Classification

Classification

- The problem of discriminating between different classes of objects
  - In our case: email spam vs. non-spam

- Classification process:
  - Find examples for which you know the class (training set)
  - Find a set of features that discriminate between the examples within the class and outside the class
  - Create a function that given the features decides the class
  - Apply the function to new examples.

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)

What is classification?

A machine learning task that deals with identifying the class to which an instance belongs

A classifier performs classification

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

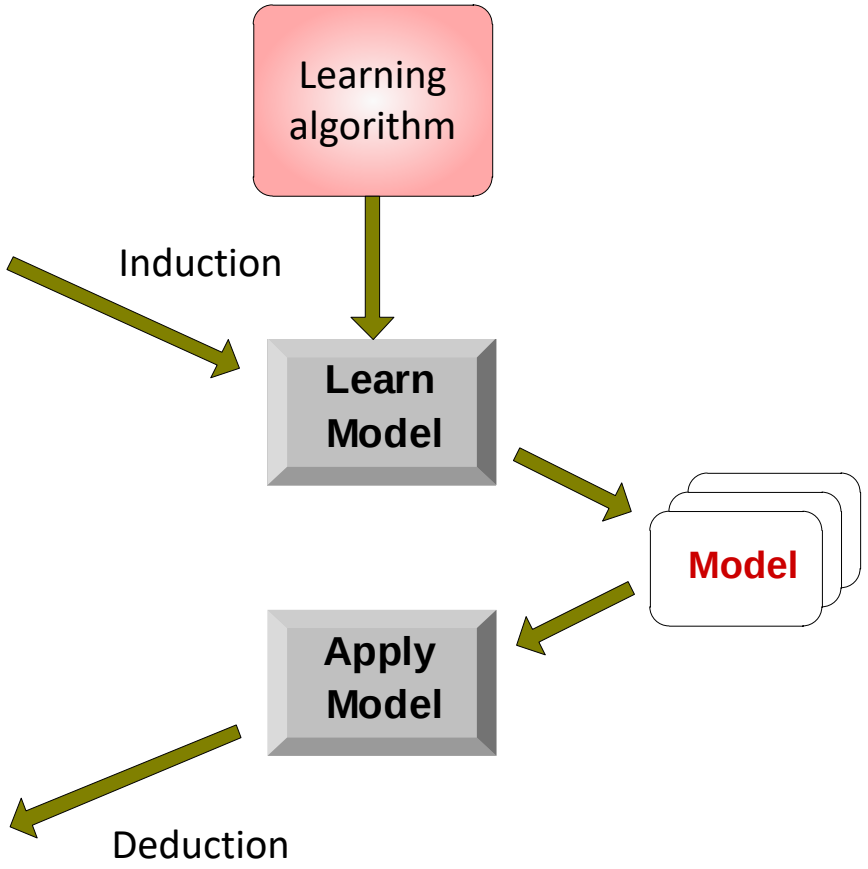
General approach to classification

- Training set consists 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

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

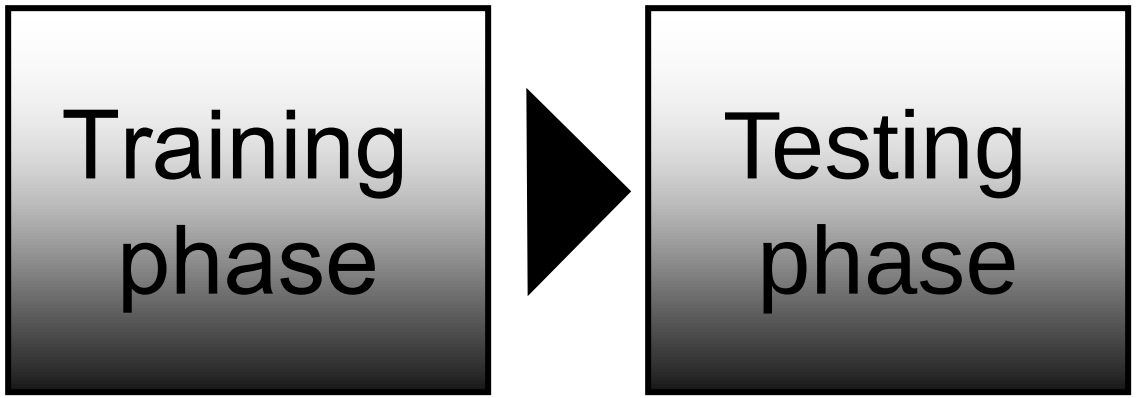
Training Set



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

Classification learning



Learning the classifier from the available data

‘Training set’

(Labeled)

Testing how well the classifier performs

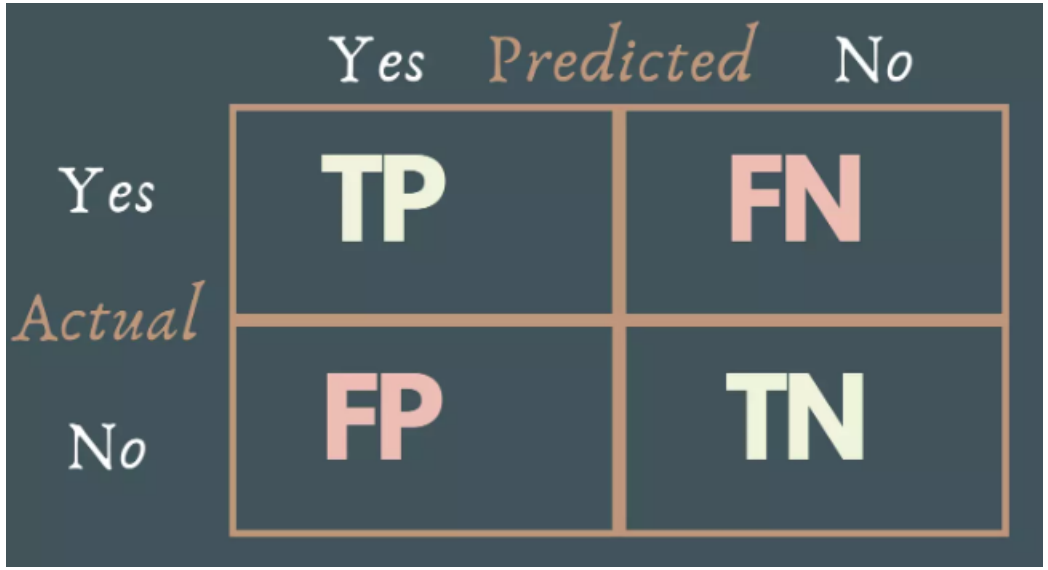
‘Testing set’

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}

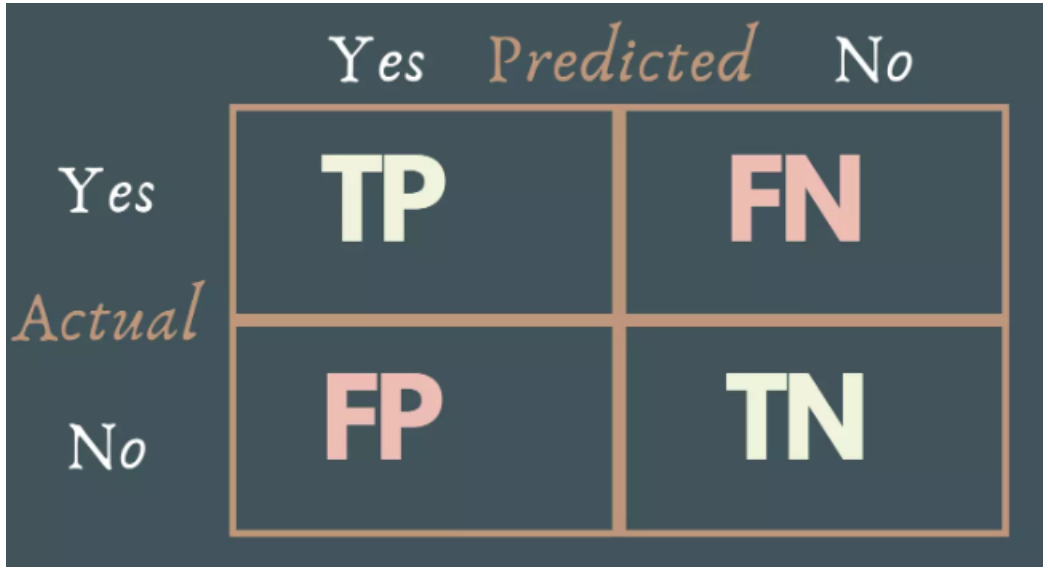
Accuracy



Total number of correct/true predictions among all the predictions.

$$\text{accuracy} = \frac{TP + TN}{n}$$

Recall

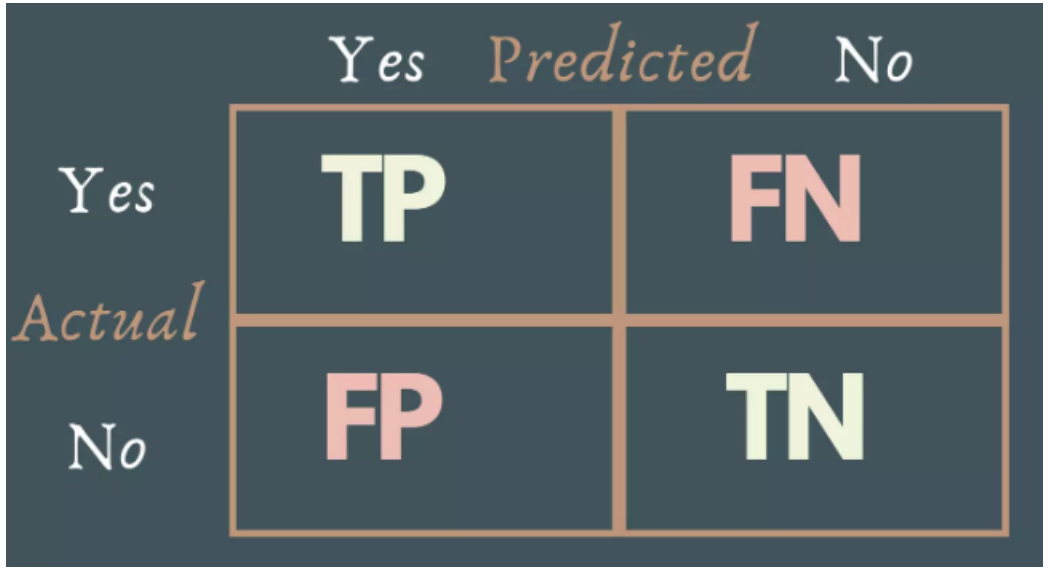


Measures classifiers' completeness.

Positive cases correctly predicted among all actual positive cases.

$$\text{recall} = \frac{TP}{TP + FN}$$

Precision



Measures classifiers' exactness.

Positive cases correctly predicted among all predicted positive cases.

$$\text{precision} = \frac{TP}{TP + FP}$$

F1 score

$$\text{F1 score} = \frac{\text{precision x recall}}{\text{precision + recall}} \text{ x 2}$$

$$\text{F1 Score} = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

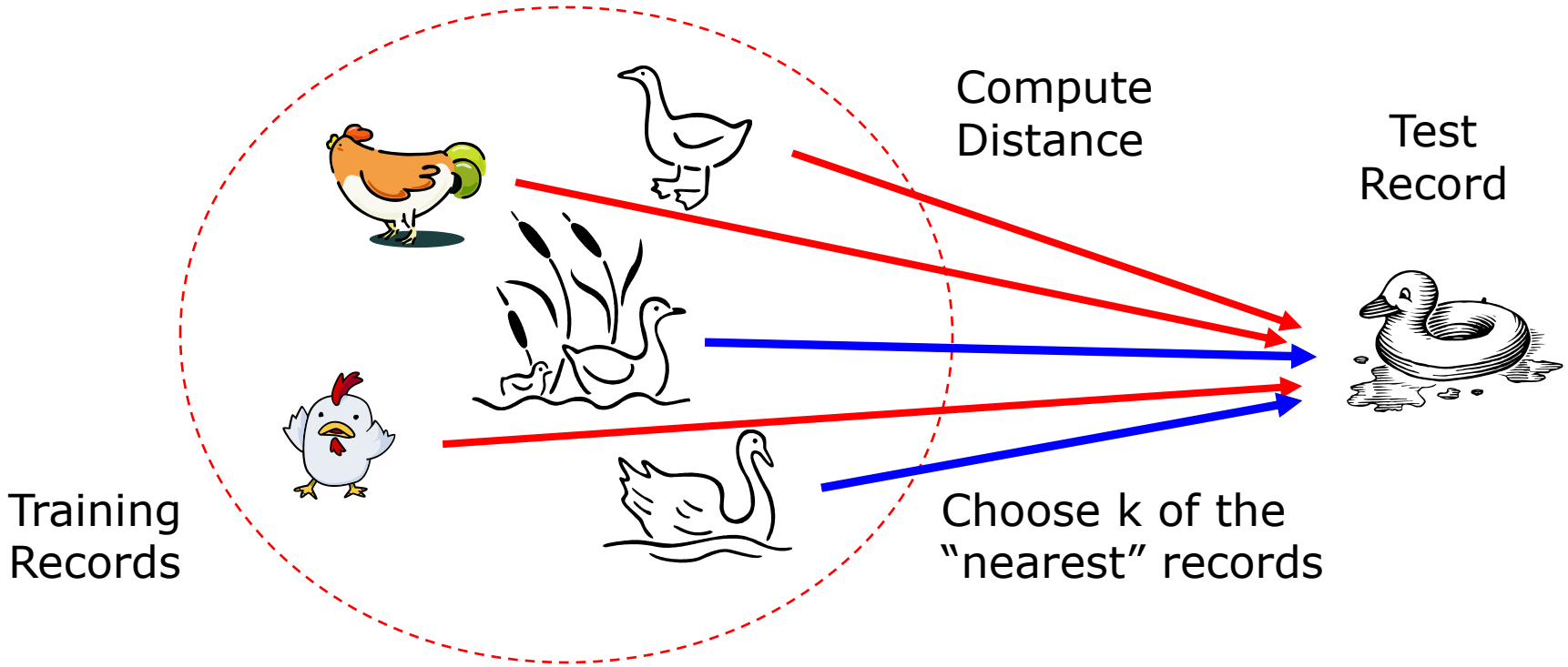
Classification Techniques

- K-Nearest Neighbor
- Naïve Bayes
- Decision Tree
- Random Forest
- Neural Networks
- Support Vector Machines

