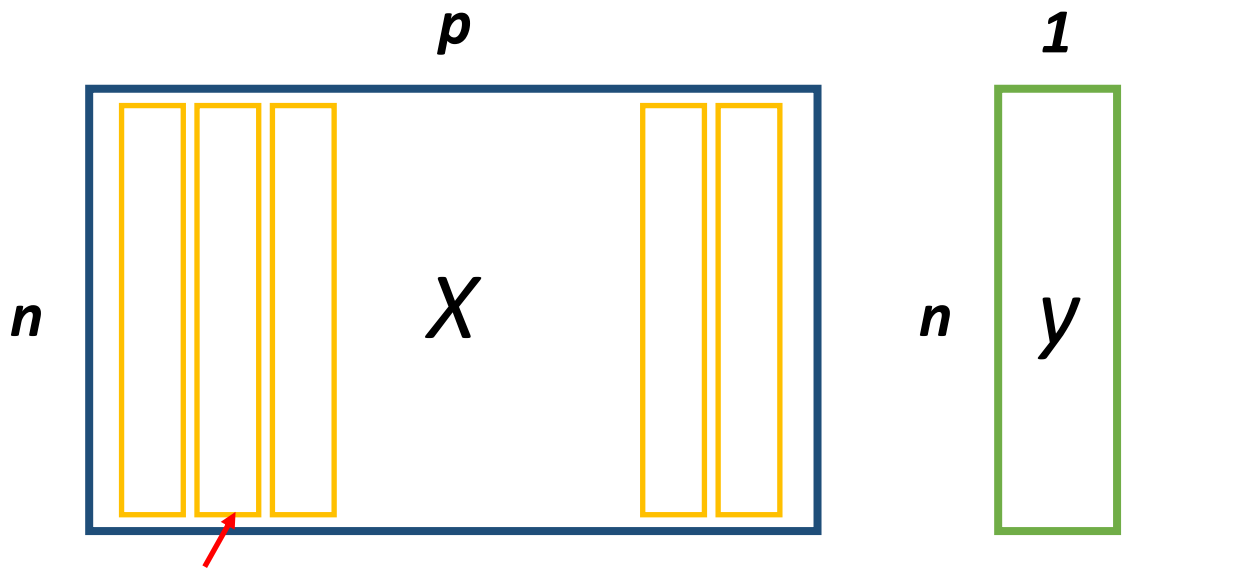
Decision Tree

Instructor: Junghye Lee

Department of Industrial Engineering
junghyelee@unist.ac.kr

Contents

Types of Variables



Standard 1

Discrete variable: nominal and ordinal

Continuous variable: continuous

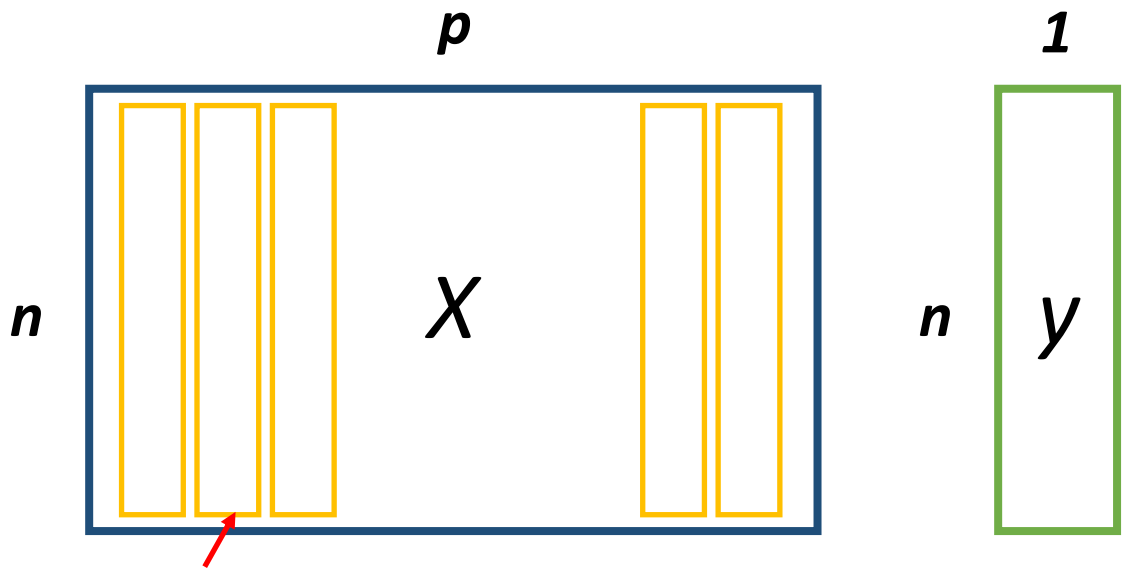
Standard 2

Numeric variable: continuous and ordinal

Nominal variable: nominal

* Nominal = Categorical

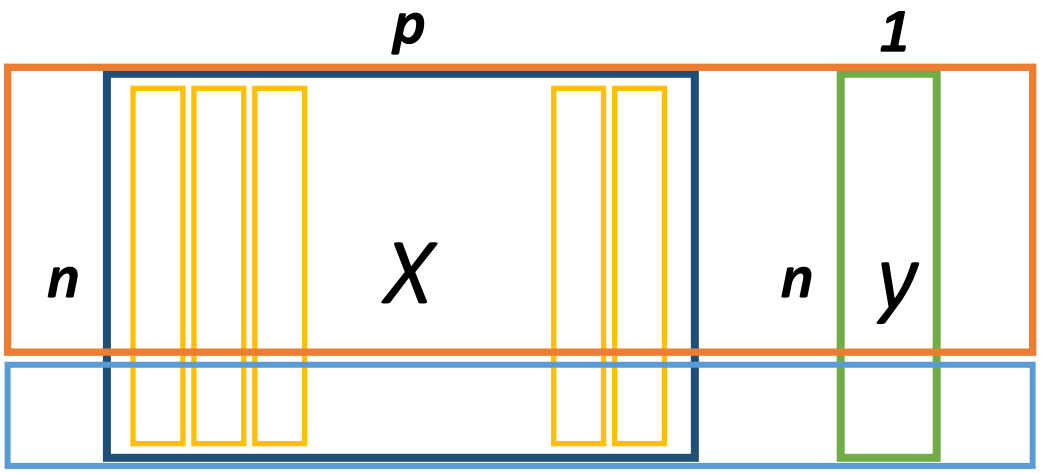
Types of Variables



Transformation into dummy variable

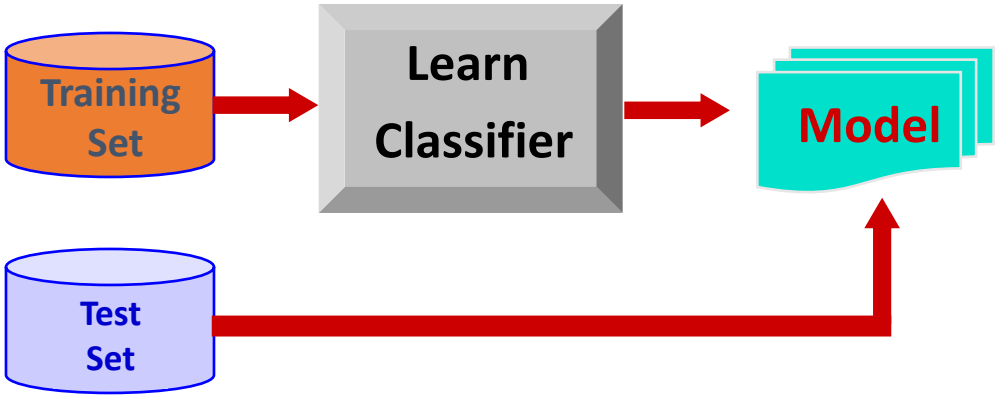
