Machine Learning and Data Mining
Decision Trees and Model Evaluation

Patricia Hoffman, PhD

Most Slides taken from

Chapter 4 of Introduction to Data Mining

By Tan, Steinbach, Kumar

Decision Trees Resources

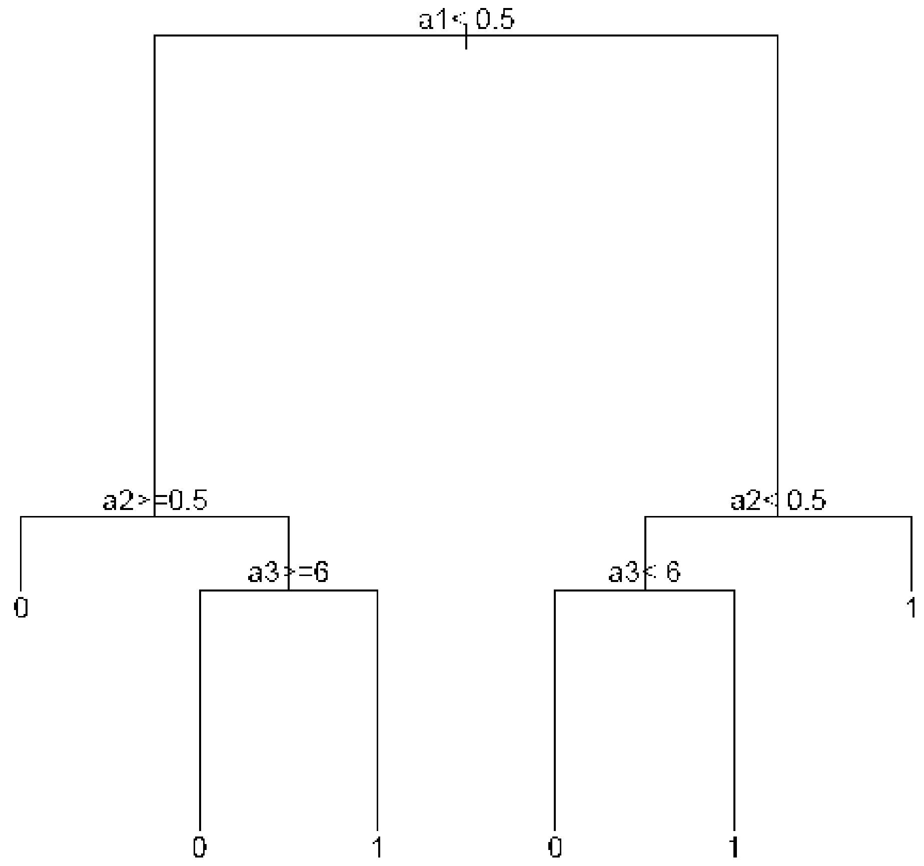
- Slides from Tan, Steinbach, Kumar
  - http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap4_basic_classification.pdf
- rpart Documentation:
  - http://cran.r-project.org/web/packages/rpart/rpart.pdf
- Understanding rpart formula:
  - Http://127.0.0.1:27898/library/stats/html/formula.html
- Algorithm for Big Data – PLANET:
  - Lecture http://fora.tv/2009/08/12/Josh_Herbach_PLANET_MapReduce_and_Tree_Learning

Recursive Partitioning and Regression Trees: Example Code

- First Example of rpart: SimpleTreeTab4_8.r
  - Use rpart to create a tree
- Classification with rpart: SonarRpartXval.r
  - Result is a Class Prediction
  - Cross Validation
  - Model Complexity Curve
  - Confusion Table
- Regression with rpart: ConcreteRpartXval.r
  - Result Predict Concrete Strength

rpart output: SimpleTreeTab4_8.r

Instance,	a1,	a2,	a3,	Target
1,	T,	T,	1.0,	1
2,	T,	T,	6.0,	1
3,	T,	F,	5.0,	0
4,	F,	F,	4.0,	1
5,	F,	T,	7.0,	0
6,	F,	T,	3.0,	0
7,	F,	F,	8.0,	0
8,	T,	F,	7.0,	1
9,	F,	T,	5.0,	0



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

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

Learning algorithm

Model

Training Set

Test Set

Decision Tree Classification Task

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

Tree Induction algorithm

Training Set

Test Set

Model

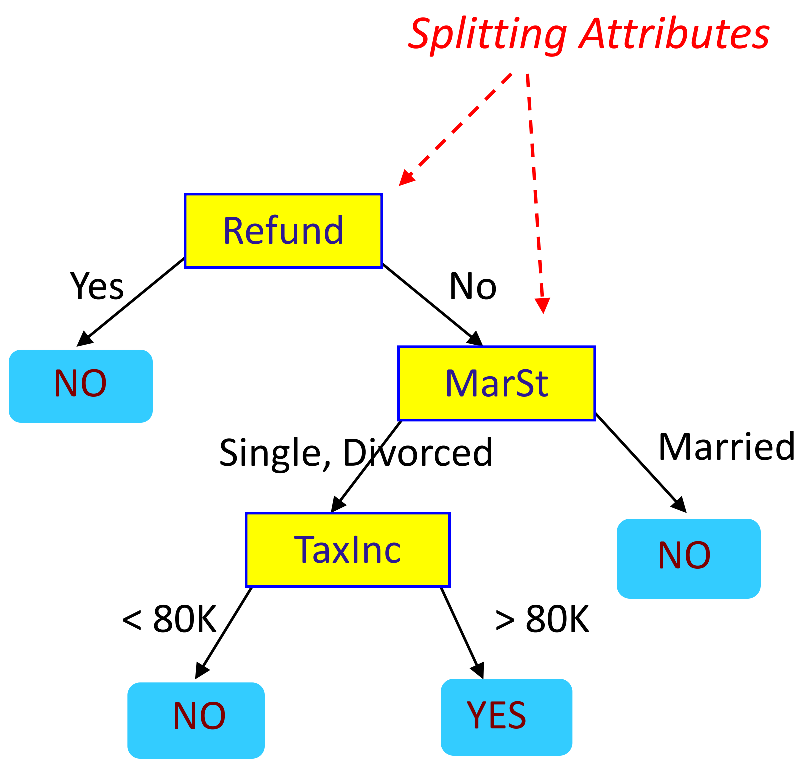
Decision Tree

Example of a Decision Tree

categorical categorical continuous class

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

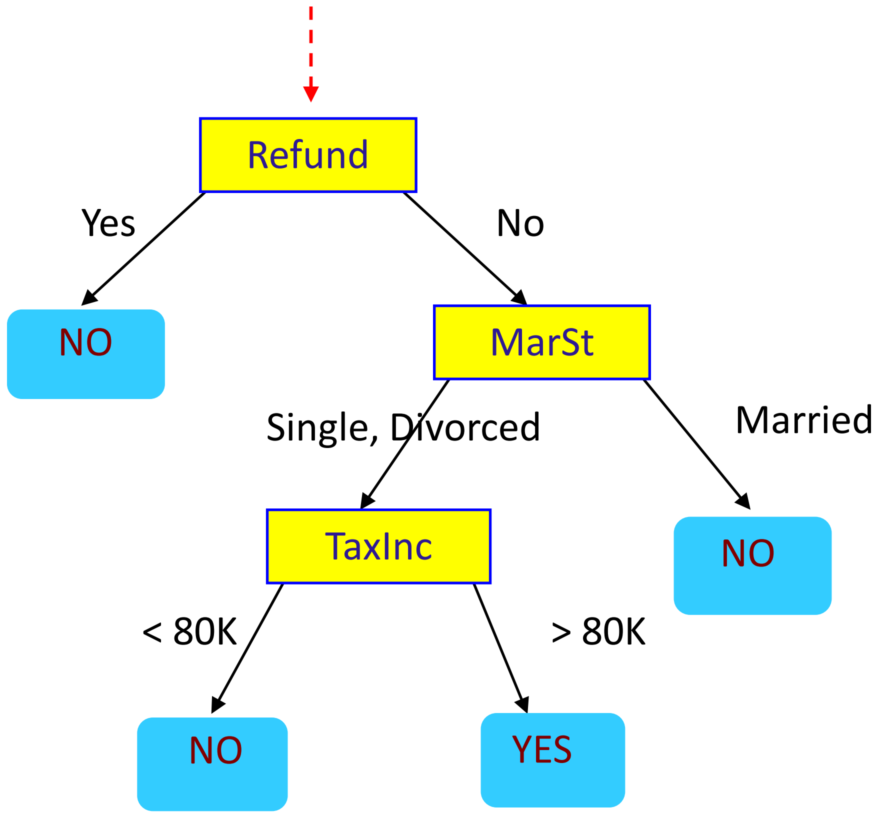
Training Data



Model: Decision Tree

Apply Model to Test Data

Start from the root of tree.