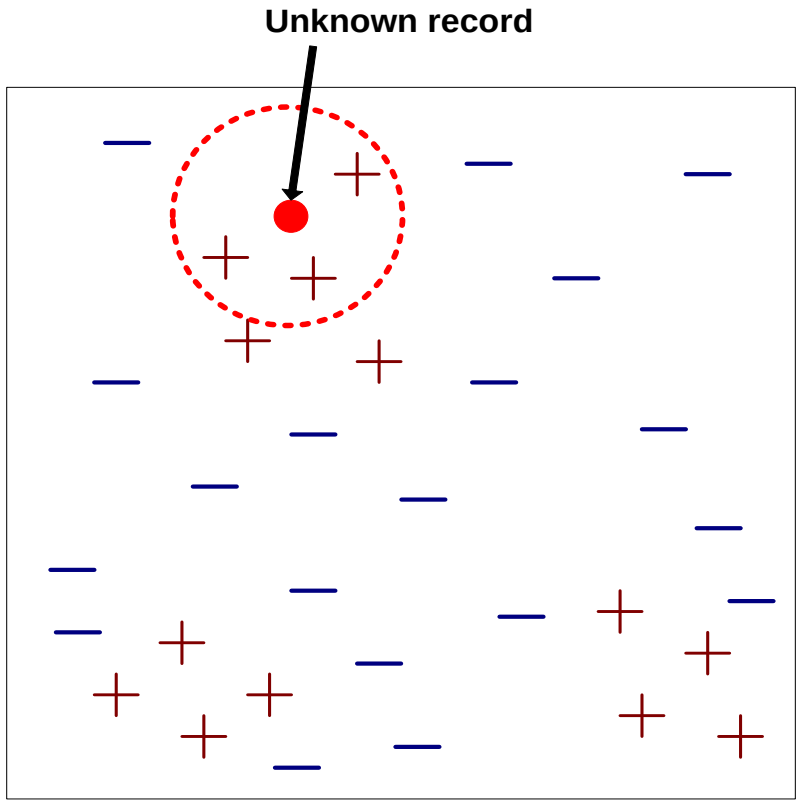
K-Nearest Neighbor Classifiers

K-Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



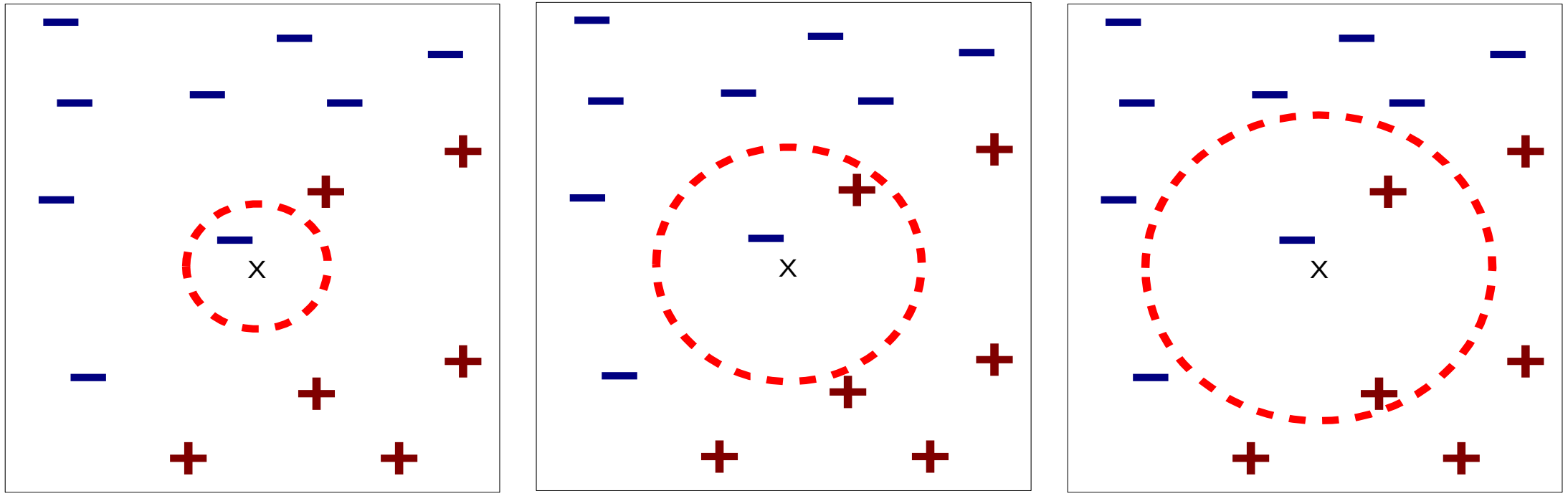
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Note: In practice, *k* is usually odd, so as to avoid ties

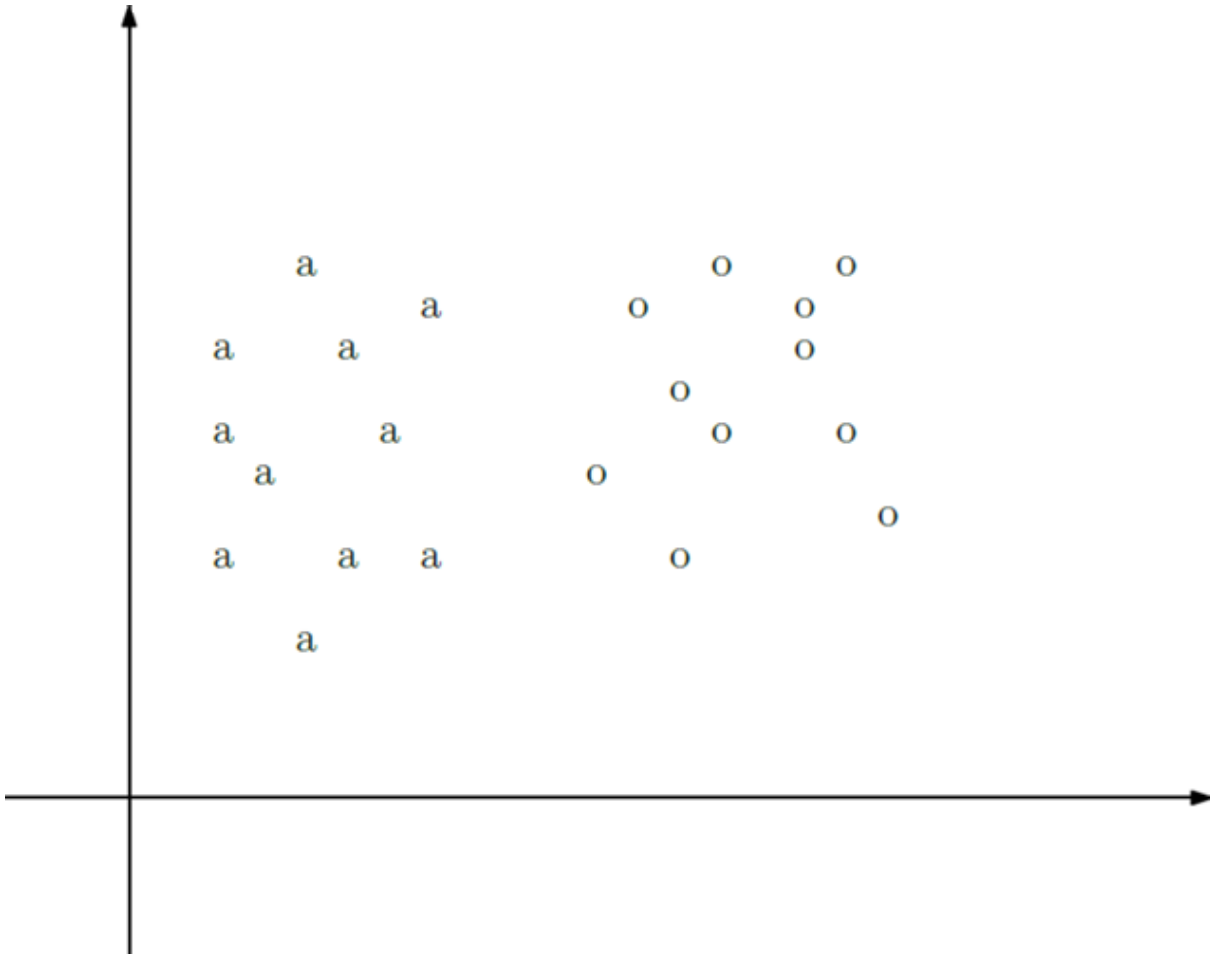
Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

K-Nearest-Neighbors (Example)

