DATA MINING
SUPERVISED LEARNING

Regression

Classification

- Decision Trees
- Evaluation

Supervised learning

- In supervised learning, except for the feature variables that describe the data, we also have a target variable
- The goal is to learn a function (model) that can predict the value of the target variable given the features
  - We learn the function using a labeled training set.
- Regression: The target variable is numerical and continuous
  - The price of a stock, the grade in a class, the height of a child, the life expectancy etc
- Classification: The target variable is discrete
  - Will the stock go up or down? Will the student pass or fail? Is a transaction fraudulent or not? What is the topic of an article?
- Predictive modeling is in the heart of the data science revolution.

LINEAR REGRESSION

Regression

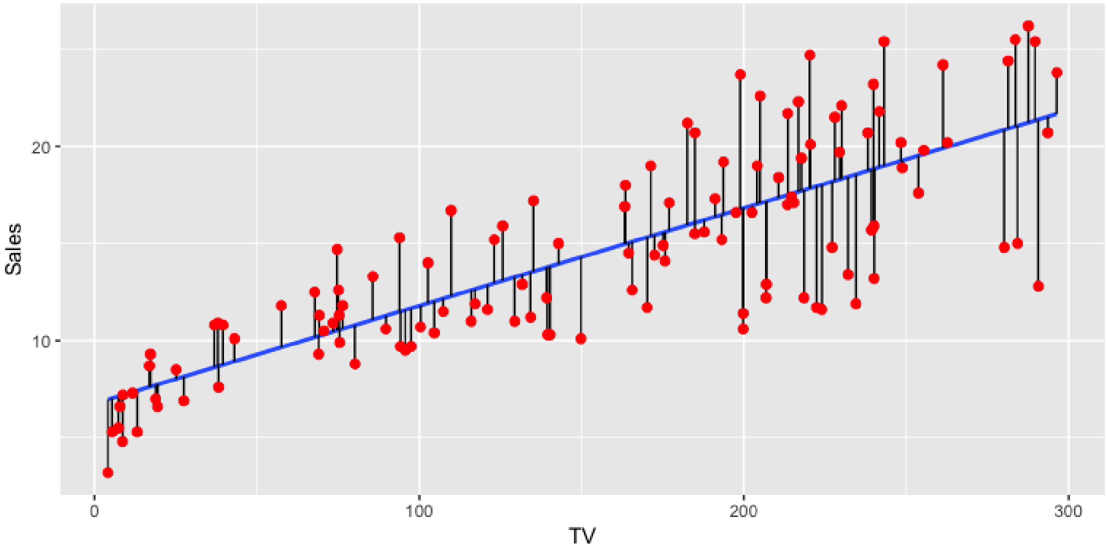
- We assume that we have k feature variables:
  - Also known as covariates, or independent variables
- The target variable is also known as dependent variable.
- We are given a dataset of the form $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ where, $\boldsymbol{x}_i$ is a k-dimensional feature vector, and y_i a real value
- We want to learn a function f which given a feature vector $\boldsymbol{x}_i$ predicts a value $y_i' = f(\boldsymbol{x}_i)$ that is as close as possible to the value y_i
- Minimize sum of squares:

$$\sum_i \left(y_i - f(\boldsymbol{x}_i)\right)^2$$

Linear regression

- The simplest form of f is a linear function
- In linear regression the function f is typically of the form:

$$f(\boldsymbol{x_i}) = w_0 + \sum_{j=1}^{k} w_j x_{ij}$$



One-dimensional linear regression

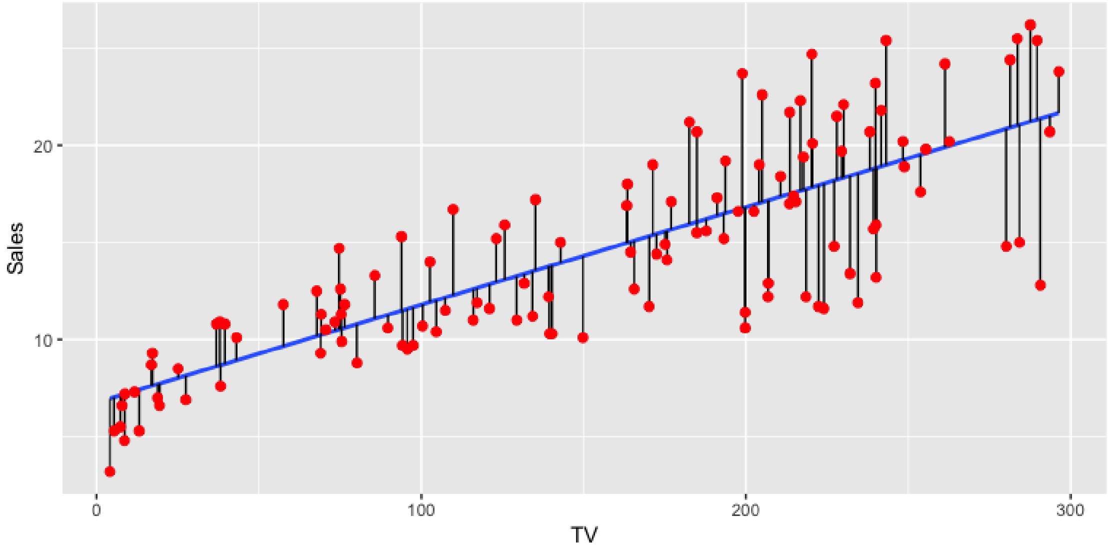
In the simplest case we have a single variable and the function is of the form:

$$f(x_i) = w_0 + w_1 x_i$$

Minimizing the error gives:

$$w_0 = \bar{y} - w_1 \bar{x}$$

$$w_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = r_{xy} \frac{\sigma_y}{\sigma_x}$$



$\bar{x}$: mean value of x_i's

$\bar{y}$: mean value of y_i's

r_{xy}: correlation coefficient between $\boldsymbol{x}, \boldsymbol{y}$

Multiple linear regression

- In the general case we have k features, and $\boldsymbol{x_i}$,$\boldsymbol{w}$ are vectors.
- We simplify the notation:

$$\boldsymbol{x_i} = (1, x_{i1}, \dots, x_{ik})$$
$$\boldsymbol{w} = (w_0, w_1, \dots, w_k)$$
$$f(\boldsymbol{x_i}, \boldsymbol{w}) = \boldsymbol{x_i}^T \boldsymbol{w}$$

- Let X be the $n \times (k+1)$ matrix with vectors $\boldsymbol{x_i}$ as rows.
- Let $\boldsymbol{y} = (y_1, \dots, y_n)$
- We can write the SSE function as:

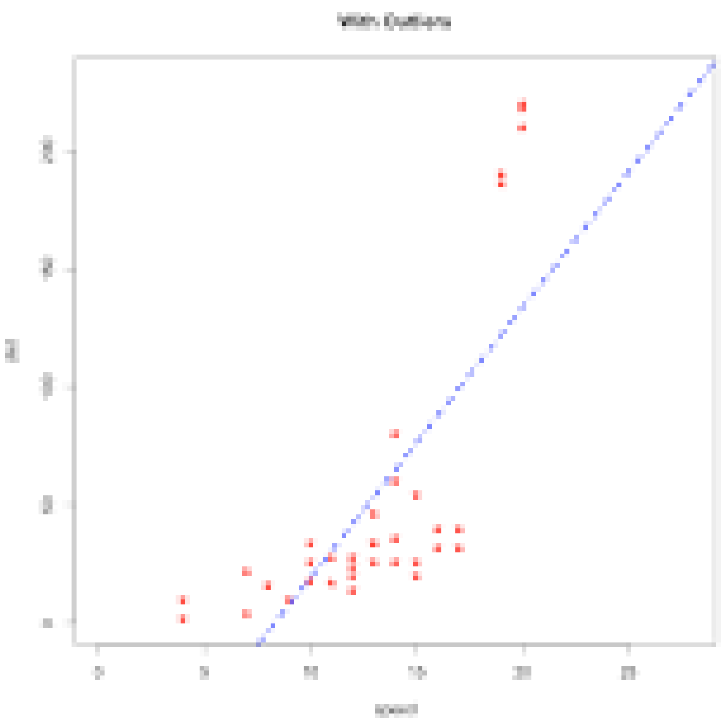
$$SSE = \|X\boldsymbol{w} - \boldsymbol{y}\|^2$$

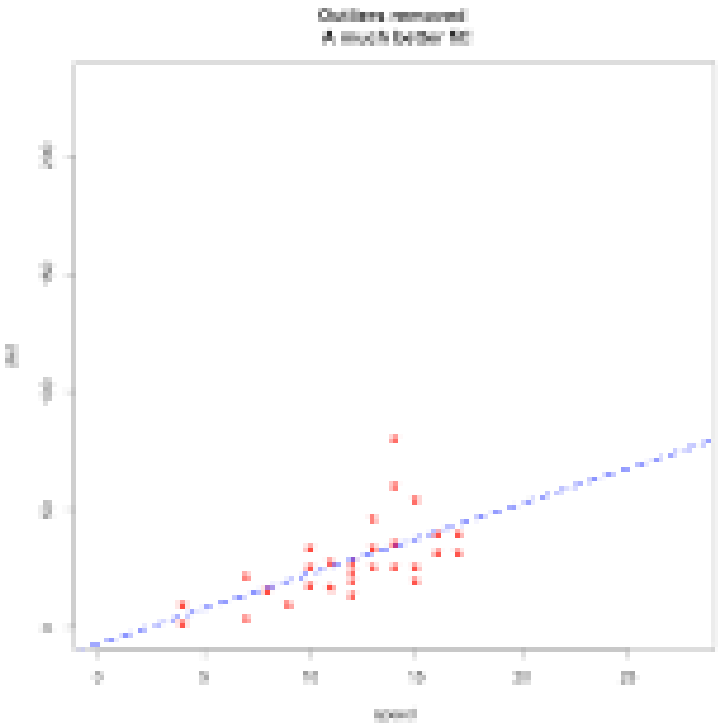
- There is a closed-form solution for $\boldsymbol{w}$:

$$\boldsymbol{w} = \left(X^T X\right)^{-1} X^T \boldsymbol{y}$$

- Matrix inversion may be too expensive. Other optimization techniques are often used to find the optimal vector (e.g., Gradient Descent)

Outliers





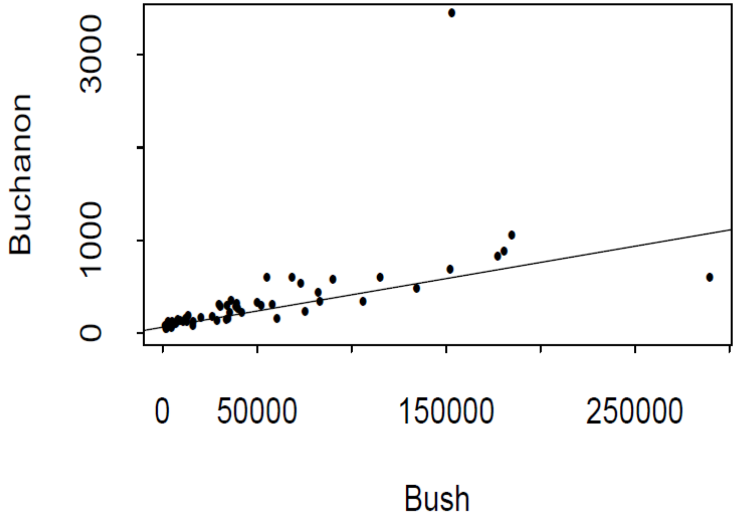
- Regression is sensitive to outliers:
  - The line will “tilt” to accommodate very extreme values
- Solution: remove the outliers
  - But make sure that they do not capture useful information

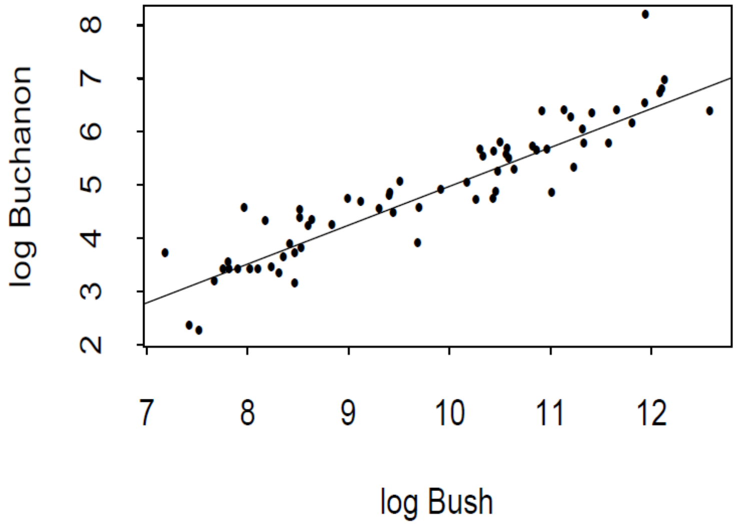
Normalization

- In the regression problem some times our features may have very different scales:
  - For example: predict the GDP of a country using as features the percentage of home owners and the income
  - The weights in this case will not be interpretable
- Solution: Normalize the features by replacing the values with the z-scores

More complex models

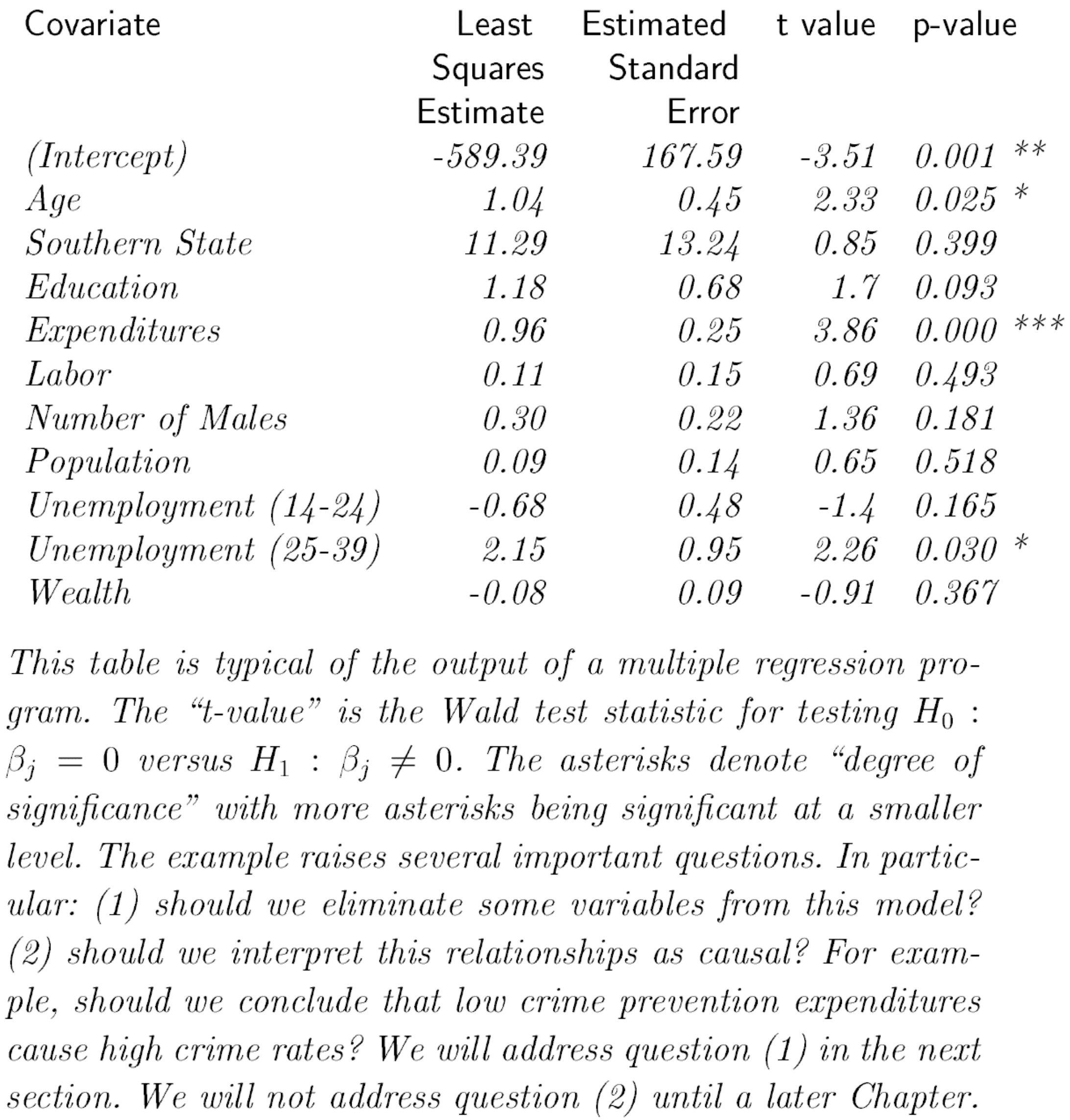
- The model we have is linear with respect to the parameters $\boldsymbol{w}$ but the features we consider may be non-linear functions of the $\boldsymbol{x}_i$ values.
- To capture more complex relationships we can take a transformation of the input (e.g., logarithm $\log x_{ij}$), or add polynomial terms (e.g., x_{ij}^2).
  - However this may increase a lot the number of features





Interpretation and significance

- A regression model is useful for making predictions for new data.
- The coefficients for the linear regression model are also useful for understanding the effect of the independent variables to the value of the dependent variable
  - The w_j value is the effect of the increase of x_{ij} by one to the value y_i
- We can also compute the significance of the value of w_j by testing the null hypothesis that $w_j = 0$



Covariate	Least Squares Estimate	Estimated Standard Error	t value	p-value
(Intercept)	*-589.39*	*167.59*	*-3.51*	*0.001 ***
Age	*1.04*	*0.45*	*2.33*	*0.025 **
Southern State	*11.29*	*13.24*	*0.85*	*0.399*
Education	*1.18*	*0.68*	*1.7*	*0.093*
Expenditures	*0.96*	*0.25*	*3.86*	*0.000 ****
Labor	*0.11*	*0.15*	*0.69*	*0.493*
Number of Males	*0.30*	*0.22*	*1.36*	*0.181*
Population	*0.09*	*0.14*	*0.65*	*0.518*
Unemployment (14-24)	*-0.68*	*0.48*	*-1.4*	*0.165*
Unemployment (25-39)	*2.15*	*0.95*	*2.26*	*0.030 **
Wealth	*-0.08*	*0.09*	*-0.91*	*0.367*

This table is typical of the output of a multiple regression program. The "t-value" is the Wald test statistic for testing $H_0 : \beta_j = 0$ versus $H_1 : \beta_j \neq 0$. The asterisks denote "degree of significance" with more asterisks being significant at a smaller level. The example raises several important questions. In particular: (1) should we eliminate some variables from this model? (2) should we interpret this relationships as causal? For example, should we conclude that low crime prevention expenditures cause high crime rates? We will address question (1) in the next section. We will not address question (2) until a later Chapter.

Predicting Crime rate

CLASSIFICATION

Classification

- Similar to the regression problem we have features and a target variable that we want to model/predict
- The target variable is now discrete. It is often called the class label
  - In the simplest case, it is a binary variable.

Example: Catching tax-evasion

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**

Tax-return data for year 2011

A new tax return for 2012
Is this a cheating tax return?

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**

An instance of the classification problem: learn a method for discriminating between records of different classes (cheaters vs non-cheaters)

Classification

- **Classification** is the task of ***learning a target function* f** that maps attribute set **x** to one of the predefined class labels **y**

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**

One of the attributes is the class attribute

In this case: Cheat

Two class labels (or classes): Yes (1), No (0)

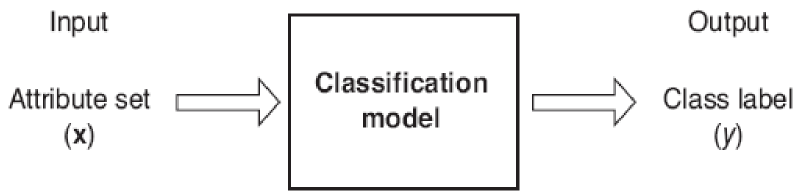


Figure 4.2. Classification as the task of mapping an input attribute set $\mathbf{x}$ into its class label y.