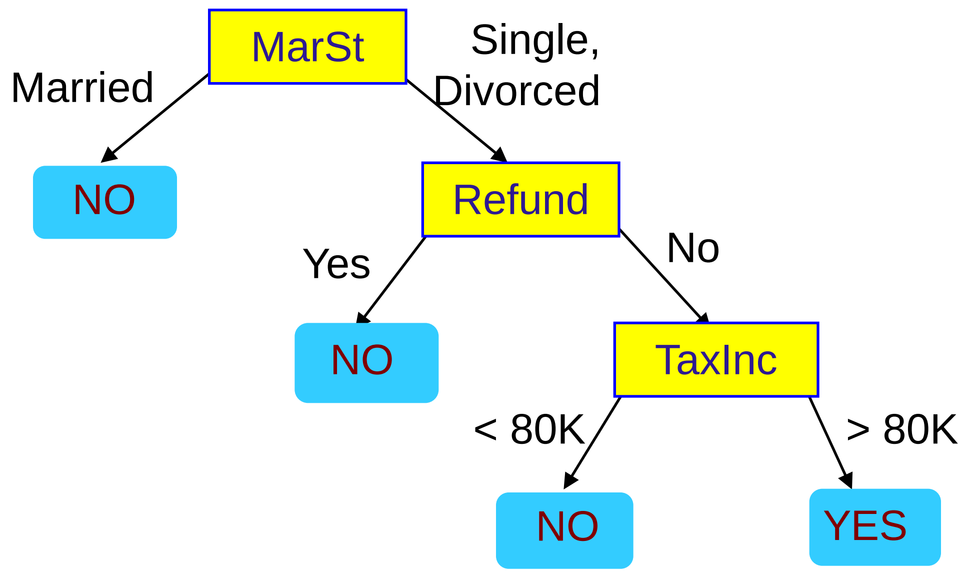
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

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!

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

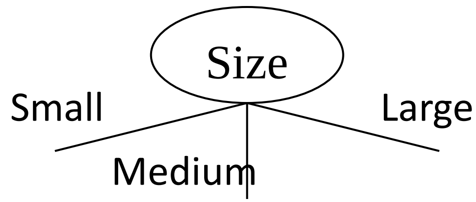
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

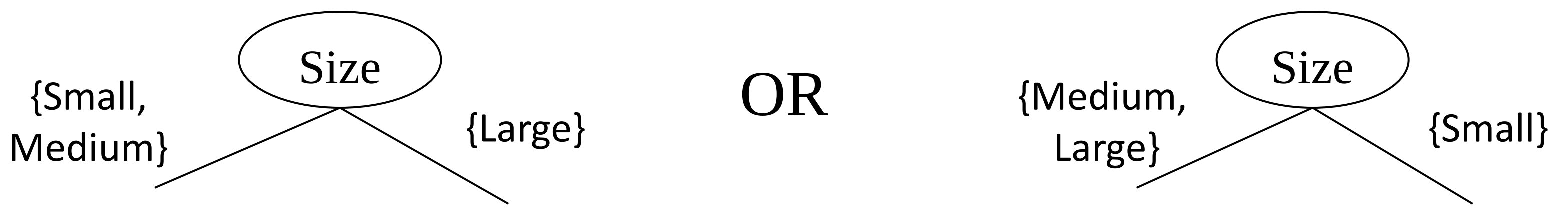
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Ordinal Attributes

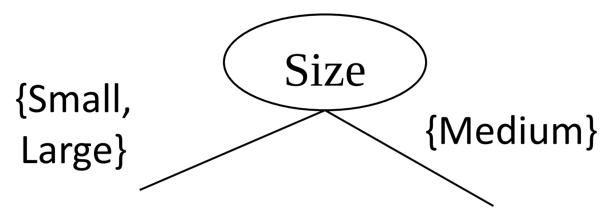
- Multi-way split: Use as many partitions as distinct values.



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



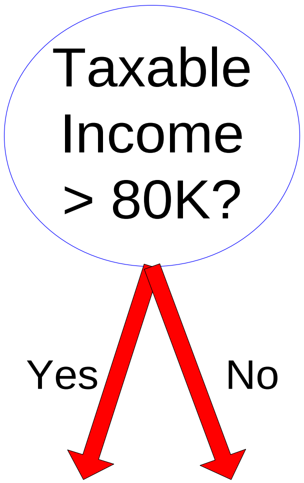
- What about this split?



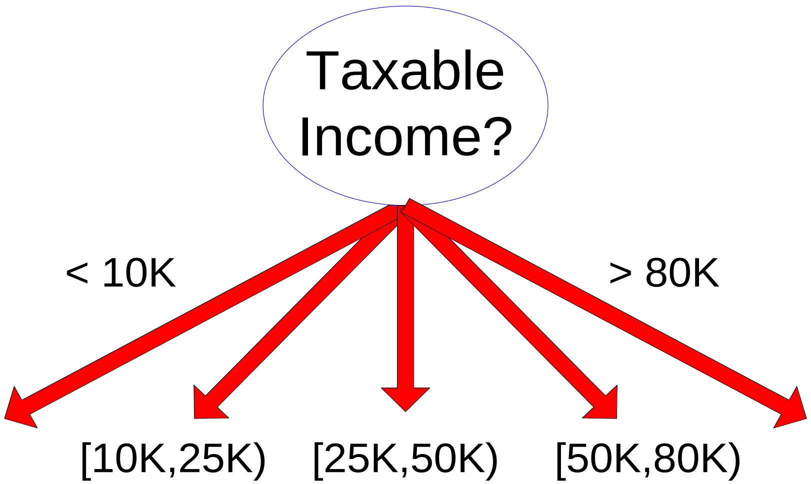
Splitting Based on Continuous Attributes

- 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

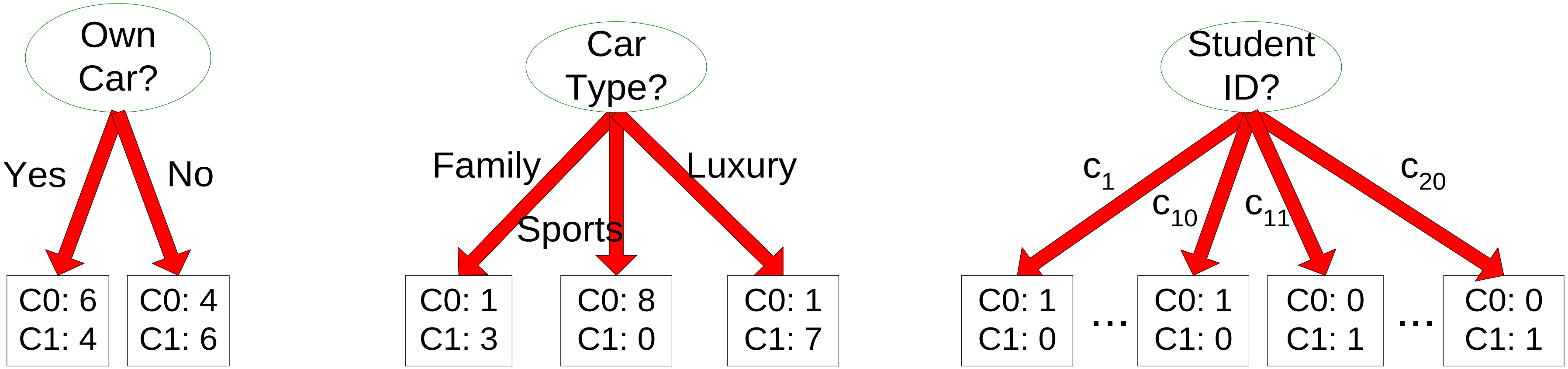
- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- 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

Student ID	Gender	Car Type	Shirt Size	Class
1	m	Family	Small	C0
2	m	Sports	Medium	C0
3	m	Sports	Medium	C0
4	m	Sports	Large	C0
5	m	Sports	Extra Large	C0
6	m	Sports	Extra Large	C0
7	f	Sports	Small	C0
14	m	Luxury	Extra Large	C1
15	f	Luxury	Small	C1
16	f	Luxury	Small	C1
17	f	Luxury	Medium	C1
18	f	Luxury	Medium	C1
19	f	Luxury	Medium	C1
20	f	Luxury	Large	C1

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

$$\text{Entropy}(\text{node}) = -\sum_{i=0}^{c-1} p(i|\text{node}) \log_2 p(i|\text{node})$$

$$\text{Gini}(\text{node}) = 1 - \sum_{i=0}^{c-1} [p(i|\text{node})]^2$$

$$\text{Classification Error}(\text{node}) = 1 - \max_i [p(i|\text{node})]$$

(NOTE: $p(\,i \mid \text{node})$ is the probability or relative frequency of class i at the node).

Purity Gain

After you have made a split – How much purity have you Gained?

$$\text{Purity Gain} = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j)$$

$N =$	The total number of records in the parent node
$N(vj) =$	The number of records in the child node $\boxed{vj}$
$k =$	The number of attribute values

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

$1-(0/6)^2 - (6/6)^2 = 0$

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

$1-(2/6)^2 - (4/6)^2 = 0.444$

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

$1-(1/6)^2 - (5/6)^2 = 0.278$

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

$1-(3/6)^2 - (3/6)^2 = 0.500$

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

Goal: Choose the split which maximizes the Purity Gain

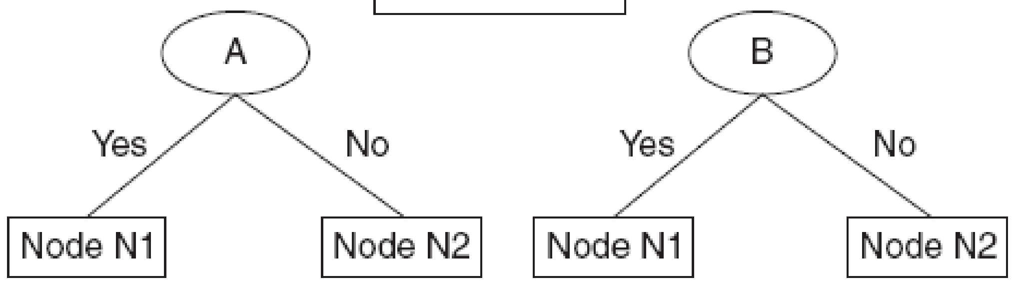
$$\boxed{PurityGain}$$

$$= Gini(parent) - \sum_{j=1}^{k} \frac{N(vj)}{N} Gini(vj)$$

$N =$	The total number of records in the parent node
$N(vj) =$	The number of records in the child node $\boxed{vj}$
$k =$	The number of attribute values

What is the best split?

