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

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

Accuracy = 80%

Cost = 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 proportional to cost if

1. $C(\text{Yes}|\text{No})=C(\text{No}|\text{Yes}) = q$
2. $C(\text{Yes}|\text{Yes})=C(\text{No}|\text{No}) = p$

$$N = a + b + c + d$$

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

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

Cost-Sensitive Measures

- 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)

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

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

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

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

Class-imbalanced data measures

- Sensitivity (Recall, true positive rate):
  - Sn = TP/(TP+FN)
- Specificity (True negative rate)
  - Sp = TN/(TN+FP)
- G-mean

$$G - mean = \sqrt{(Sn \cdot Sp)}$$

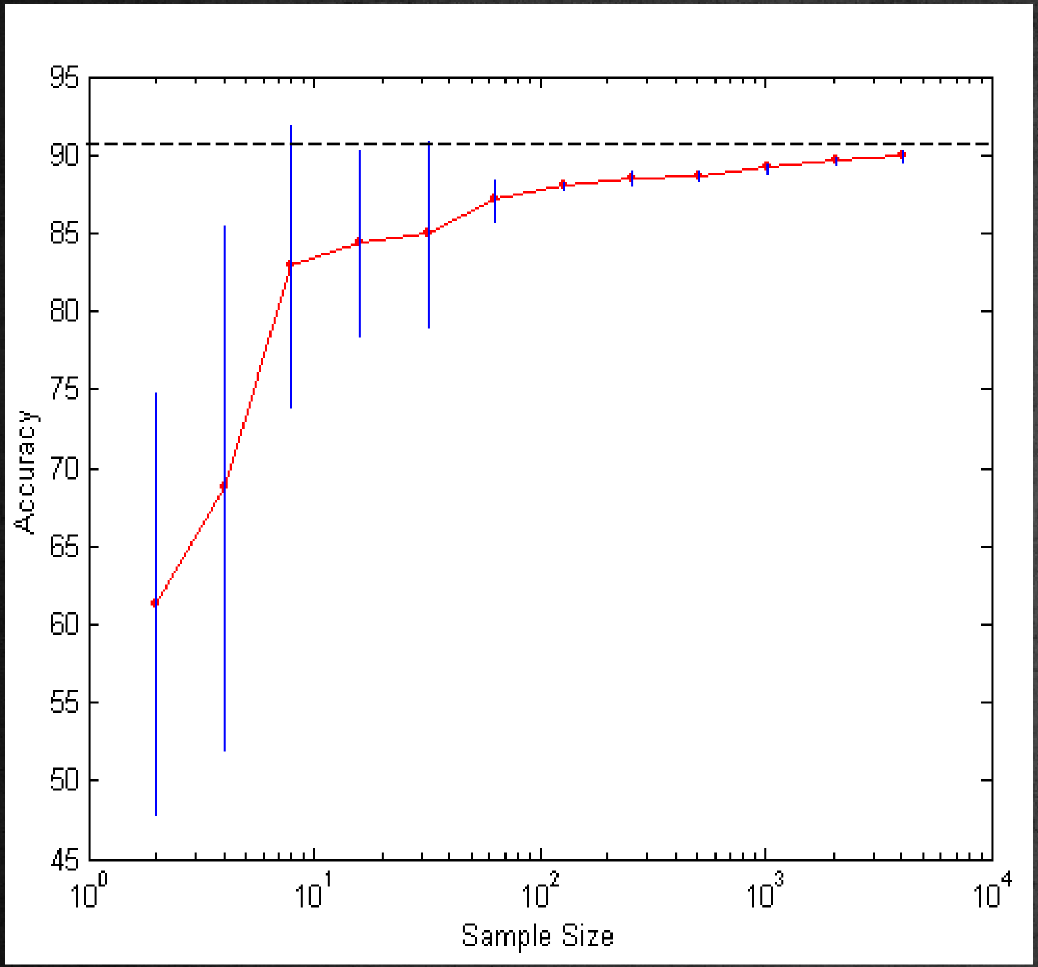
Model Evaluation

- 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

Learning Curve



- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley et al.)
  - Geometric sampling (Provost et al.)
- Effect of small sample size:
- Bias in the estimate
- Variance of estimate

Methods of Estimation

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Random subsampling
  - Repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Stratified sampling
  - oversampling vs undersampling
- Bootstrap
  - Sampling with replacement

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?