Why classification?

- The target function f is known as a classification model
- Descriptive modeling: Explanatory tool to distinguish between objects of different classes (e.g., understand why people cheat on their taxes, or what makes a hipster)
- Predictive modeling: Predict a class of a previously unseen record

Examples of Classification Tasks

- Predicting tumor cells as benign or malignant
- Classifying credit card transactions as legitimate or fraudulent
- Categorizing news stories as finance, weather, entertainment, sports, etc
- Identifying spam email, spam web pages, adult content
- Understanding if a web query has commercial intent or not

Classification is everywhere in data science
Big data has the answers to all questions.

General approach to classification

- Obtain a training set consisting of records with known class labels

- Training set is used to build a classification model

- A labeled test set of previously unseen data records is used to evaluate the quality of the model.

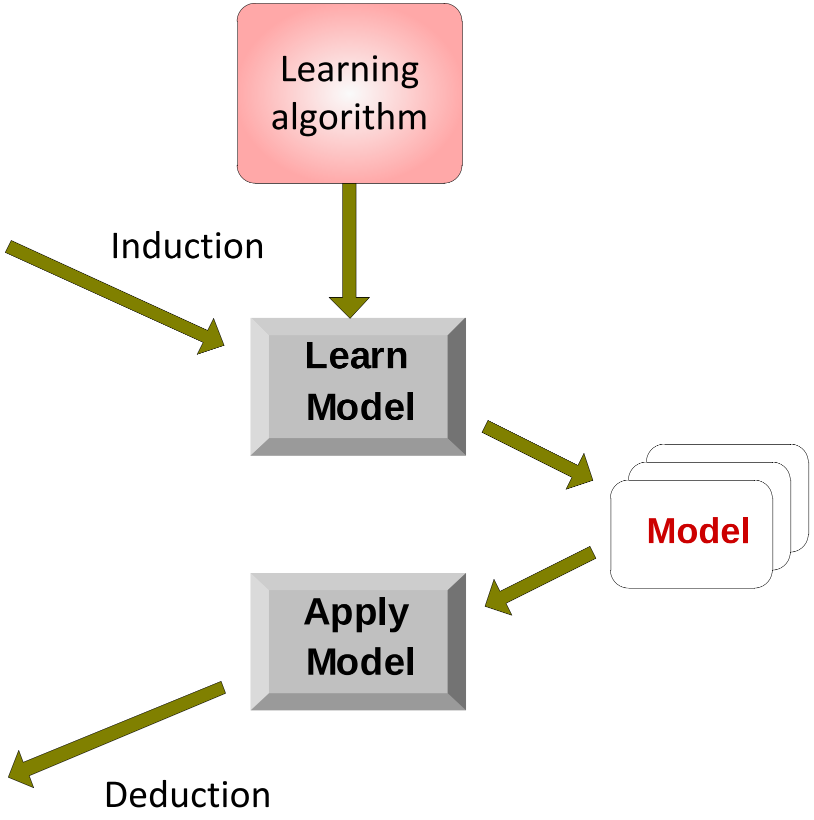
- The classification model is applied to new records with unknown class labels

- Important intermediate step: Decide on what features to use

Illustrating Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set



Learning algorithm

Induction

Learn Model

Model

Apply Model

Deduction

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Evaluation of classification models

- Counts of test records that are correctly (or incorrectly) predicted by the classification model
- **Confusion matrix**

		Predicted Class	
		Class = 1	**Class = 0**
Actual Class	**Class = 1**	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

$$\text{Accuracy} = \frac{\#\,\text{correct predictions}}{\text{total}\,\#\,\text{of predictions}} = \frac{f_{11} + f_{00}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

$$\text{Error rate} = \frac{\#\,\text{wrong predictions}}{\text{total}\,\#\,\text{of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}$$

Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
- Logistic Regression

Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

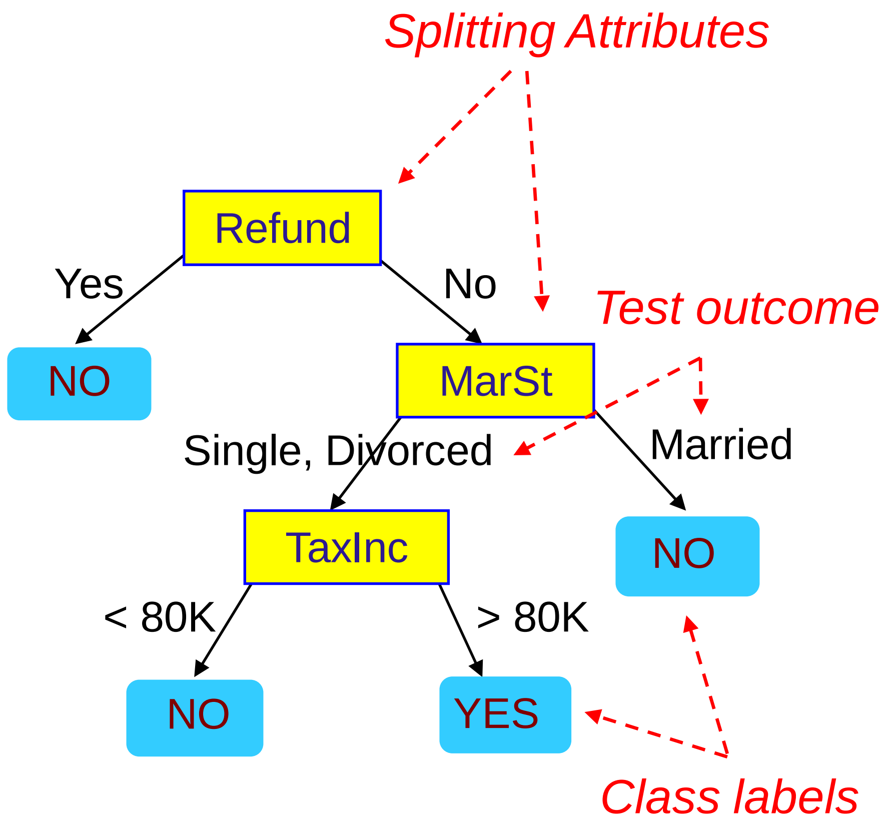
Decision Trees

- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution

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



Splitting Attributes

Refund

Yes

No

NO

MarSt

Test outcome

Single, Divorced

Married

TaxInc

NO

< 80K

> 80K

NO

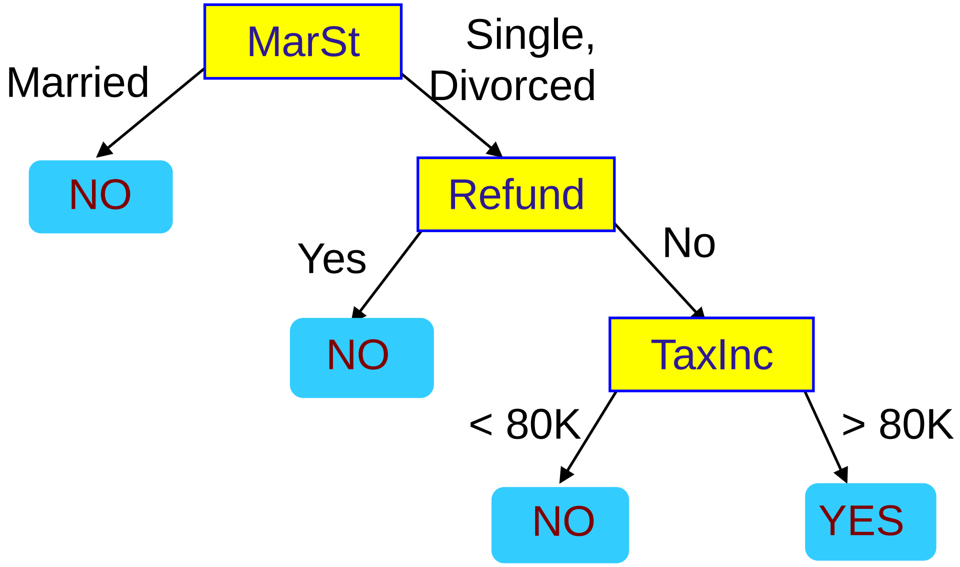
YES

Class labels

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!

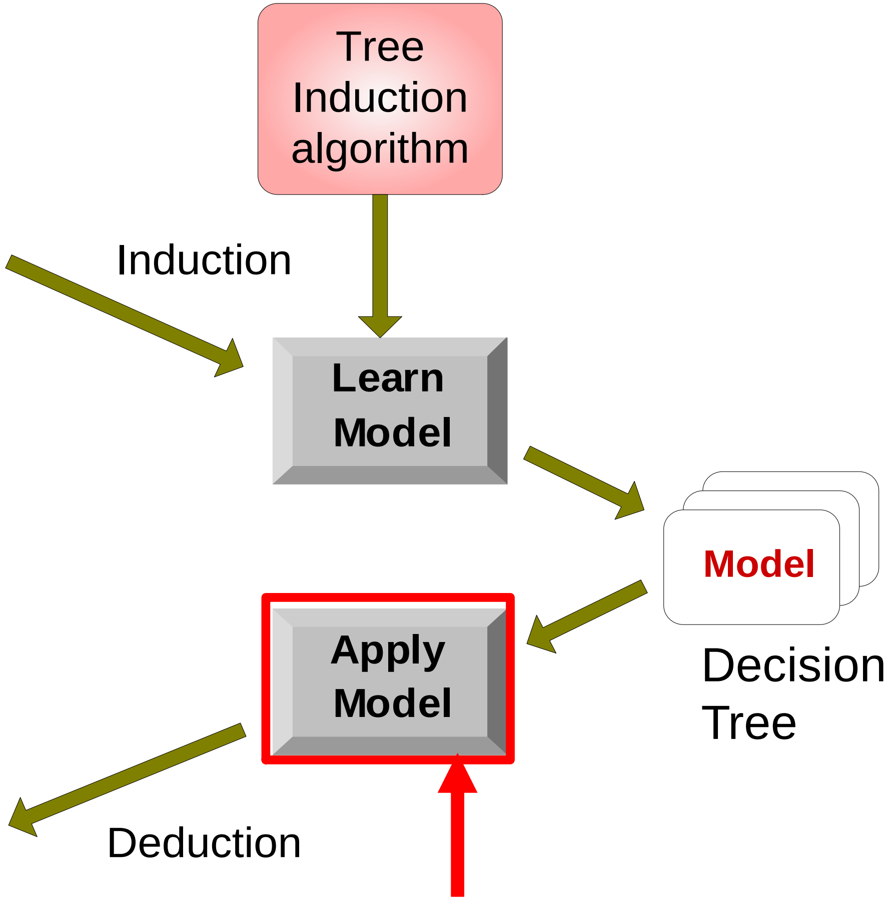
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Tree Induction algorithm

Induction

Learn Model

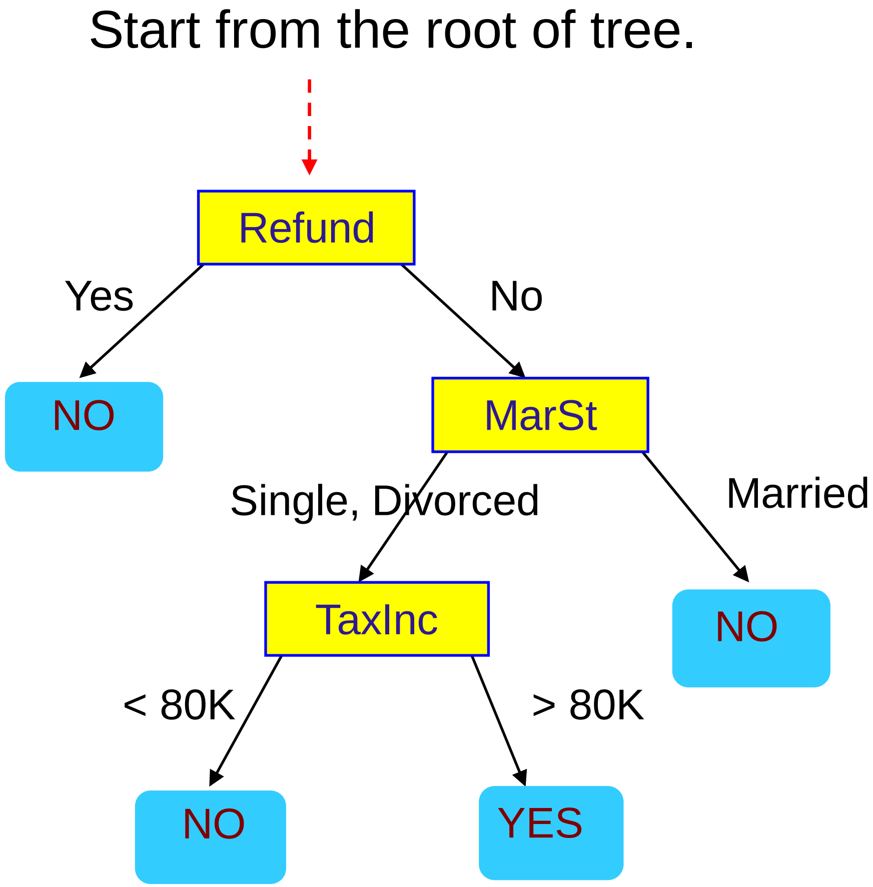
Model

Decision Tree

Apply Model

Deduction

Apply Model to Test Data



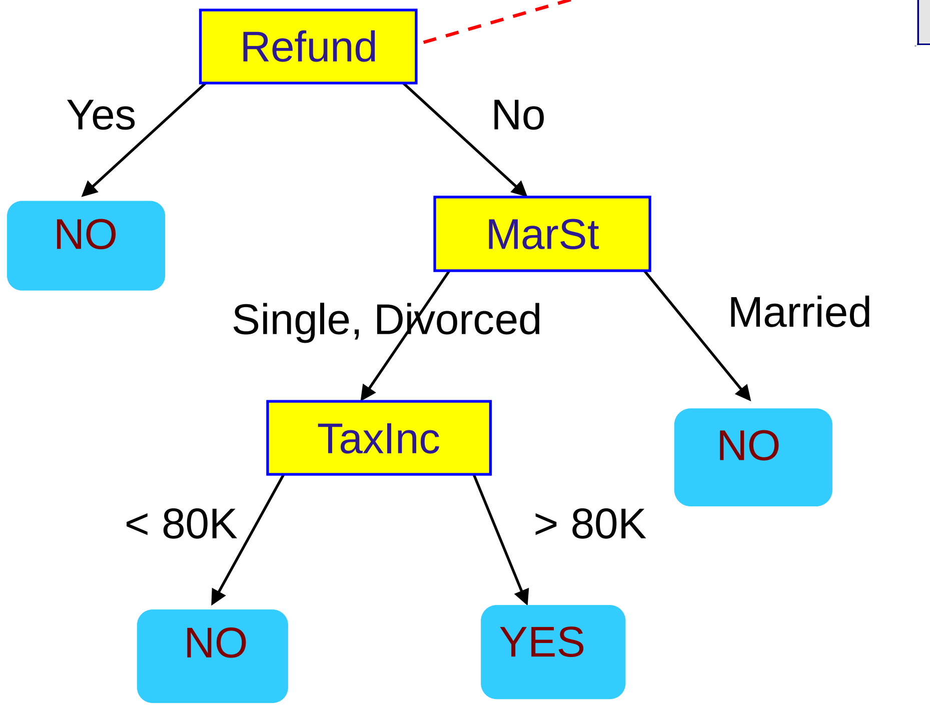
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**

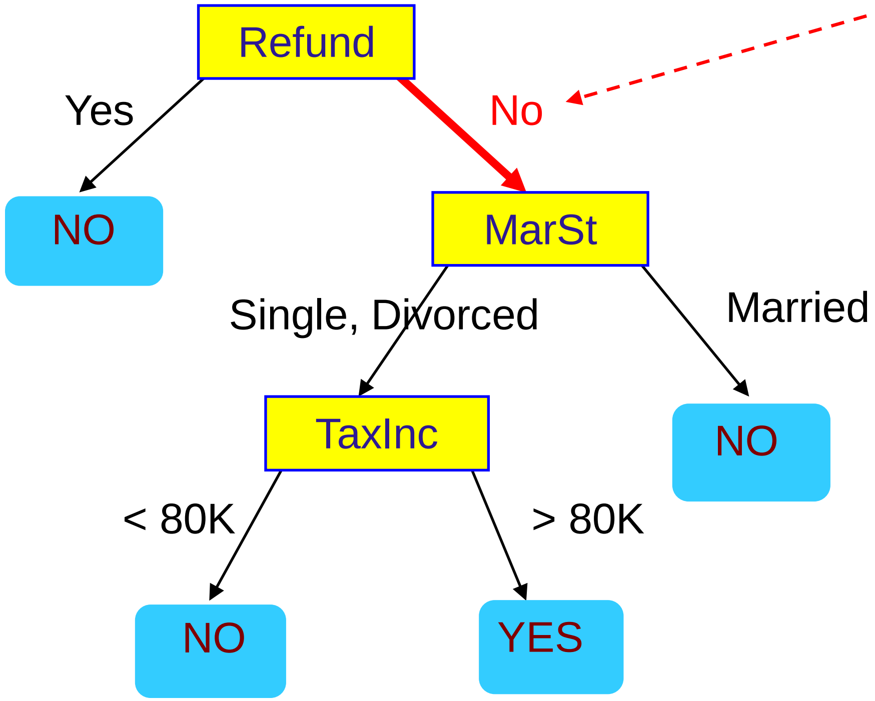
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data



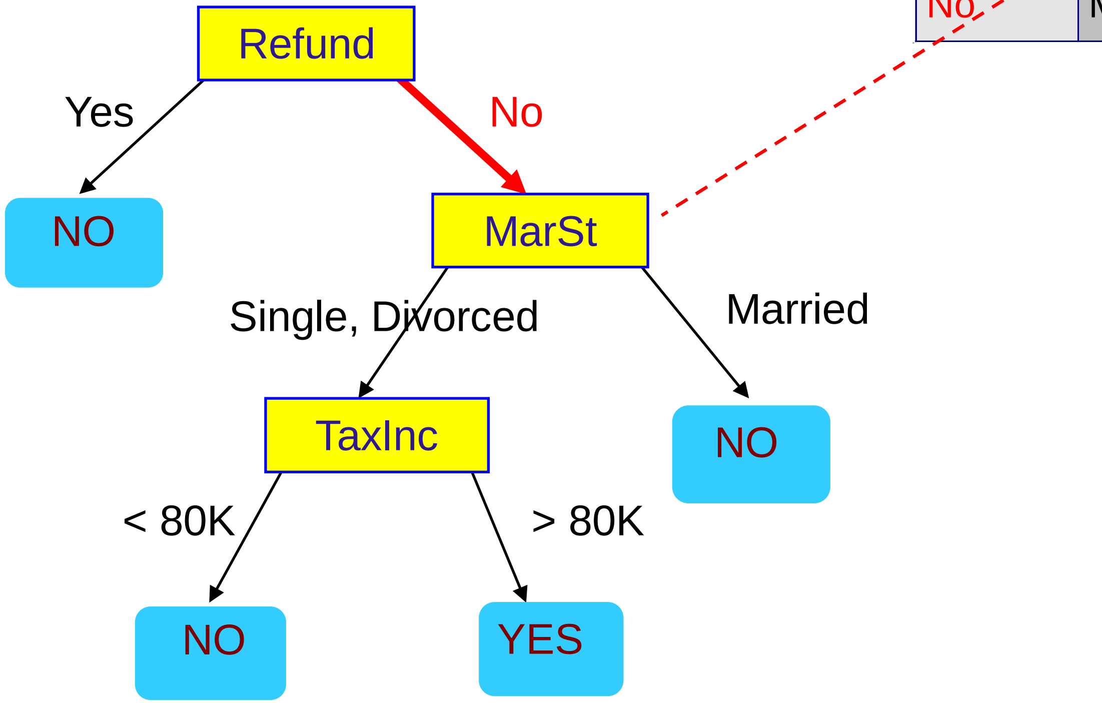
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