X	y	Problem
Nominal	Nominal	Classification
Nominal	Continuous	Regression
Continuous	Nominal	Classification
Continuous	Continuous	Regression

Training and Test Sets



80%: training set

20%: test set

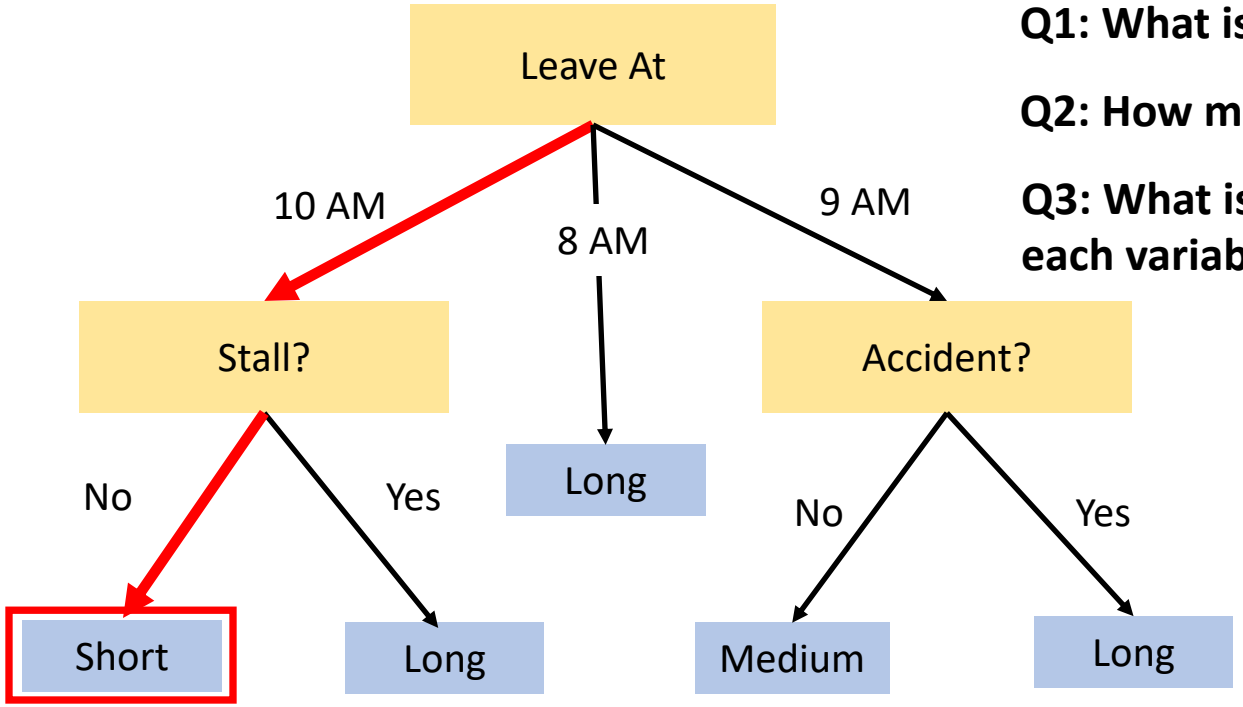


Regression/Classification

What is a Decision Tree?

- **Predicting commute time**

If we leave at 10 AM and there are no cars stalled on the road, what will our commute time be?



Q1: What is a D.V.?

Q2: How many I.V.s?

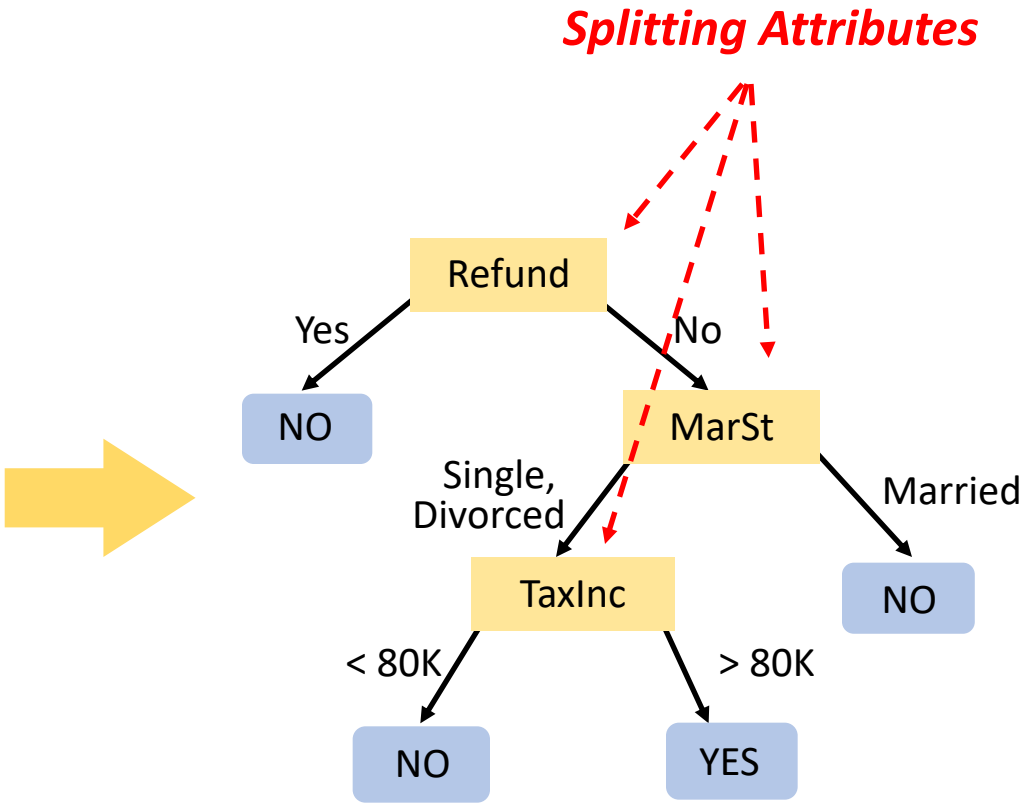
Q3: What is the type of each variable?