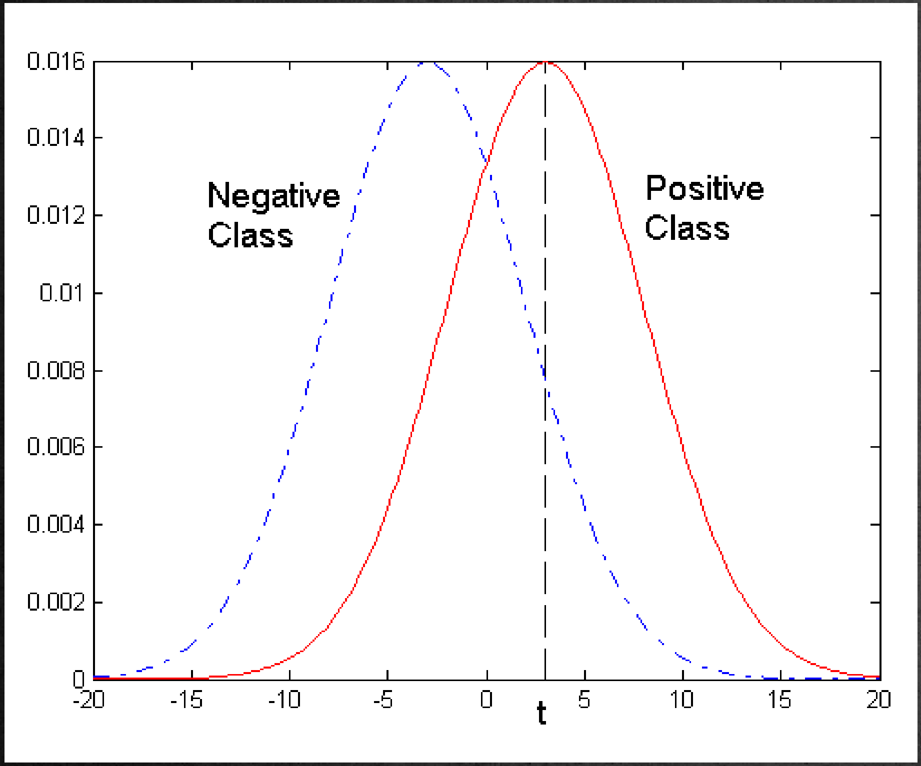
ROC (Receiver Operating Characteristic)

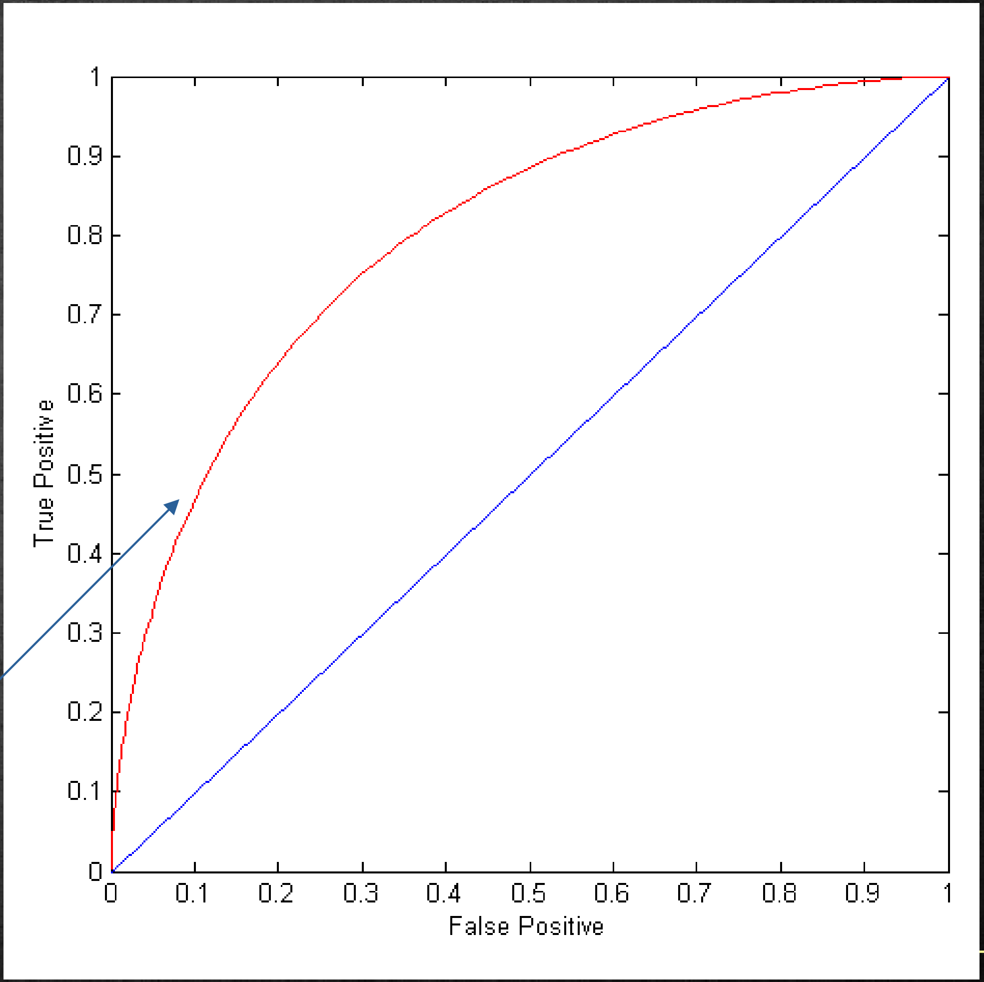
- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TP rate (Sn) on the y-axis against FP rate (1-Sp) on the x-axis
- Performance of each classifier represented as a point on the ROC curve
  - Changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