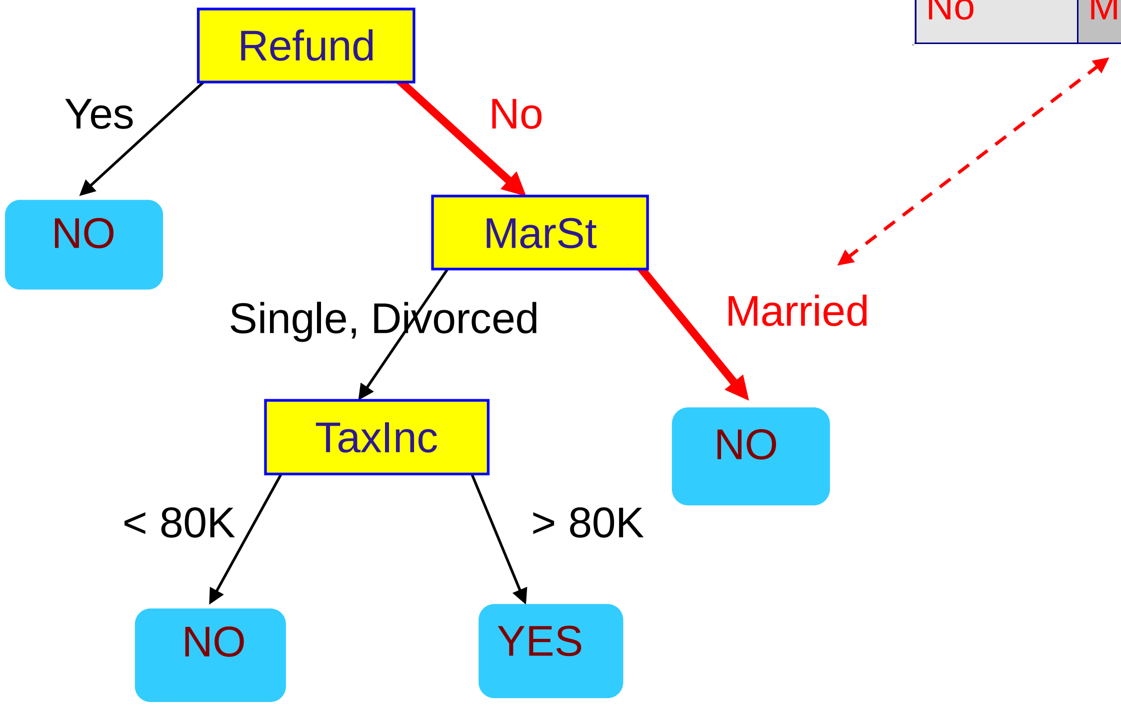
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

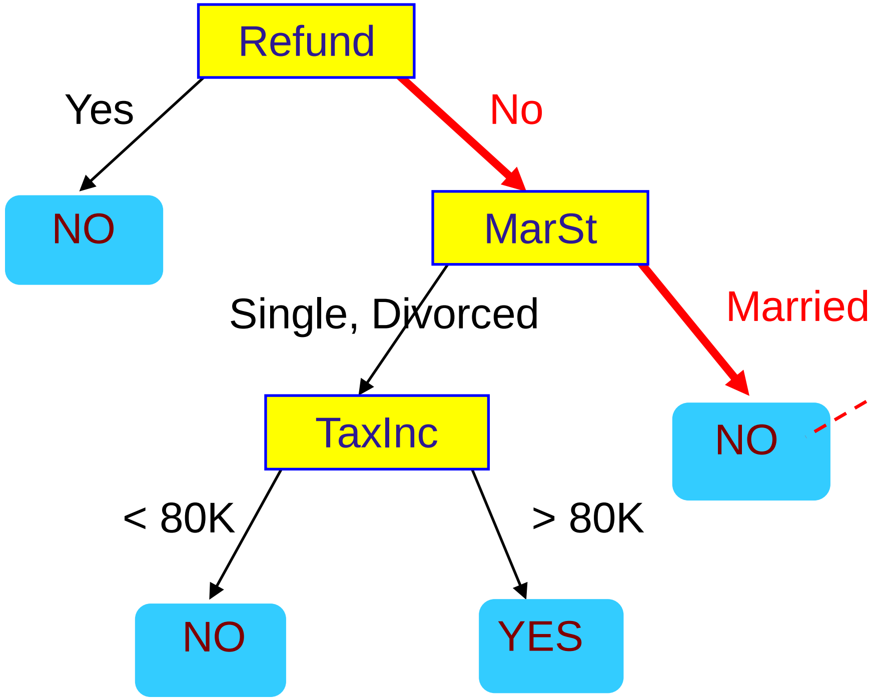
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

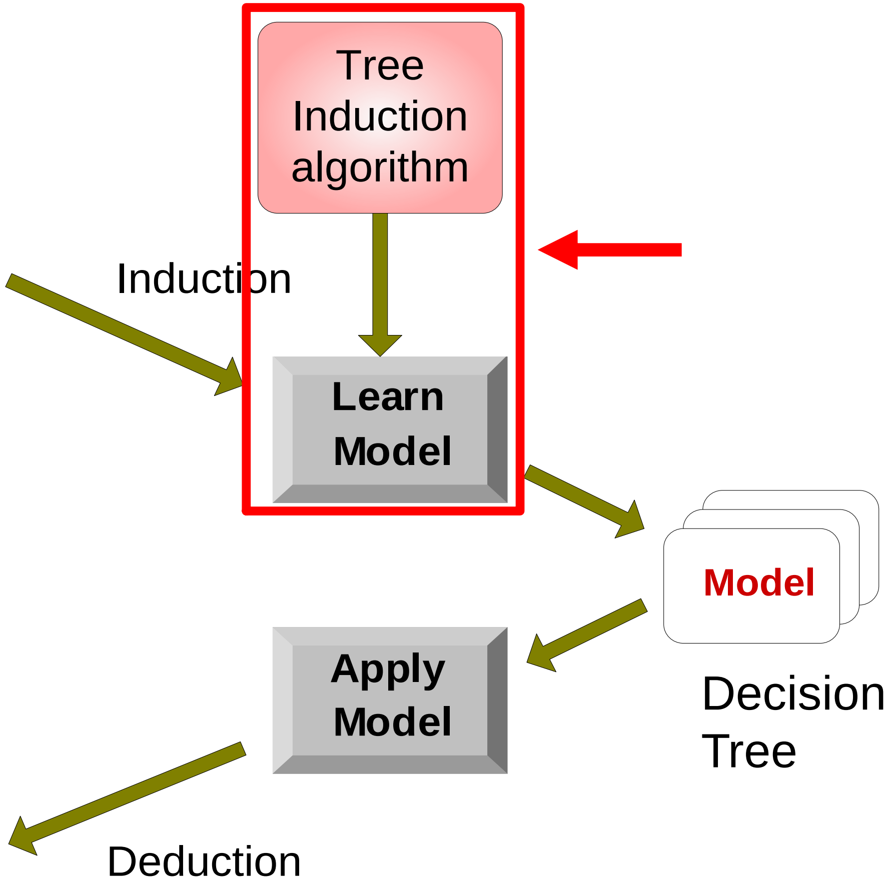
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Tree Induction

- Goal: Find the tree that has low classification error in the training data (training error)

- Finding the best decision tree (lowest training error) is NP-hard

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

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

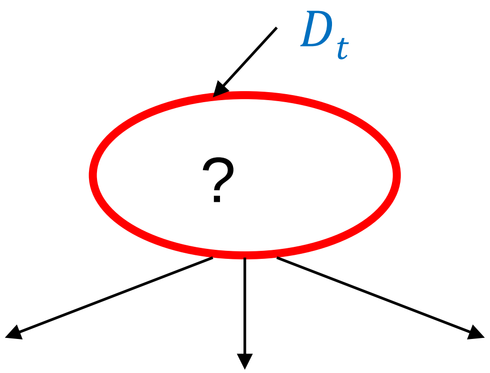
General Structure of Hunt's Algorithm

- 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 contains records with the same attribute values, then t is a leaf node labeled with the majority class y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets.

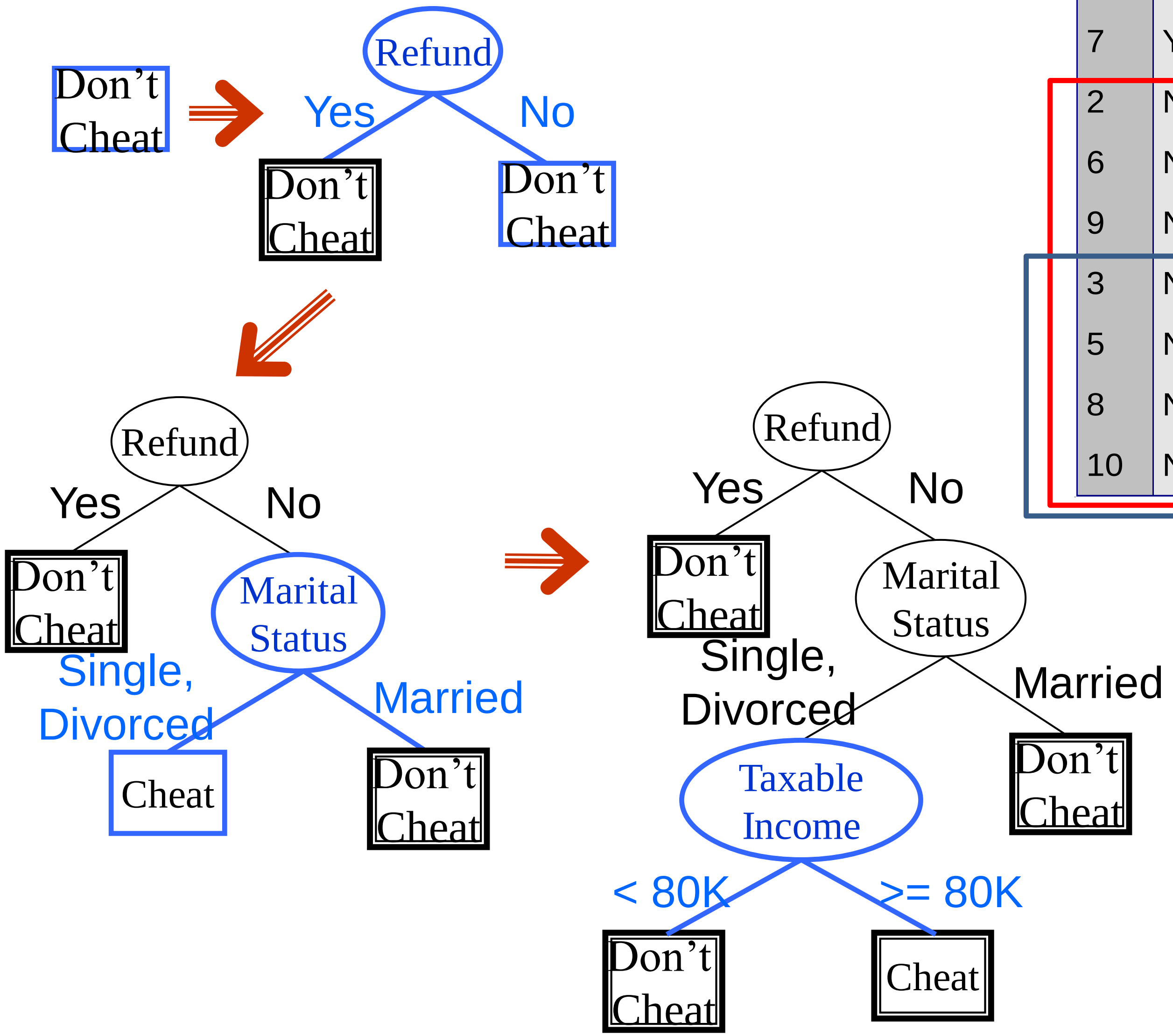
- Recursively apply the procedure to each subset.

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
4	Yes	Married	120K	No
7	Yes	Divorced	220K	No
2	No	Married	100K	No
6	No	Married	60K	No
9	No	Married	75K	No
3	No	Single	70K	No
5	No	Divorced	95K	Yes
8	No	Single	85K	Yes
10	No	Single	90K	Yes



Constructing decision-trees (pseudocode)

```
GenDecTree(Sample S, Features F)
1.  If stopping_condition(S,F) = true then
    a.  leaf = createNode()
    b.  leaf.label= Classify(S)
    c.  return leaf
2.  root = createNode()
3.  root.test_condition = findBestSplit(S,F)
4.  V = {v| v a possible outcome of root.test_condition}
5.  for each value vϵV:
    a.  $S_v$: = {s | root.test_condition(s) = v and s ϵ S};
    b.  child = GenDecTree($S_v$ ,F) ;
    c.  Add child as a descent of root and label the edge (root→child) as v
6.  return root
```

Tree Induction

- Issues
  - How to **Classify** a leaf node
    - Assign the majority class
    - If leaf is empty, assign the default class – the class that has the highest popularity (overall or in the parent node).
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

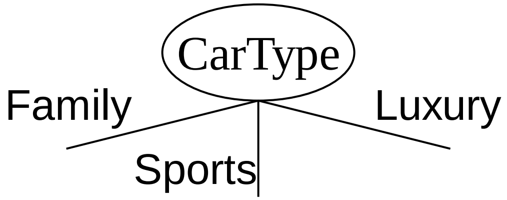
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

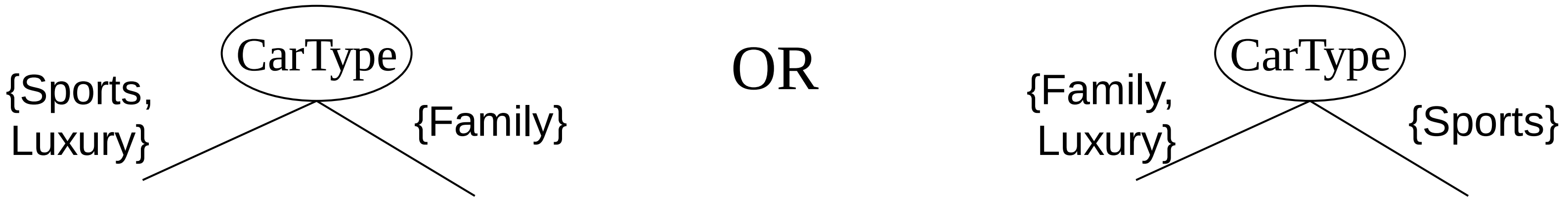
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

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

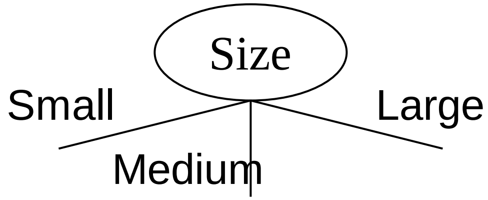


- Binary split: Divides values into two subsets.
  Need to find optimal partitioning.

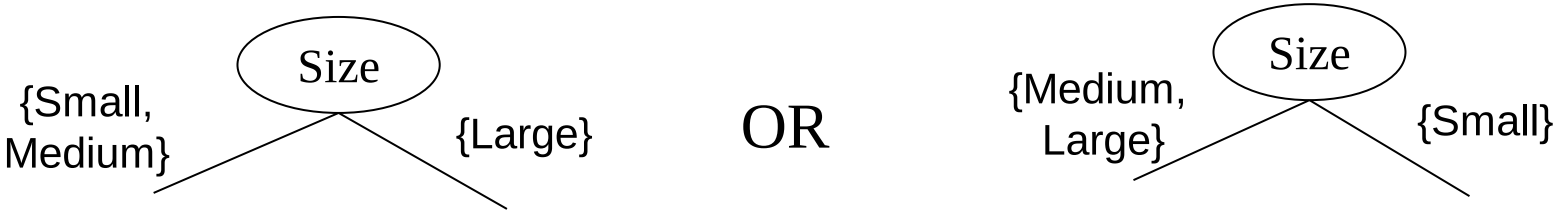


Splitting Based on Ordinal Attributes

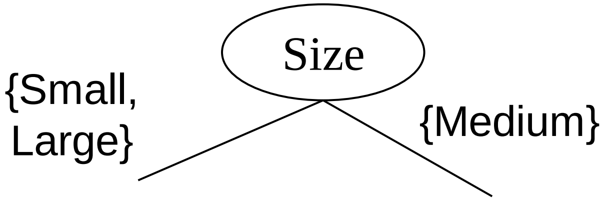
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets – respects the order. Need to find optimal partitioning.



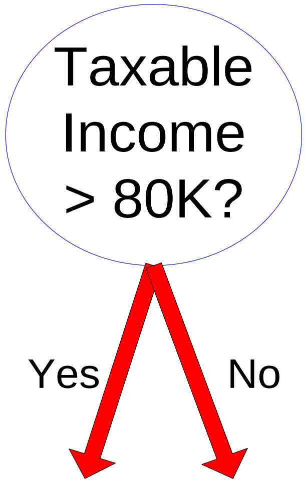
- What about this split?



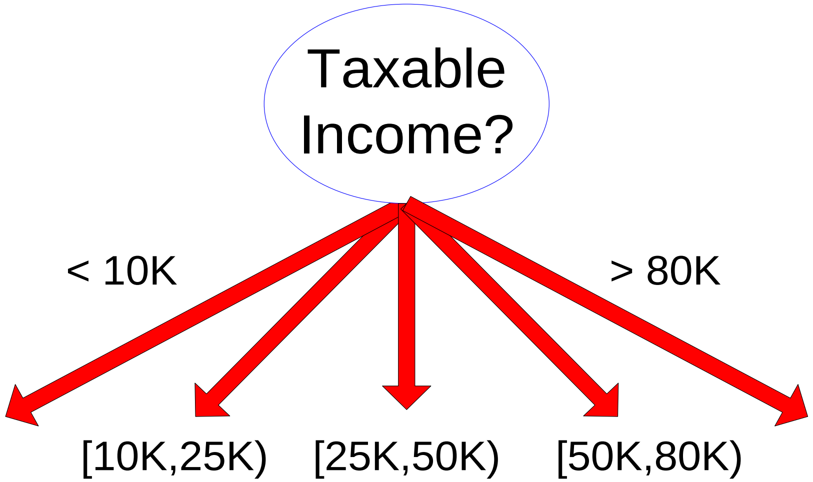
Splitting Based on Continuous Attributes

- 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 computationally intensive

Splitting Based on Continuous Attributes



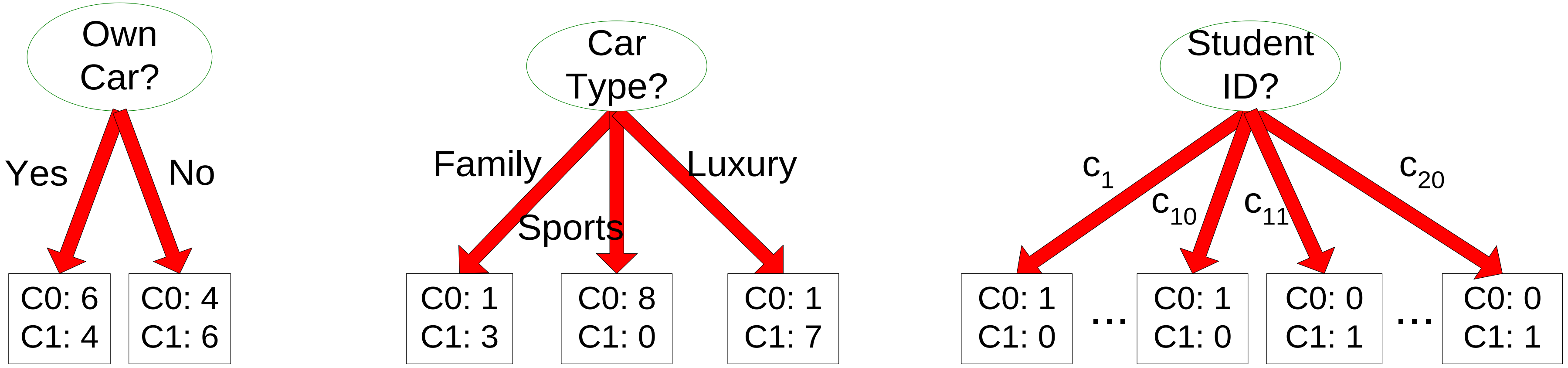
(i) Binary split



(ii) Multi-way split

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:
  - Creation of nodes with homogeneous class distribution is preferred
- Need a measure of node impurity:

C0: 5
C1: 5

Non-homogeneous,

High degree of impurity

C0: 9
C1: 1

Homogeneous,

Low degree of impurity

- Ideas?