Rule-based prediction model

Another Example of a Decision Tree

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

Training Data

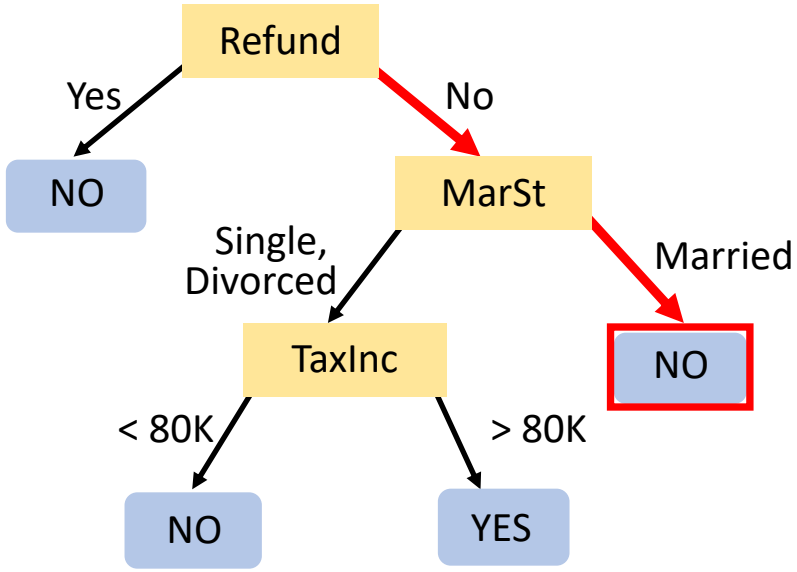


Decision Tree

Another Example of a Decision Tree

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Test Data



Assign Cheat to "No"

Decision Tree originated from training data

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.

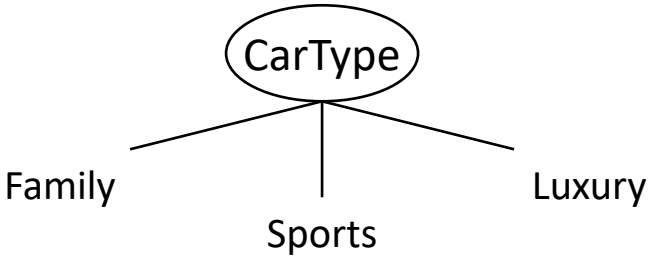
- Issues
  - Determine how to split the records

    How to specify the attribute test condition?

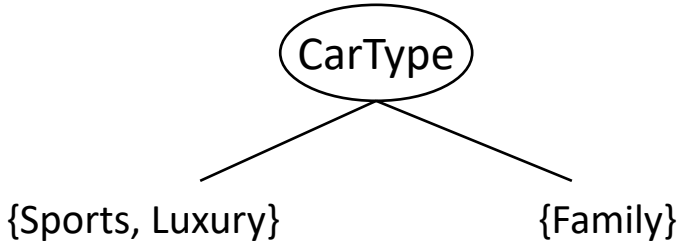
    How to determine the best split?
  - Determine when to stop splitting

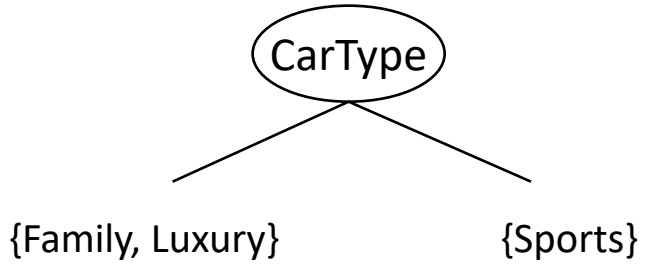
Splitting based on Nominal Attributes

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



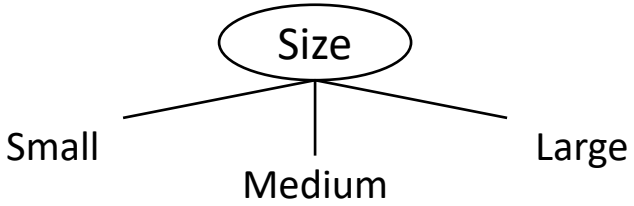
- Binary split
  - Divides values into two subsets
  - Need to find optimal partitioning



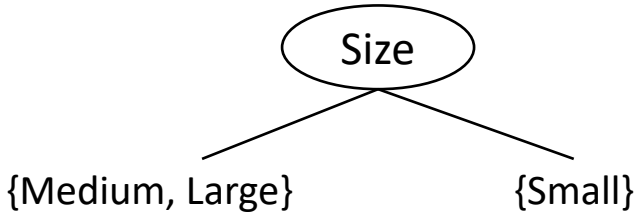


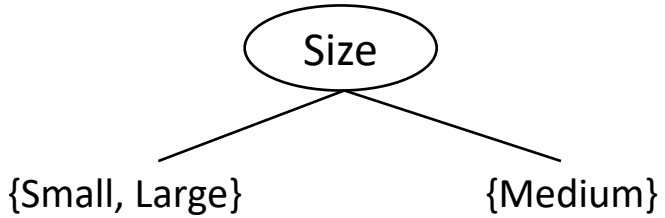
Splitting based on Ordinal Attributes

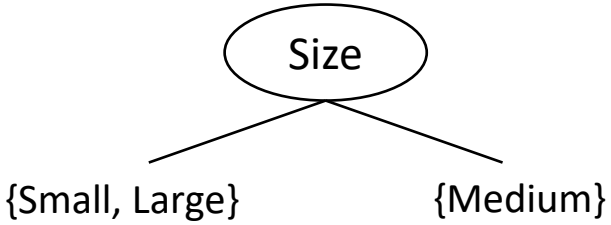
- Multi-way split
  - Use as many partitions as distinct values.



