Data Science - Lecture 10

Practical Issues of Classification

Dr. Faisal Kamiran

Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Data Fragmentation

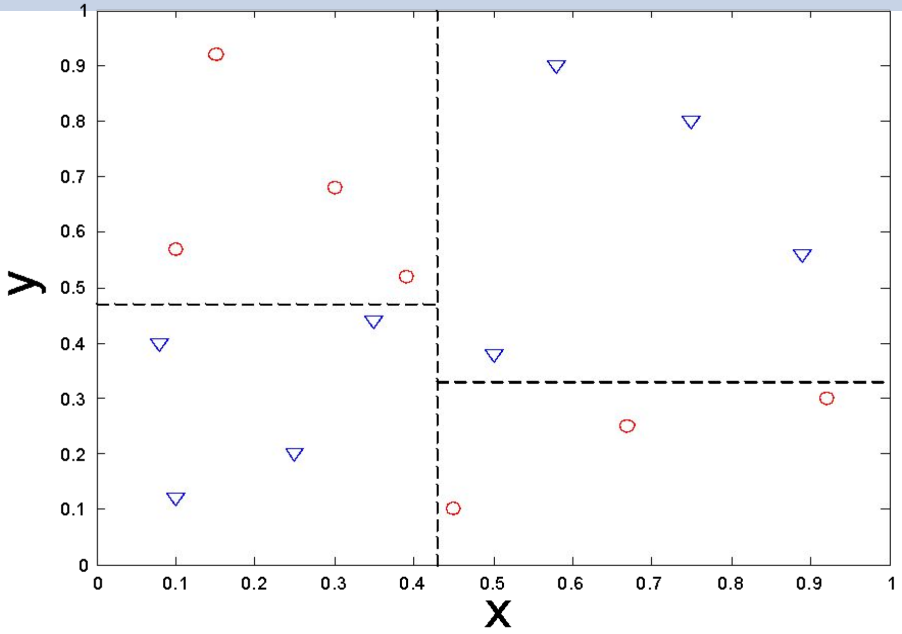
Practical Issues of Classification

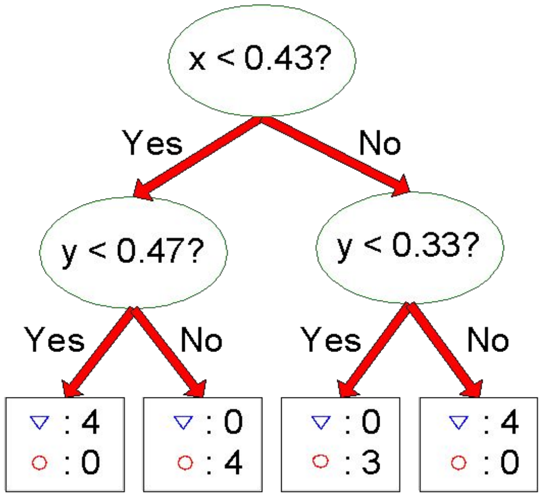
- Underfitting and Overfitting

- Missing Values

- Data Fragmentation

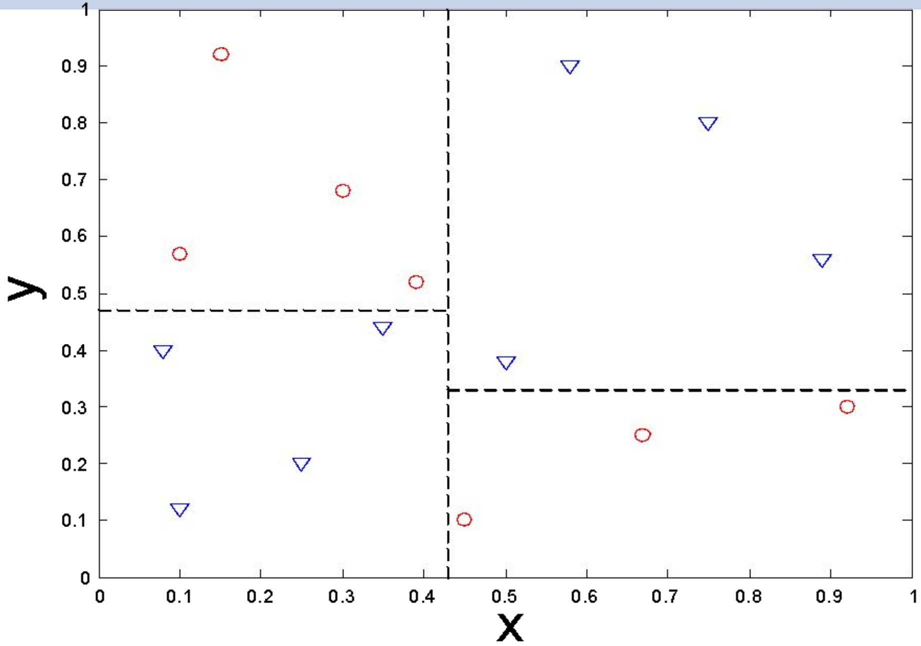
Decision Boundary

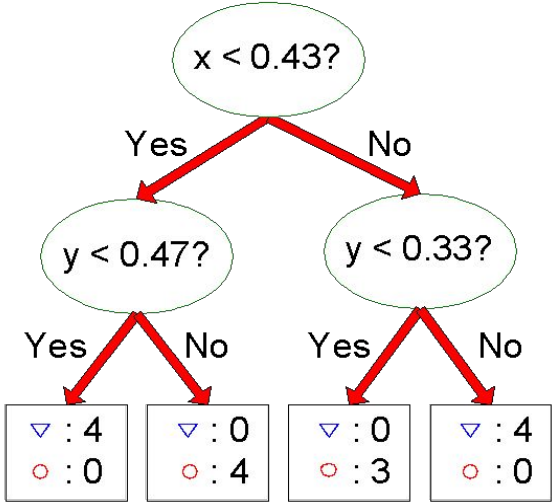