Measuring Node Impurity

- We are at a node D_t and the samples belong to classes $\{1, \dots, c\}$
  - $p(i|t)$: fraction of records associated with node D_t belonging to class $\boldsymbol{i}$
- Impurity measures:

$$Entropy(D_t) = -\sum_{i=1}^{c} p(i|t) \log p(i|t)$$

  - Used in ID3 and C4.5

$$Gini(D_t) = 1 - \sum_{i=1}^{c} p(i|t)^2$$

$$Classification\ Error(D_t) = 1 - \max p(i|t)$$

  - Used in CART, SLIQ, SPRINT.

Gain

- Gain of an attribute split into children $\{v_1, \ldots, v_k\}$: compare the impurity of the parent node with the average impurity of the child nodes

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

- Maximizing the gain

⇔ Minimizing the weighted average impurity of children nodes

⇔ Maximizing average purity

- If **I() = Entropy(),** then Δ_{info} is called **information gain**

Example

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Gini = $1 – P(C1)^2 – P(C2)^2 = 1 – 0 – 1 = 0$

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Gini = $1 – (1/6)^2 – (5/6)^2 = 0.278$

Entropy = $– (1/6) \log_2 (1/6) – (5/6) \log_2 (1/6) = 0.65$

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

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

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

Gini = $1 – (2/6)^2 – (4/6)^2 = 0.444$

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

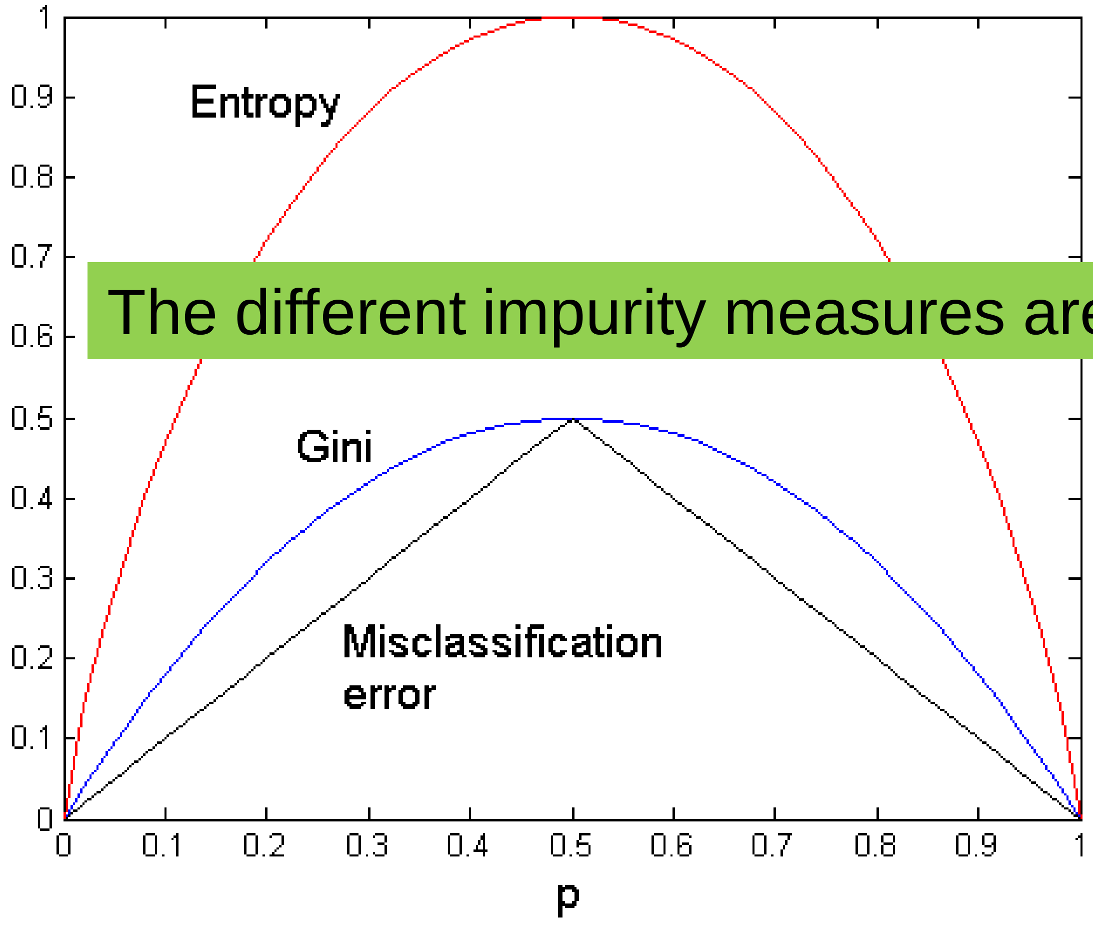
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Impurity measures

- All of the impurity measures take value zero (minimum) for the case of a pure node where a single value has probability 1
- All of the impurity measures take maximum value when the class distribution in a node is uniform.

Comparison among Splitting Criteria

For a 2-class problem:



The different impurity measures are consistent

Categorical Attributes

- For binary values split in two
- For multivalued attributes, for each distinct value, gather counts for each class in the dataset
  - Use the count matrix to make decisions

Multi-way split

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

Two-way split
(find best partition of values)

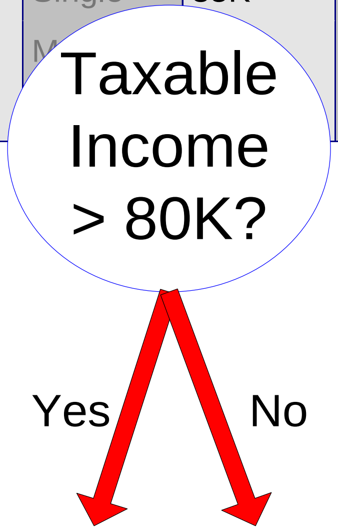
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

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

Continuous Attributes

- Use Binary Decisions based on one value
- 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 $A \geq v$
- Exhaustive method to choose best v
  - For each v, scan the database to gather count matrix and compute the impurity 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	M		No
10	No			Yes



Continuous Attributes

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

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	

Splitting based on impurity

- Impurity measures favor attributes with large number of values

- A test condition with large number of outcomes may not be desirable
  - # of records in each partition is too small to make predictions

Splitting based on INFO

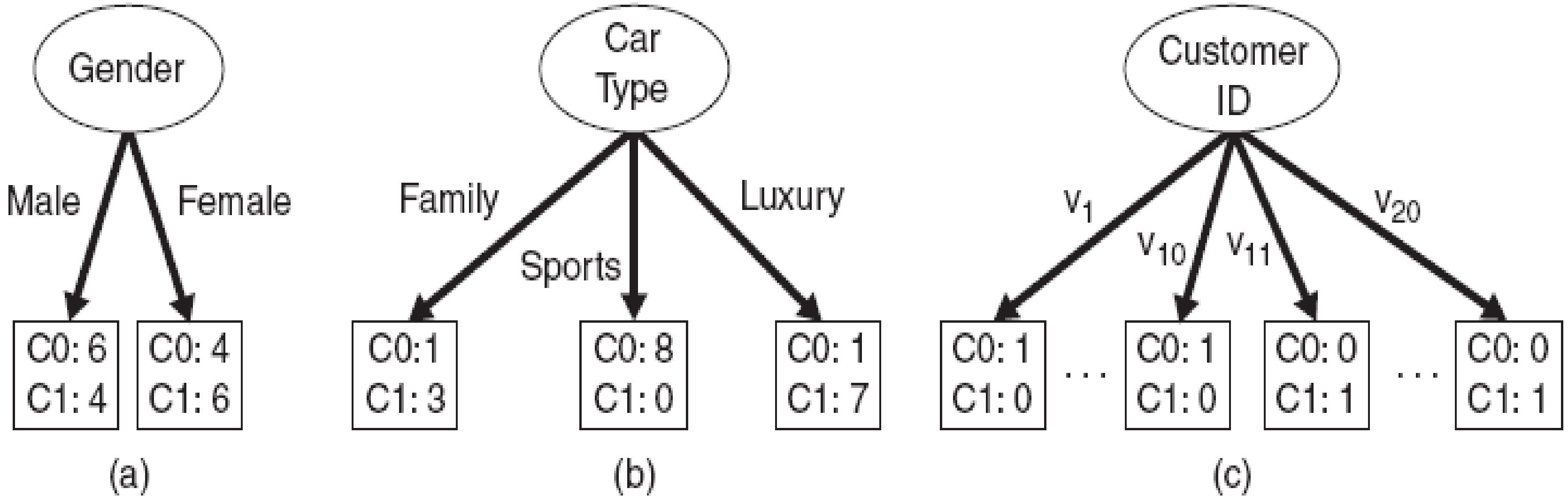


Figure 4.12. Multiway versus binary splits.

Gain Ratio

- Splitting using information gain

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

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 partition (SplitINFO). Higher entropy partition (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of impurity

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

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

OTHER CLASSIFICATION ISSUES

Expressiveness

Overfitting

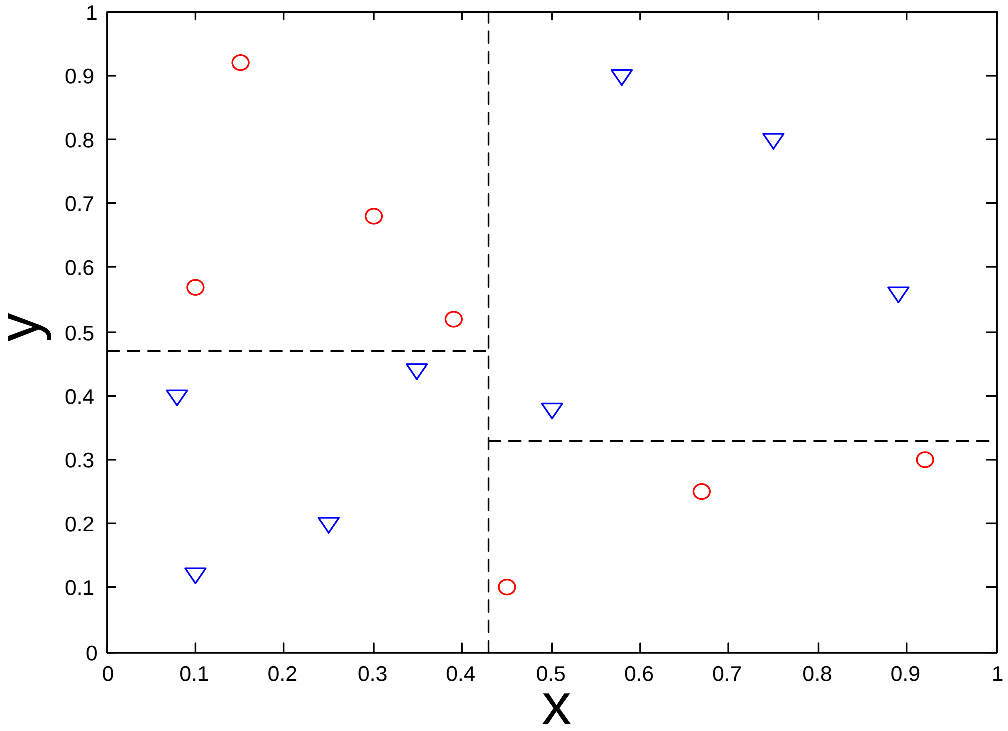
Evaluation

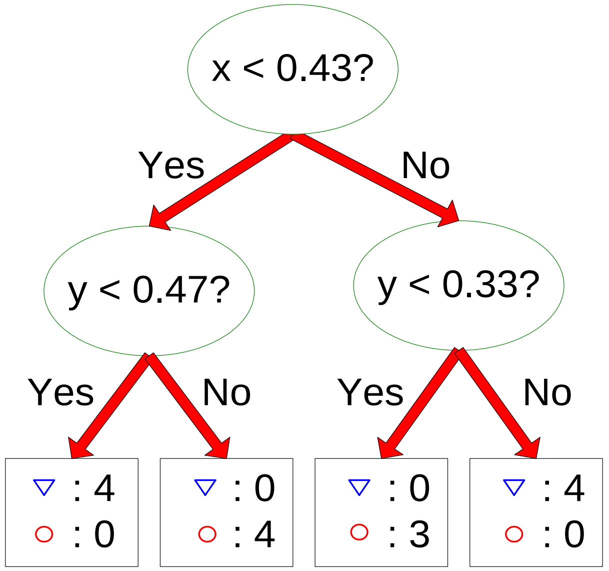
EXPRESSIVENESS

Expressiveness

- A classifier defines a function that discriminates between two (or more) classes.
- The expressiveness of a classifier is the class of functions that it can model, and the kind of data that it can separate
  - When we have discrete (or binary) values, we are interested in the class of boolean functions that can be modeled
  - If the data-points are real vectors we talk about the decision boundary that the classifier can model

Decision Boundary

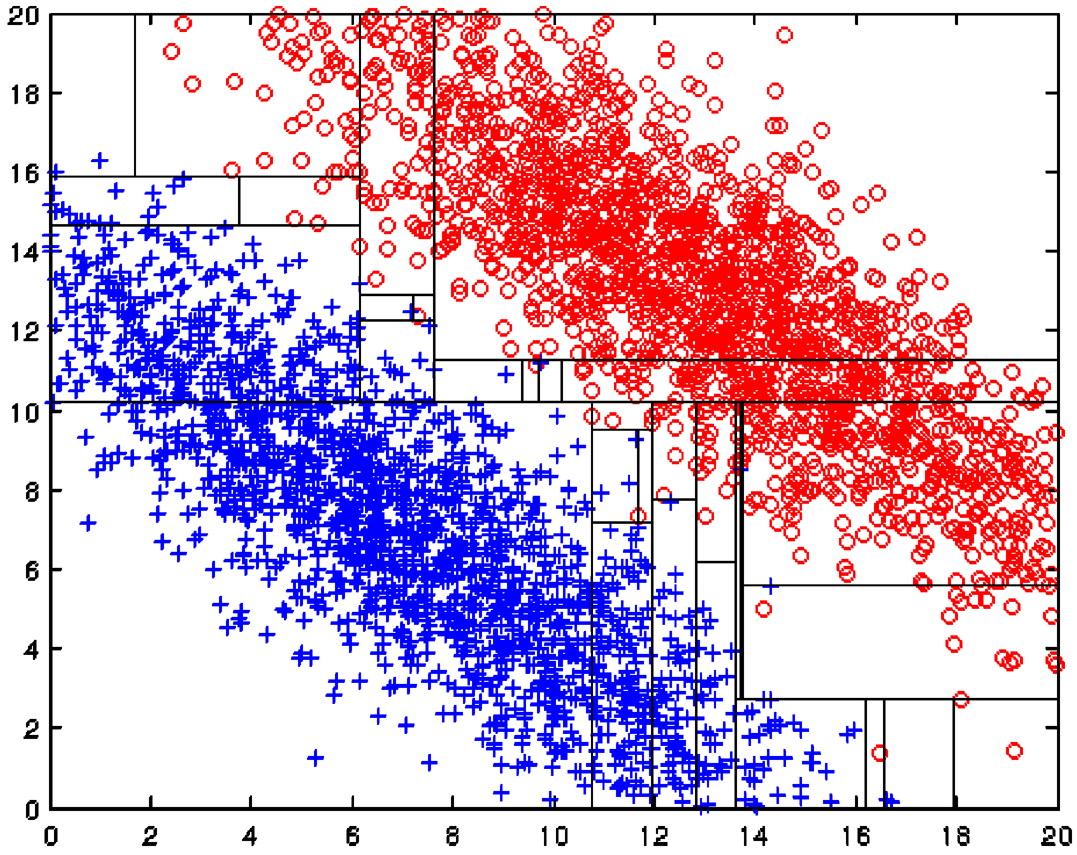




- Border line between two neighboring regions of different classes is known as decision boundary

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

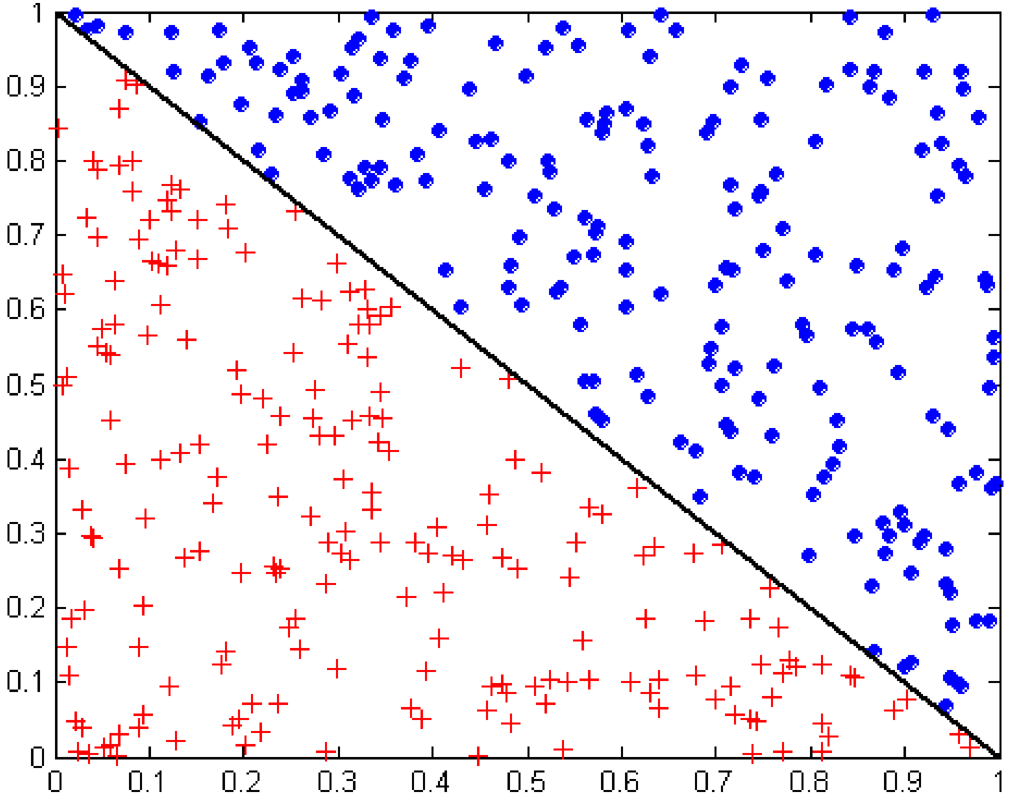
Limitations of single attribute-based decision boundaries

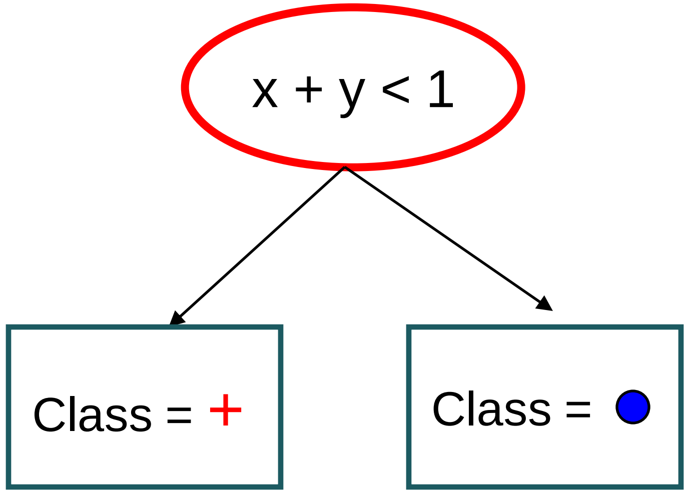


Both positive (+) and negative (o) classes generated from skewed Gaussians with centers at (8,8) and (12,12) respectively.