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







Splitting based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute

    Static – discretizes once at the beginning (pre-processing)

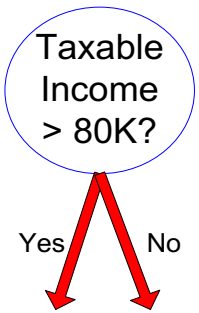
    – one attribute at a time (independently)

    Dynamic – responds when the learner requires so, during the building of the model

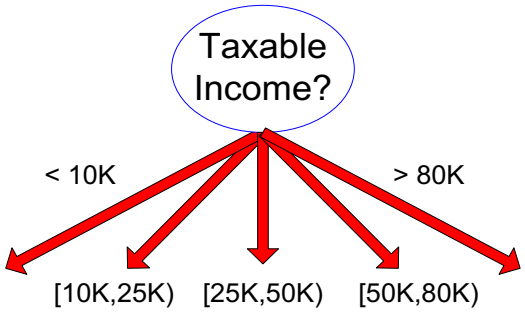
    – interdependencies among attributes
  - Binary Decision: $(A < v)$ or $(A \geq v)$

    consider all possible splits and finds the best cut

    can be more computation-intensive.



(i) Binary split



(ii) Multi-way split

Tree Induction

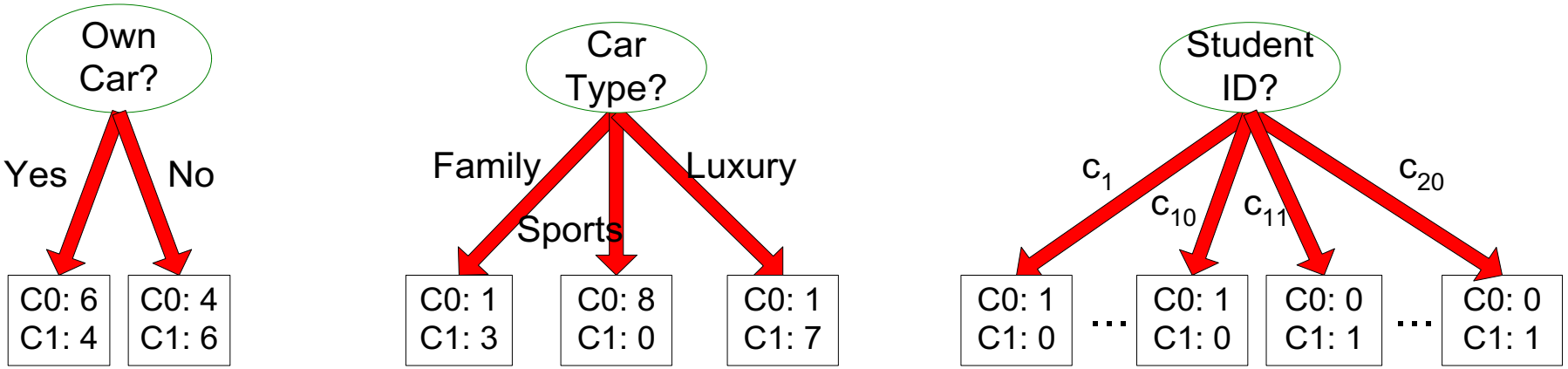
- Greedy strategy
  - Split the records based on an attribute test that optimizes certain criterion.

- 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?

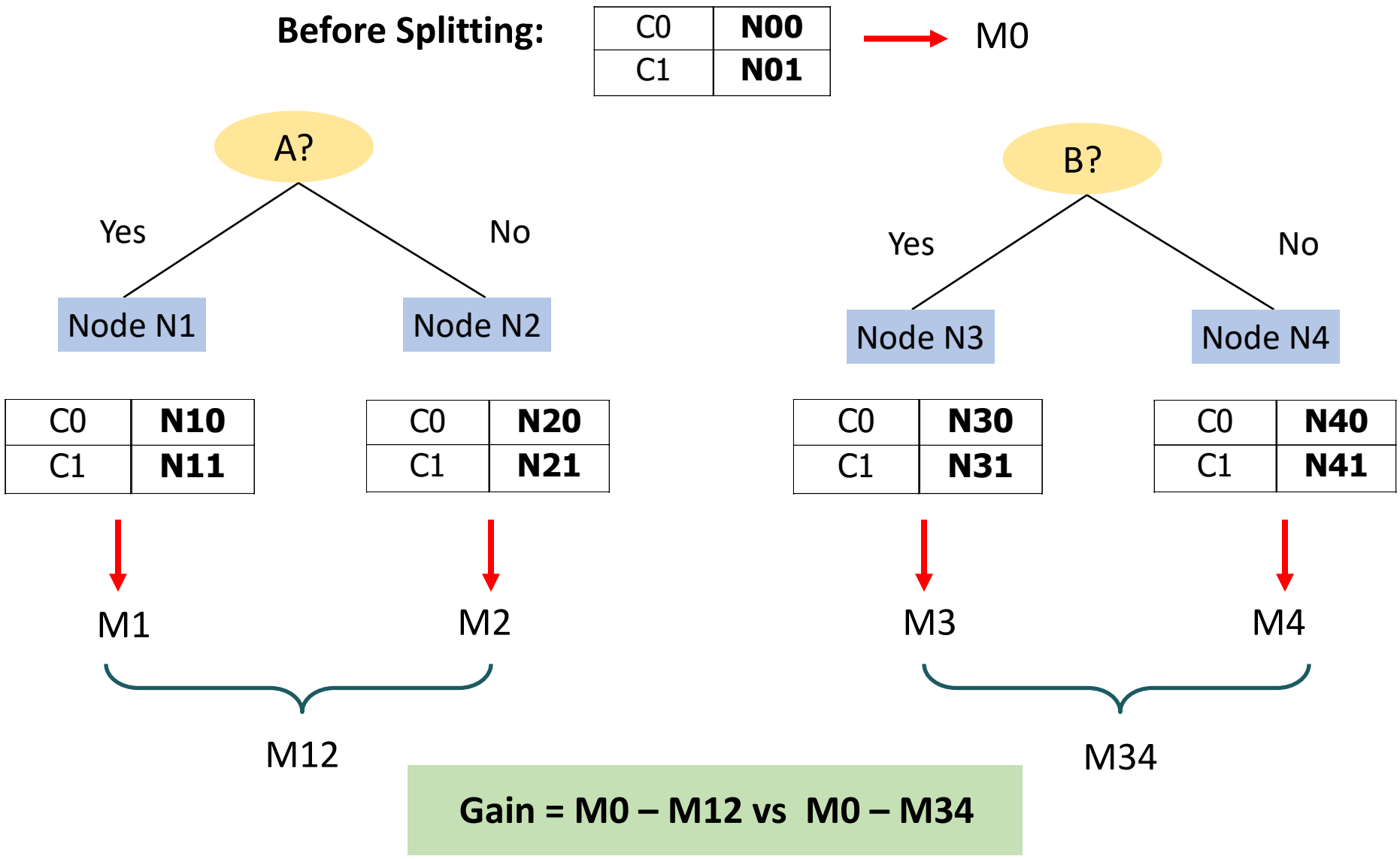
How to determine the Best Split

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

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

- Measures of node impurity
  - Gini Index
  - Entropy
  - Misclassification error