ROC Curve

1-dimensional data set containing 2 classes (positive and negative)

any points located at x > t is classified as positive



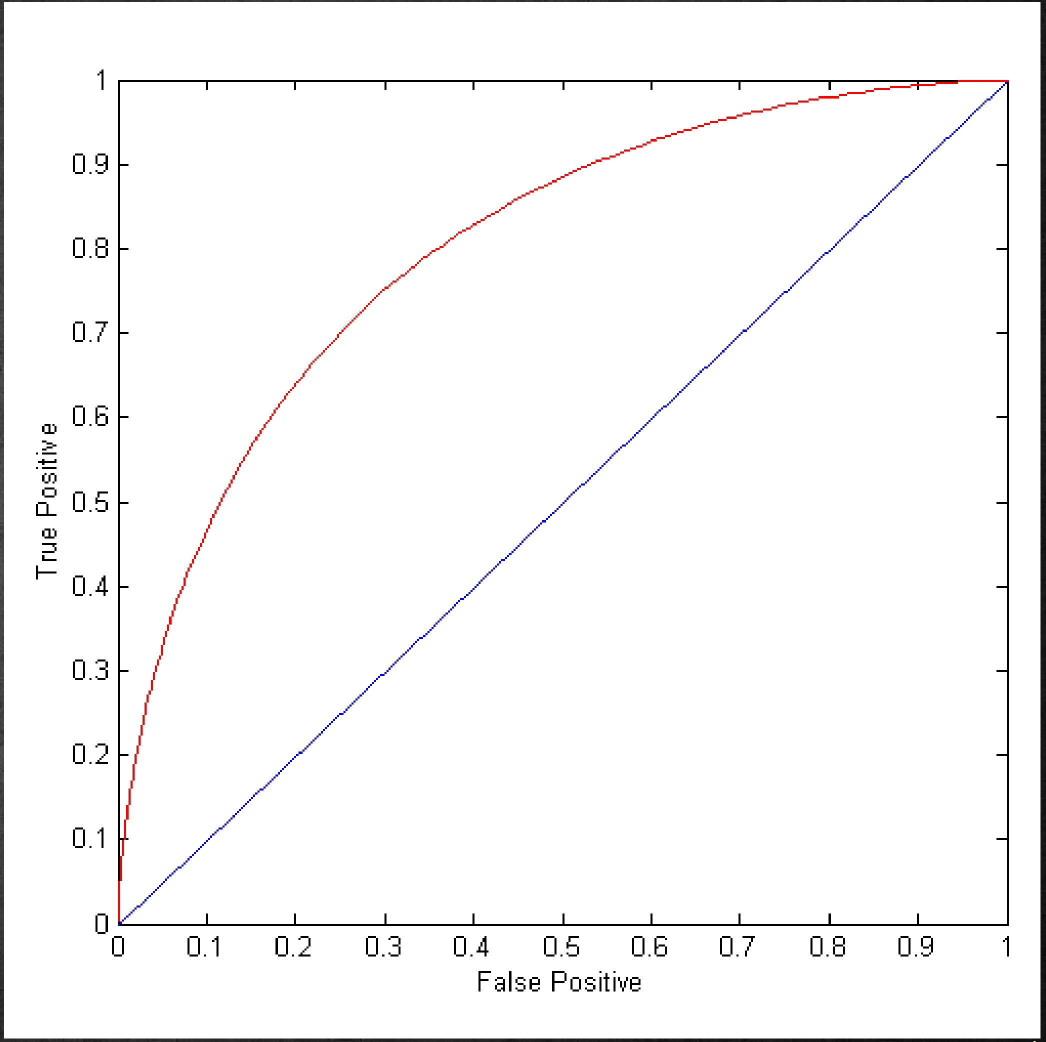


At threshold t:

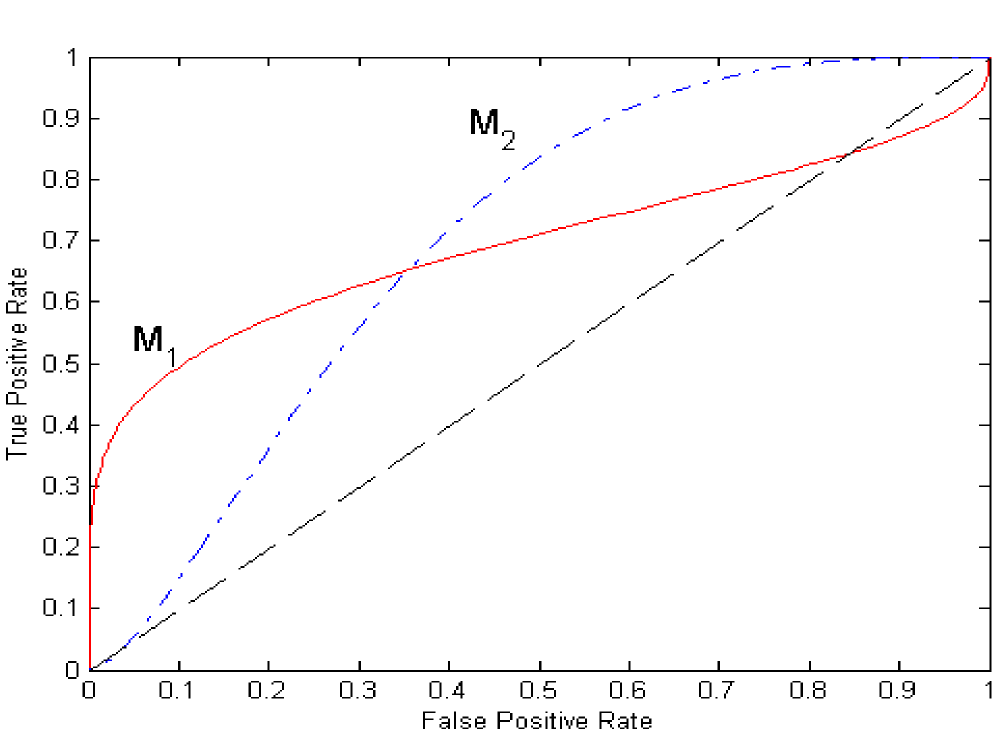
TP=0.5, FN=0.5, FP=0.12, FN=0.88

ROC Curve

- (TP rate = TP/P, FP rate = Fp/N):
- (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
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve
- Ideal:
  - Area = 1
- Random guess:
  - Area = 0.5