The resulting boundary is very complex.

Oblique Decision Trees





x + y < 1

Class = +

Class = ●

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

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

- Less expressive for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

OVERFITTING

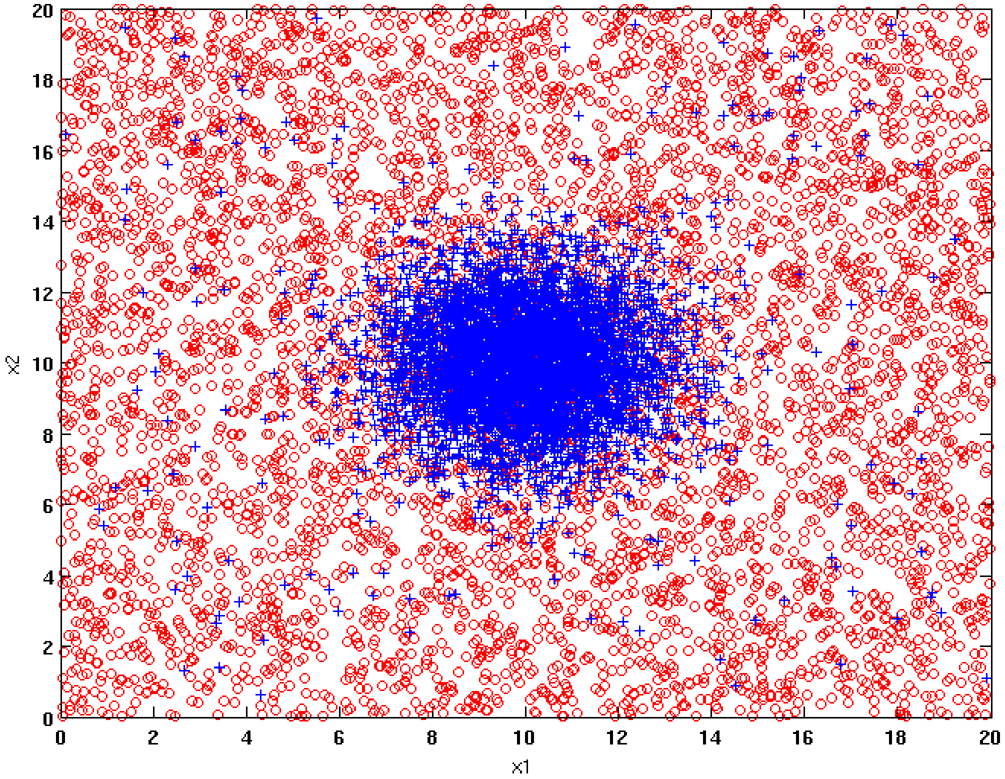
Classification Errors

- Training errors (apparent errors)
  - Errors committed on the training set

- Test errors
  - Errors committed on the test set

- Generalization errors
  - Expected error of a model over random selection of records from same distribution

Example Data Set



Two class problem:

+ : 5400 instances

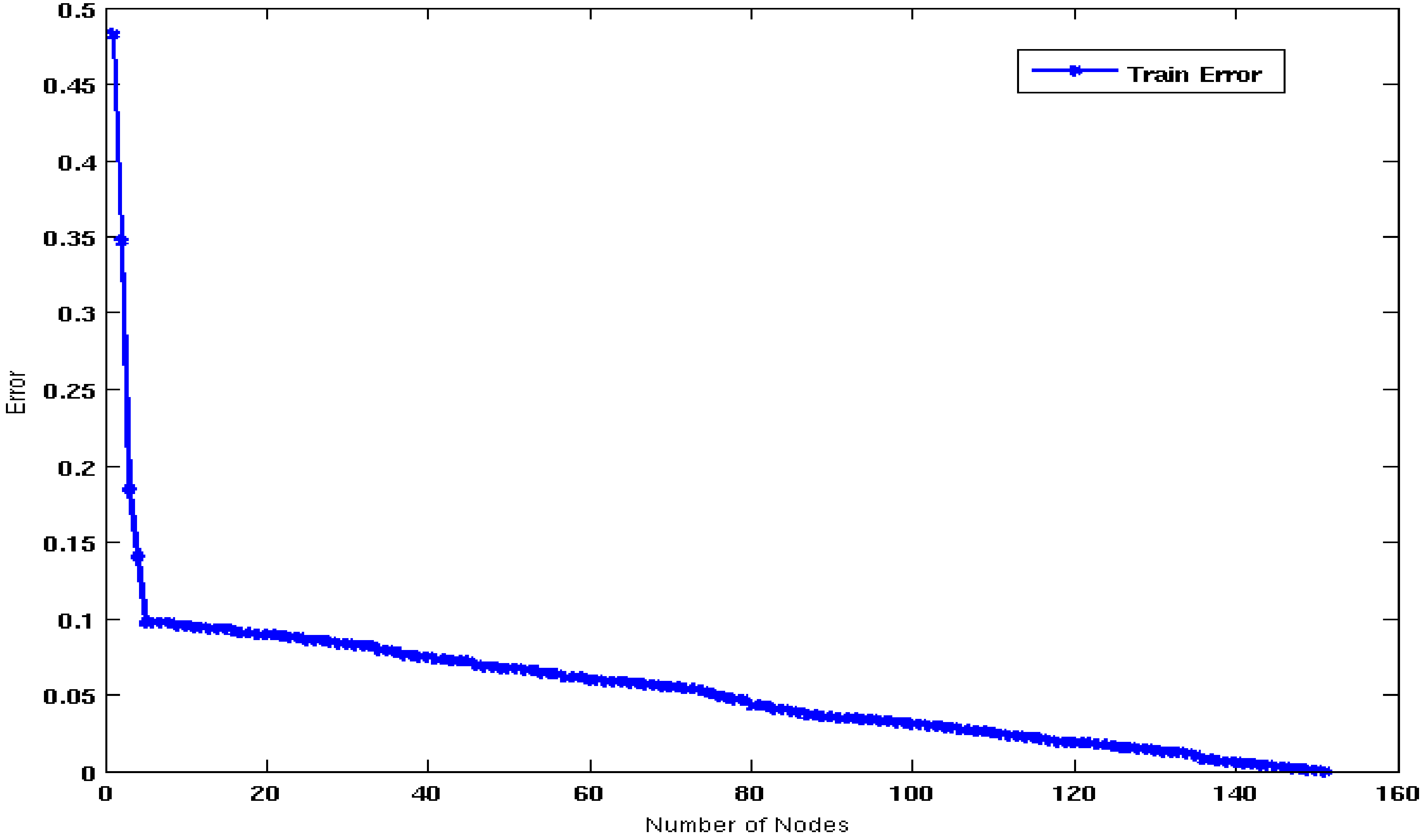
- **5000 instances generated from a Gaussian centered at (10,10)**
- **400 noisy instances added**

o : 5400 instances

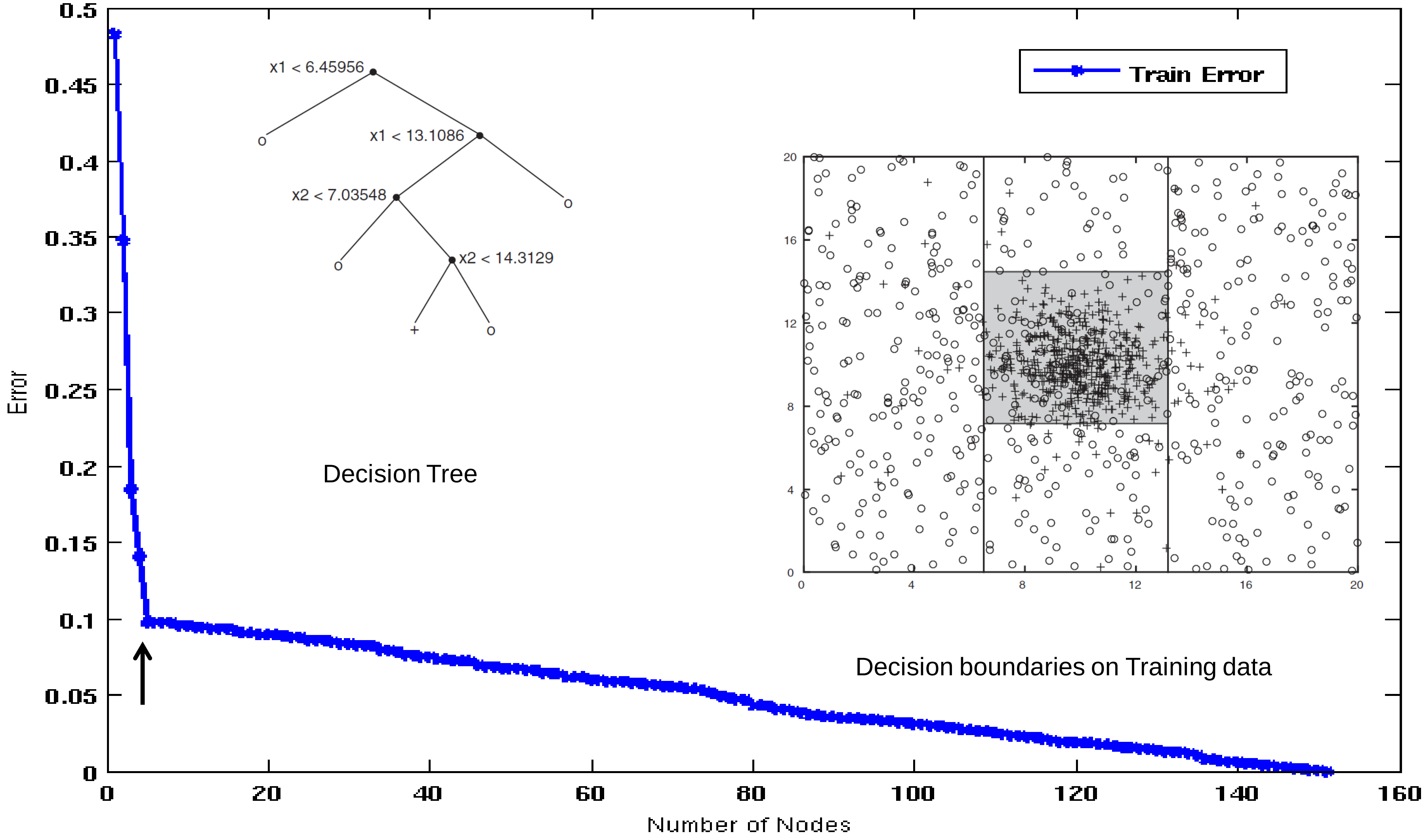
- **Generated from a uniform distribution**

10 % of the data used for training and 90% of the data used for testing

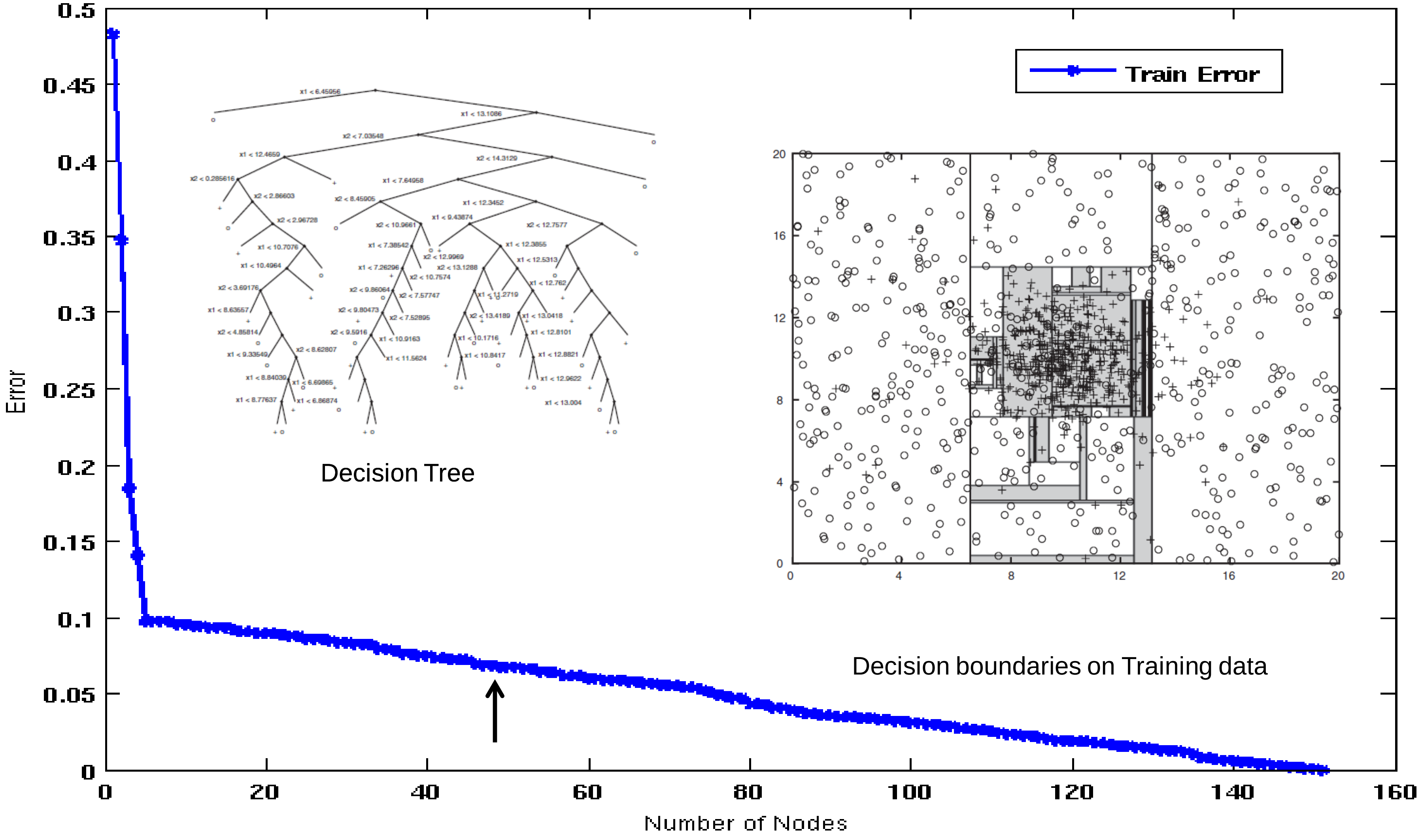
Increasing number of nodes in Decision Trees



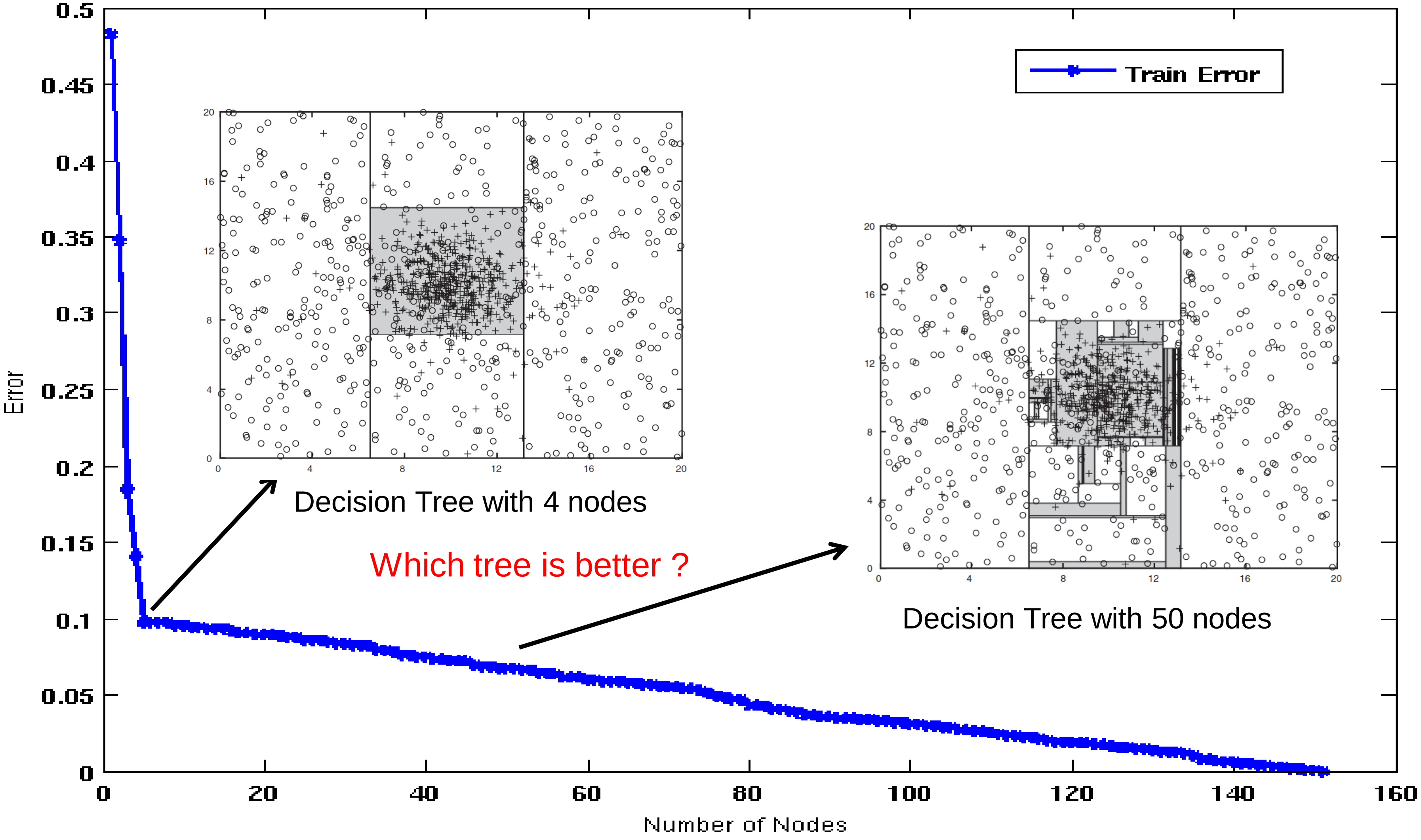
Decision Tree with 4 nodes



Decision Tree with 50 nodes



Which tree is better?

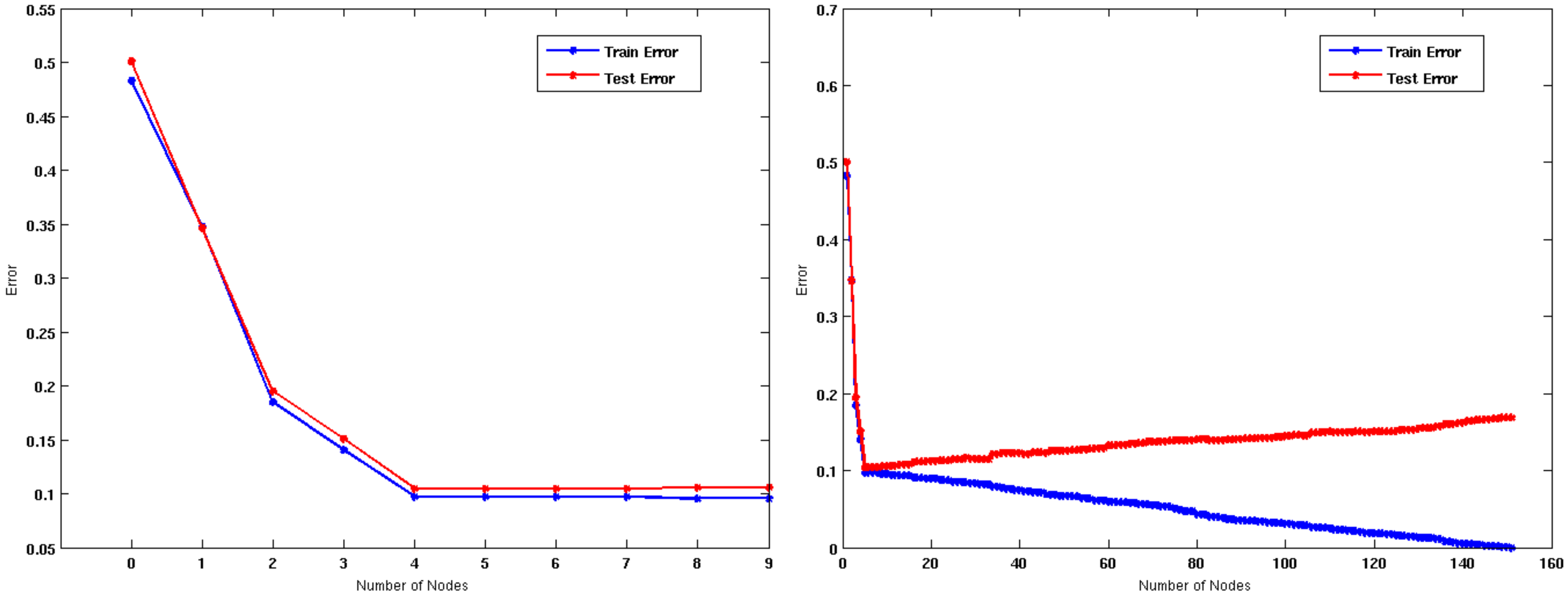


Decision Tree with 4 nodes

Which tree is better ?