How to Find the Best Split



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 when records are equally distributed among all classes, implying least pure information
  - Minimum when all records belong to one class, implying most pure 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

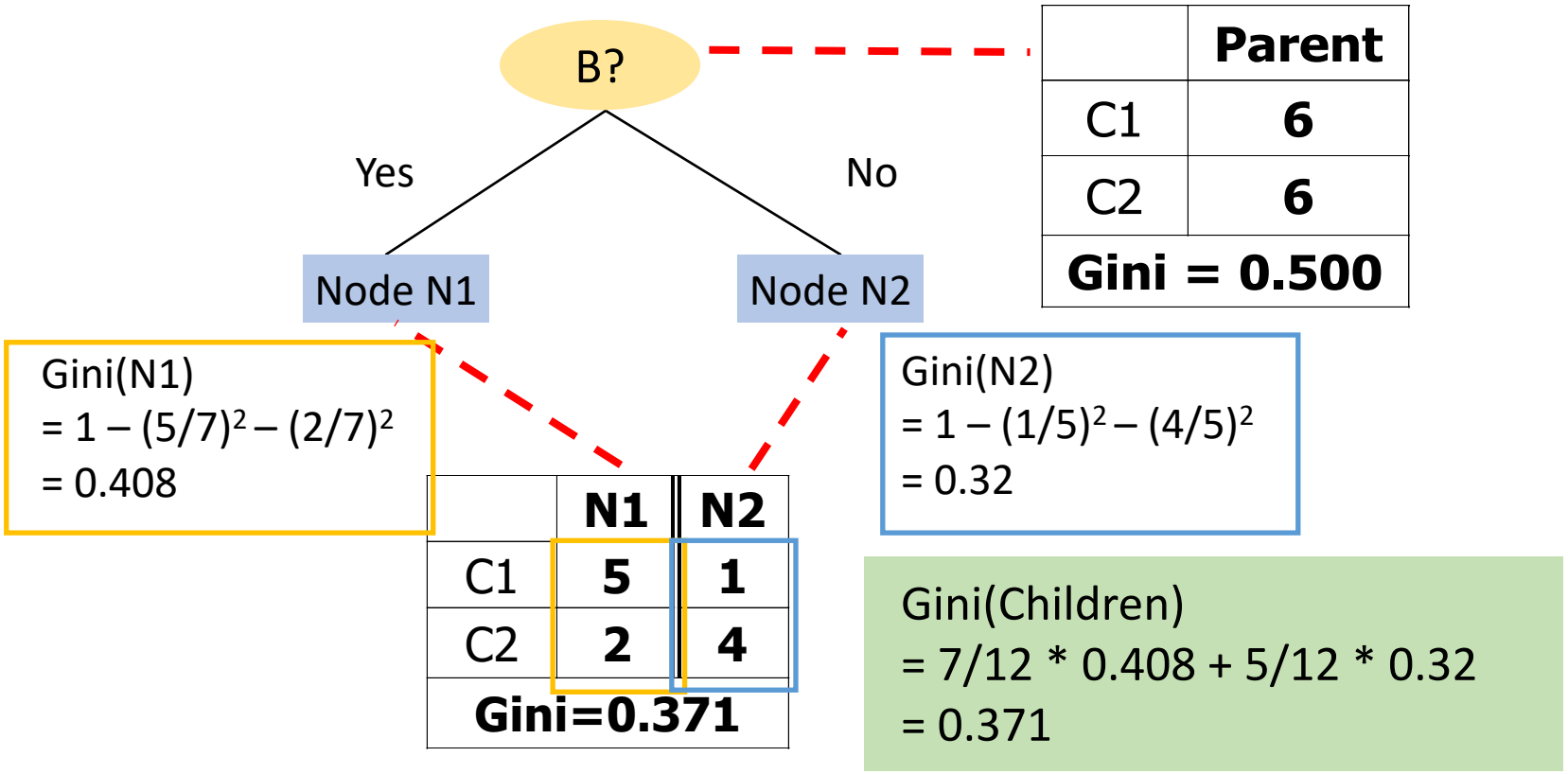
- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

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

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	

	N1	N2
C1	5	1
C2	2	4
Gini=0.371		

Categorical Attributes: Gini Index

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

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

Multi-way split

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

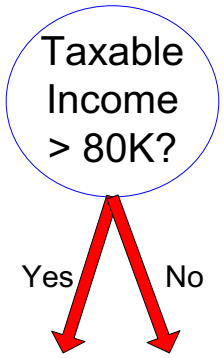
	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Two-way split
(find best partition of values)

Continuous Attributes: 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 $A \geq v$
- 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: 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 Criterion

- Entropy at a given node t:

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

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

- Measures homogeneity of a node.

  Maximum when records are equally distributed among all classes implying most information

  Minimum when all records belong to one class, implying least information

- Entropy based computations are similar to the Gini index computations

Splitting based on Information

- 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 Information

- Gain Ratio:

$$GainRATIO_{split} = \frac{Gain_{split}}{SplitINFOR}$$

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

Parent Node, p is split into k partitions

n_i is 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

Another Splitting Criterion

- Classification error at a node t:

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

- Measures misclassification error made by a node
  - Maximum when records are equally distributed among all classes, implying least accurate information
  - Minimum when all records belong to one class, implying most accurate information