How to construct an ROC curve

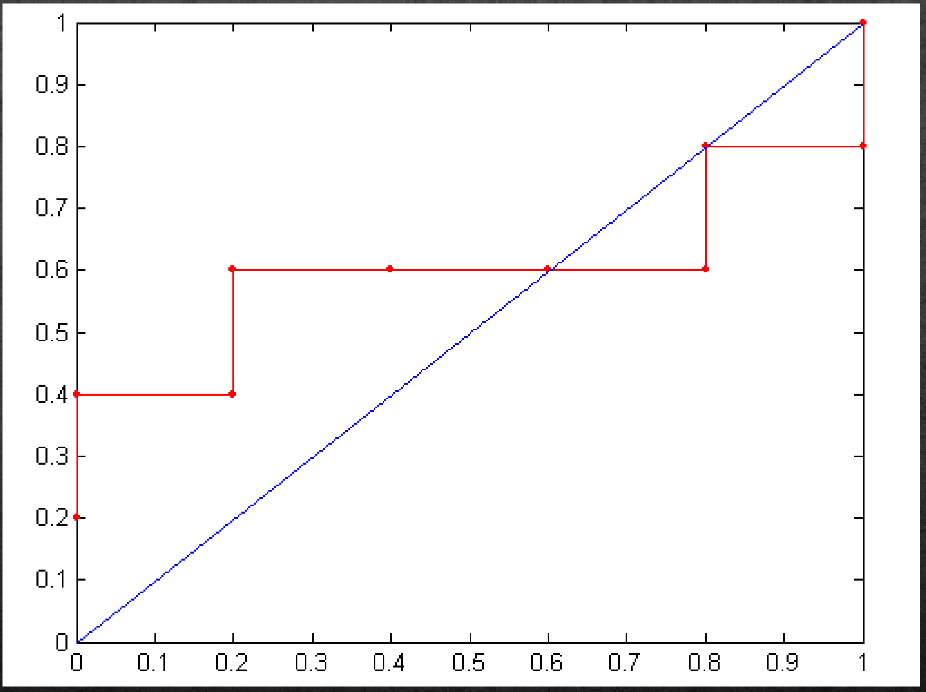
Instance	P(+ | A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Use classifier that produces posterior probability for each test instance P(+|A)
- Sort the instances according to P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Count the number of TP, FP,
- TN, FN at each threshold
- TP rate, TPR = TP/(TP+FN)
- FP rate, FPR = FP/(FP + TN)

How to construct an ROC curve

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
→ TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
→ FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC Curve:



Test of Significance

- Given two models:
  - Model M_1: accuracy = 85%, tested on 30 instances
  - Model M_2: accuracy = 75%, tested on 5000 instances
- Can we say M_1 is better than M_2?
  - How much confidence can we place on accuracy of M_1 and M_2?
  - Can the difference in performance measure be explained as a result of random fluctuations in the test set?

Confidence Interval for Accuracy

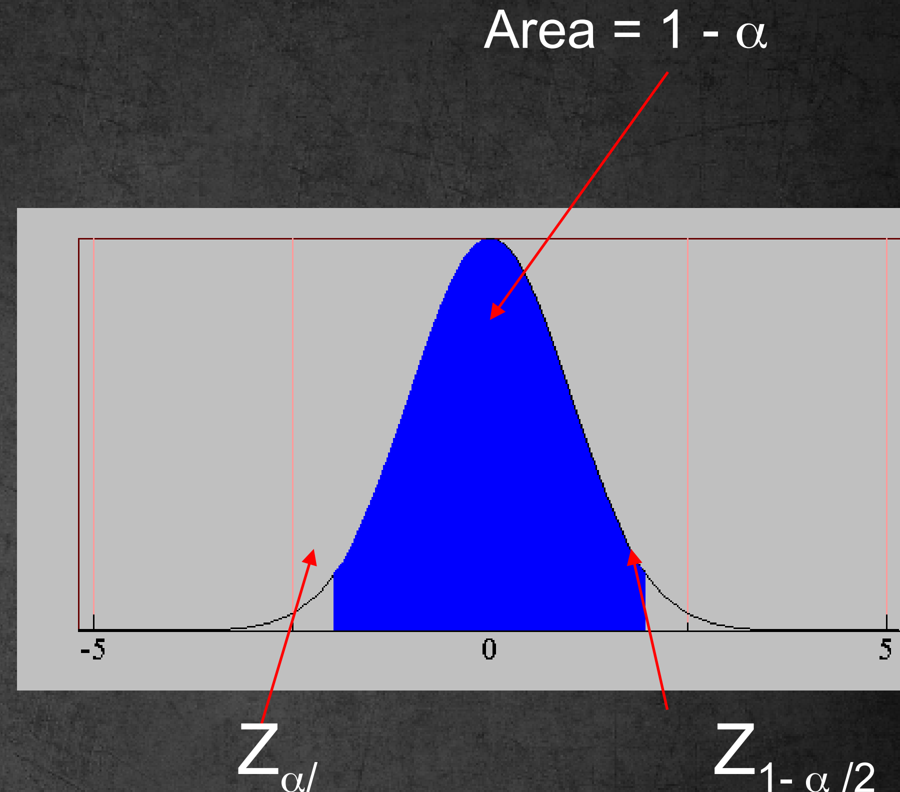
- Prediction can be regarded as a Bernoulli trial
  - A Bernoulli trial has 2 possible outcomes
  - Possible outcomes for prediction: correct or wrong
  - Collection of Bernoulli trials has a Binomial distribution:
    - $x \sim Bin(N, p)$ x: number of correct predictions
    - e.g: Toss a fair coin 50 times, how many heads would turn up?
      Expected number of heads = N x p = 50 x 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),
  - acc has a normal distribution with mean p and variance p(1-p)/N