Decision Tree with 50 nodes

Model Overfitting



- As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing

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

Overfitting: when model is too complex, training error is small but test error is large

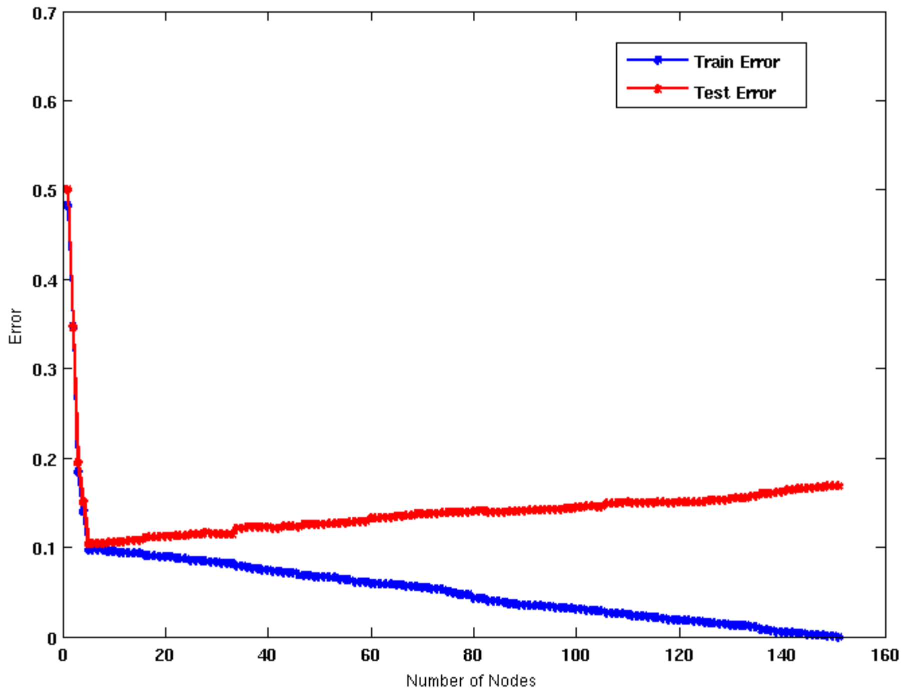
Bias – Variance tradeoff

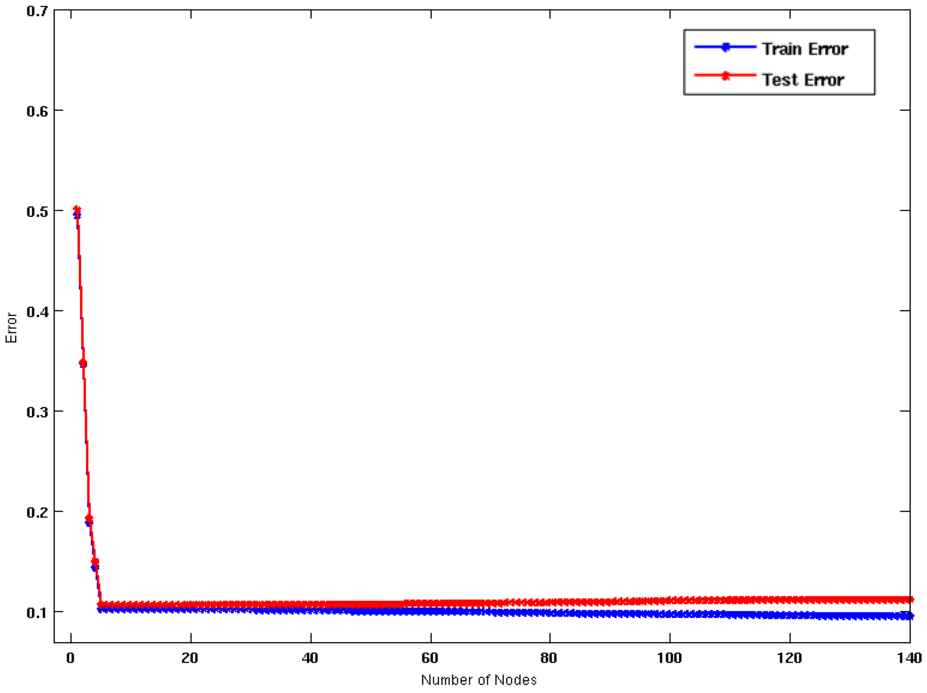
- Bias: Measures how good the model is with respect to the training data
  - High Bias: Underfitting.
  - We have a poor model (e.g., a tree with a single decision node)

- Variance: Measures how sensitive the model error is with respect to changes in the training data
  - High Variance: Overfitting.
  - We have a very specific model (e.g., a tree with a single sample per leaf). Small changes in the data cause errors in the model

- There is a tradeoff between these two: decreasing one will increase the other.

Model Overfitting

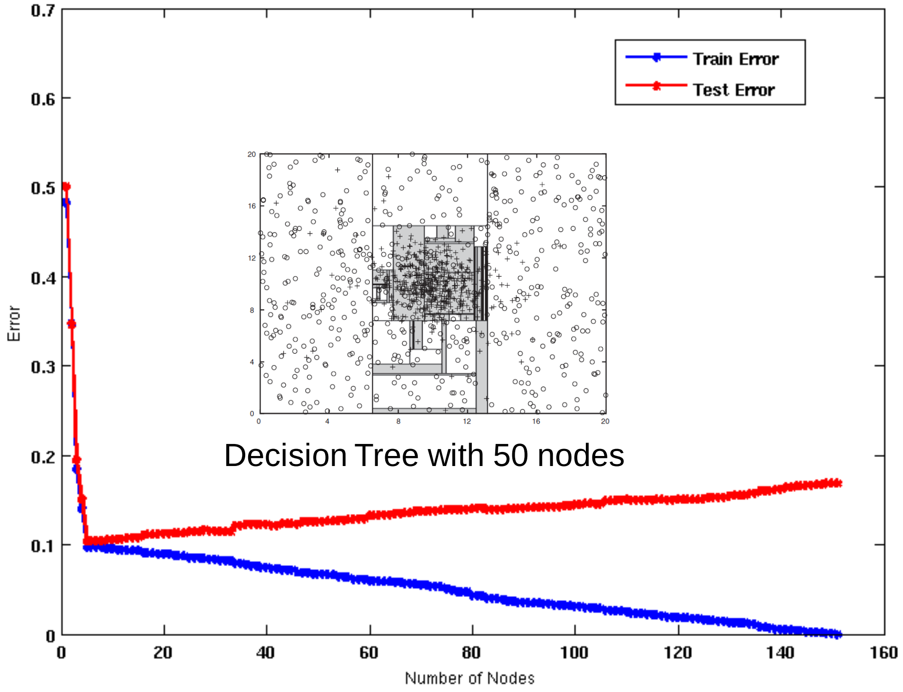




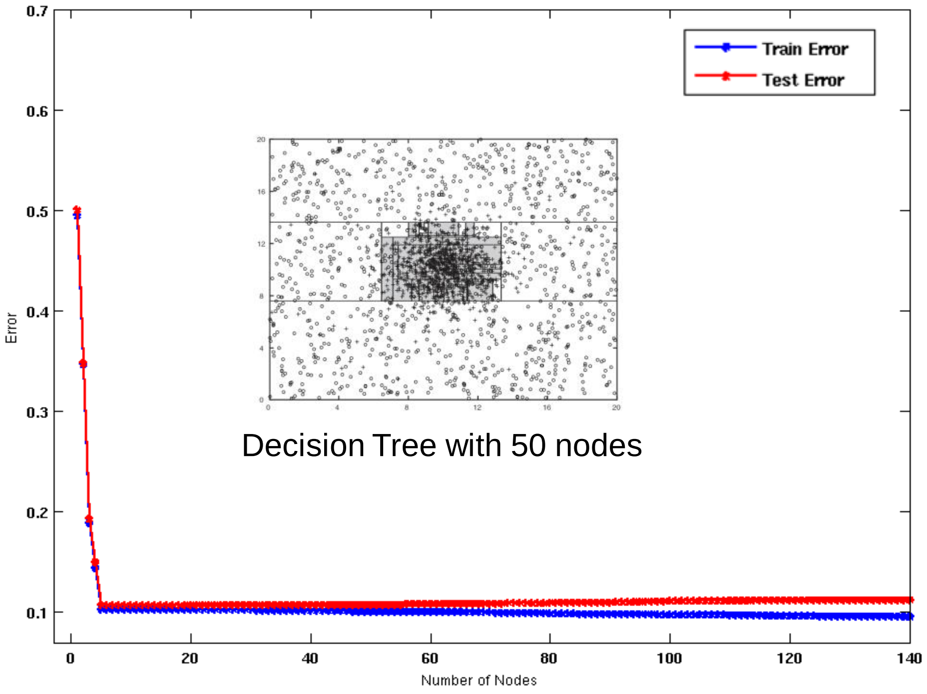
Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

Model Overfitting



Decision Tree with 50 nodes



Decision Tree with 50 nodes

Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

Reasons for Model Overfitting

- Limited Training Size

- High Model Complexity
  - Multiple Comparison Procedure

Effect of Multiple Comparison Procedure

- Consider the task of predicting whether stock market will rise/fall in the next 10 trading days

- Random guessing:

  $P(correct) = 0.5$

- Make 10 random guesses in a row:

$$P(\# correct \geq 8) = \frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0547$$

Day 1	Up
Day 2	Down
Day 3	Down
Day 4	Up
Day 5	Down
Day 6	Down
Day 7	Up
Day 8	Up
Day 9	Up
Day 10	Down

Effect of Multiple Comparison Procedure

- Approach:
  - Get 50 analysts
  - Each analyst makes 10 random guesses
  - Choose the analyst that makes the most number of correct predictions

- Probability that at least one analyst makes at least 8 correct predictions

$$P(\#correct \geq 8) = 1 - (1 - 0.0547)^{50} = 0.9399$$

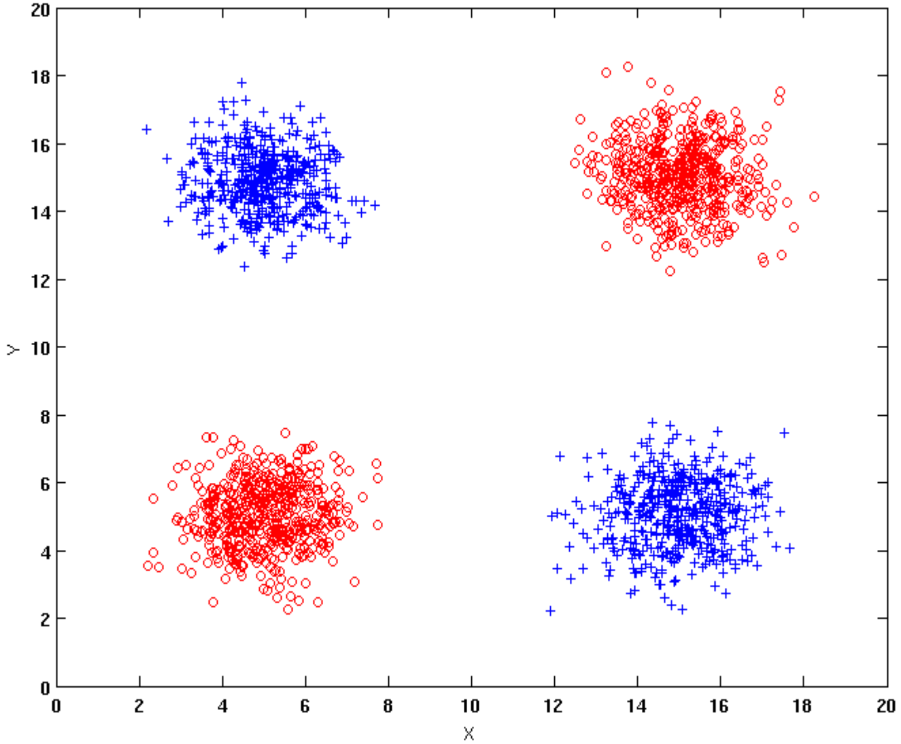
Effect of Multiple Comparison Procedure

- Many algorithms employ the following greedy strategy:
  - Initial model: M
  - Alternative model: $M' = M$,
    where γ is a component to be added to the model (e.g., a test condition of a decision tree)
  - Keep M' if improvement, $\Delta(M, M') > \alpha$

- Often times, γ is chosen from a set of alternative components, $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$

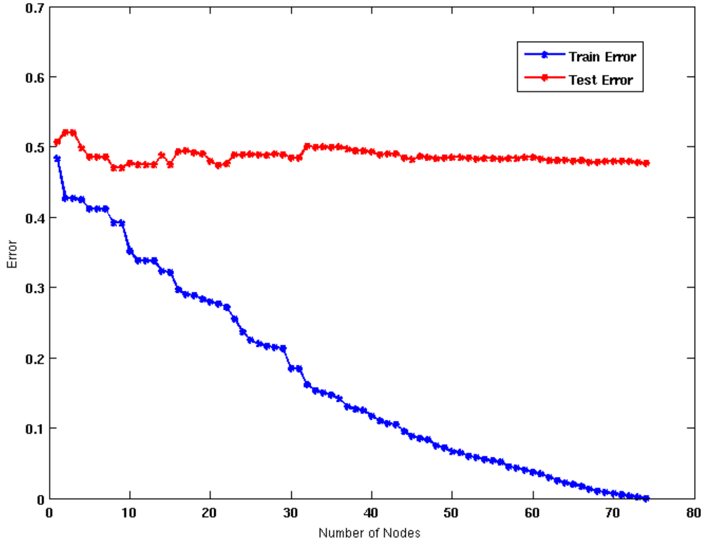
- If many alternatives are available, one may inadvertently add irrelevant components to the model, resulting in model overfitting

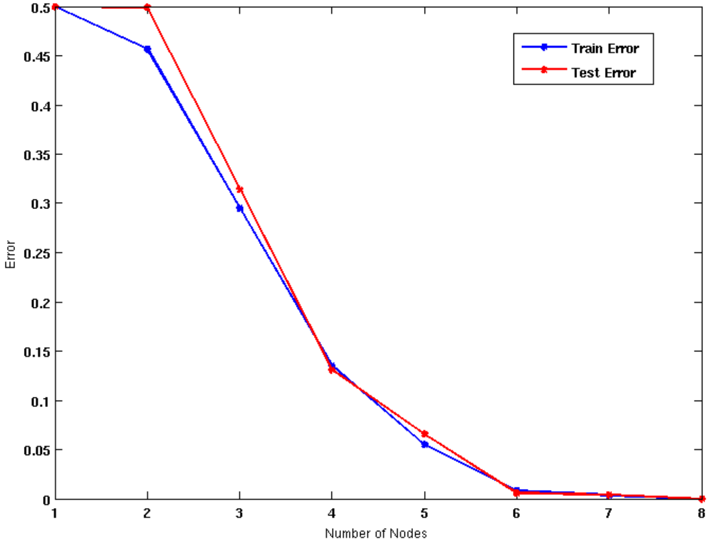
Effect of Multiple Comparison - Example



Use additional 100 noisy variables generated from a uniform distribution along with X and Y as attributes.

Use 30% of the data for training and 70% of the data for testing





Using only X and Y as attributes

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- Training error no longer provides a good estimate of test error, that is, how well the tree will perform on previously unseen records
- We say that the model does not generalize well

- Generalization: The ability of the model to predict data points that it has not already seen.

- Need ways for estimating generalization errors

Model Selection

- Performed during model building
- Purpose is to ensure that model is not overly complex (to avoid overfitting)
- Need to estimate generalization error
  - Using Validation Set
  - Incorporating Model Complexity

Model Selection: Using Validation Set

- Divide training data into two parts:
  - Training set:
    - Use for model building
  - Validation set:
    - Use for estimating generalization error
    - Note: validation set is not the same as test set since it affects the creation of the model (e.g. in tuning a parameter)

- Drawback:
  - Less data available for training

Occam’s Razor

- Occam’s razor: All other things being equal, the simplest explanation/solution is the best.
  - A good principle for life as well

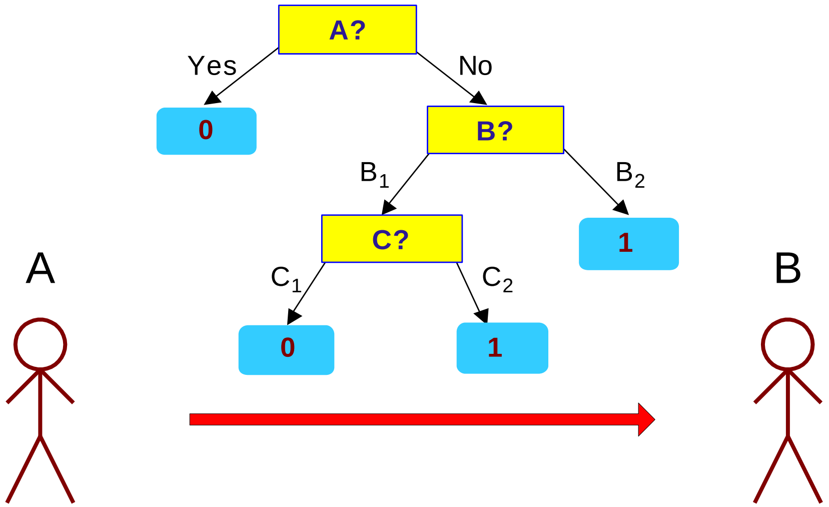
- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

- 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

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1



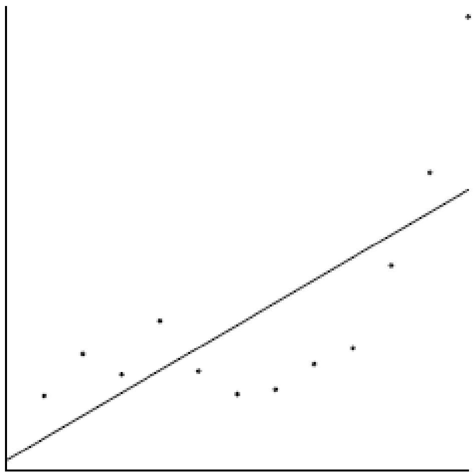
X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- Cost(Model,Data) = Cost(Model) + Cost(Data|Model)
  - Search for the least costly model.

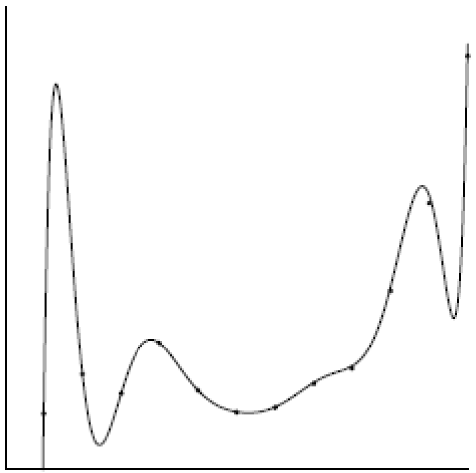
- Cost(Model) encodes the decision tree
  - node encoding (number of children) plus splitting condition encoding.
- Cost(Data|Model) encodes the misclassification errors.