	Parent
C0	6
C1	6
Gini = 0.500	



	N1	N2
C0	4	2
C1	3	3
Gini = 0.486		

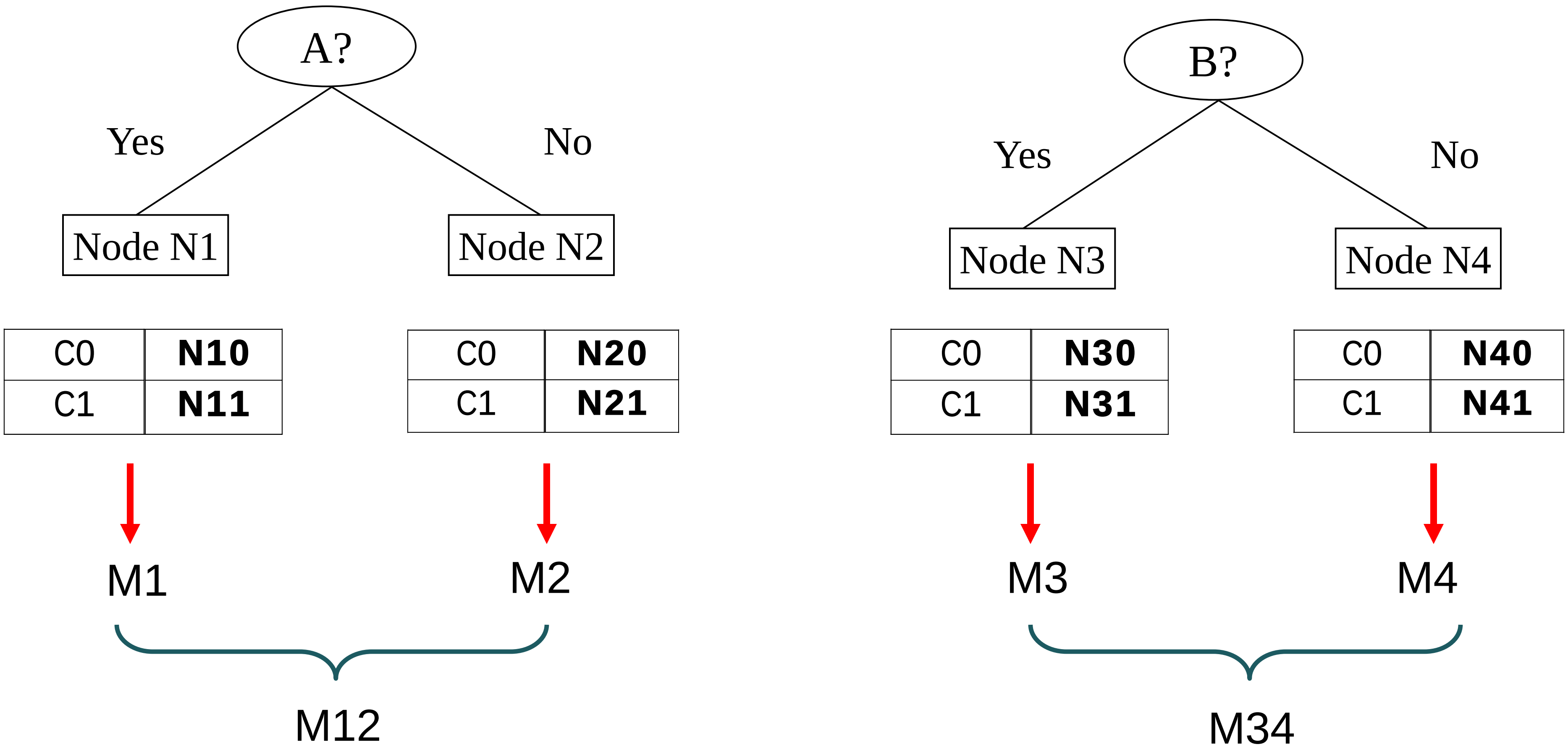
	N1	N2
C0	1	5
C1	4	2
Gini = 0.375		

How to Find the Best Split

Before Splitting:

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

→ M0



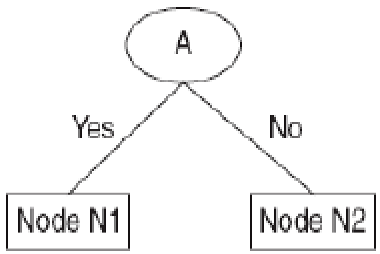
Gain = M0 – M12 vs M0 – M34

Calculate Purity Gain for Split A

	Parent
C0	6
C1	6
Gini = 0.500	

$$\text{Gini}(\text{node}) = 1 - \sum_{i=0}^{c-1} [p(i|\text{node})]^2$$

$$\text{Purity Gain} = \text{Gini}(\text{parent}) - \sum_{j=1}^{k} \frac{N(\nu_j)}{N} \text{Gini}(\nu_j)$$



	N1	N2
C0	4	2
C1	3	3

Gini for node N1 = $1 - (4/7)^2 – (3/7)^2 = 0.4898$
Gini for node N2 = $1 - (2/5)^2 – (3/5)^2 = 0.48$

Purity Gain for Split A = 0.5 – (7/12)0.4898 – (5/12)0.48
= 0.5 – 0.486 = 0.014

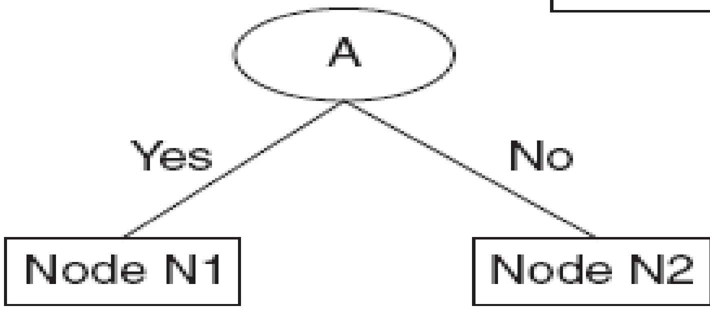
Purity Gain for Split A is 0.014

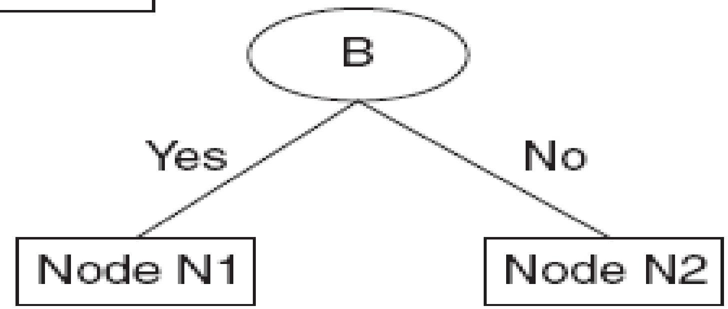
Select Split with greatest Purity Gain

$$\mathrm{Gini}(node) = 1 - \sum_{i=0}^{c-1} [p(i|node)]^2$$

	Parent
C0	6
C1	6
Gini = 0.500	

$$\text{Purity Gain} = \mathrm{Gini}(parent) - \sum_{j=1}^{k} \frac{N(v_j)}{N} \mathrm{Gini}(v_j)$$





	N1	N2
C0	4	2
C1	3	3

Purity Gain = 0.014

	N1	N2
C0	1	5
C1	4	2

Purity Gain = 0.125

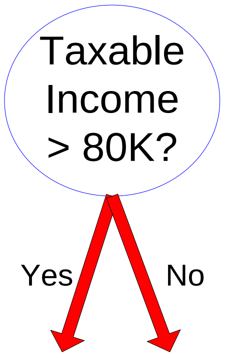
Choose Split B

Gain for Split A = 0.014 while for Node B = 0.5 – 0.375 = 0.125

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 $A \geq v$
- 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: 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:

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

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

  – Measures 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

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

= -(1/6) (-2.58) – (5/6)(-.26)