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

$$1-\alpha$$



- Confidence Interval for p:

2

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

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, $Z_{\alpha/2}=1.96$

$1-\alpha$	Z
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

Comparing Performance of 2 Models

- Given two models, say M1 and M2, 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
  - Assume D_1 and D_2 are independent
  - If n_1 and n_2 are sufficiently large, then

$$e_1 \sim N(\mu_1, \sigma_1)$$
$$e_2 \sim N(\mu_2, \sigma_2)$$

  - 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: $d = e_1 - e_2$
  - $d \sim N(d_t, \sigma_t)$, where d_t is the true difference
  - Since D_1 and D_2 are independent, their variance adds up:

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

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

  - At $(1-\alpha)$ confidence level,

$$d_t = d \pm Z_{\alpha/2}\hat{\sigma}_t$$

An Illustrative Example

- Given: M_1: $n_1 = 30$, $e_1 = 0.15$
  M_2: $n_2 = 5000$, $e_2 = 0.25$
- $d = |e_2 - e_1| = 0.1$ (2-sided test)

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

- 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

Comparing Performance of 2 Algorithms

- Each learning algorithm may produce k models:
  - L_1 may produce M_{11} , M_{12}, ..., M_{1K}
  - L_2 may produce M_{21} , M_{22}, ..., M_{2K}
- If models are generated on the same test sets D_1,D_2, ..., D_k (e.g., via cross-validation)
  - For each set: compute $d_j = e_{1j} - e_{2j}$
  - d_j has mean d_t and variance σ_t
  - Estimate:

$$\hat{\sigma}_t^2 = \frac{\sum_{j=1}^{k} (d_j - \overline{d})^2}{k(k-1)}$$

$$d_t = d \pm t_{1-\alpha, k-1} \hat{\sigma}_t$$

Non parametric tests: 1 vs. 1 over N problems

- Wilcoxon's test
  - 2 algorithms over N problems
  - d_i is the difference of the performance in i-th set

$$R^+ = \sum_{d_i>0} \mathrm{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \mathrm{rank}(d_i) \qquad R^- = \sum_{d_i<0} \mathrm{rank}(d_i) + \frac{1}{2}\sum_{d_i=0} \mathrm{rank}(d_i).$$

  - Being T the smaller of the sums:

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}$$

  is normally distributed.