Example

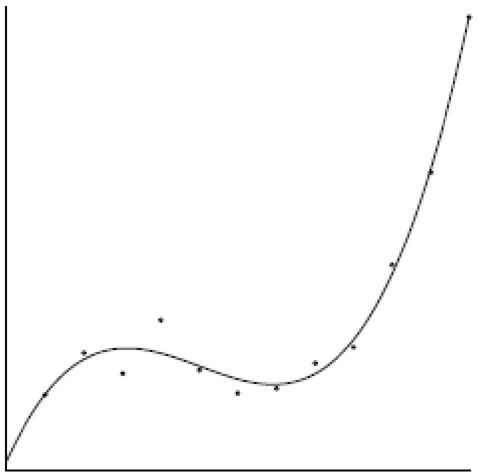
- Regression: find a polynomial for describing a set of values
  - Model complexity (model cost): polynomial coefficients
  - Goodness of fit (data cost): difference between real value and the polynomial value



Minimum model cost
High data cost



High model cost
Minimum data cost



Low model cost
Low data cost

MDL avoids overfitting automatically!

Source: Grunwald et al. (2005) *Tutorial on MDL*.

Model Selection: Incorporating Model Complexity

- Occam's razor: All other things being equal, the simplest explanation/solution is the best.
  - A good principle for life as well

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

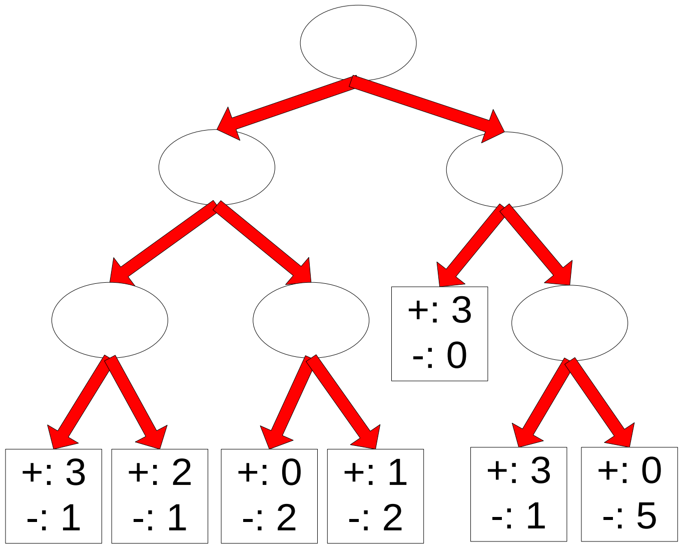
- For complex models, there is a greater chance that it was fitted accidentally

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

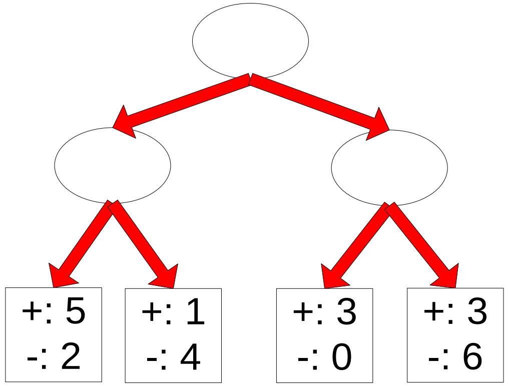
Gen. Error(Model) = Train. Error(Model, Train. Data)
+ α × Complexity(Model)

Estimating the Complexity of Decision Trees

- Resubstitution Estimate:
  - Using training error as an optimistic estimate of generalization error
  - Referred to as optimistic error estimate



Decision Tree, T_L



Decision Tree, T_R

$$e(TL) = 4/24$$

$$e(TR) = 6/24$$

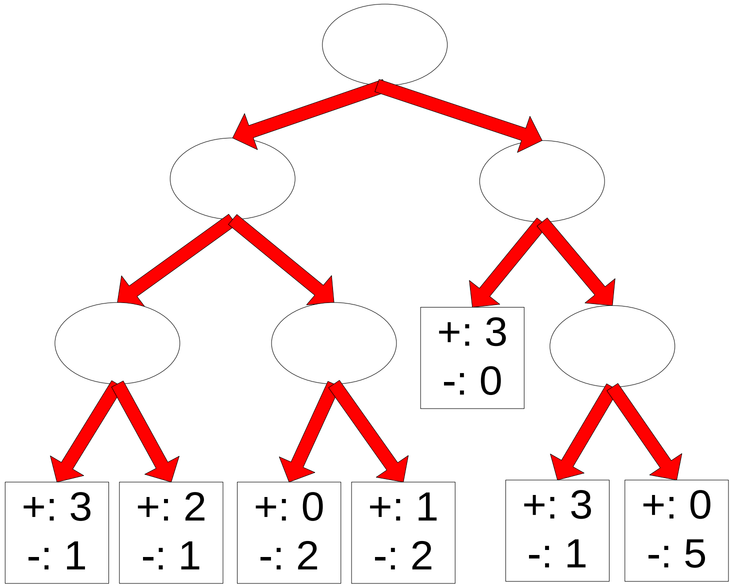
Estimating the Complexity of Decision Trees

- Pessimistic Error Estimate of decision tree T with k leaf nodes:

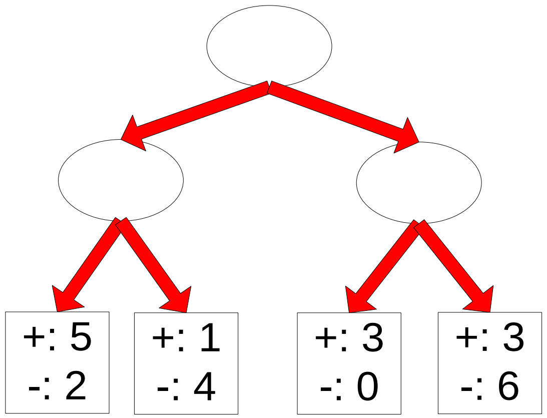
$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

  - $err(T)$: error rate on all training records
  - Ω: trade-off hyper-parameter (similar to α)
    - Relative cost of adding a leaf node
  - k: number of leaf nodes
  - N_{train}: total number of training records

Estimating the Complexity of Decision Trees: Example



Decision Tree, T_L



Decision Tree, T_R

$$e(TL) = 4/24$$

$$e(TR) = 6/24$$

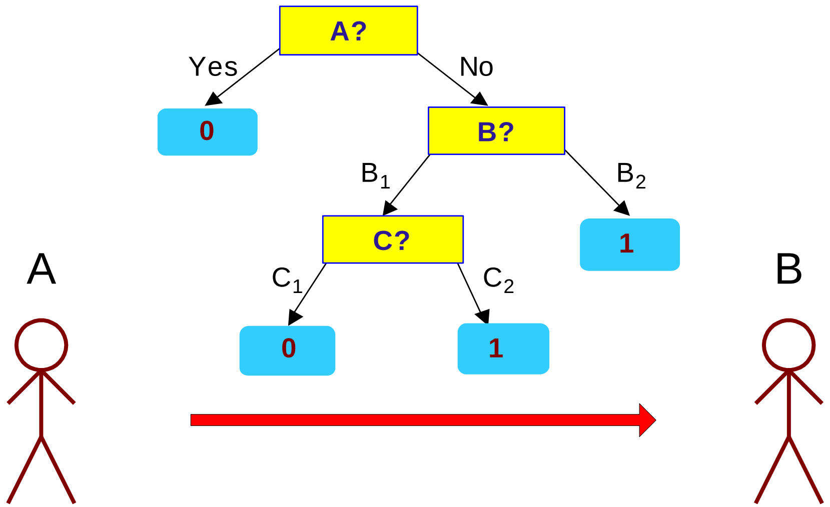
$$\Omega = 1$$

$$e_{gen}(TL) = 4/24 + 1 * 7/24 = 11/24 = 0.458$$

$$e_{gen}(TR) = 6/24 + 1 * 4/24 = 10/24 = 0.417$$

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1



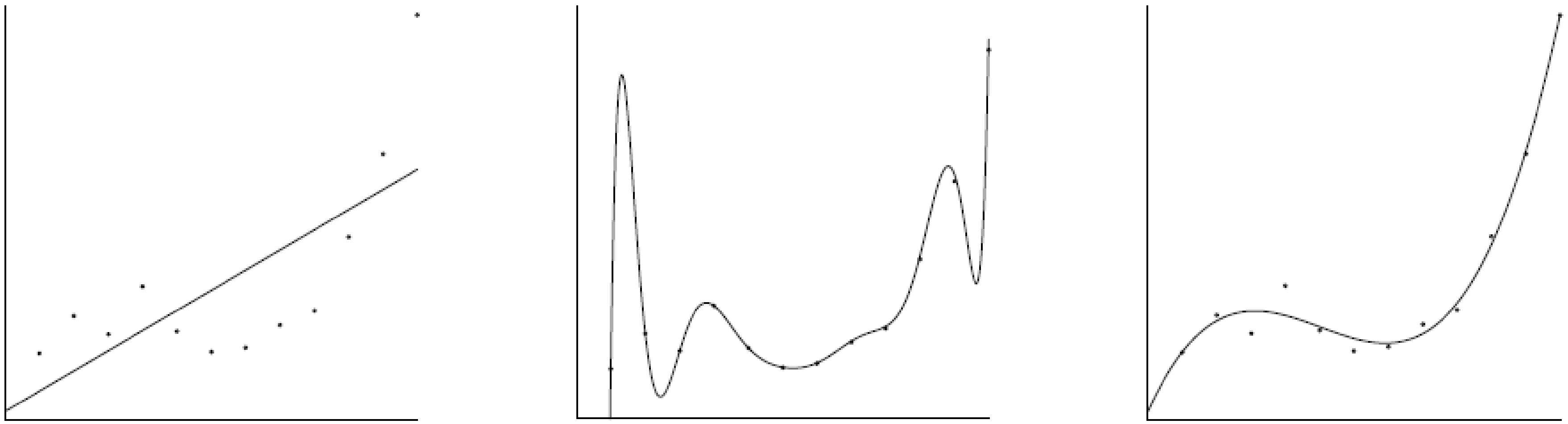
X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- Cost(Model,Data) = Cost(Model) + Cost(Data|Model)
  - Cost is the number of bits needed for encoding.
  - Search for the least costly model.

- Cost(Model) encodes the decision tree
  - node encoding (number of children) plus splitting condition encoding.
- Cost(Data|Model) encodes the misclassification errors.

Example

- Regression: find a polynomial for describing a set of values
  - Model complexity (model cost): polynomial coefficients
  - Goodness of fit (data cost): difference between real value and the polynomial value



Minimum model cost
High data cost

High model cost
Minimum data cost

Low model cost
Low data cost

MDL avoids overfitting automatically!

Source: Grunwald et al. (2005) *Tutorial on MDL*.

Model selection for Decision Trees

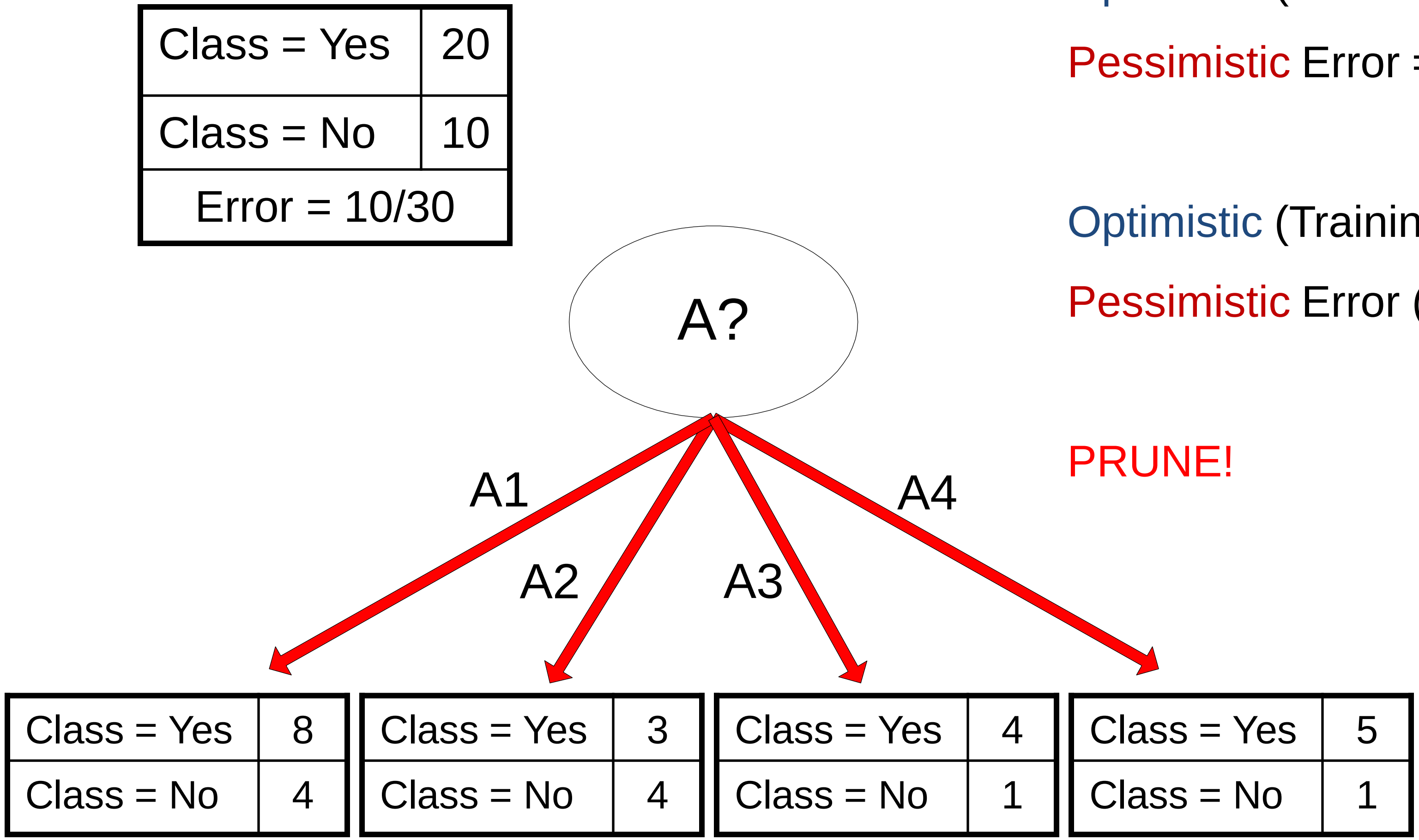
- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instance classes 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).

Model selection for Decision Trees

- 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 (subtree pruning) or by the most probable subtree (subtree raising).
  - Class label of leaf node is determined from majority class of instances in the sub-tree

  - Can use MDL for post-pruning
    - NP hard problem

Example of Post-Pruning



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

Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Optimistic (Training) Error (Before splitting) = 10/30

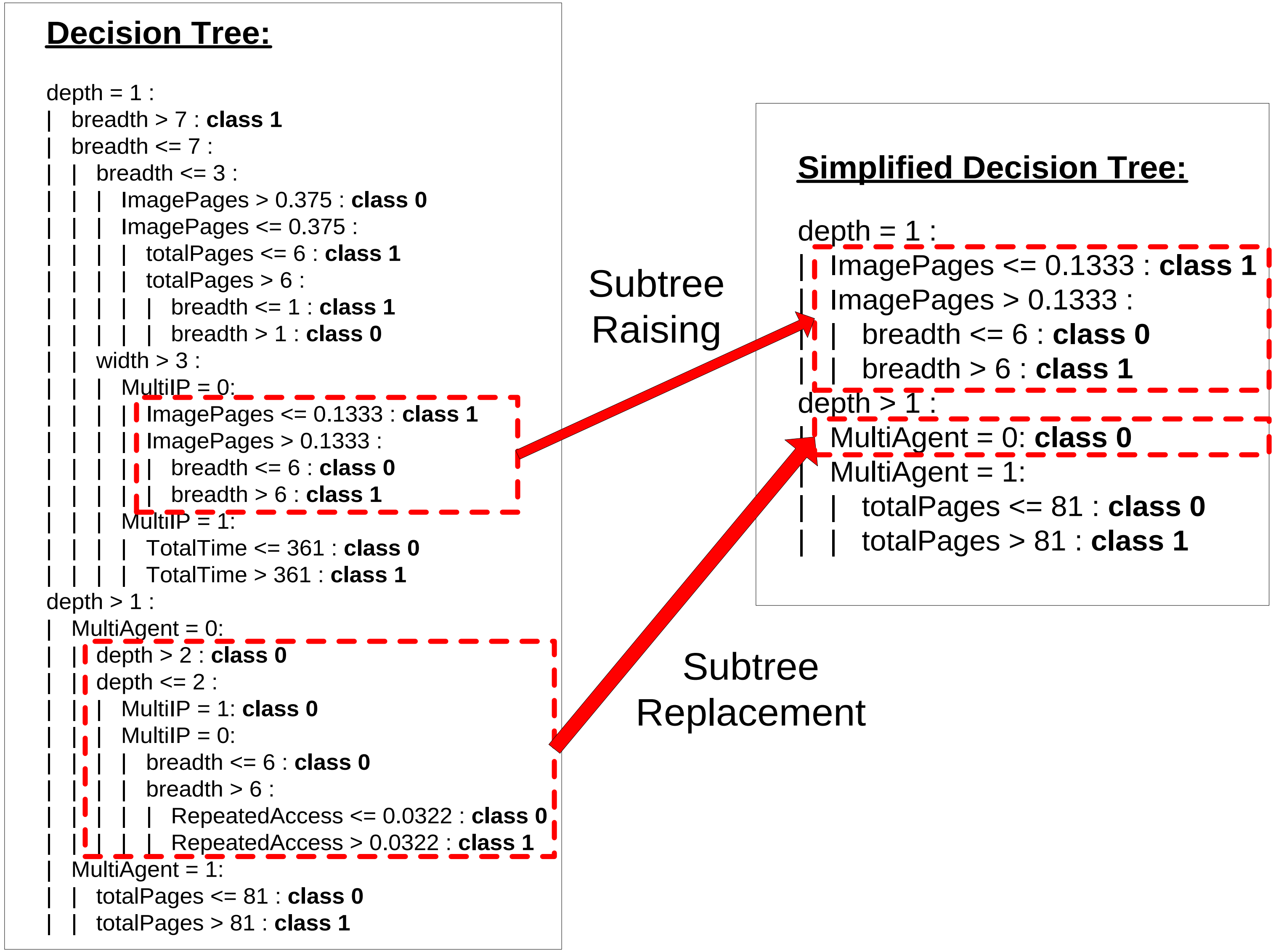
Pessimistic Error = (10 + 0.5)/30 = 10.5/30

Optimistic (Training) Error (After splitting) = 9/30

Pessimistic Error (After splitting) = (9 + 4 × 0.5)/30 = 11/30

PRUNE!