C1	**0**
C2	**6**
Error=0.000	

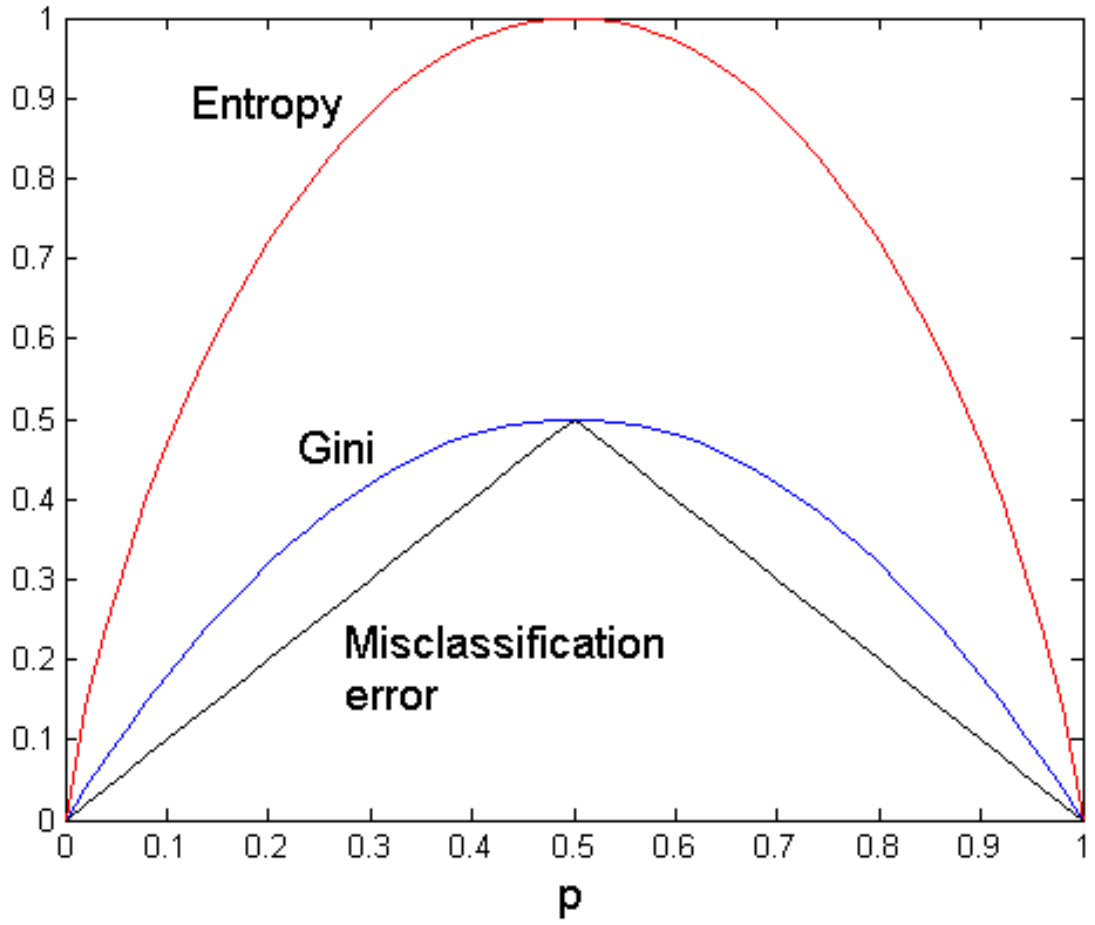
C1	**1**
C2	**5**
Error=1/6	

C1	**2**
C2	**4**
Error=2/6	

C1	**3**
C2	**3**
Error=1/2	

Comparison among Splitting Criteria

- For a two-class problem:



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 (completed)
- Stop expanding a node when all the records have similar attribute values (no hope)
- 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

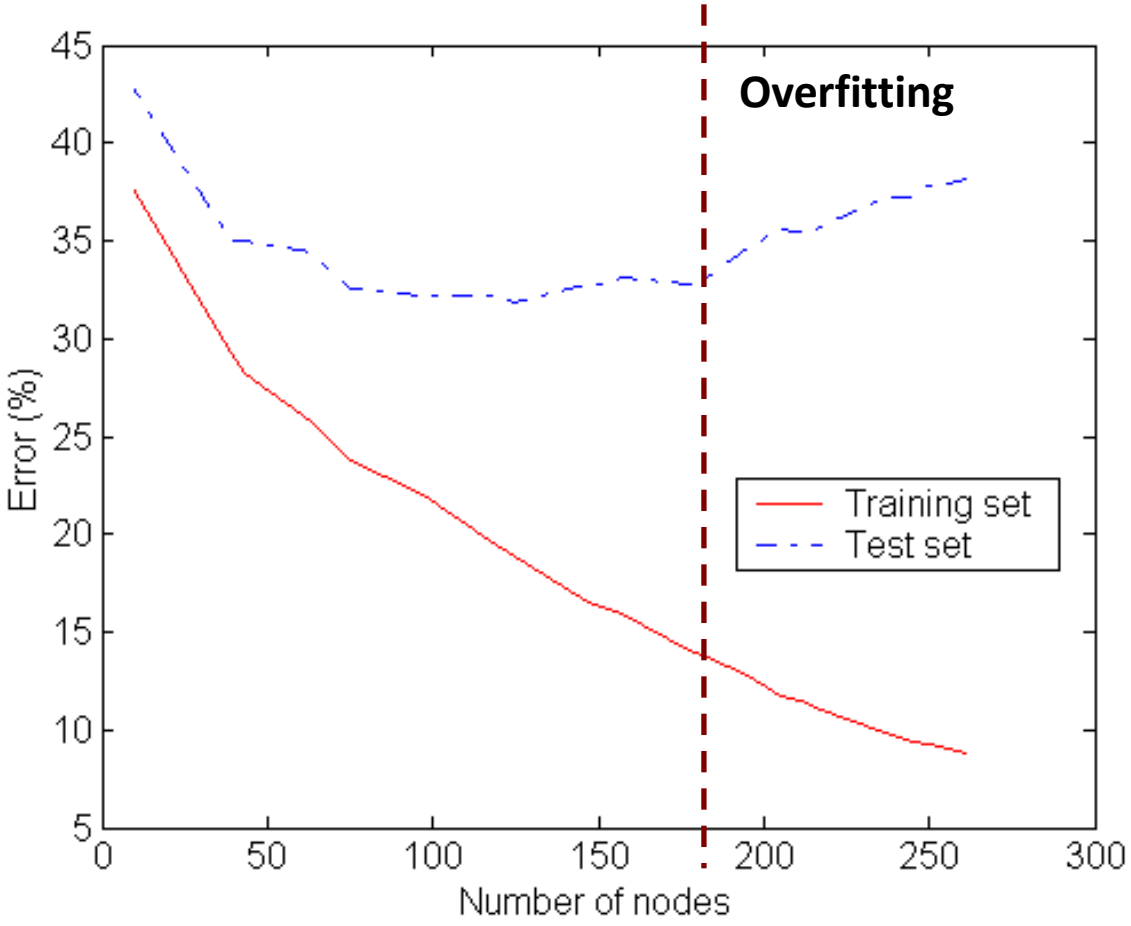
- Example: C4.5
  - Simple depth-first construction
  - Uses Information Gain
  - Sorts continuous attributes at each node