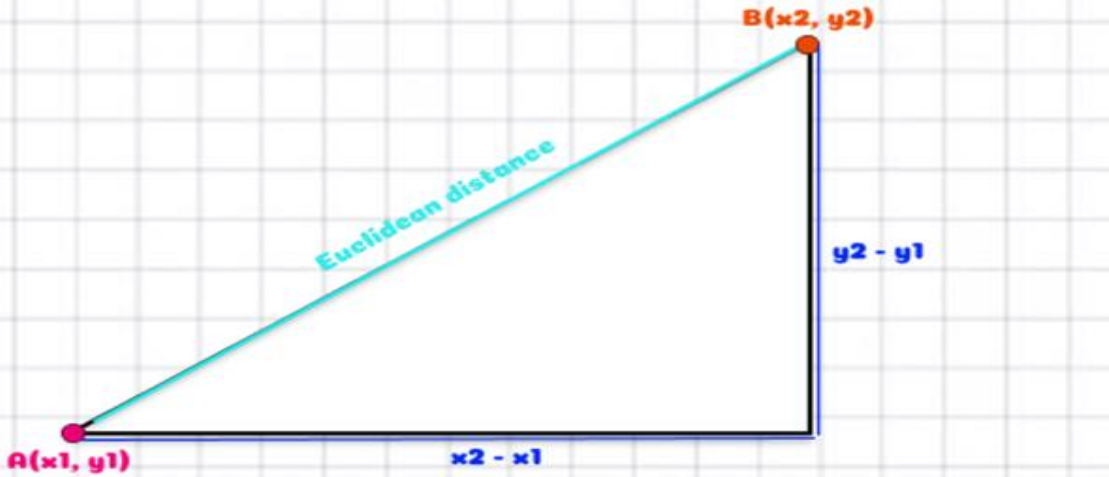


Popular Distance Metric

- Euclidean Distance
- Manhattan Distance
- Cosine distance

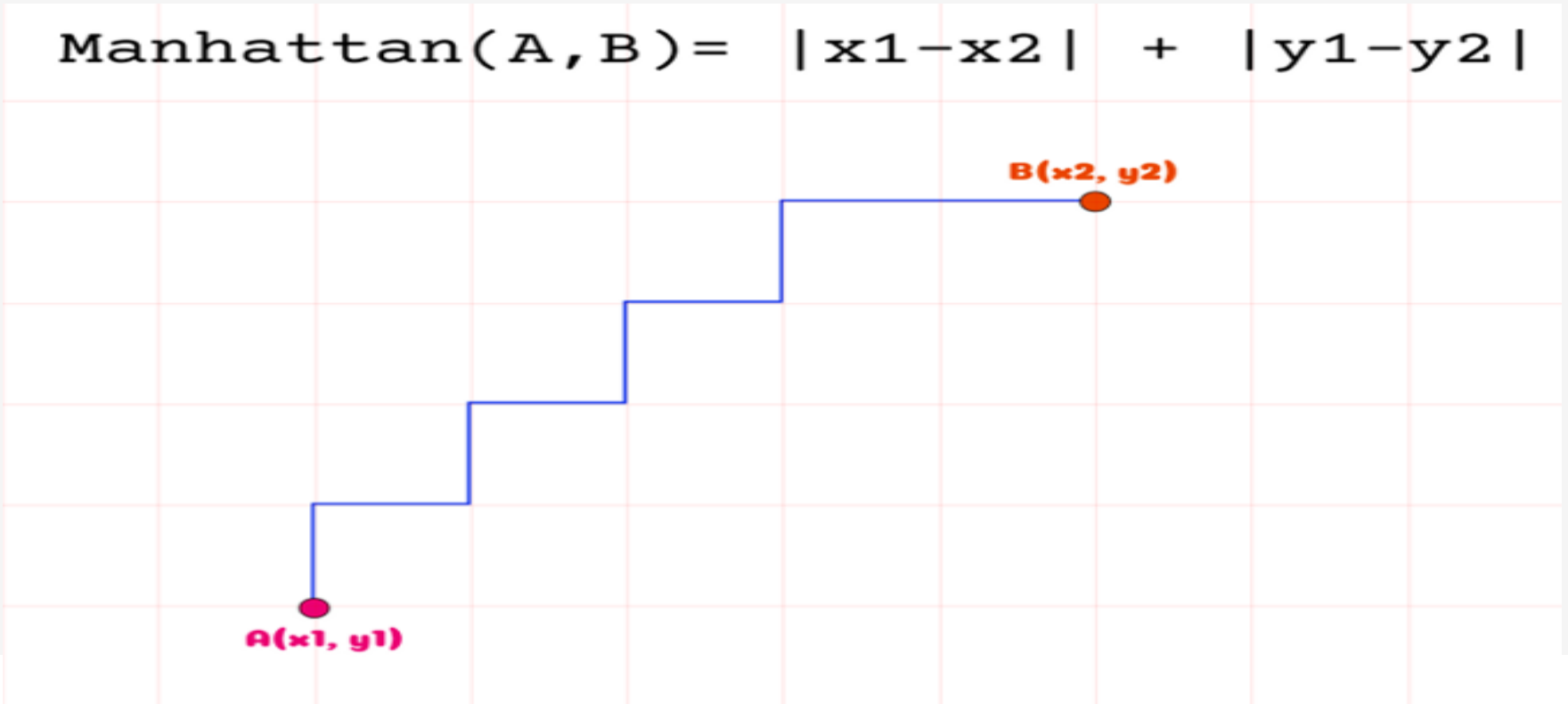
Euclidean Distance represents the shortest distance between two data points.

$$Euclidean\,(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

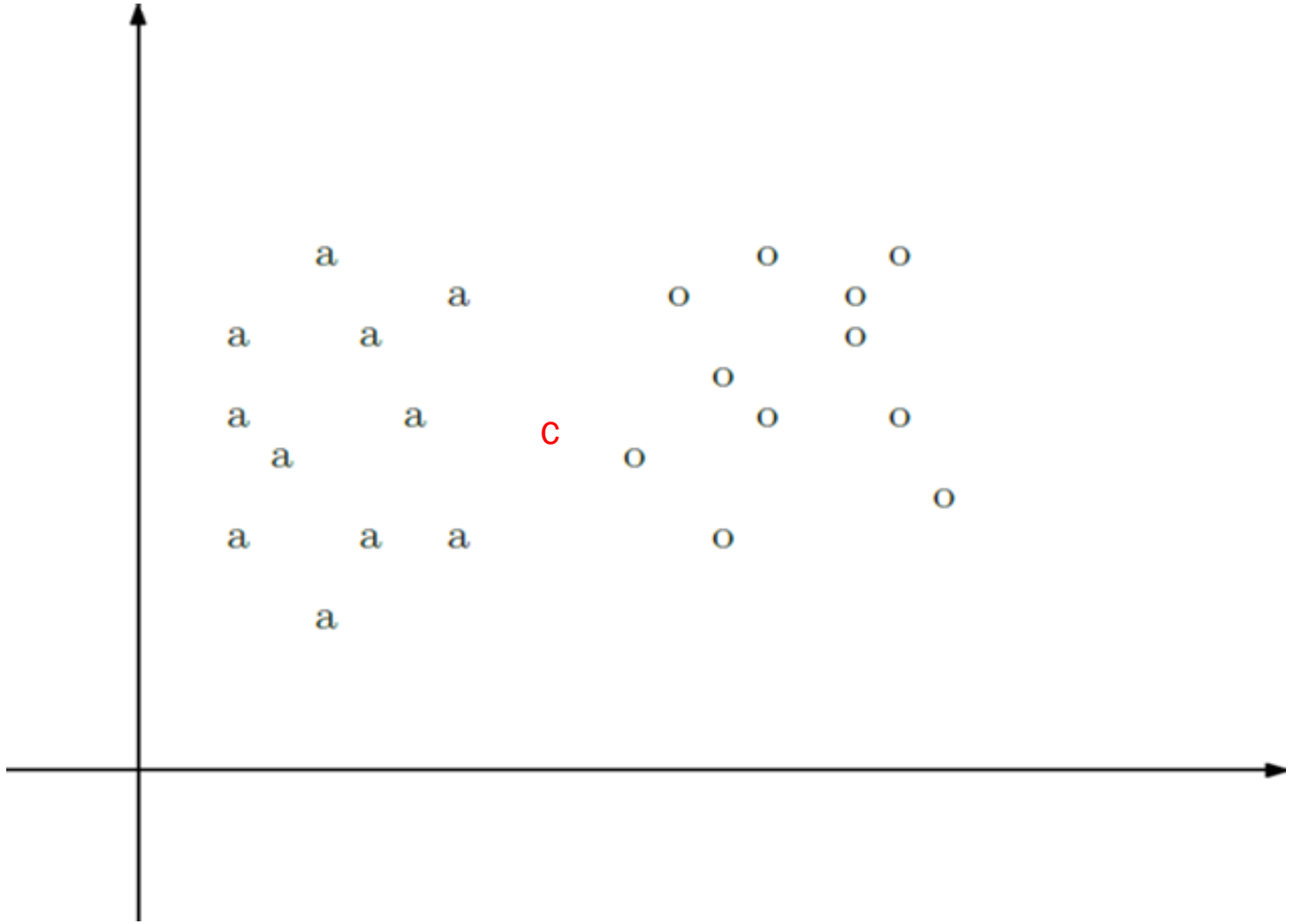


Manhattan Distance is the sum of absolute differences between points across all the dimensions.

$$Manhattan(A,B) = |x1 - x2| + |y1 - y2|$$



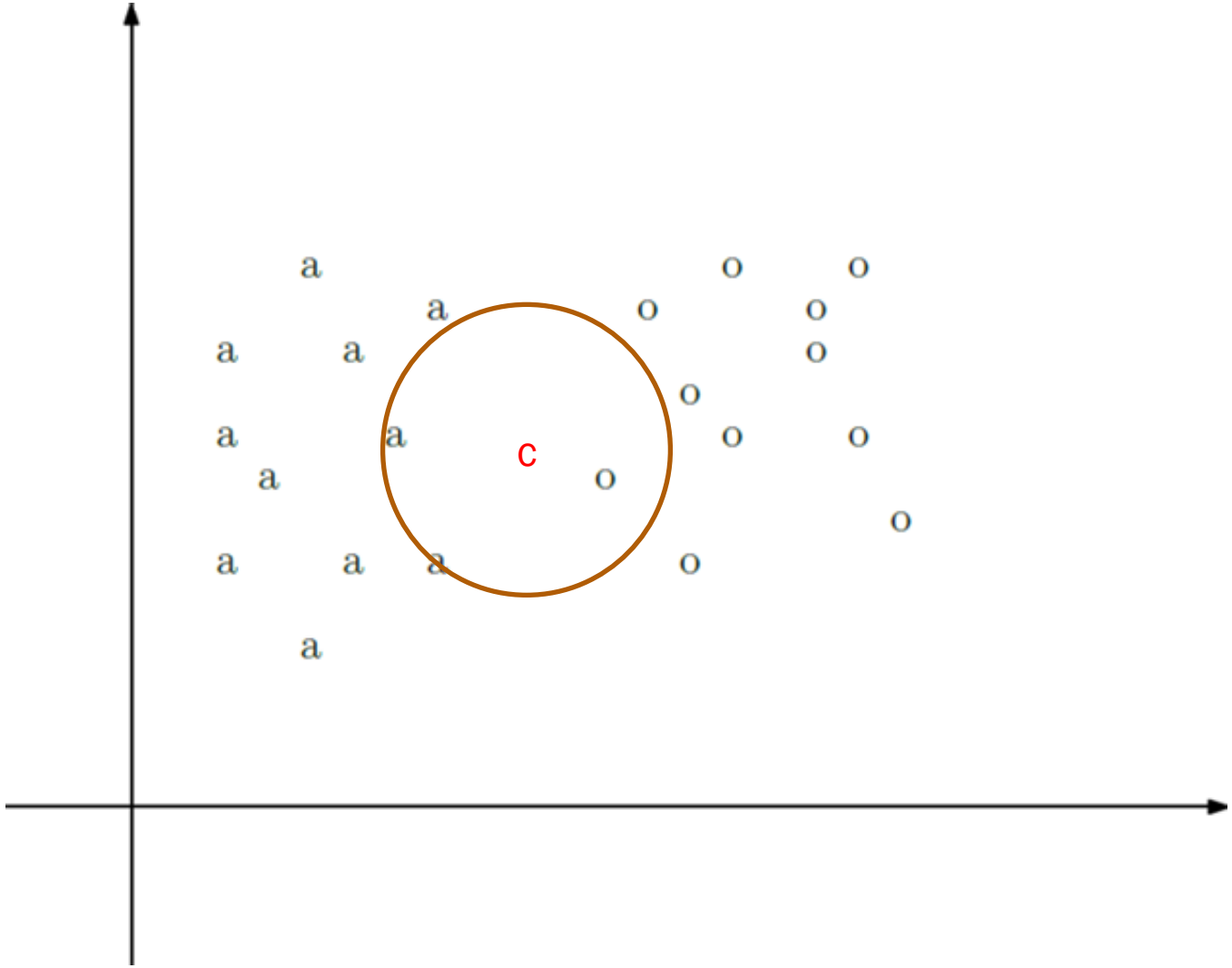
What is the most possible label for c?



What is the most possible label for c?

- Solution: Looking for the nearest K neighbors of c.
- Take the majority label as c's label
- Let's suppose k = 3:

What is the most possible label for c?



What is the most possible label for c?

- The 3 nearest points to c are: a, a and o.
- Therefore, the most possible label for c is a.