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

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_{i} p(i \mid t)$$

- 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

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

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

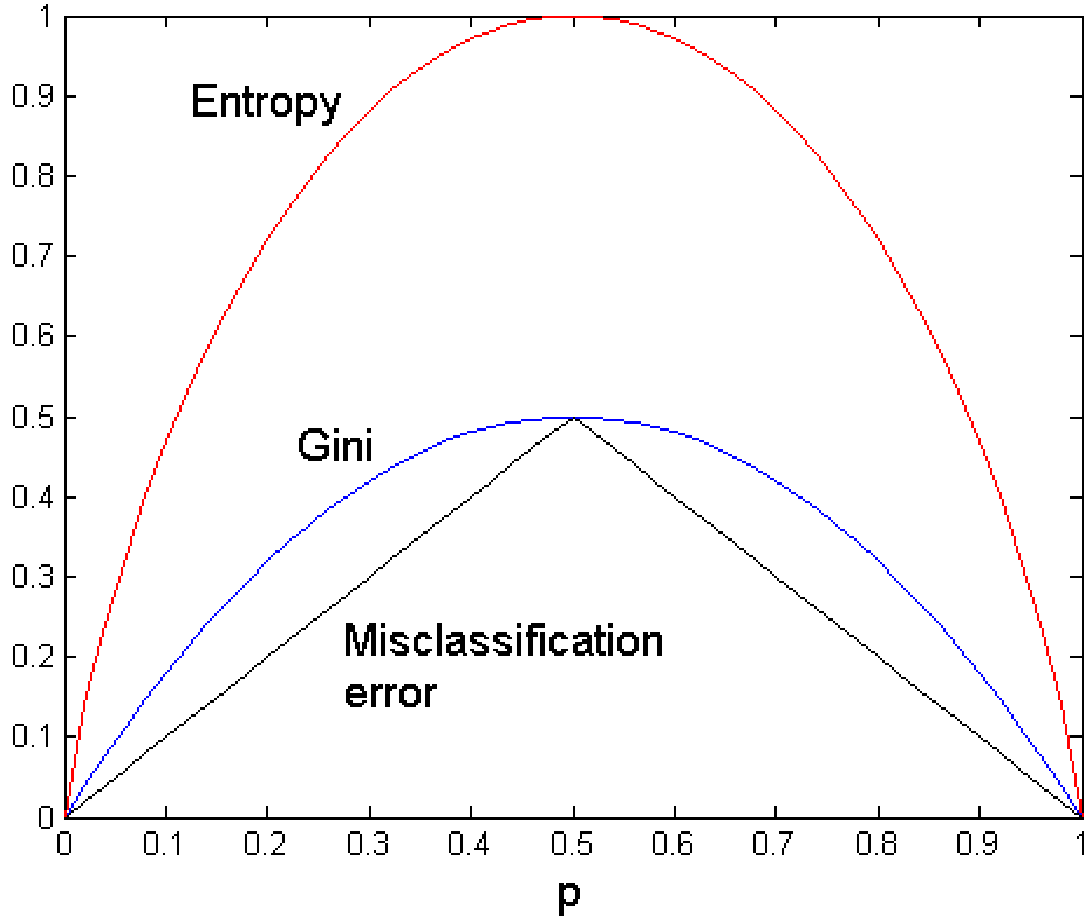
C1	**2**
C2	**4**

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 2-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

- 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
- Disadvantage:
  - Weak learner
    - Slight change in data results in very different tree.

Practical Issues of Classification

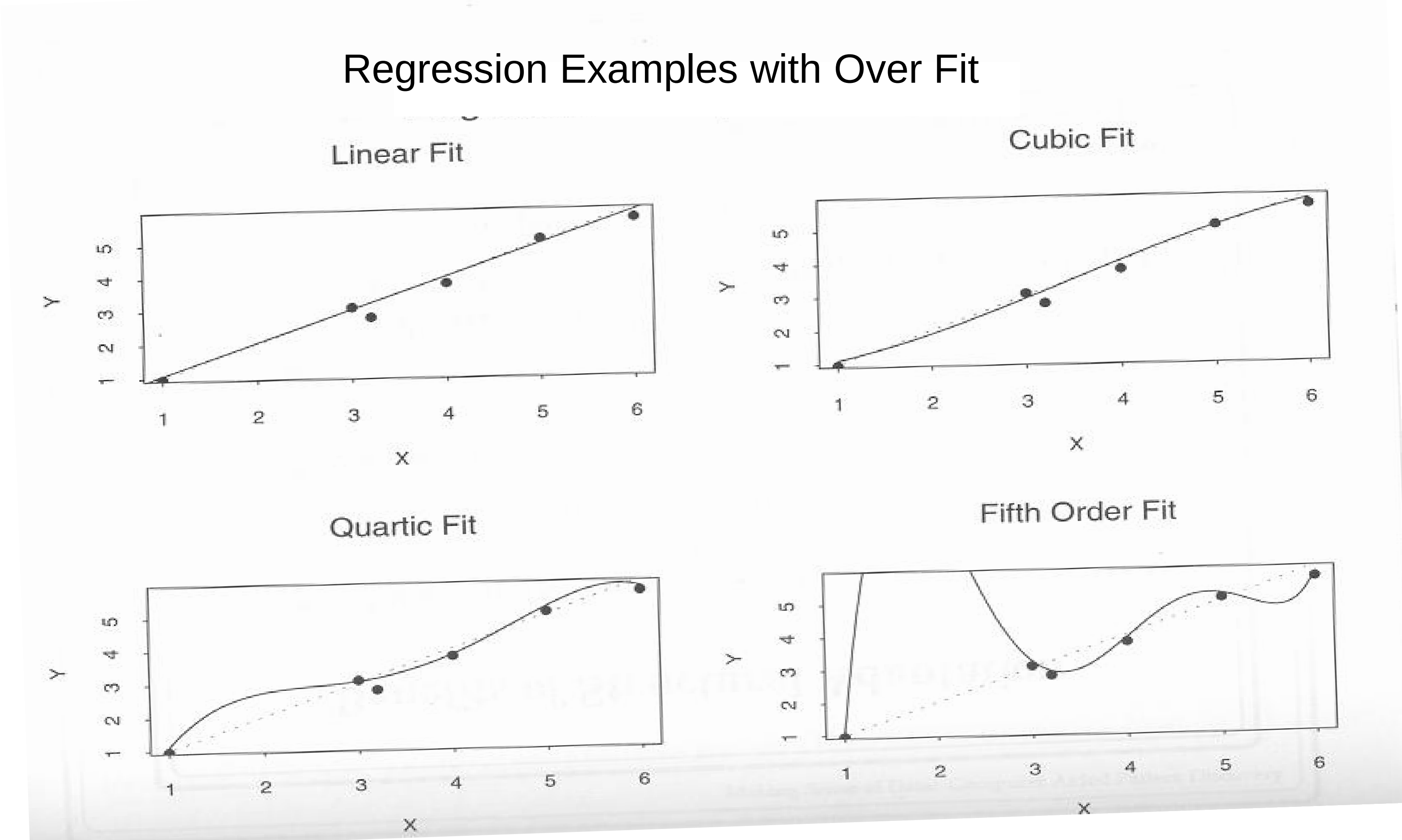
- Underfitting and Overfitting

- Missing Values

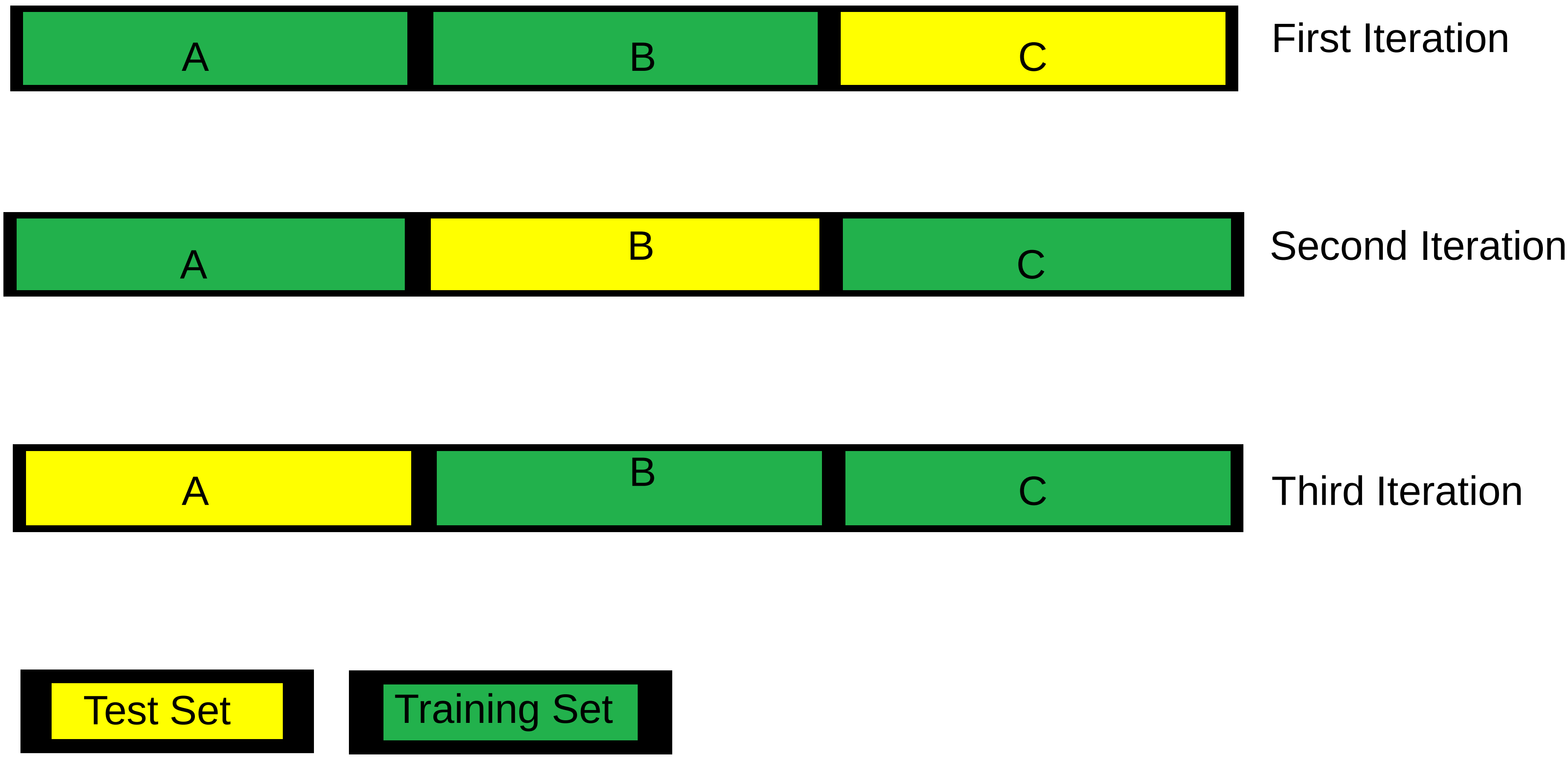
- Costs of Classification

Which Model is Best for this Data?