Practical Issues of Classification

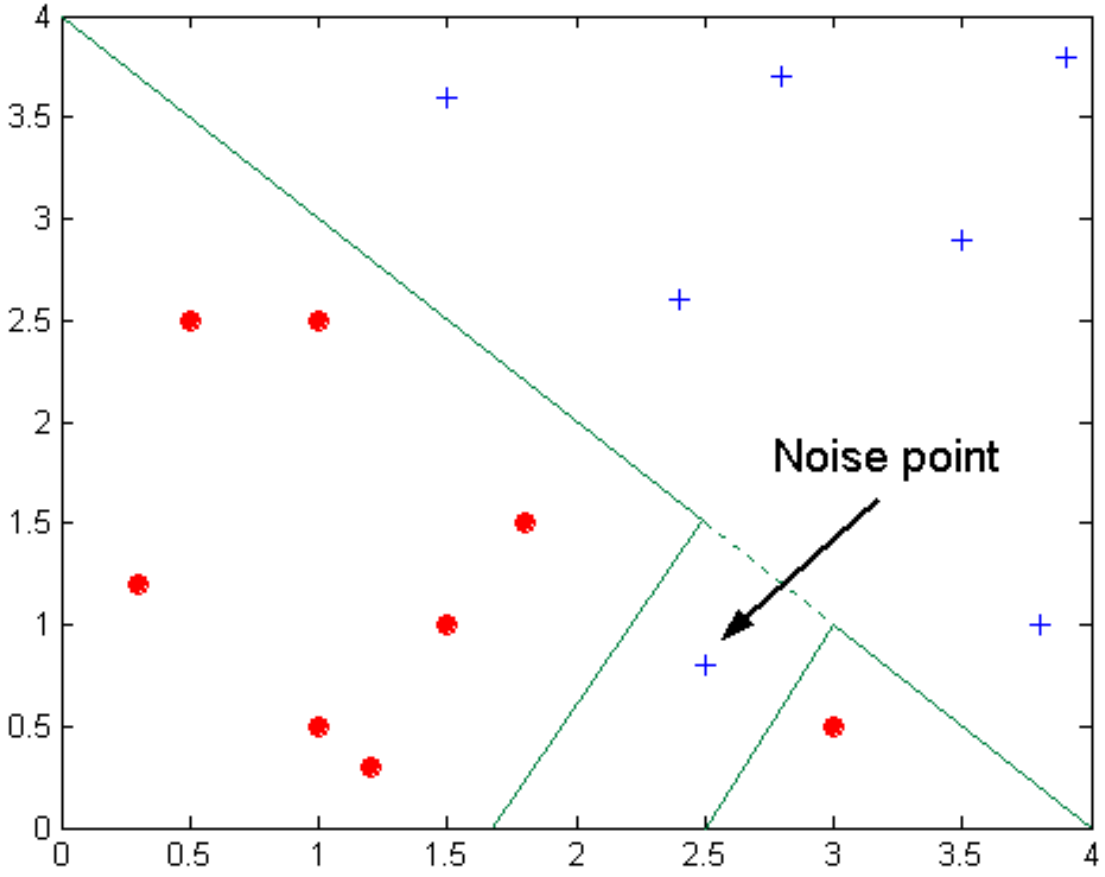
- Underfitting and overfitting
- Classification error
- Missing values

Underfitting and Overfitting

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



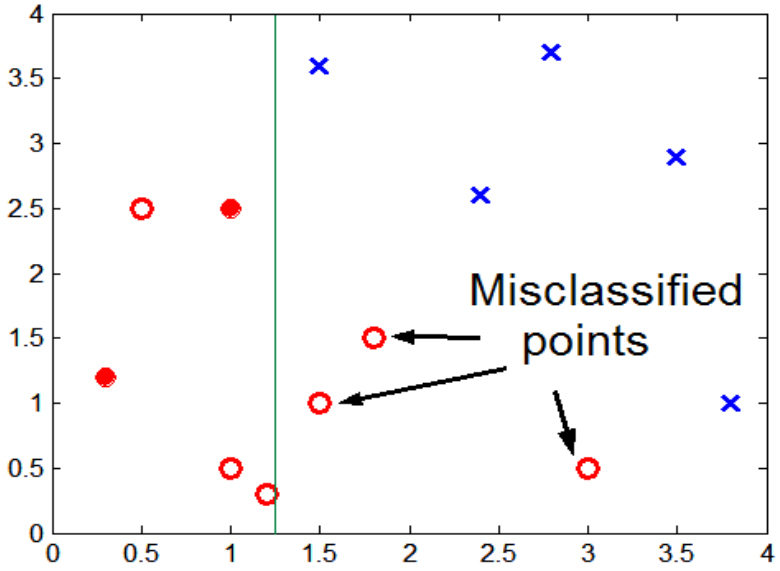
Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Instances

- 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.



Methods for Estimating the Error

- 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:

    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 + 30×0.5)/1000 = 2.5%
  - Reduced error pruning (REP):

    uses validation data set to estimate generalization error

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.

❖ **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

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 (completed)
    - Stop if all the attribute values are the same (no hope)
  - 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 (with 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

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

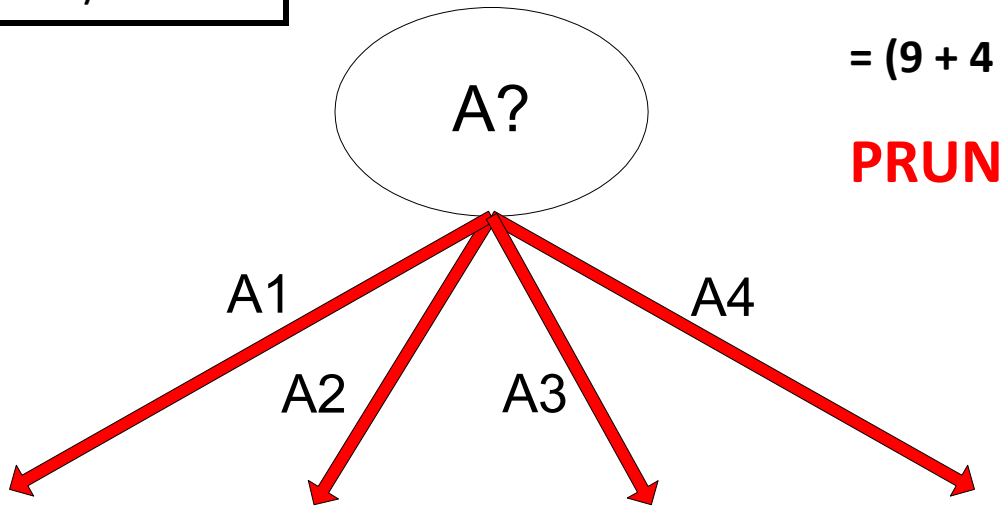
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Examples of Post-pruning

- Optimistic error?

  Don’t prune for both cases

- Pessimistic error?