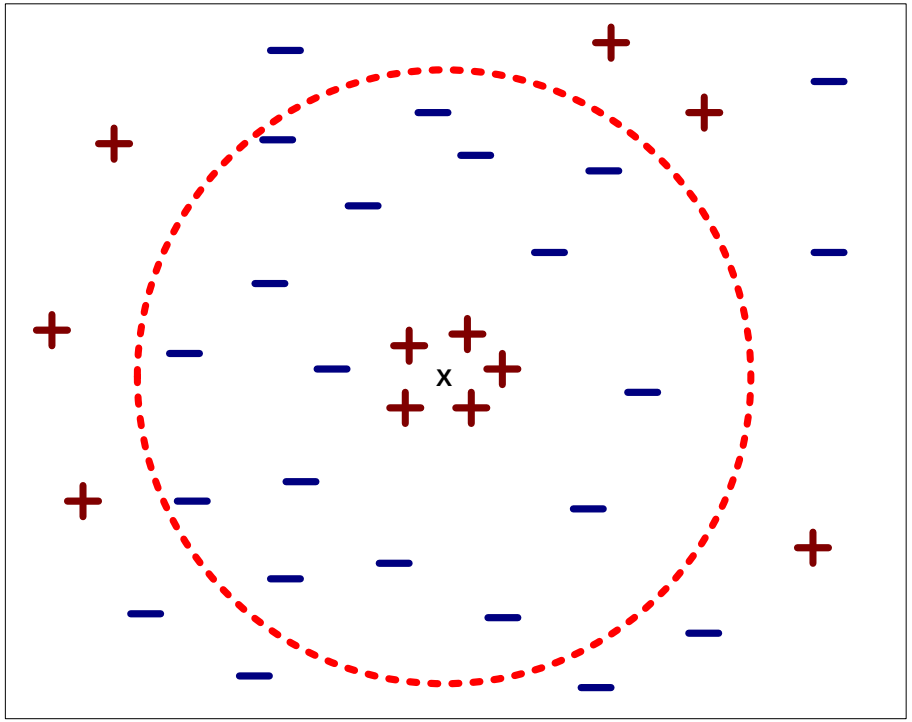
Nearest Neighbor Classification: Issues

- The value of *k*, the number of nearest neighbour to retrieve
- Choice of Distance Metric to compute distance between records

Nearest Neighbor Classification: Issues Contd…

- **Choosing the value of k**:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes

Rule of thumb:
K=sqrt(N)
N: number of training points



Decision Tree Classifier

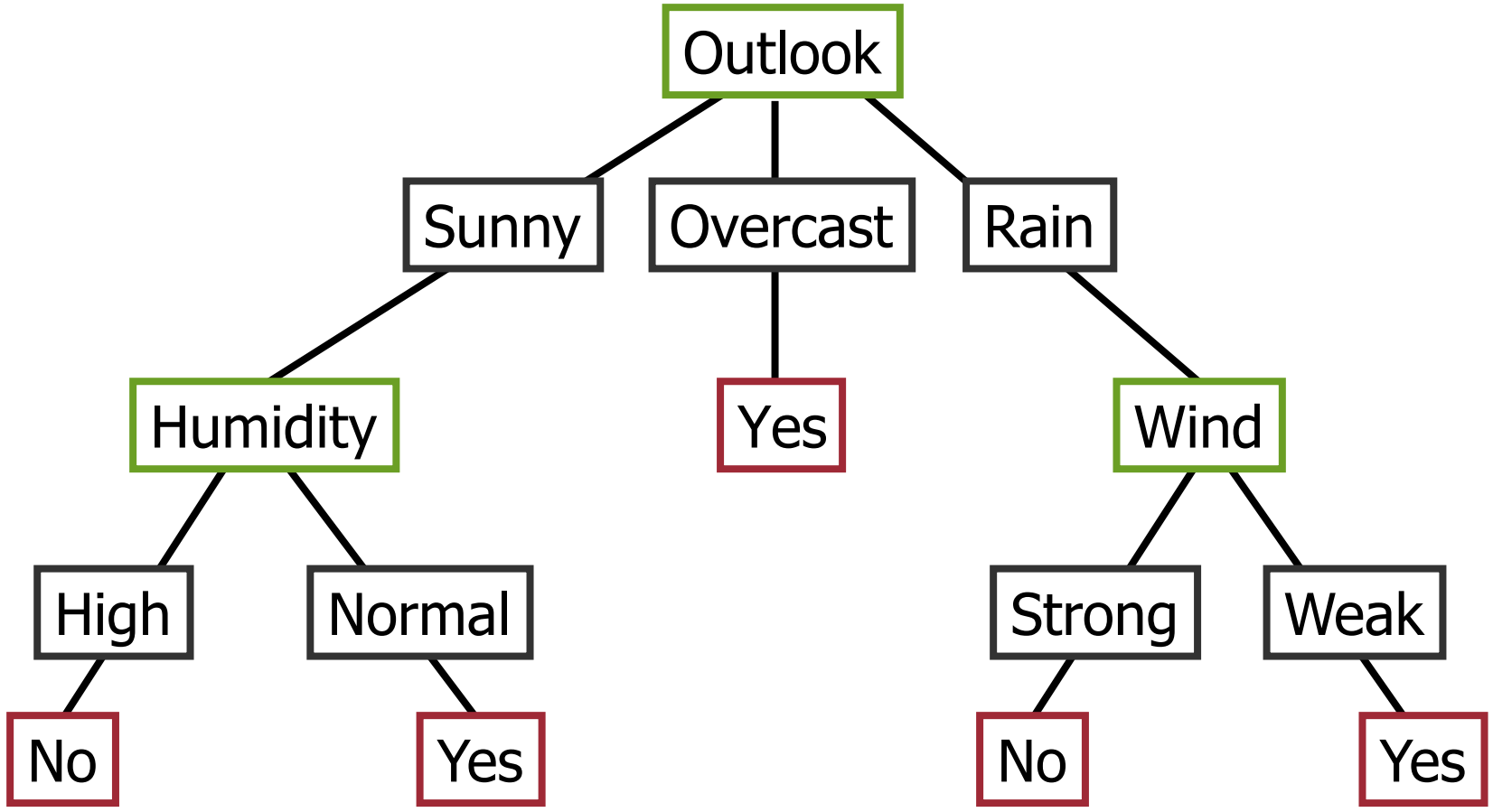
Decision Tree Classifier

- Decision tree is tree structured classifier
- Has two types of nodes:
  - Decision nodes
  - Leaf nodes
- Decision node specify a choice or tests an attribute
- Leaf node indicate the classification or value of an example

Training Examples

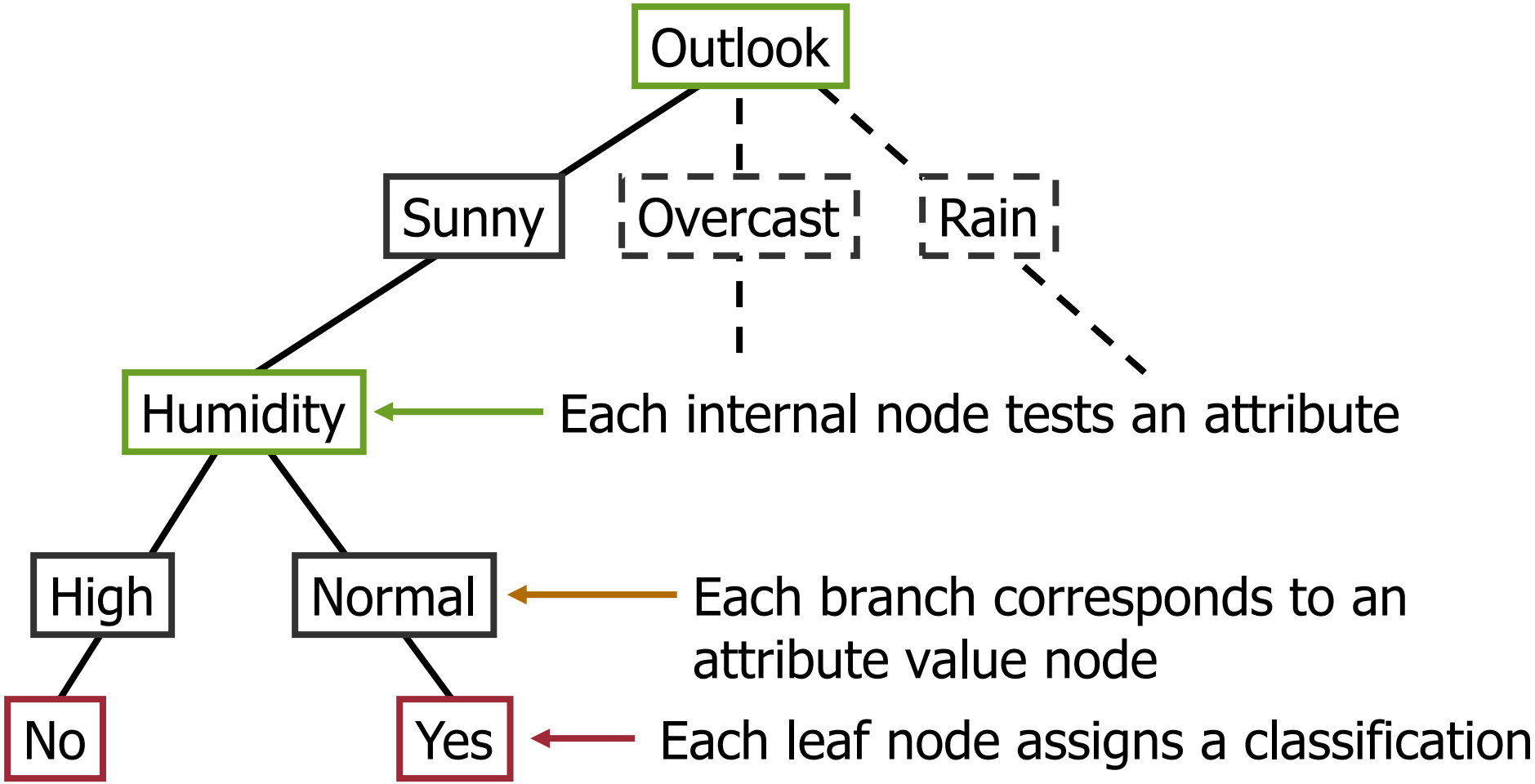
Day	Outlook	Temp.	Humidity	Wind	EnjoySport
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Example 1 of Decision Tree



Example 1 of Decision Tree

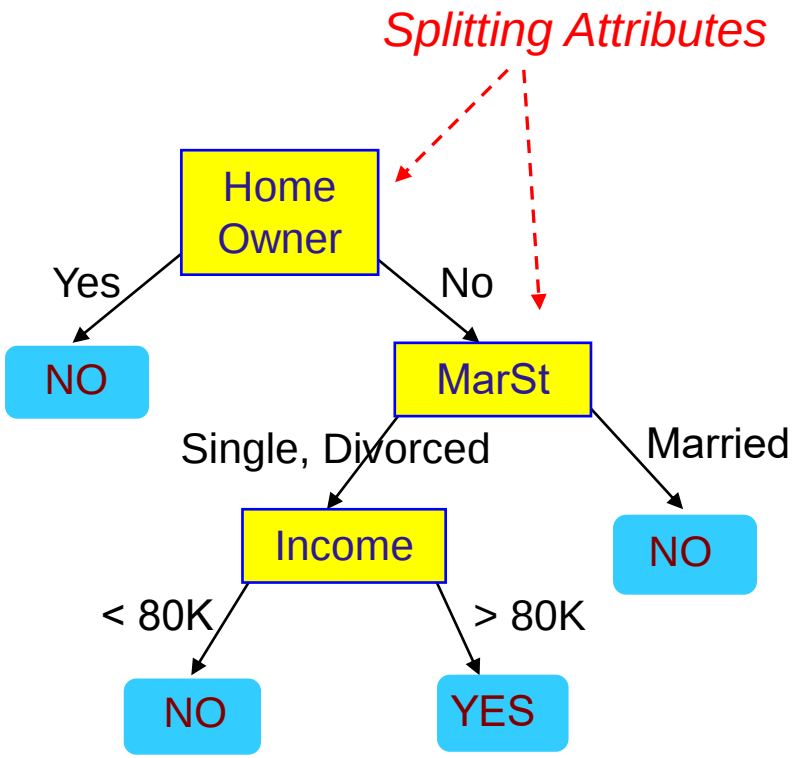
Contd…



Example 2 of a Decision Tree

	categorical	categorical	continuous	class
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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

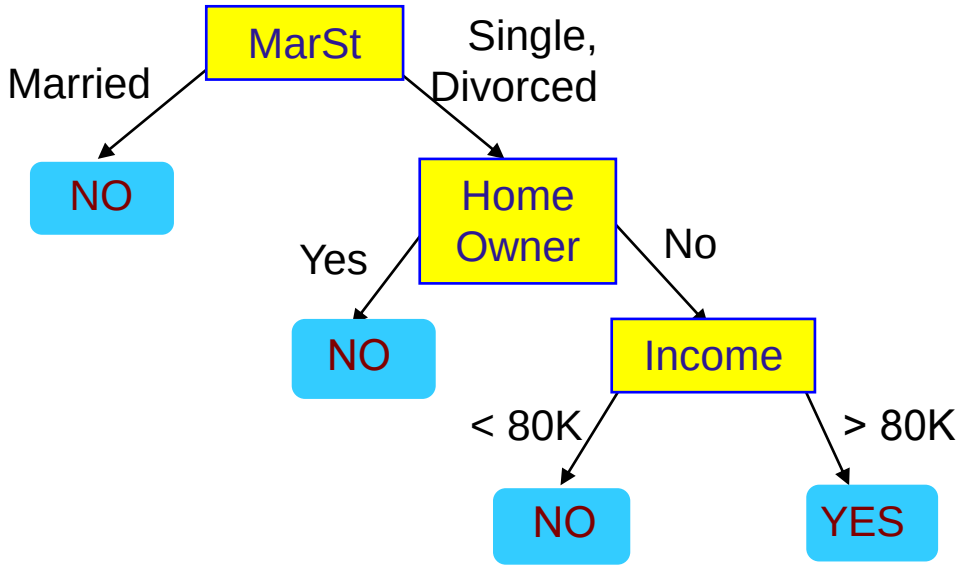
Training Data



Model: Decision Tree

Example 3 of Decision Tree

	categorical	categorical	continuous	class
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
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