Complexity increases with Order of Polynomial



Cross Validation – Three Fold



Cross Validation – Three Fold

- Divide the Data Set into three parts. In each iteration one third of the data are designated as the Test Set. The remaining 2/3 form the Training Set.

- First Iteration: The model is built using Training Data Set (A + B). When this model is applied to the Training Set, the first training error is calculated as this rms error. The test error for the first iteration is calculated by applying this first model to the first iteration Test Set (C).

- Repeat this process three times. Average the rms training errors. Average the rms test errors.

Model Complexity

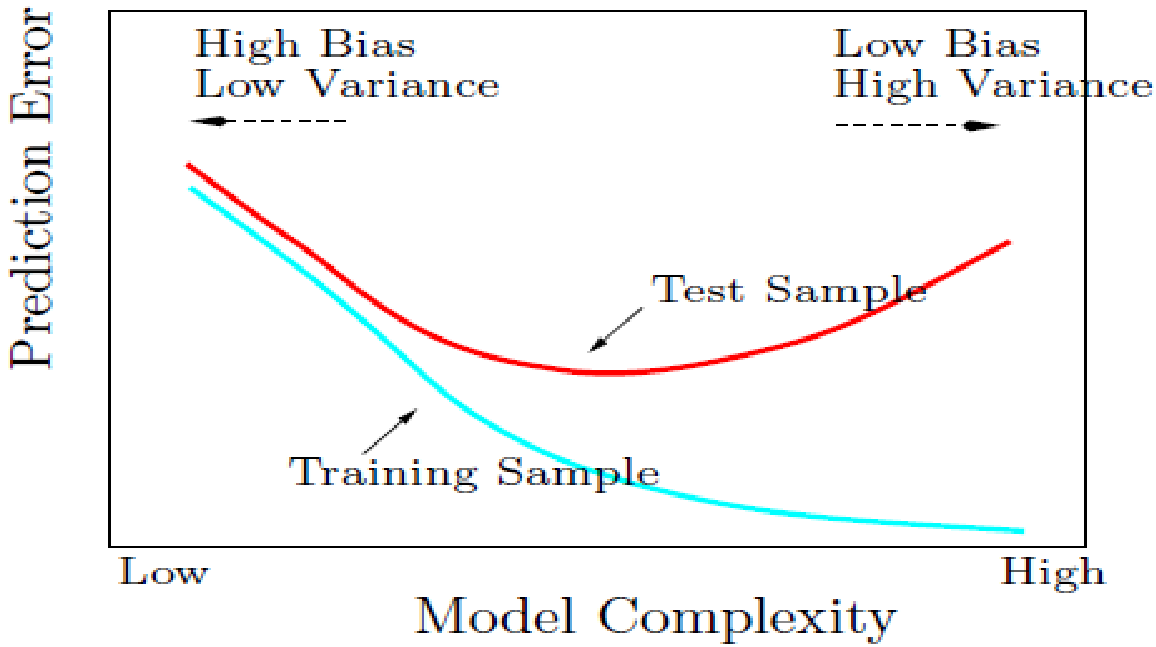
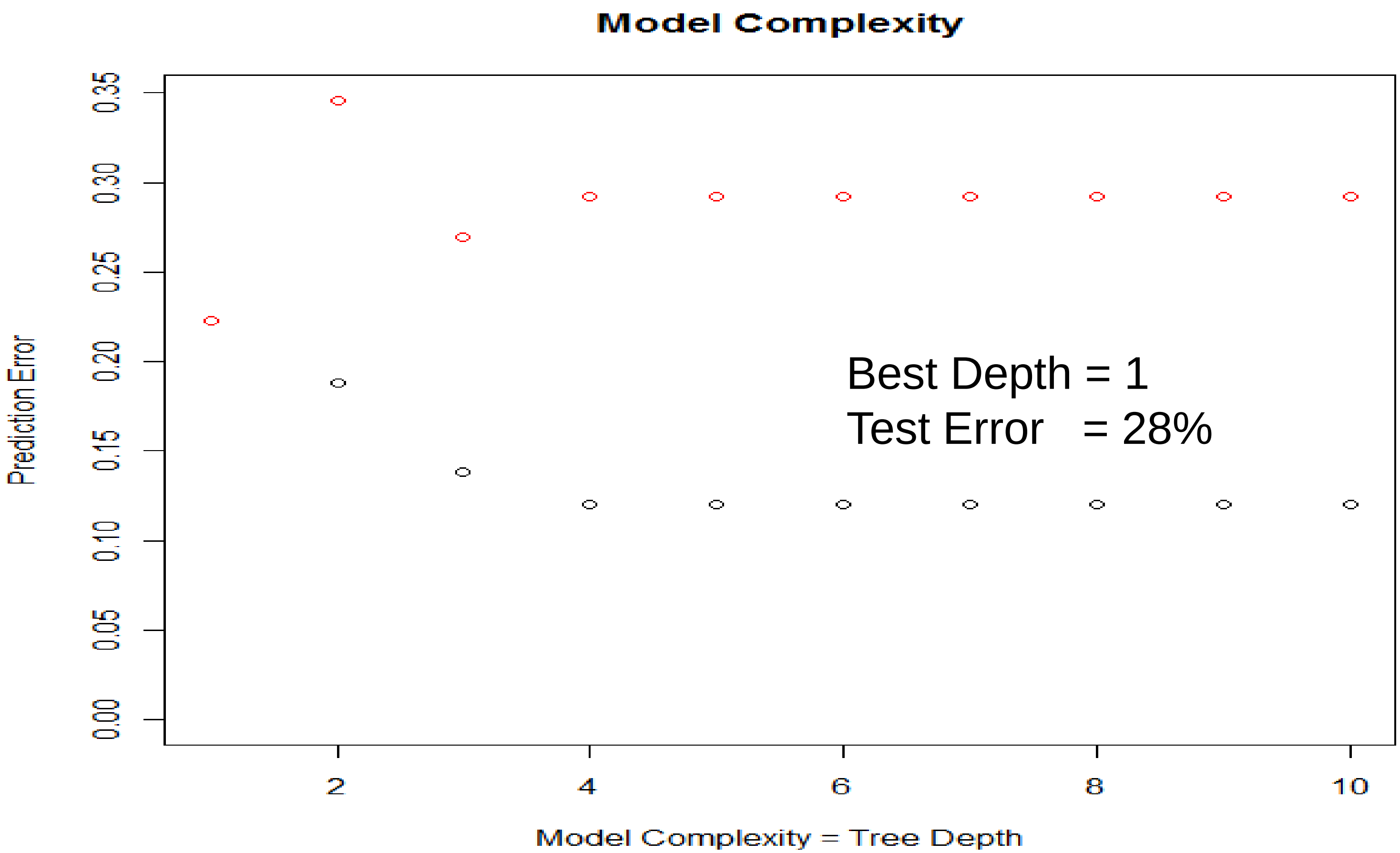