Examples of Post-pruning



Decision Tree:

```
depth = 1 :
|   breadth > 7 : class 1
|   breadth <= 7 :
|   |   breadth <= 3 :
|   |   |   ImagePages > 0.375 : class 0
|   |   |   ImagePages <= 0.375 :
|   |   |   |   totalPages <= 6 : class 1
|   |   |   |   totalPages > 6 :
|   |   |   |   |   breadth <= 1 : class 1
|   |   |   |   |   breadth > 1 : class 0
|   |   width > 3 :
|   |   |   MultiIP = 0:
|   |   |   |   ImagePages <= 0.1333 : class 1
|   |   |   |   ImagePages > 0.1333 :
|   |   |   |   |   breadth <= 6 : class 0
|   |   |   |   |   breadth > 6 : class 1
|   |   |   MultiIP = 1:
|   |   |   |   TotalTime <= 361 : class 0
|   |   |   |   TotalTime > 361 : class 1
depth > 1 :
|   MultiAgent = 0:
|   |   depth > 2 : class 0
|   |   depth <= 2 :
|   |   |   MultiIP = 1: class 0
|   |   |   MultiIP = 0:
|   |   |   |   breadth <= 6 : class 0
|   |   |   |   breadth > 6 :
|   |   |   |   |   RepeatedAccess <= 0.0322 : class 0
|   |   |   |   |   RepeatedAccess > 0.0322 : class 1
|   MultiAgent = 1:
|   |   totalPages <= 81 : class 0
|   |   totalPages > 81 : class 1
```

Simplified Decision Tree:

```
depth = 1 :
|   ImagePages <= 0.1333 : class 1
|   ImagePages > 0.1333 :
|   |   breadth <= 6 : class 0
|   |   breadth > 6 : class 1
depth > 1 :
|   MultiAgent = 0: class 0
|   MultiAgent = 1:
|   |   totalPages <= 81 : class 0
|   |   totalPages > 81 : class 1
```

MODEL EVALUATION

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

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Metrics for Performance Evaluation…

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

- Most widely-used metric:

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

Precision-Recall

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

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

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

$$\text{F-measure (F)} = \frac{1}{\left(\frac{1/r+1/p}{2}\right)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c} = \frac{2TP}{2TP+FP+FN}$$

Assumption: The class YES is the one we care about.

- 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)**

More Measures of Classification Performance

		PREDICTED CLASS	
ACTUAL CLASS		Yes	No
	Yes	TP	FN
	No	FP	TN

α is the probability that we reject the null hypothesis when it is true.

This is a Type I error or a false positive (FP).

β is the probability that we accept the null hypothesis when it is false.

This is a Type II error or a false negative (FN).

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

$$ErrorRate = 1 - accuracy$$

$$Precision = Positive\ Predictive\ Value = \frac{TP}{TP + FP}$$

$$Recall = Sensitivity = TP\ Rate = \frac{TP}{TP + FN}$$

$$Specificity = TN\ Rate = \frac{TN}{TN + FP}$$

$$FP\ Rate = \boxed{\alpha} = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN\ Rate = \boxed{\beta} = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

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 TPR (true positive rate) (on the y-axis) against FPR (false positive rate) (on the x-axis)

Look at the positive predictions of the classifier and compute:

$$TPR = \frac{TP}{TP + FN}$$

What fraction of true positive instances are predicted correctly? (1-Type II error rate)

$$FPR = \frac{FP}{FP + TN}$$

What fraction of true negative instances were predicted incorrectly? (Type I error rate)

We want to strike a balance between these two

		PREDICTED CLASS	
Actual		Yes	No
	Yes	a (TP)	b (FN)
	No	c (FP)	d (TN)

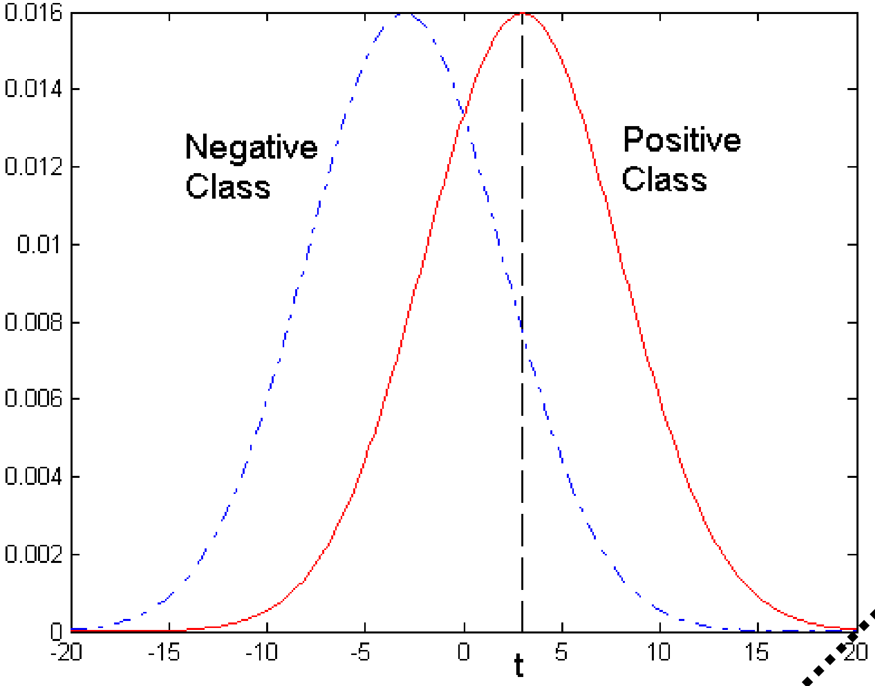
ROC (Receiver Operating Characteristic)

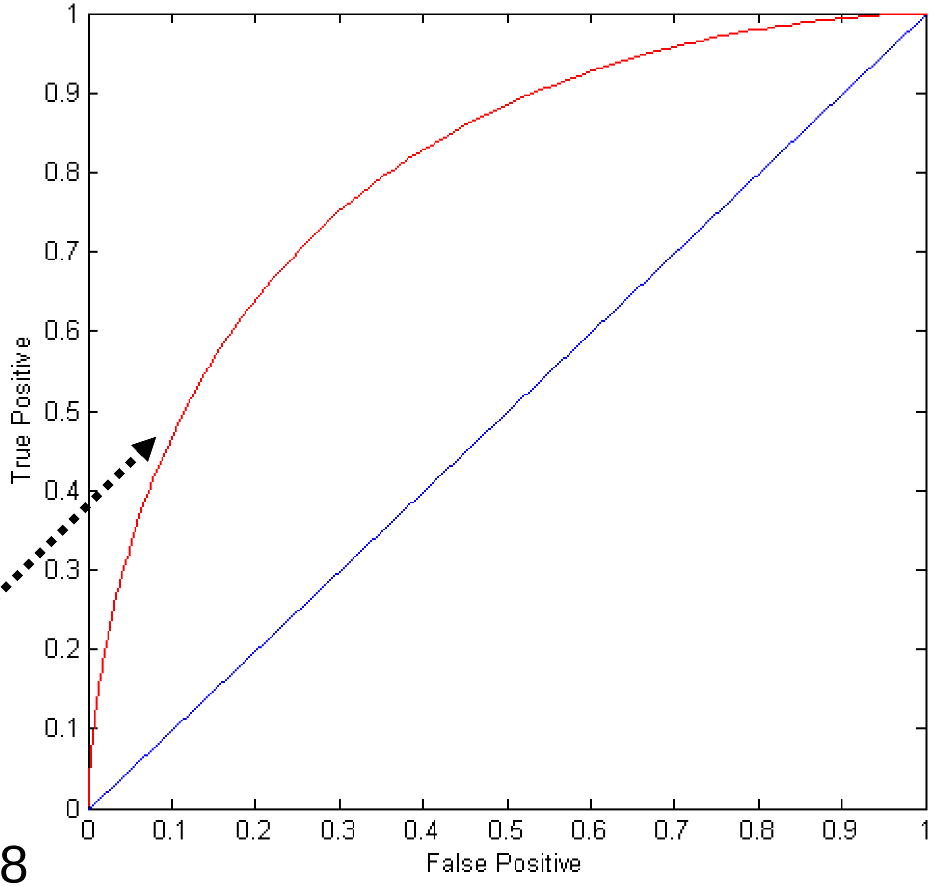
- Performance of a classifier represented as a point on the **ROC** curve

- Changing some parameter of the 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**:

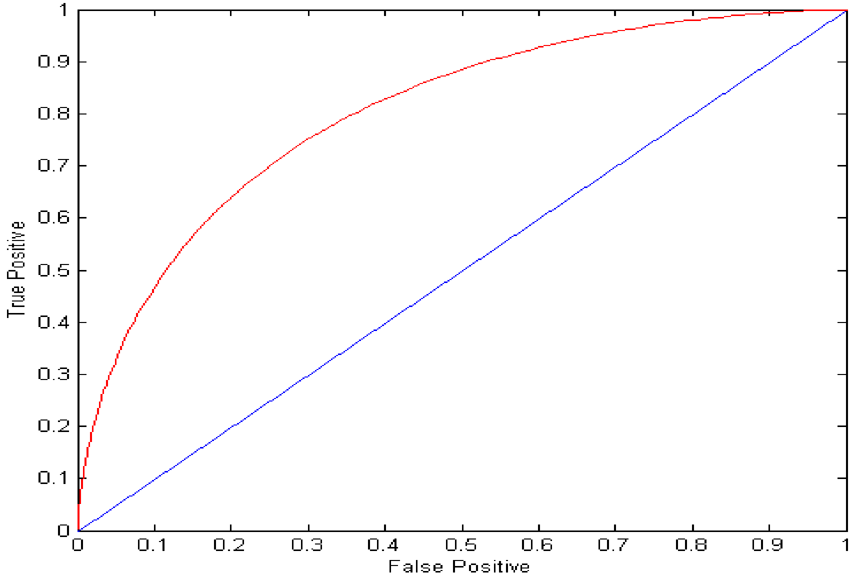
TP=0.5, FN=0.5, FP=0.12, FN=0.88

ROC Curve

(TP,FP):

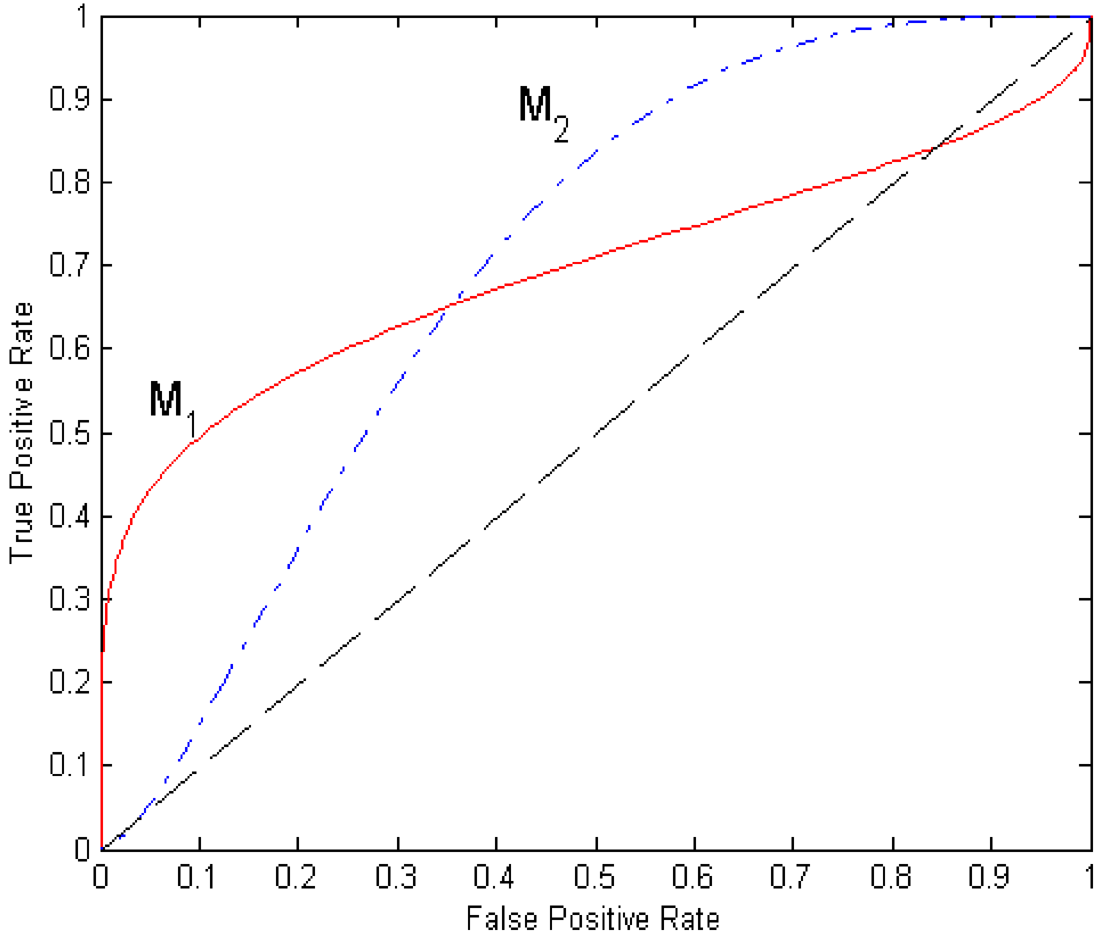
- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



		PREDICTED CLASS	
		Yes	No
Actual	Yes	a (TP)	b (FN)
	No	c (FP)	d (TN)

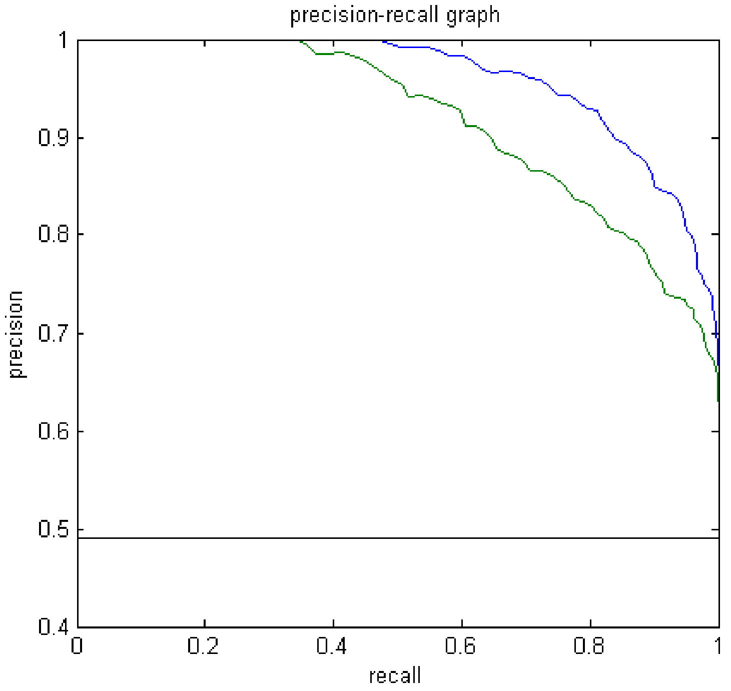
Using ROC for Model Comparison



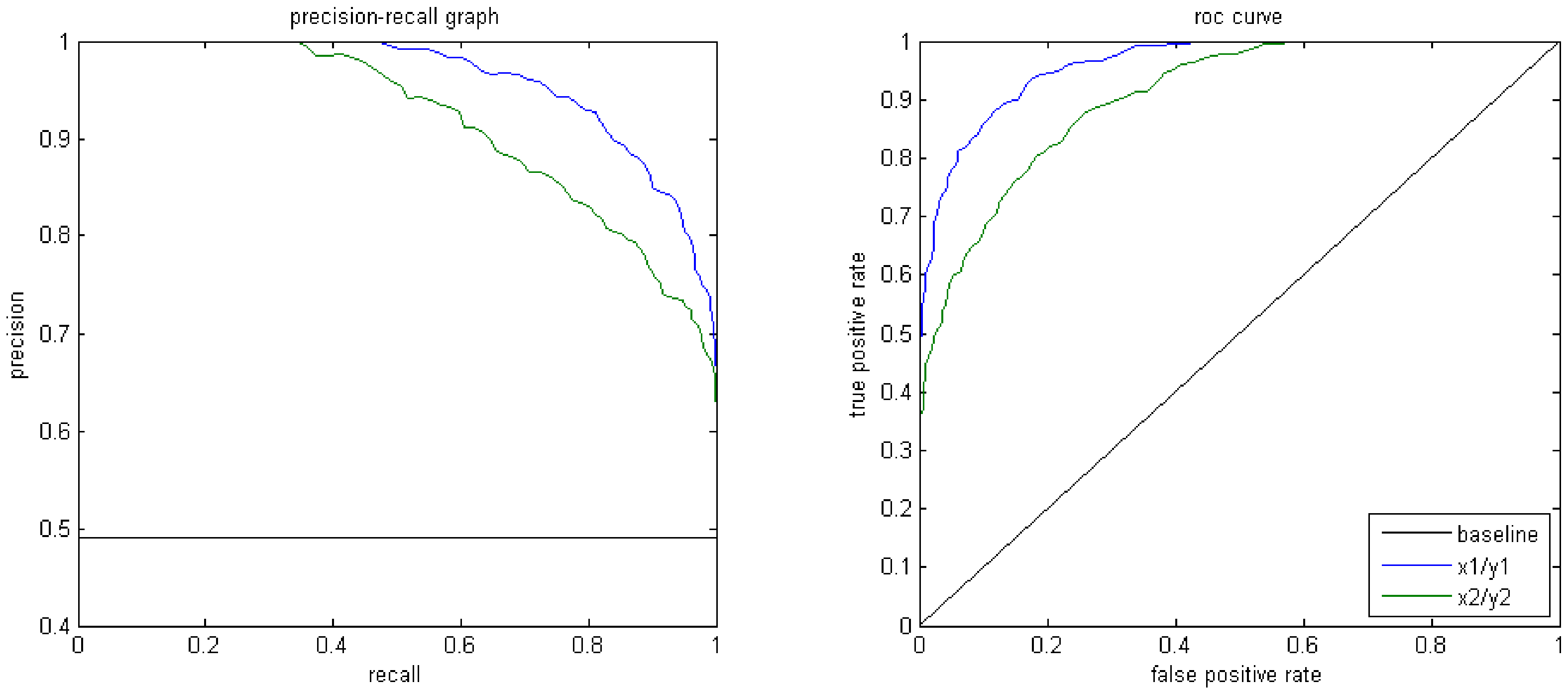
- No model consistently outperform the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve (AUC)
  - Ideal: Area = 1
  - Random guess:
    - Area = 0.5

Precision-Recall plot

- Usually for parameterized models, it controls the precision/recall tradeoff



ROC curve vs Precision-Recall curve



Area Under the Curve (AUC) as a single number for evaluation

Methods of Performance Estimation

- Holdout
  - Reserve **2/3** for training and **1/3** for testing
- Random subsampling
  - One sample may be biased -- 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**
  - Guarantees that each record is used the same number of times for training and testing
- Bootstrap
  - Sampling with replacement
  - ~63% of records used for training, ~27% for testing

Class imbalance

- 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
  - Precision and recall are better measures

Dealing with class Imbalance

- Class imbalance is a problem in training:
  - If the class we are interested in is very rare, then the classifier will ignore it.
- Solution
  - We can balance the class distribution
    - Sample from the larger class so that the size of the two classes is the same
    - Replicate the data of the class of interest so that the classes are balanced
      - Over-fitting issues
  - We can modify the optimization criterion by using a cost sensitive metric

Cost Matrix

<table>
<tr><td></td><td colspan="3">PREDICTED CLASS</td></tr>
<tr><td rowspan="3">ACTUAL CLASS</td><td>C(i|j)</td><td><b>Class=Yes</b></td><td><b>Class=No</b></td></tr>
<tr><td><b>Class=Yes</b></td><td>C(Yes|Yes)</td><td>C(No|Yes)</td></tr>
<tr><td><b>Class=No</b></td><td>C(Yes|No)</td><td>C(No|No)</td></tr>
</table>

C(i|j): Cost of classifying class **j** example as class **i**

Weighted Accuracy

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

COST MATRIX	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	**Class=Yes**	**Class=No**
	Class=Yes	w_1 C(Yes|Yes)	w_2 C(No|Yes)
	Class=No	w_3 C(Yes|No)	w_4 C(No|No)

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

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	1	100
	-	1	1

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

Accuracy = 80%

Weighted Accuracy = 8.9%

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 90%

Weighted Accuracy= 9%

Classification Cost

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

COST MATRIX	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	**Class=Yes**	**Class=No**
	Class=Yes	w_1 C(Yes|Yes)	w_2 C(No|Yes)
	Class=No	w_3 C(Yes|No)	w_4 C(No|No)

$$\text{Classification Cost} = w_1 a + w_2 b + w_3 c + w_4 d$$

Some weights can also be negative

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	-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		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if

1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

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

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