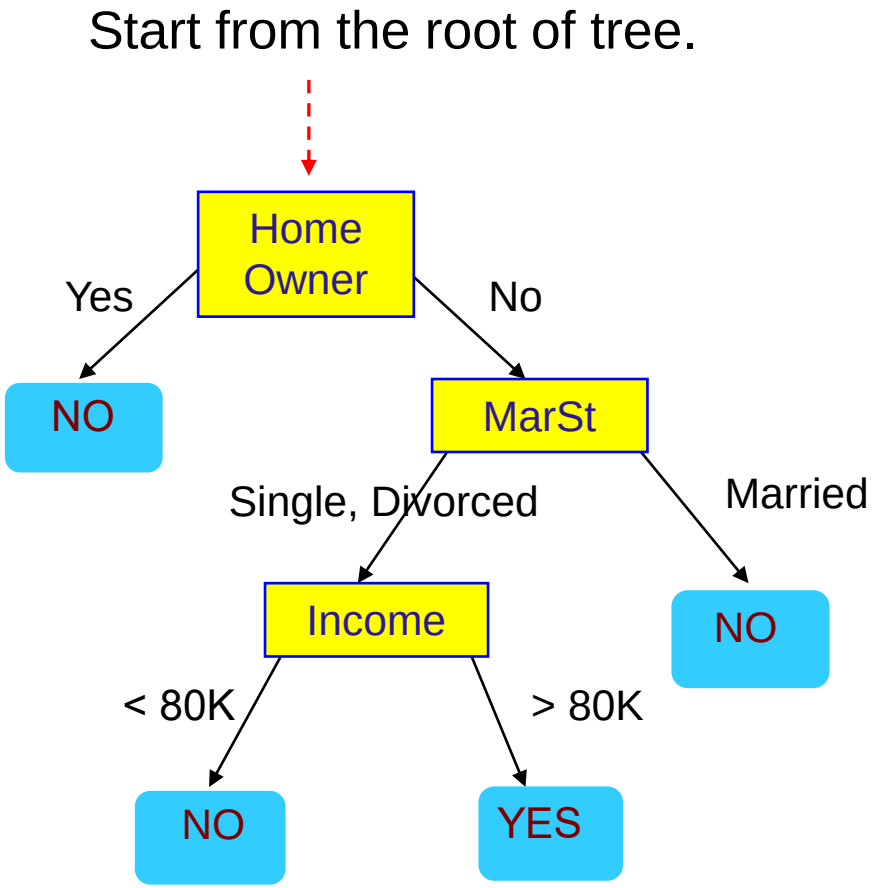
There could be more than one tree that fits the same data!

Issues

- Given some training example, what decision tree should be generated (or chosen)?

- One proposal: prefer the smallest tree (Small number of nodes or Low depth) that is consistent with the data (Bias)

Apply Model to Test Data



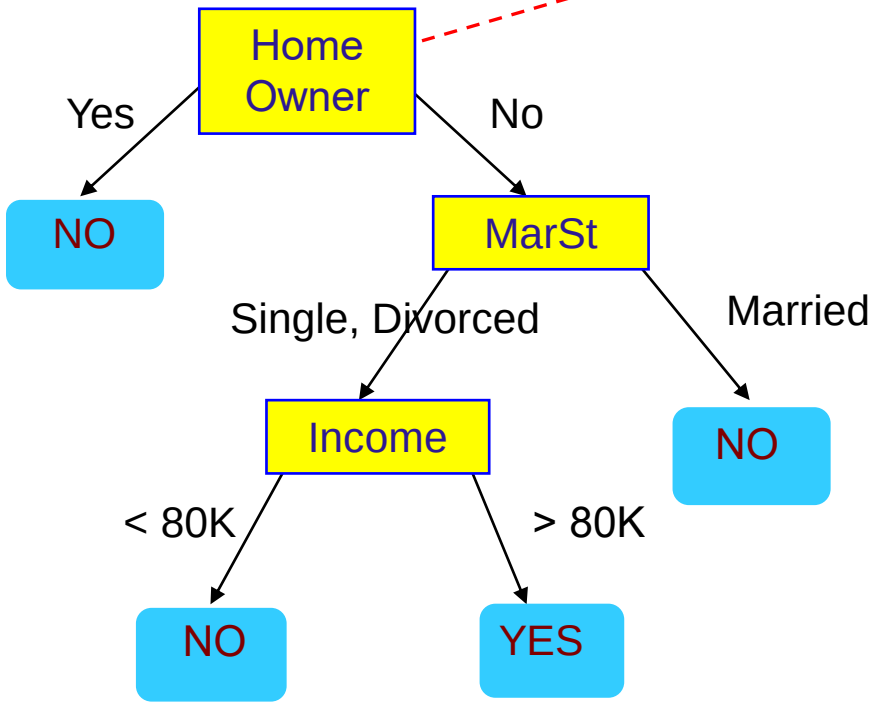
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	**?**

Apply Model to Test Data

Test Data

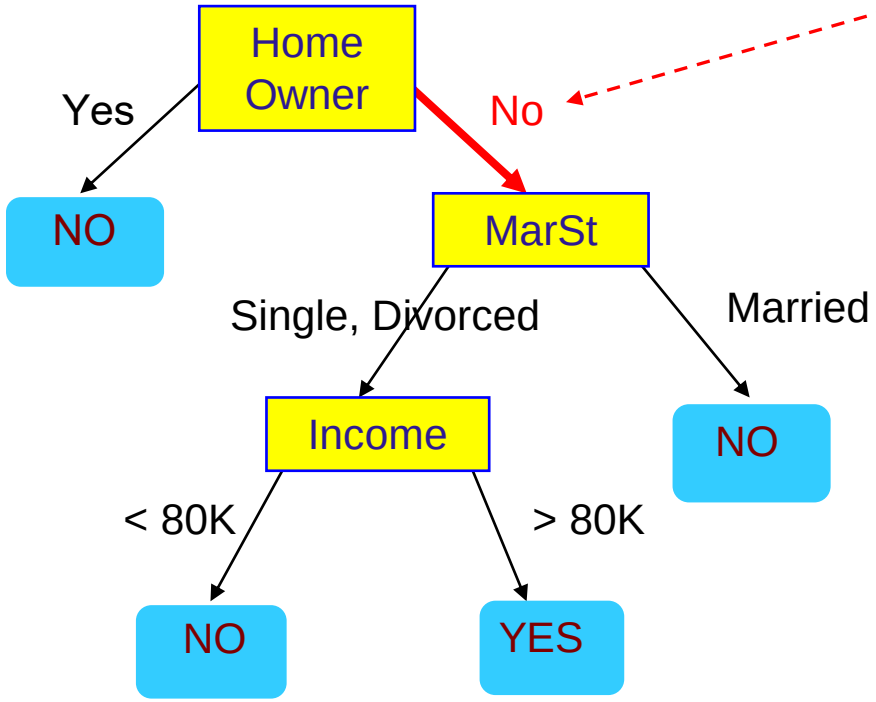
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

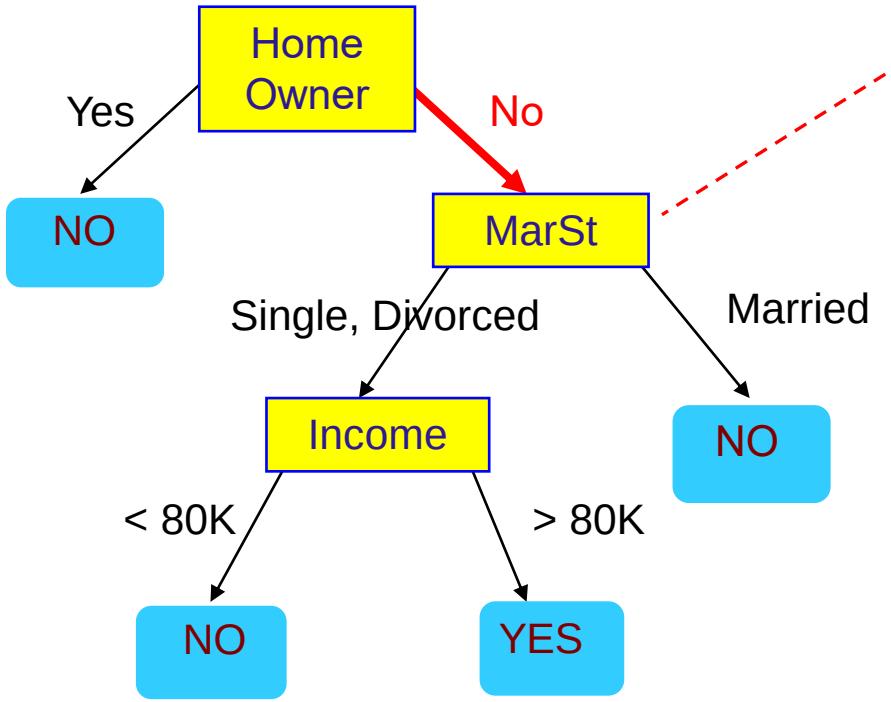
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data

Test Data

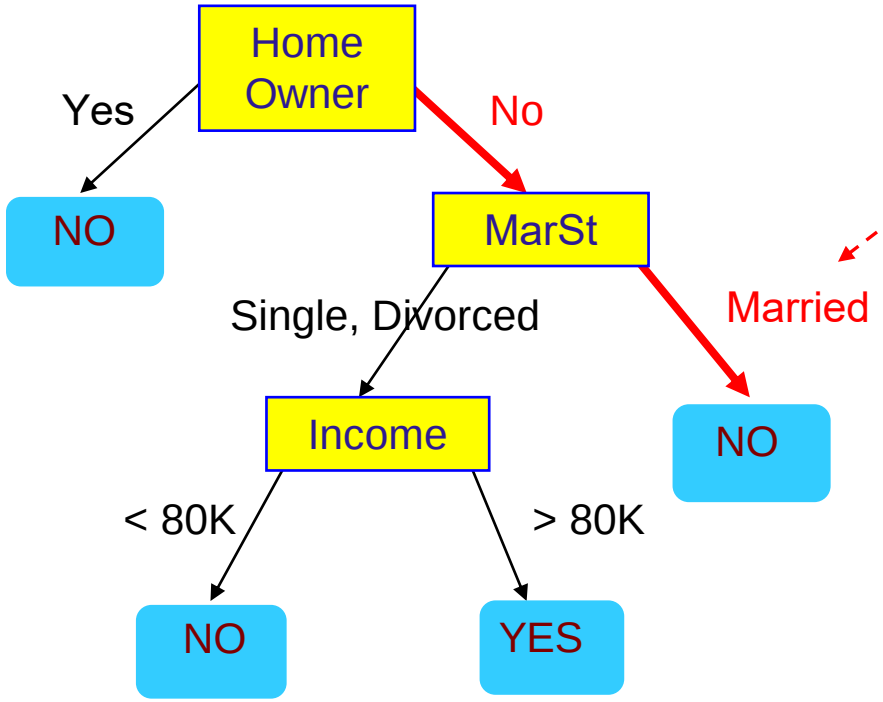
Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



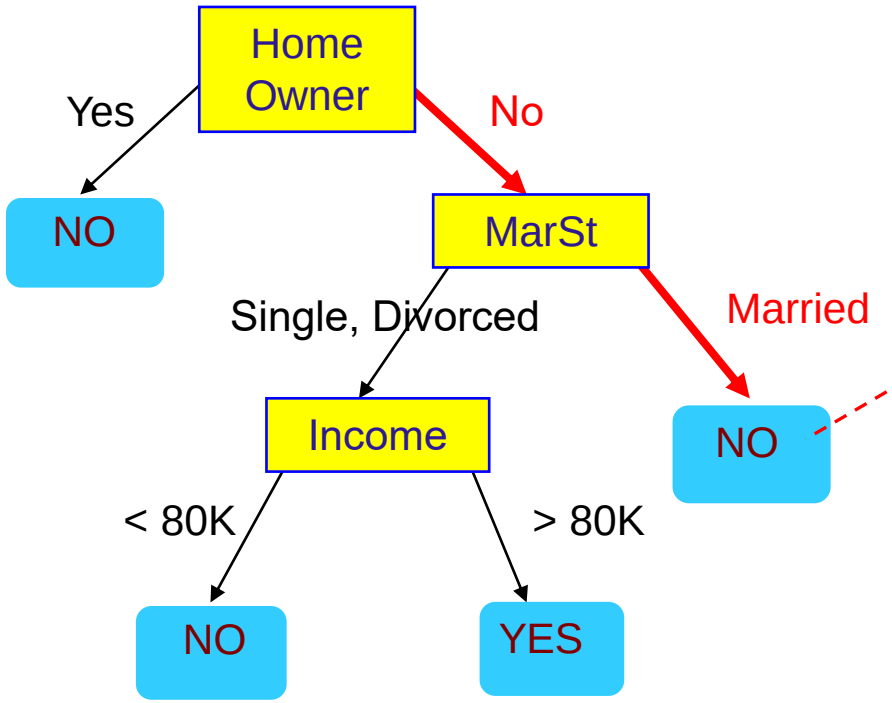
Apply Model to Test Data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?