  - A p-value is obtained from the value of z and compared with a critical value α

Non parametric tests: 1 vs. k-1 over N problems

- Holm's procedure

$$R_i = 1/N \sum_j r_j^i$$

- z is normally distributed

$$z = \frac{(R_i - R_j)}{\sqrt{\frac{k(k+1)}{6N}}}.$$

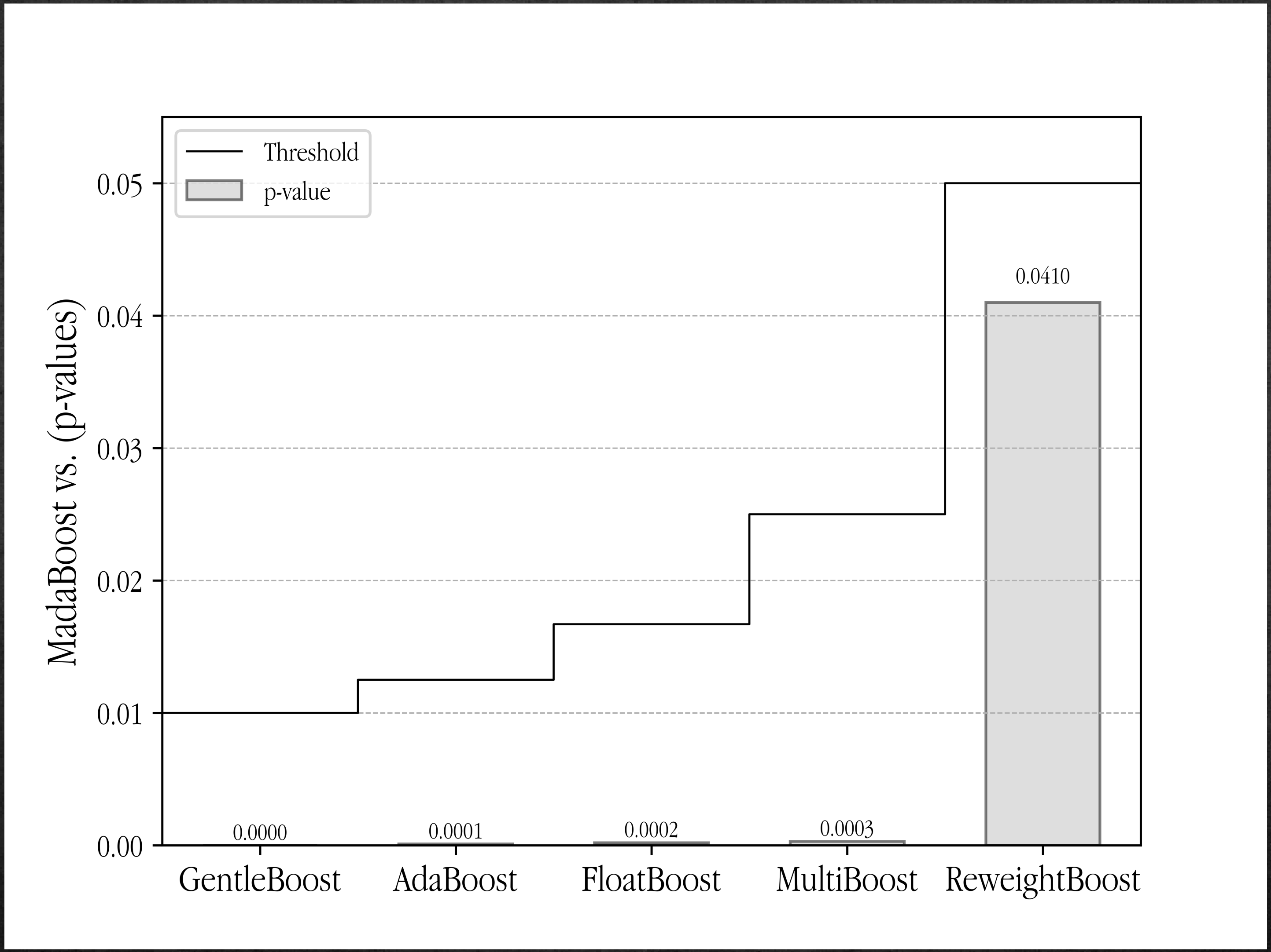
- The critical value, alpha, is adjusted using:

- p values are ordered and tested in turns

$$\alpha/(k-1)$$

Holm's procedure

- Graphical representation



Friedman test

- Multiple comparison test
- Bonferroni correction: $\alpha/(k-1)$ for k methods
  - Too conservative
- Friedman Test
  - Let r_i^j be the rank of the j-th of k algorithms on the i-th of N data sets
  - Friedman test compares the average ranks of algorithms R_j

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$$

  - is distributed according to χ_F^2 with k-1 degrees of freedom, when N and k are big enough (as a rule of a thumb, N > 10 and k > 5)

Iman – Davenport test

- Friedman test is undesirably conservative
- Iman and Davenport designed a better statistic:

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$$

which is distributed according to the F-distribution with k-1 and (k-1)(N-1) degrees of freedom.

Nemenyi test

- Based on Friedman's ranks
- Methods significantly different if rank different above critical value:

$$CD = q_\alpha \sqrt{k(k+1)\frac{6}{N}}$$

  - k: #methods, N: #datasets, q_α: critical value (Student t)