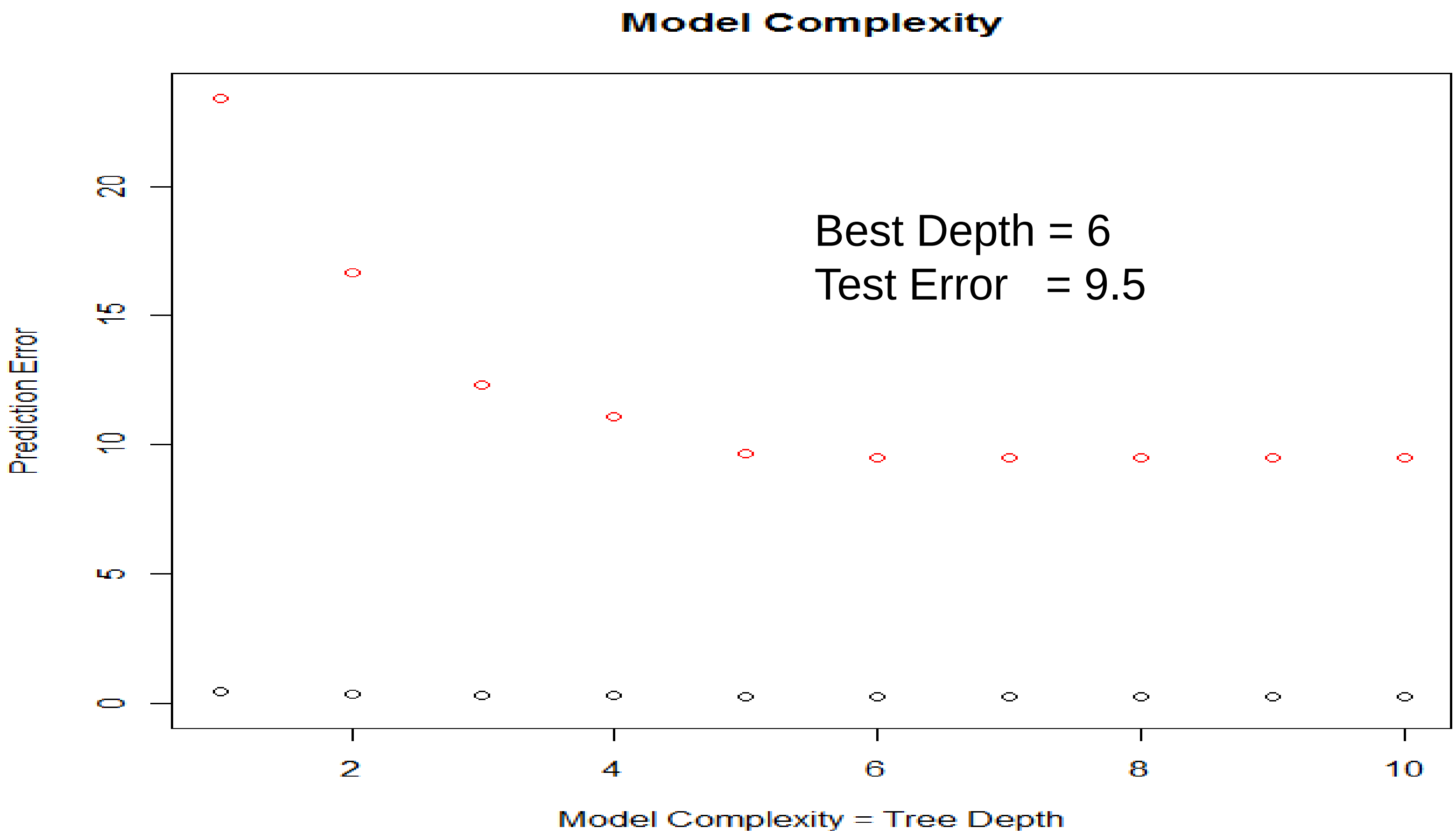
FIGURE 2.11. *Test and training error as a function of model complexity.*

From Elements of Statistical Learning by Hastie, Tibshirani, Friedman

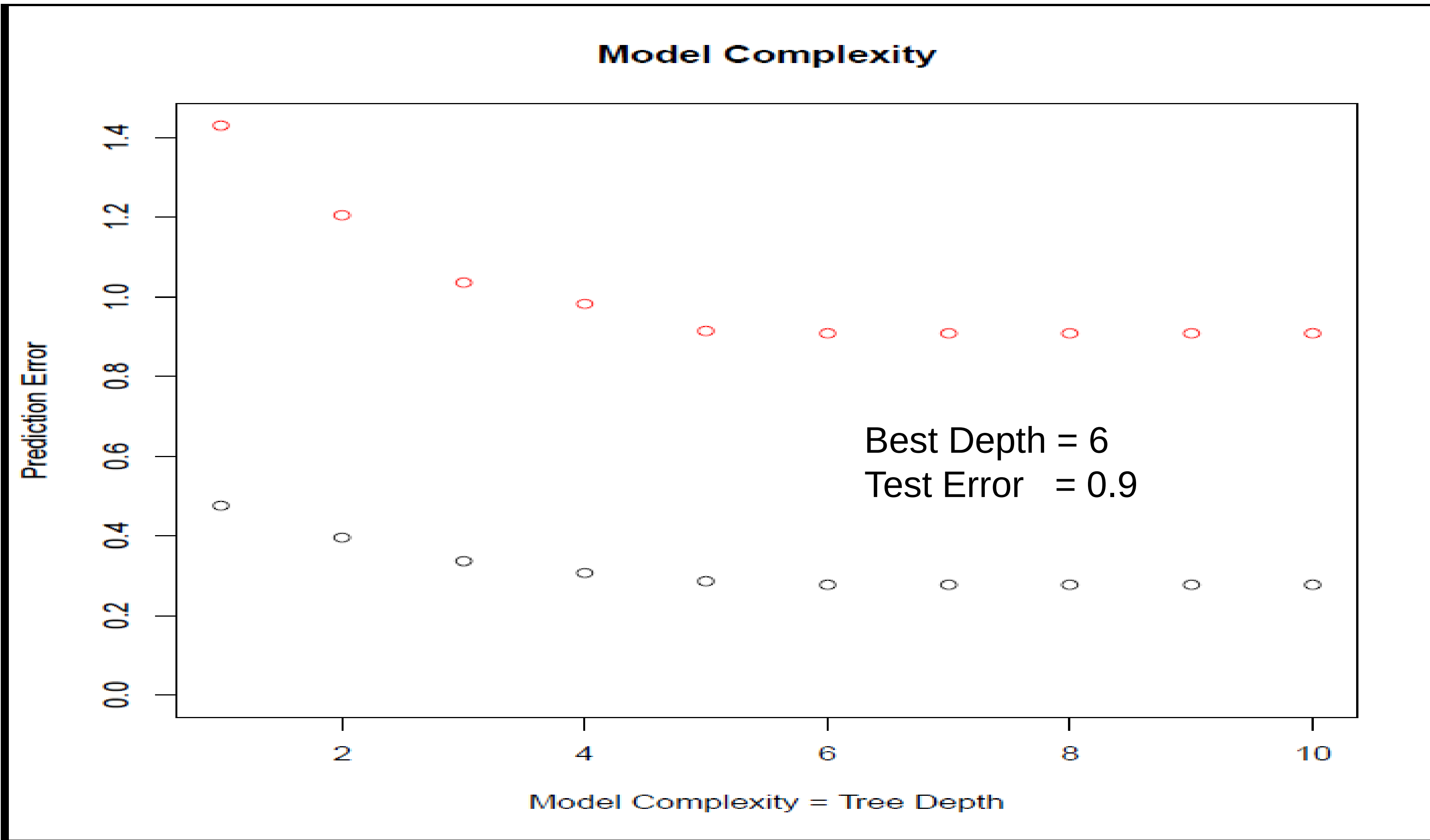
Model Complexity – Sonar Data SonarRpartXval.r



Model Complexity – Concrete Data ConcreteRpartXval.r



Model Complexity – Concrete Data ConcreteRpartXval.r



Use Cross Validation to Determine Best Fit

-A tree can overfit SO CAN ALL OTHER ML METHODS

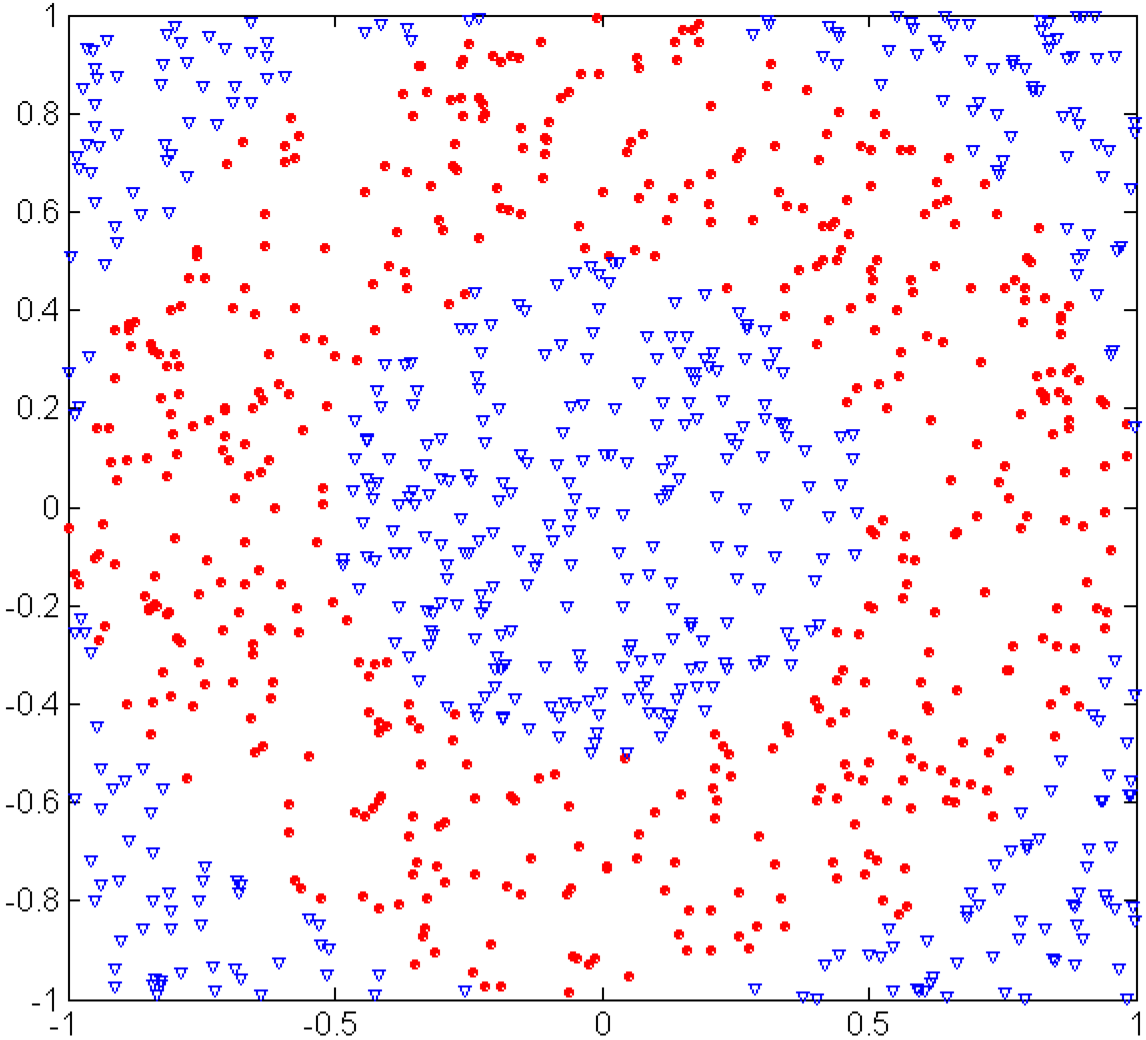
-How can we estimate the degree of underfit or overfit??