Apply Model to Test Data



Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Assign Defaulted to "No"

Top-Down Induction of Decision Trees ID3 (Framework of basic decision tree)

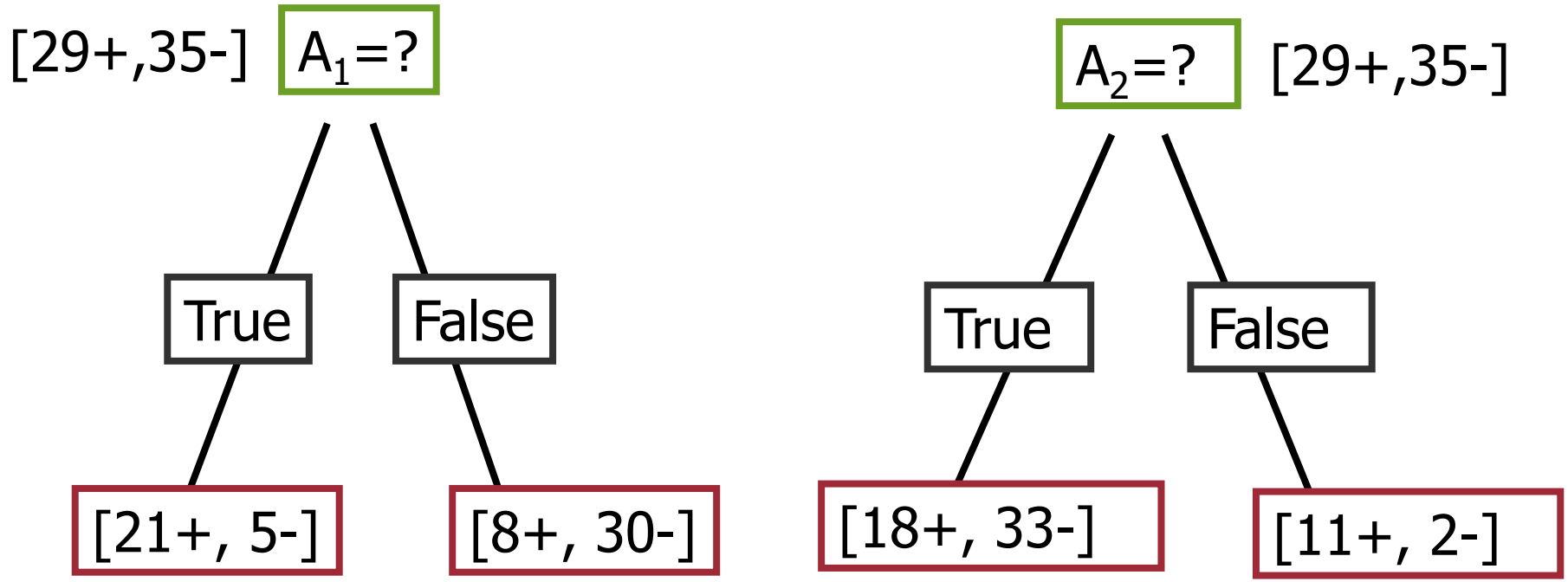
1. A ← the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A create new descendant
4. Sort training examples to leaf node according to the attribute value of the branch
5. If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

Choices

- When to stop
  - No more features
  - All the examples are classified the same
  - Too few examples to make an informative split

- Which test to split on
  - Split gives smallest error
  - With multi-valued features
    - Split on all the values or
    - Split values into half.

Which Attribute is ”best”?



Principled Criterion

- Selection of an attribute to test at each node-
  - choosing the most useful attribute for classifying examples.

- Information gain
  - Measures how well a given attribute separates the training examples according to their target classification
  - This measure is used to select among the candidate attributes at each step while growing the tree
  - Gain is measure of how much we can reduce uncertainty (values lies between 0 and 1)

Entropy

- A measure of
  - Uncertainty
  - Purity
  - Information content
- Information theory: optimal length code assigns ($-log_2 p$) bits to message having probability p
- S is a sample of training examples
  - p+ is the proportion of positive examples in S
  - p- is the proportion of negative examples in S
- Entropy of S: average optimal number of bits to encode information about certainty/ uncertainty about S

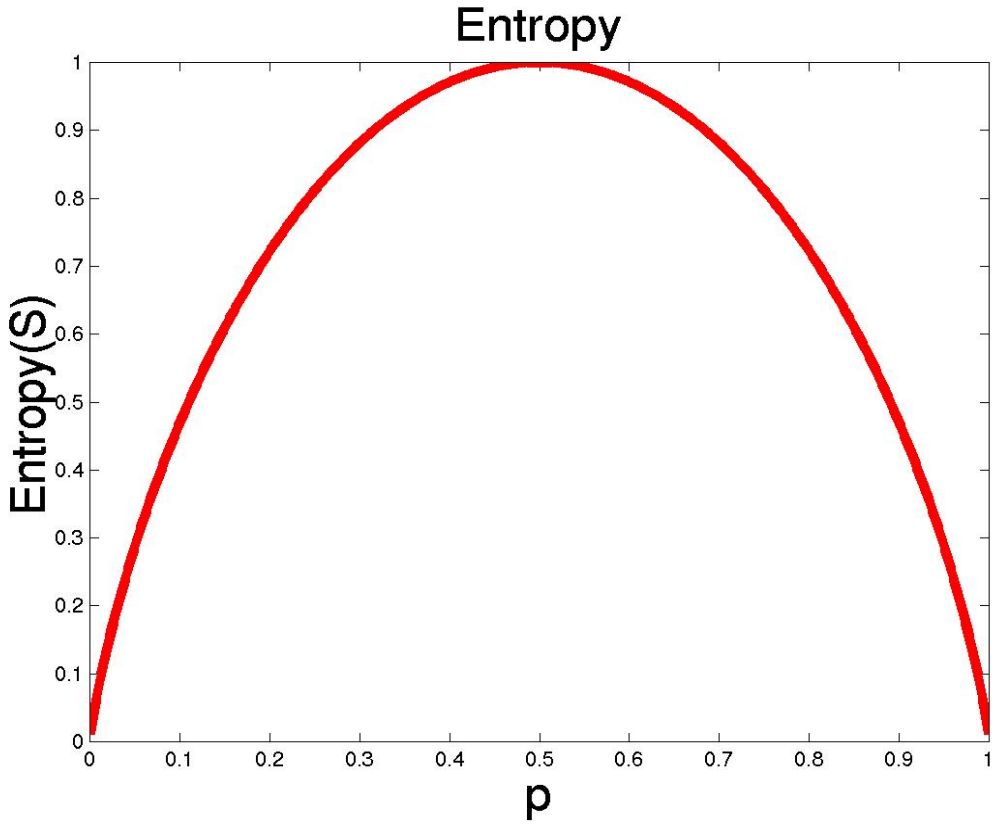
$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Entropy

- S is a sample of training examples
  - p+ is the proportion of positive examples
  - p- is the proportion of negative examples
- Entropy mesaures the inpurity of S

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

- Entropy is 0 if the outcome is ‘certain’.
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).

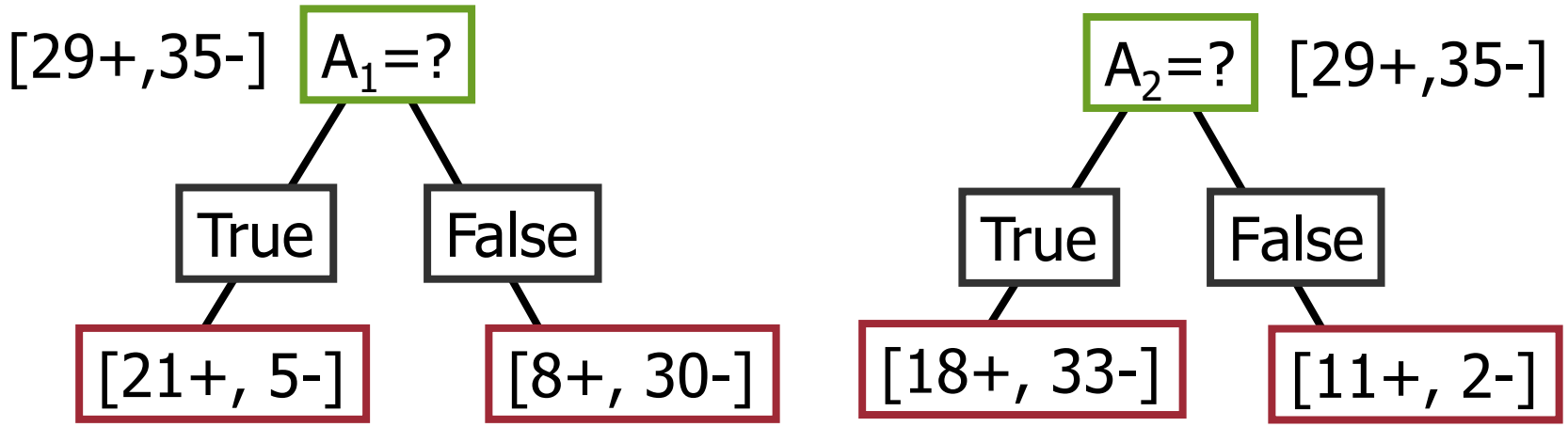


Information Gain

- Gain(S, A): expected reduction in entropy due to partitionaing S on attribute A

$$Gain(S, A)=Entropy(S) - \sum_{v \in values(A)} |S_v| / |S| \; Entropy(S_v)$$

$$Entropy([29+, 35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64$$
$$= 0.99$$



Information Gain

Entropy([21+, 5-]) = 0.71

Entropy([8+, 30-]) = 0.74

Gain(S, A_1)=Entropy(S)

-26/64*Entropy([21+,5-])

-38/64*Entropy([8+,30-])

=0.27

Entropy([18+, 33-]) = 0.94

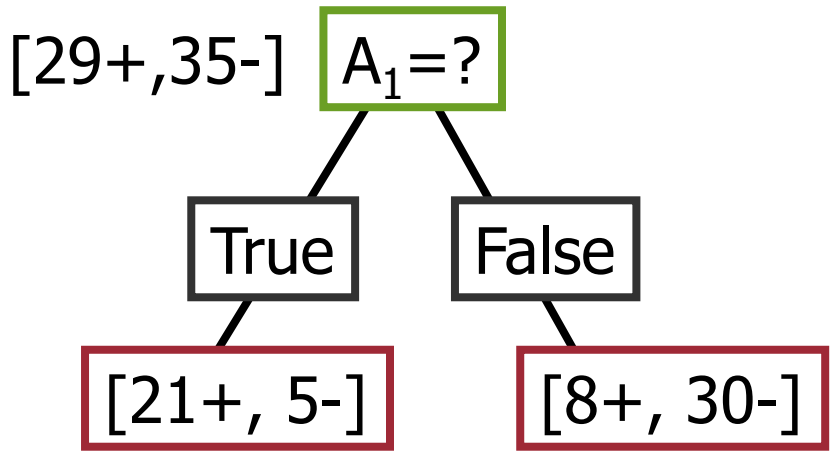
Entropy([8+, 30-]) = 0.62

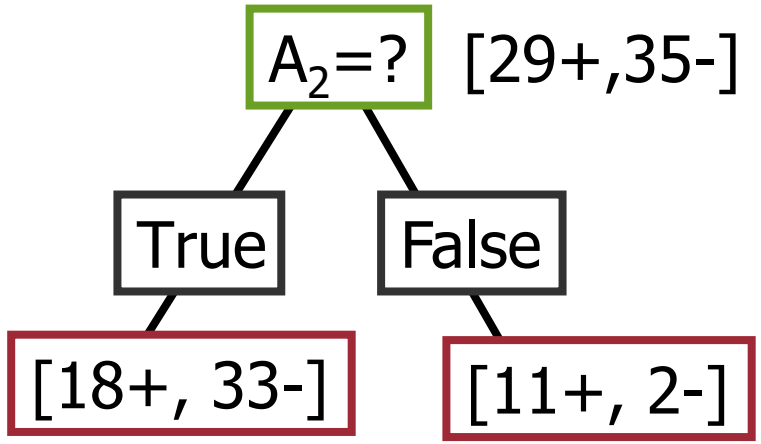
Gain(S, A_2)=Entropy(S)

-51/64*Entropy([18+,33-])

-13/64*Entropy([11+,2-])