- **Border line between two neighboring regions of different classes is known as decision boundary**

Decision Boundary





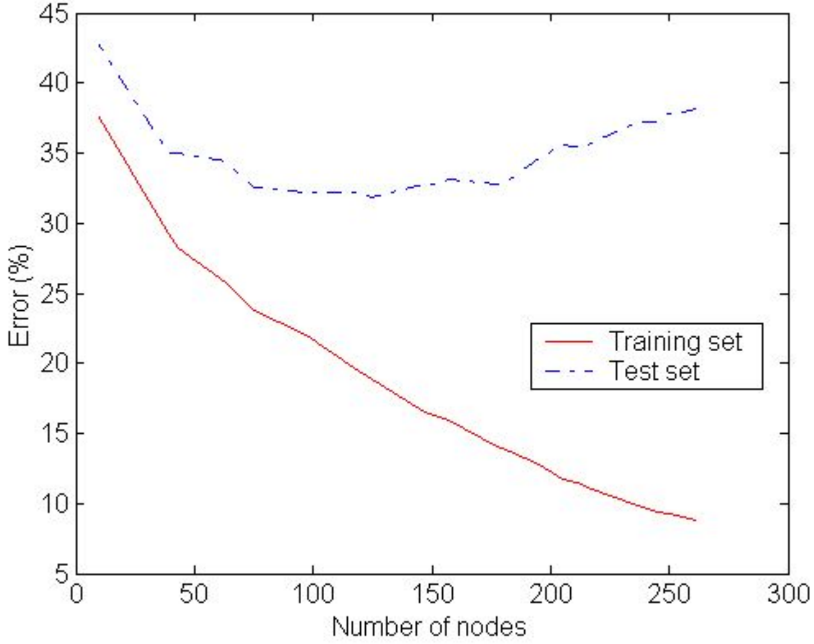
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

Overfitting and Underfitting

- **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

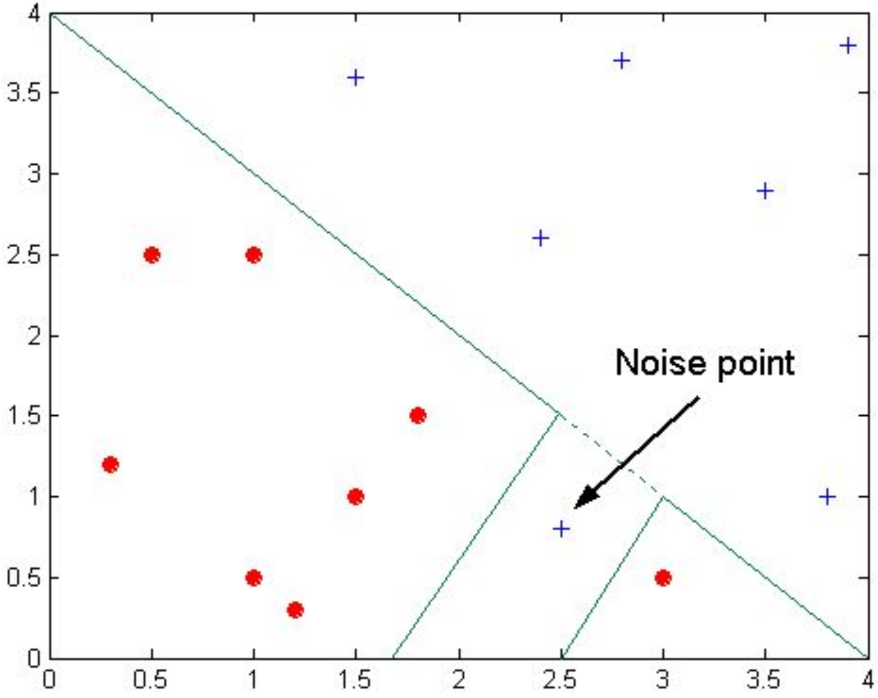
- **Underfitting:** when model is too simple, both training and test errors are large