-Holdout Method

-K-fold cross validation

-Hold out 1st set, train on 2-k, then hold out 2 and train on 1 + 3-k etc.
-Calculate average error on training set and average error on test set.

Underfitting and Overfitting



500 circular and 500 triangular data points.

Circular points:

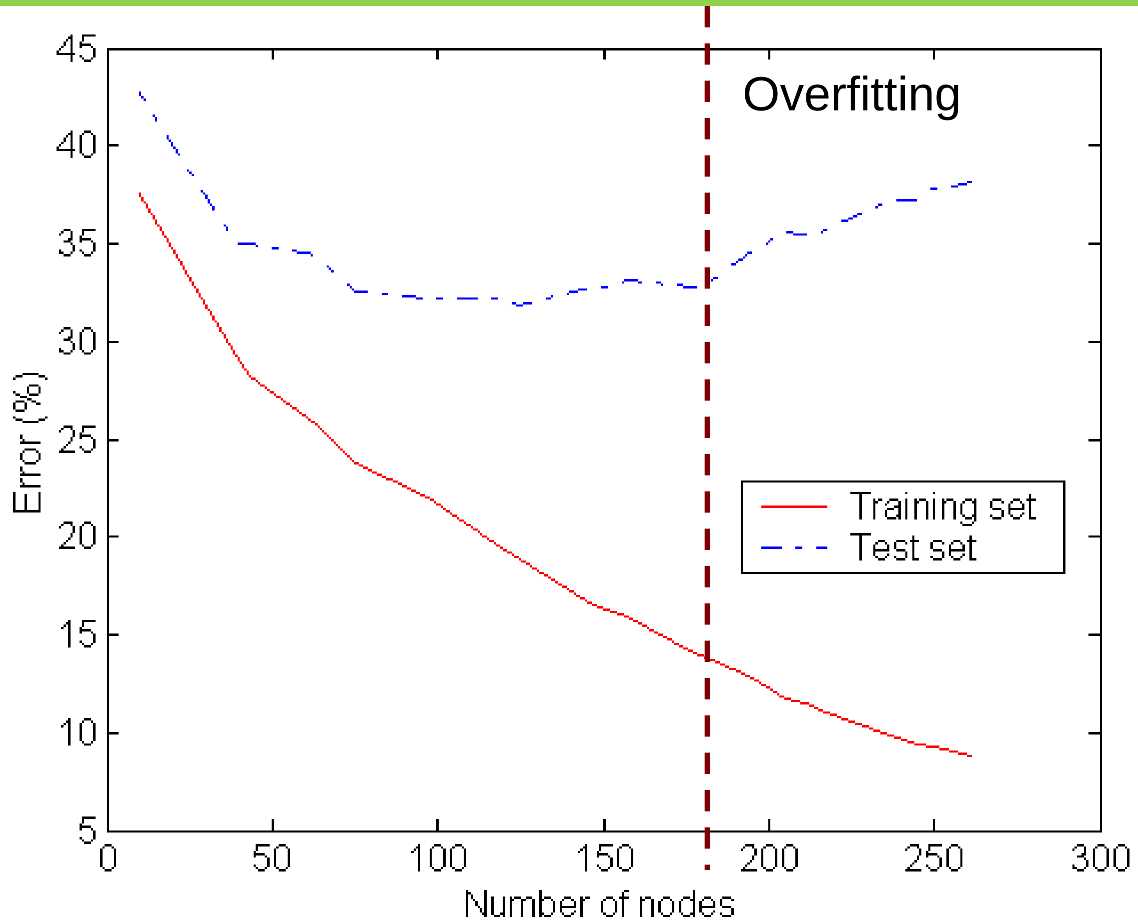
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

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

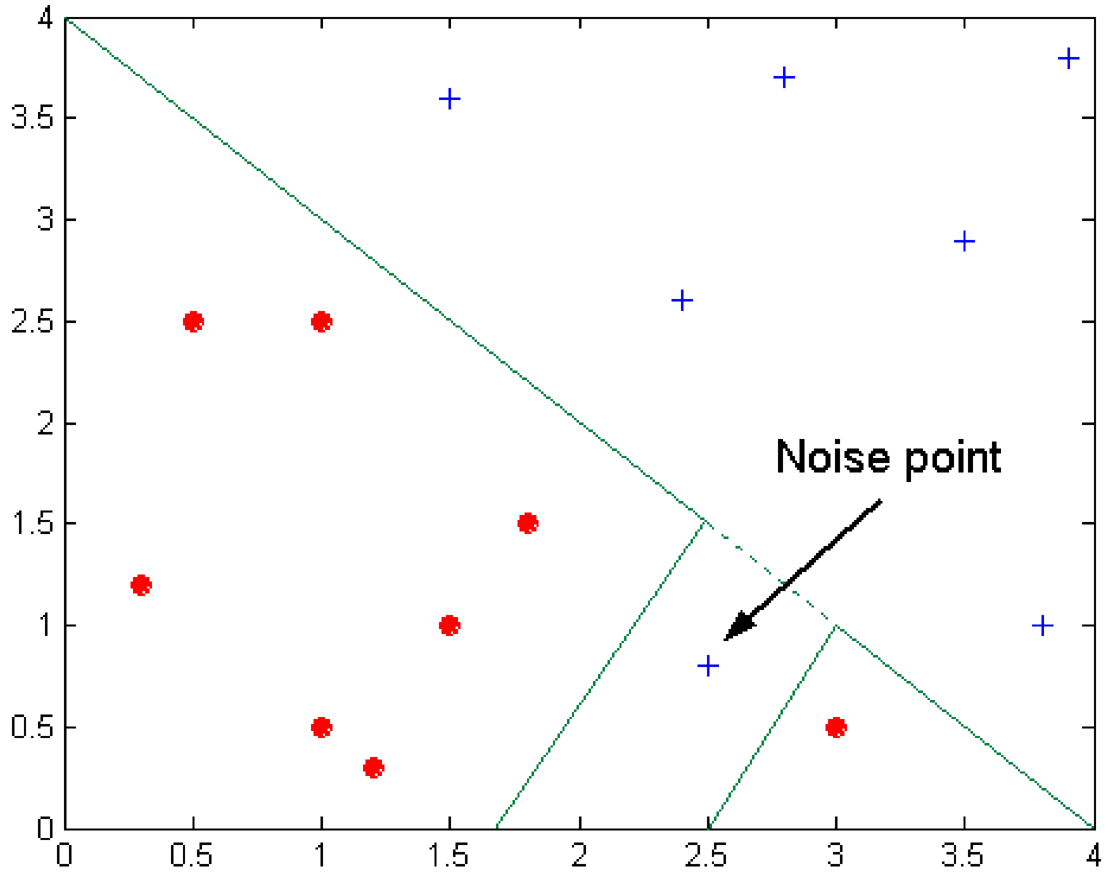
$\text{sqrt}(x_1^2+x_2^2) < 1$

Underfitting and Overfitting



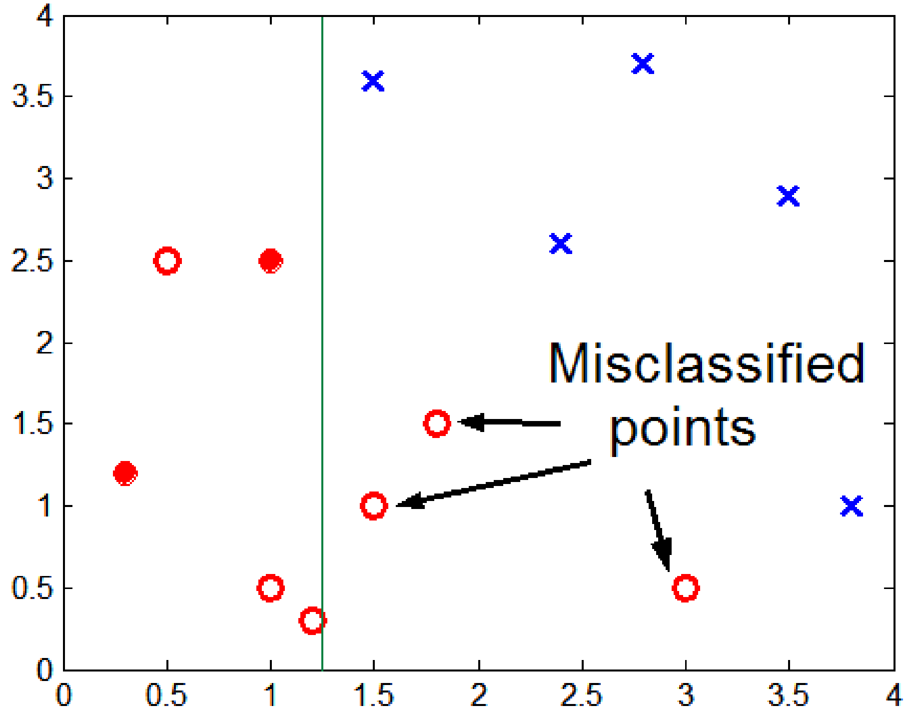
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient



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

Machine Learning –IDM Sect 4.4 -4.6

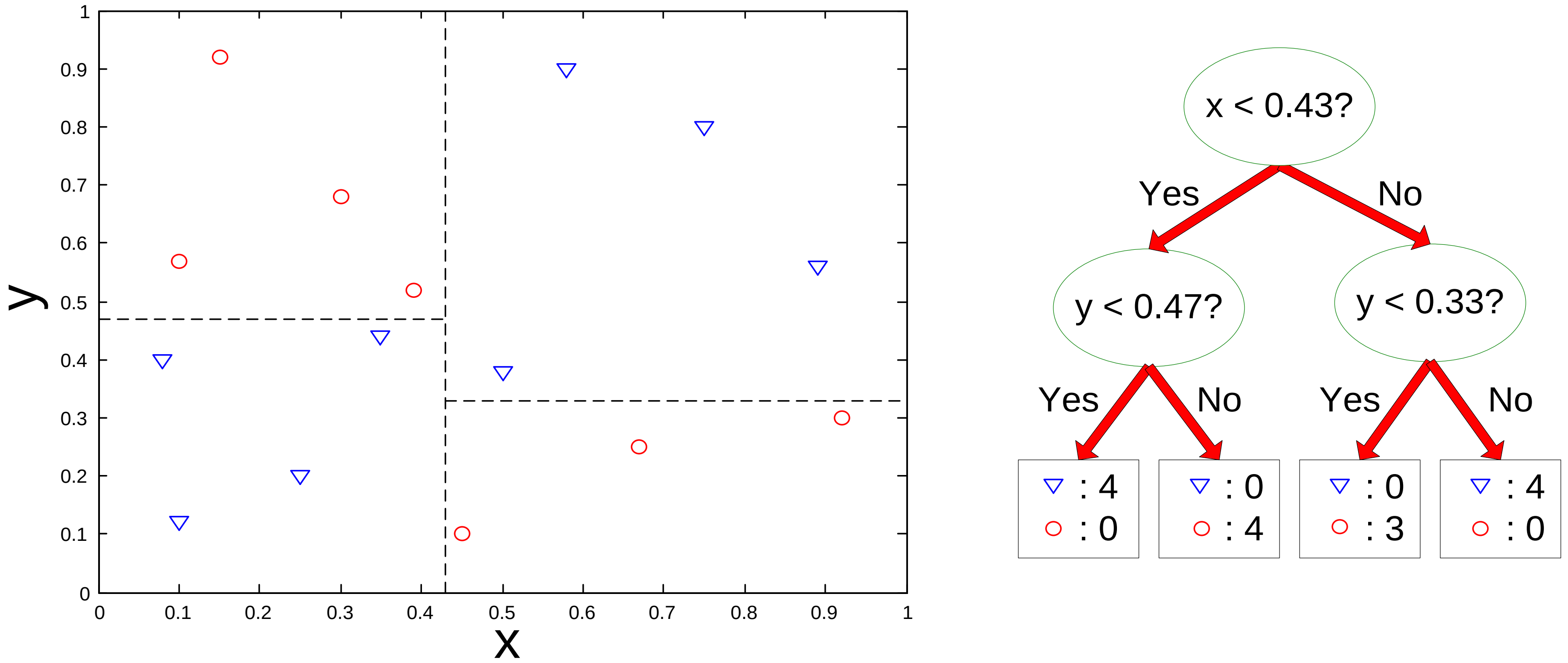
-Let's try crossvalidationwith the sonar classification tree

```
train<-read.csv("sonar_train.csv",header=FALSE)
nxval<-10
out <-matrix(nrow= nxval, ncol= 2)
I <-seq(from = 1, to = nrow(train))
for(idepthin seq(from = 1, to = 10)){
trainErr<-0.0
testErr<-0.0
for(ixvalin seq(from = 1, to = nxval)){
Iout<-which(I%%nxval== ixval%%nxval)
trainIn<-train[-Iout,]
trainOut<-train[Iout,]
yin <-as.factor(trainIn[,61])
yout<-as.factor(trainOut[,61])
xin<-trainIn[,1:60]
xout<-trainOut[,1:60]
fit <-rpart(yin~.,xin,control=rpart.control(maxdepth=idepth))
trainErr<-trainErr+ (1-sum(yin==predict(fit,xin,type= "class"))/length(yin))
testErr<-testErr+ (1-sum(yout==predict(fit,xout,type="class"))/length(yout))
}
out[idepth,1] <-trainErr/nxval
out[idepth,2] <-testErr/nxval
}
```

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

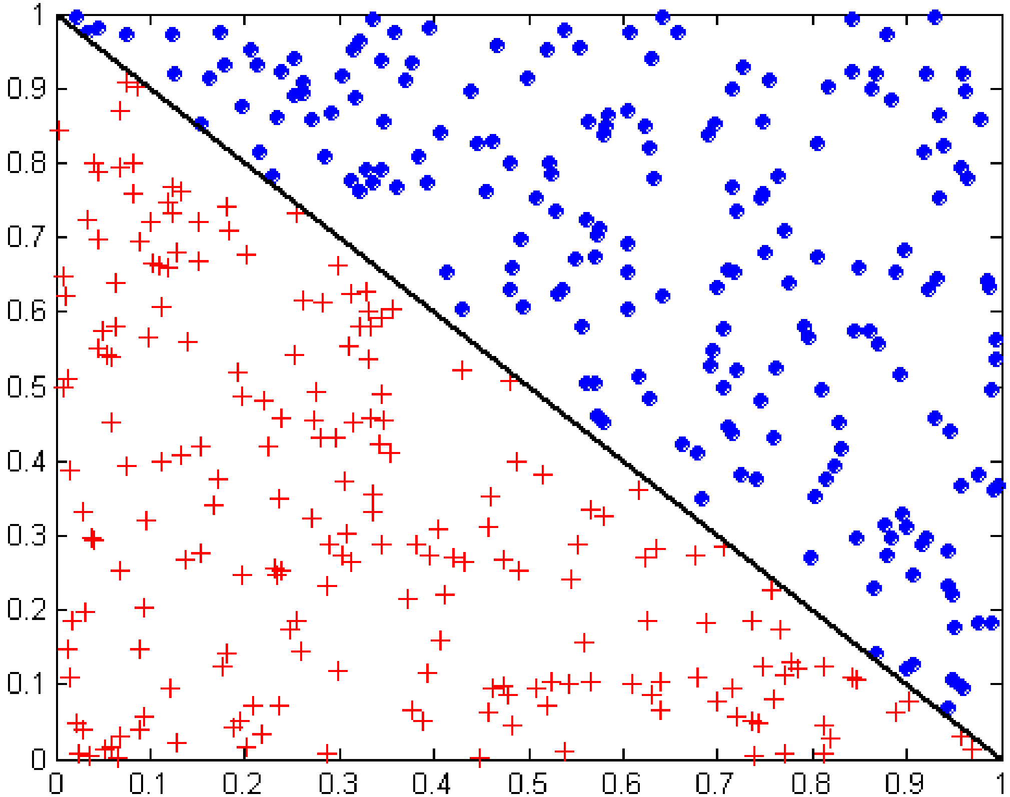
Decision Boundary

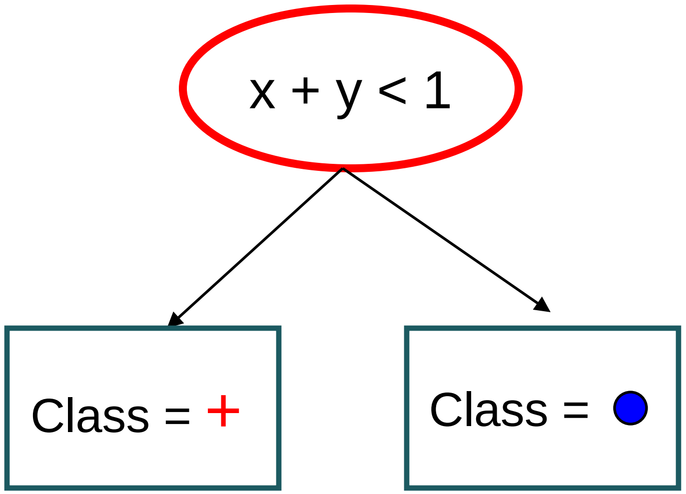


- Border line between two neighboring regions of different classes is known as decision boundary

- 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

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

Decision Tree Summary

Nonparametric Approach

– No prior assumptions made on the underlying probability distributions

Heuristic-based approach guides search in vast hypothesis space

- Computationally efficient even when the training set size is very large

Classifying a test record is extremely fast, once a decision tree has been built

Smaller-sized decision trees are relatively easy to interpret

- accuracies are comparable to other classification techniques