=0.12

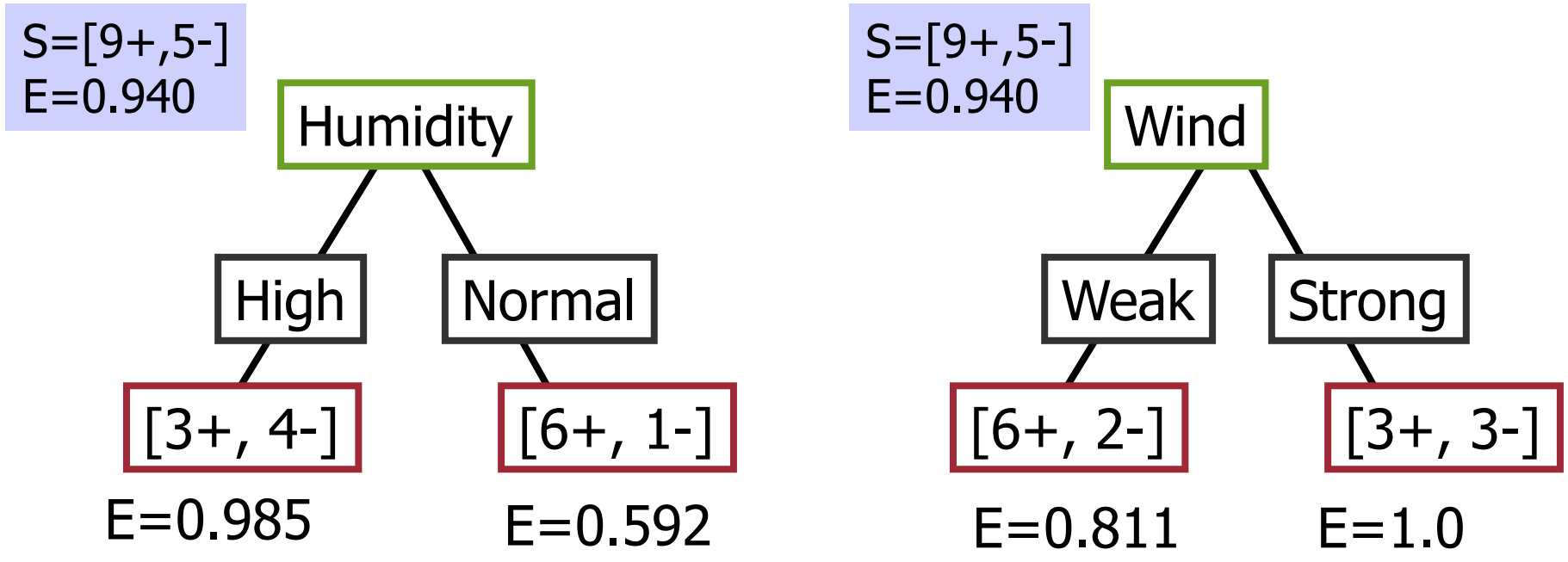




Training Examples

Day	Outlook	Temp.	Humidity	Wind	EnjoySport
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Selecting the Next Attribute

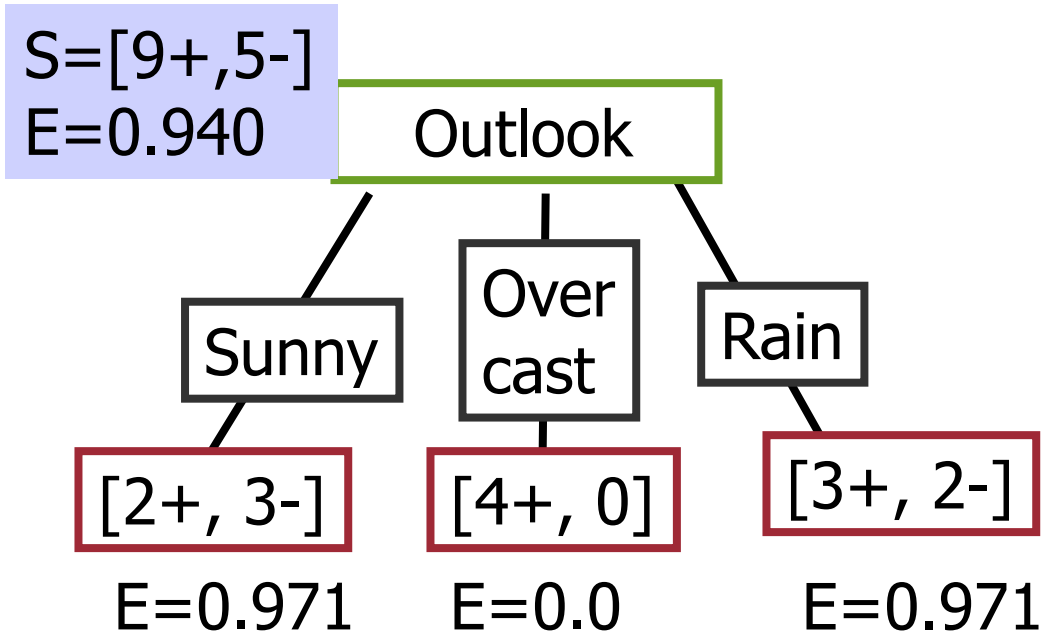


Gain(S, Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

Gain(S, Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Humidity provides greater info. gain than Wind, w.r.t. target classification

Selecting the Next Attribute



Gain(S,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247

Temp ?

Gain(S, Temp)=0.029

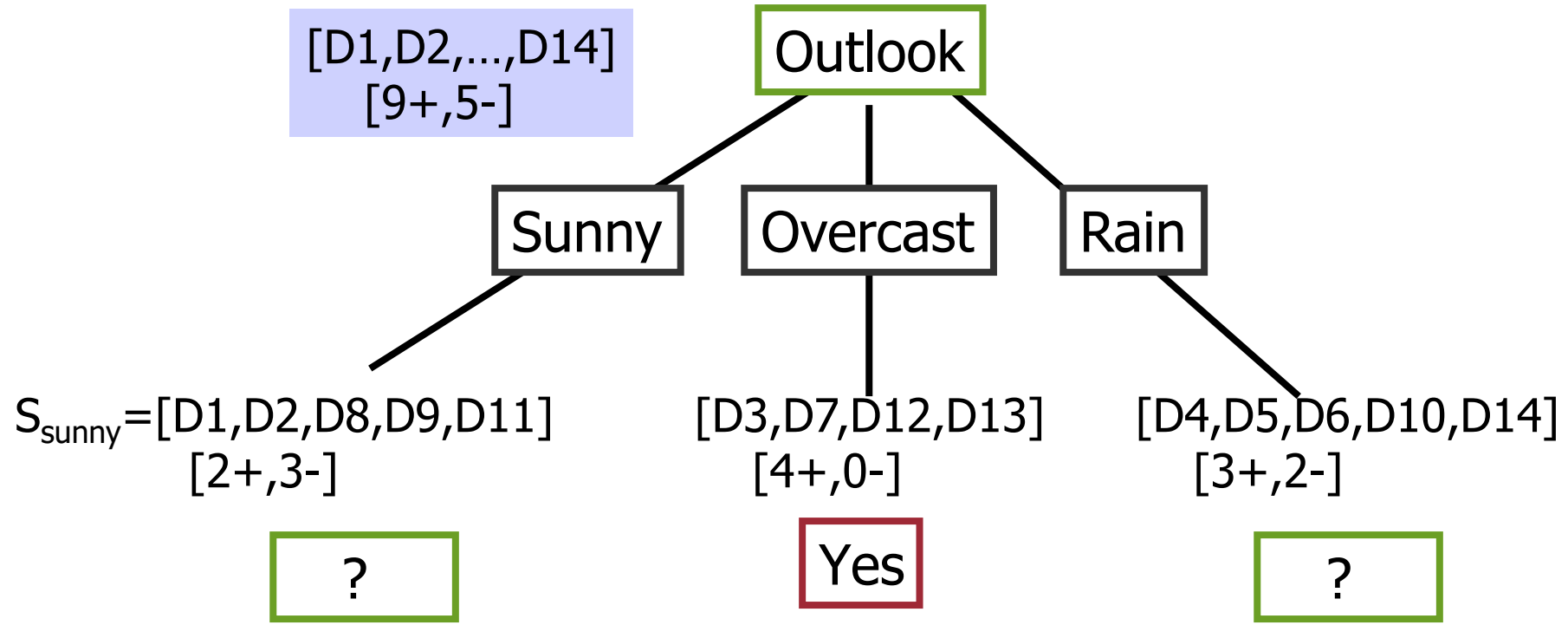
Selecting the Next Attribute

- The information gain values for the 4 attributes are:
  - Gain(S, Humidity)=0.151
  - Gain(S, Wind)=0.048
  - Gain(S,Outlook)=0.247
  - Gain(S, Temp)=0.029

where S denote the collection of training examples

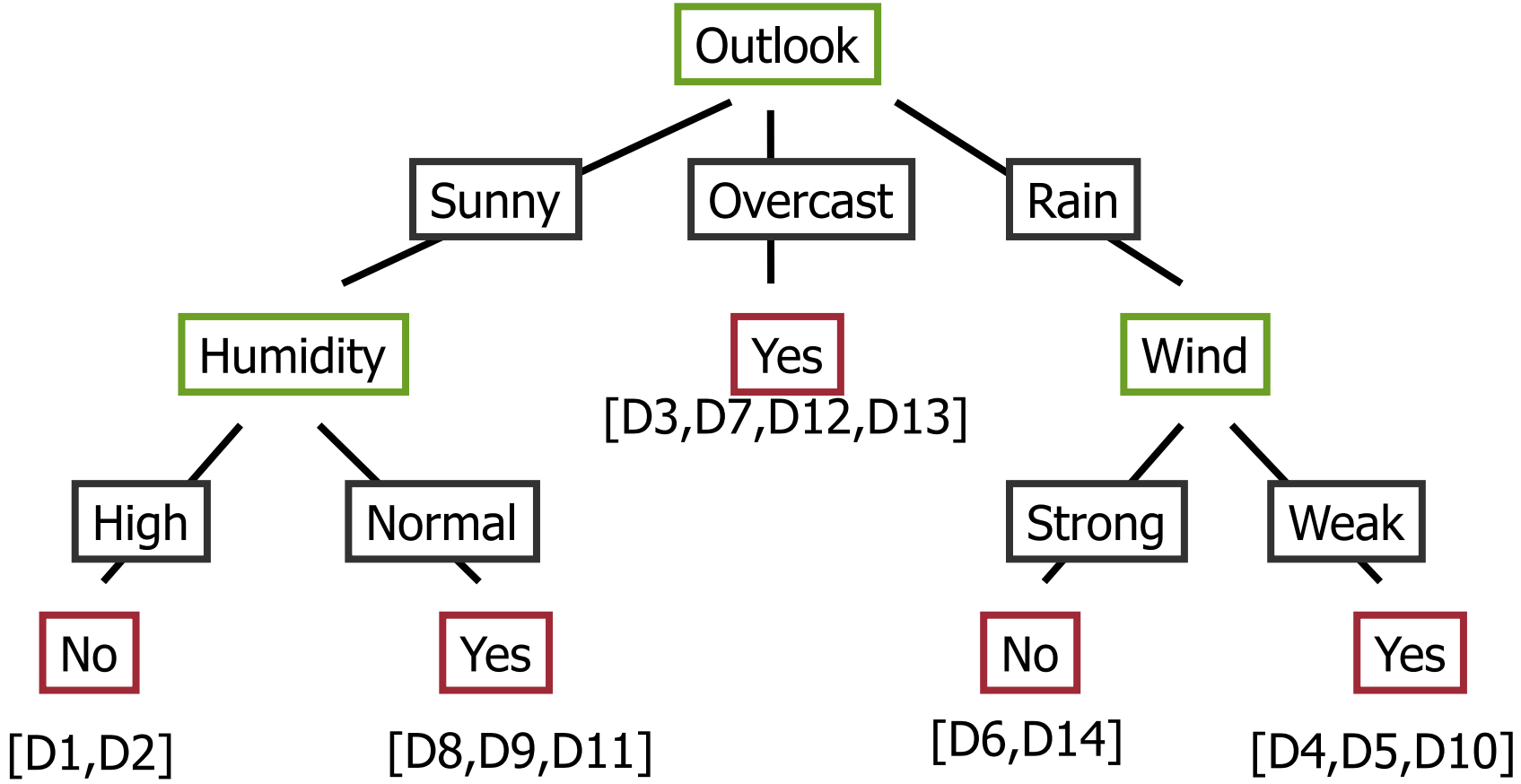
Note: $0\log_2 0=0$

ID3 Algorithm



Gain(S_{sunny} , Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970
Gain(S_{sunny} , Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570
Gain(S_{sunny} , Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019