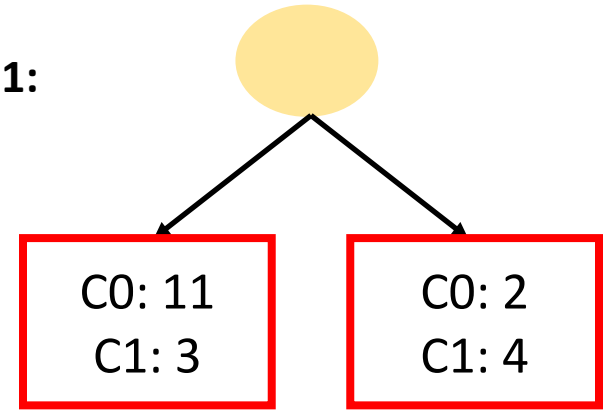
  Don’t prune case 1, prune case 2

- Reduced error pruning?

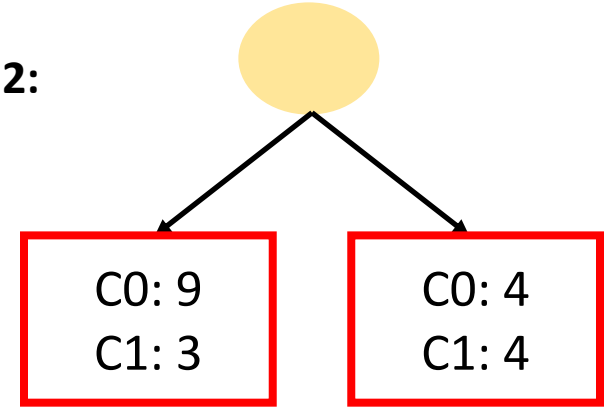
  Depends on validation set

C0	13
C1	7
Error = 7/20	

Case 1:



Case 2:



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

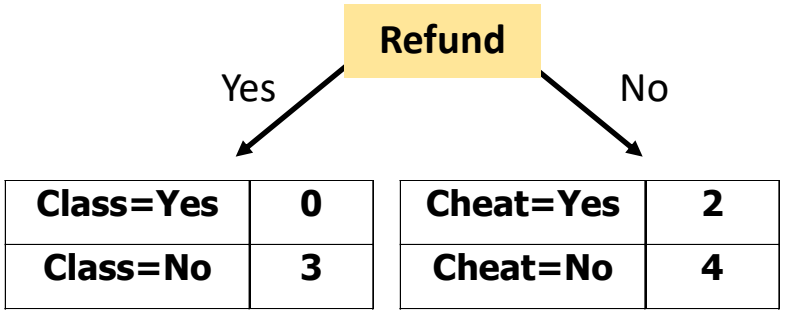
Entropy(Refund=No)
= -(2/6)log(2/6) – (4/6)log(4/6) = 0.9183

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

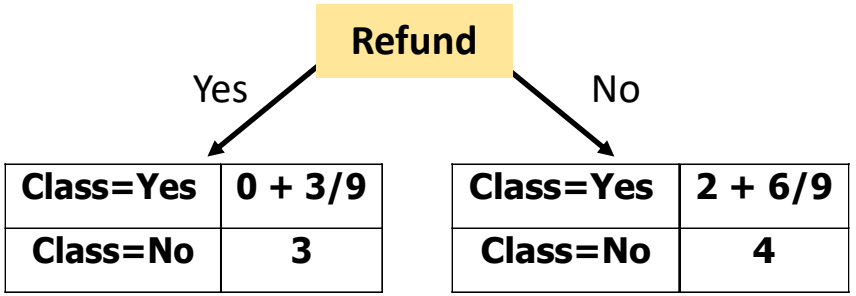
Gain = 0.8813 – 0.551 = 0.3303

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



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



Probability(Refund = Yes) = 3/9

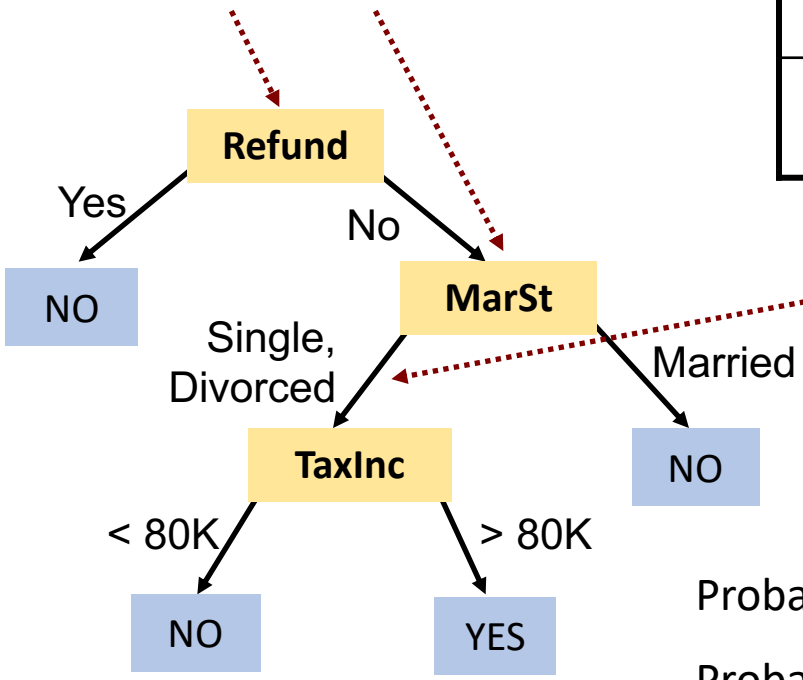
Probability(Refund = No) = 6/9

Assign record to
the left child with weight = 3/9 and
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	0	1+6/9	1	2.67
Total	3	2.67	1	6.67

Probability(Marital Status = Married) = 3/6.67

Probability(Marital Status = {Single,Divorced}) = 3.67/6.67

What about Regression Problem?

- Gini Index
- Entropy
- Classification Error

- Can we use these measures for regression?

What about Regression Problem?

- Gini Index
- Entropy
- Classification Error

- Can we use these measures for regression?

- Then what kind of measure that we can think of for regression?

What about Regression Problem?

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

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

- The first and second options are about $\boldsymbol{y}$
  - Var(p): Variance of $\boldsymbol{X}$ at the parent node.
  - MSE(p): MSE($\boldsymbol{y}$) at the parent node.

    What should be $\hat{\boldsymbol{y}}$ to result in MSE($\boldsymbol{y}$) = Var($\boldsymbol{y}$)?

What about Regression Problem?

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

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

- The first and second options are about $\boldsymbol{y}$
  - Var(p): Variance of $\boldsymbol{X}$ at the parent node.
  - MSE(p): MSE($\boldsymbol{y}$) at the parent node.

    What should be $\hat{\boldsymbol{y}}$ to result in MSE($\boldsymbol{y}$) = Var($\boldsymbol{y}$)?

    Yes, that is the $\hat{\boldsymbol{y}}$ in regression of Decision Tree!

Questions?