ID3 Algorithm



Training Examples

Day	Outlook	Temp.	Humidity	Wind	EnjoySport
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Splitting Rule: GINI Index

- GINI Index
  - Measure of node impurity

$$GINI_{node}(Node) = 1 - \sum_{c \in classes} [p(c)]^2$$

$$GINI_{split}(A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GINI(N_v)$$

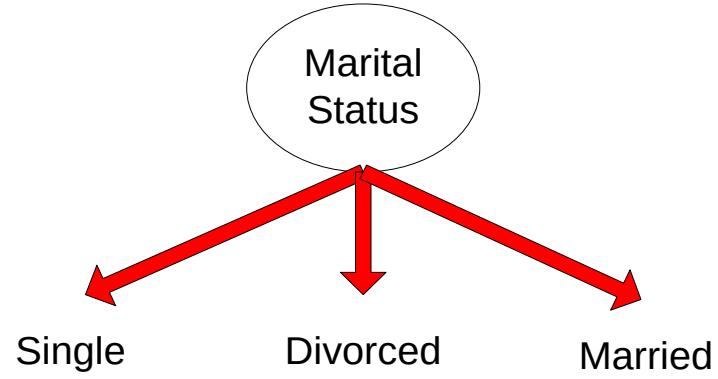
Methods for Expressing Test Conditions

- Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous

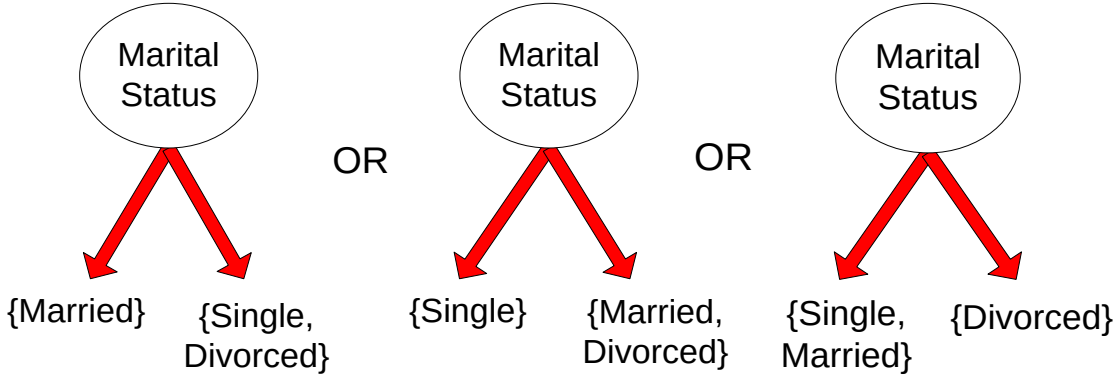
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Test Condition for Nominal Attributes

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

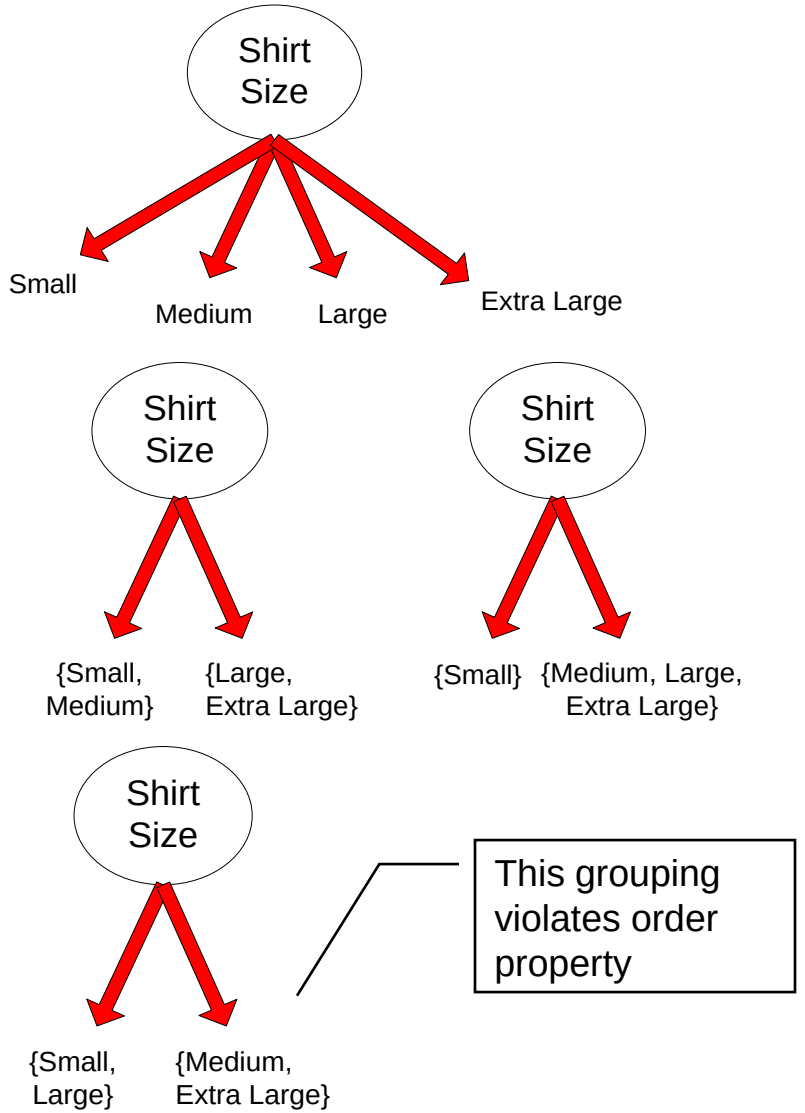


- Binary split:
  - Divides values into two subsets

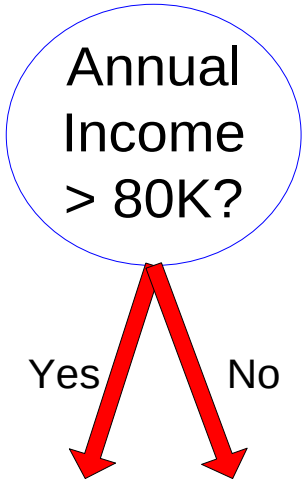


Test Condition for Ordinal Attributes

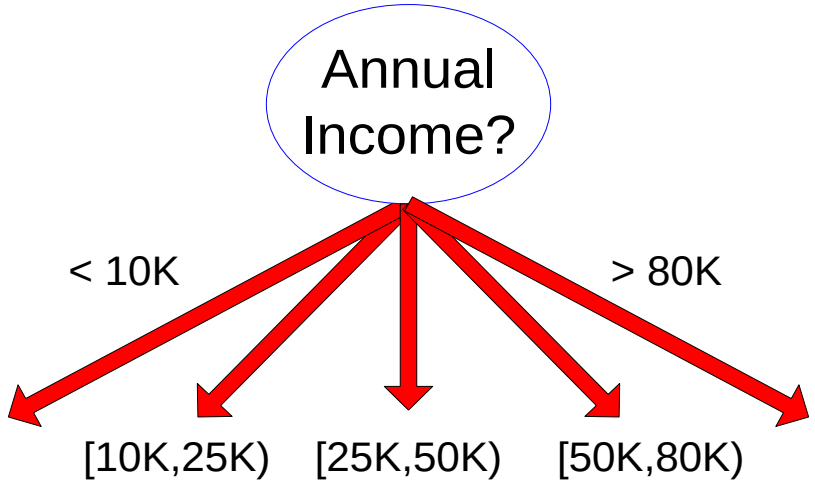
- Multi-way split:
  - Use as many partitions as distinct values
- Binary split:
  - Divides values into two subsets
  - Preserve order property among attribute values



Test Condition for Continuous Attributes



(i) Binary split



(ii) Multi-way split

Continuous Attributes – Binary Split

- For continuous attribute
  - Partition the continuous value of attribute A into a discrete set of intervals
  - Create a new Boolean attributes A_c, looking for a threshold c,

$$A_c = \begin{cases} true & if\ A_c < c \\ false & otherwise \end{cases}$$

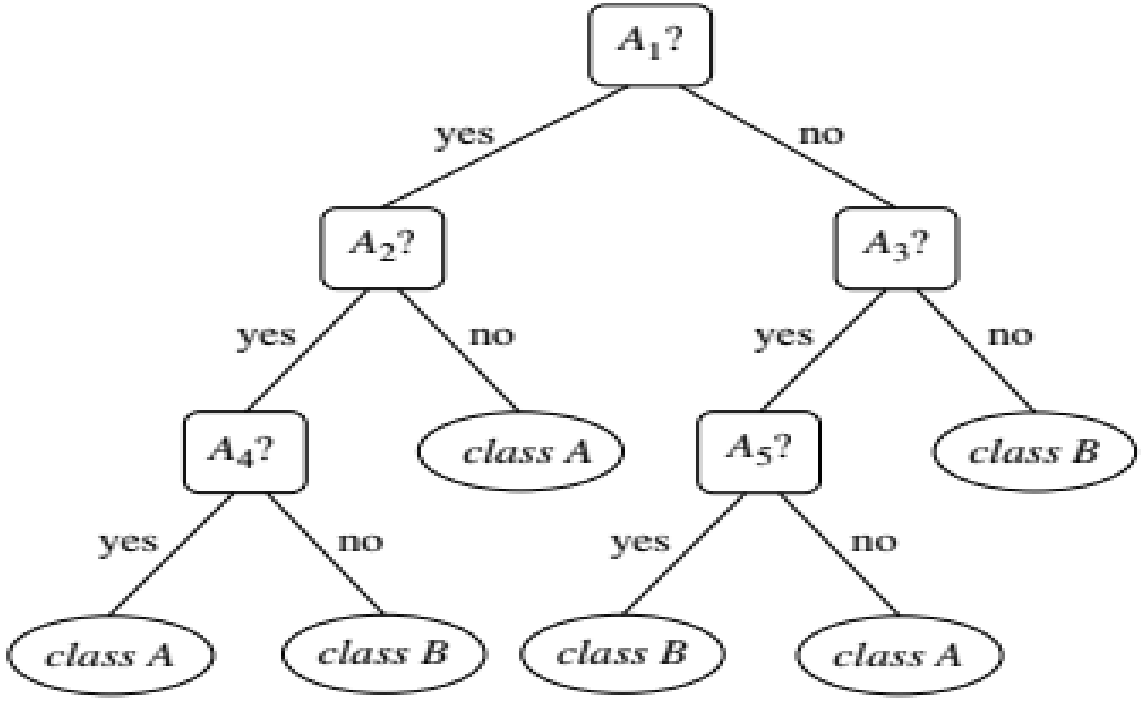
How to choose c?

- Consider all possible splits and finds the best cut

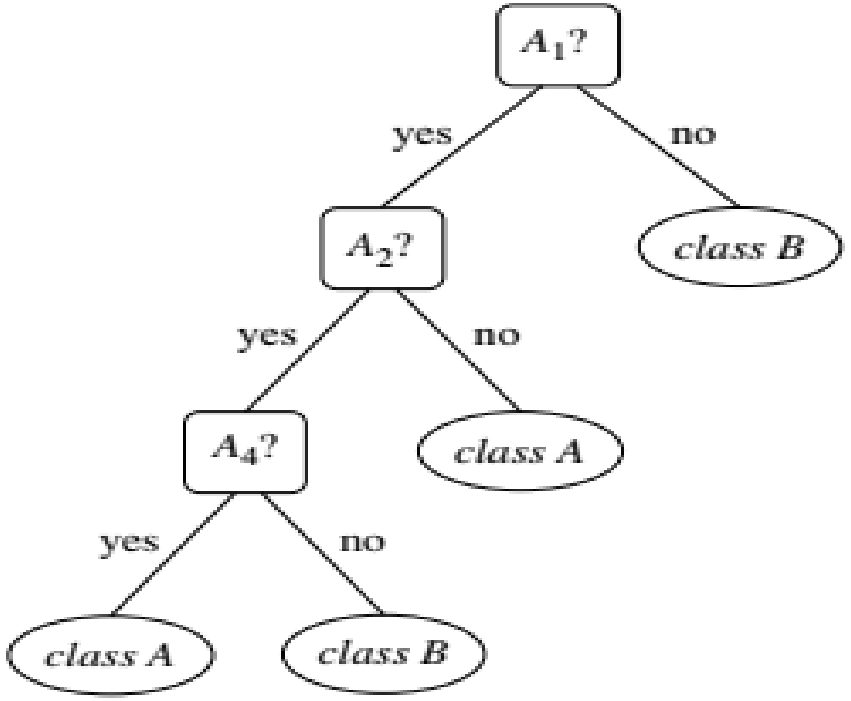
Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early*—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Example of Tree Pruning



Unpruned decision tree



Pruned Decision tree