Underfitting and Overfitting



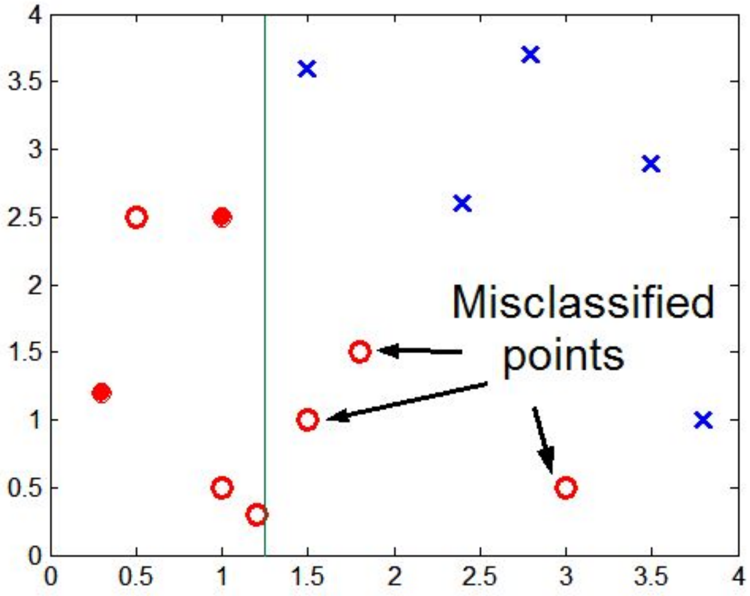
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



- 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

How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

How to Address Overfitting...

- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree

Estimating Generalization Errors

- Re-substitution errors: error on training (Σ e(t))
- Generalization errors: error on testing (Σ e'(t))

- Methods for estimating generalization errors:
  - Optimistic approach: e'(t) = e(t)

Estimating Generalization Errors

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

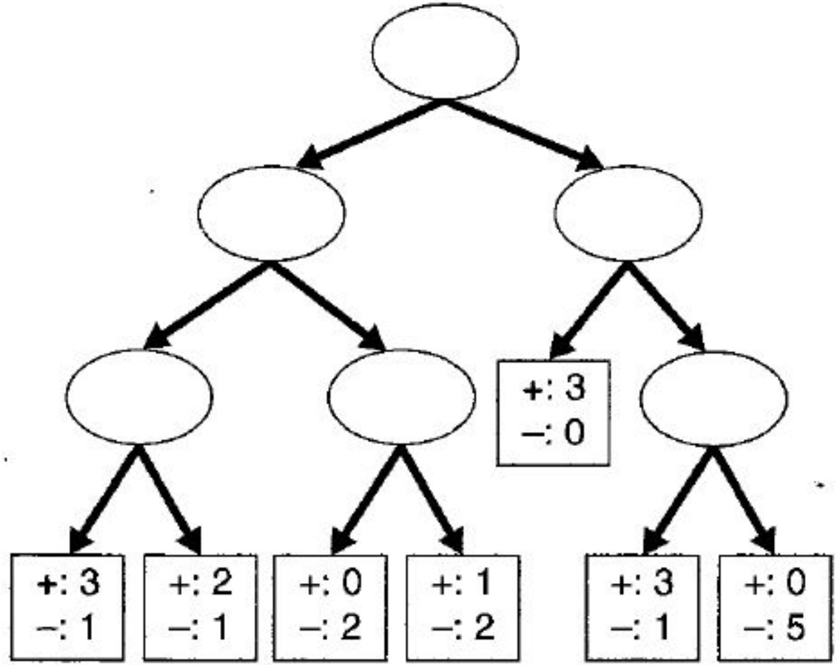
      Training error = 10/1000 = 1%

      Generalization error = $(10 + 30 \times 0.5)/1000 = 2.5\%$

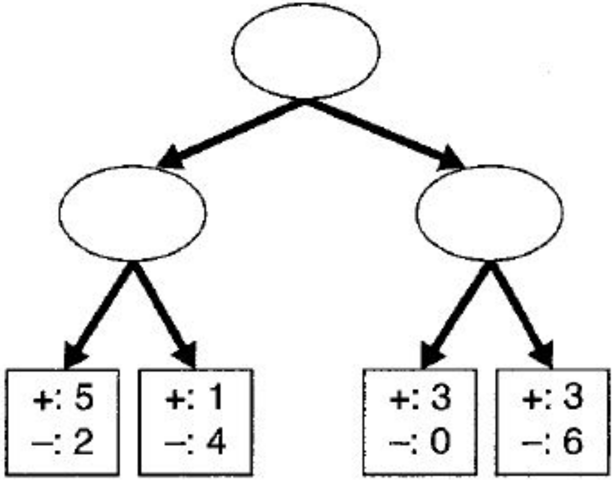
Estimating Generalization Errors

- Methods for estimating generalization errors:
    - Reduced error pruning (REP):
    - uses validation dataset to estimate generalization error
    - Validation set is part of training data used for preliminary validation of model during the learning process

Estimating Generalization Errors

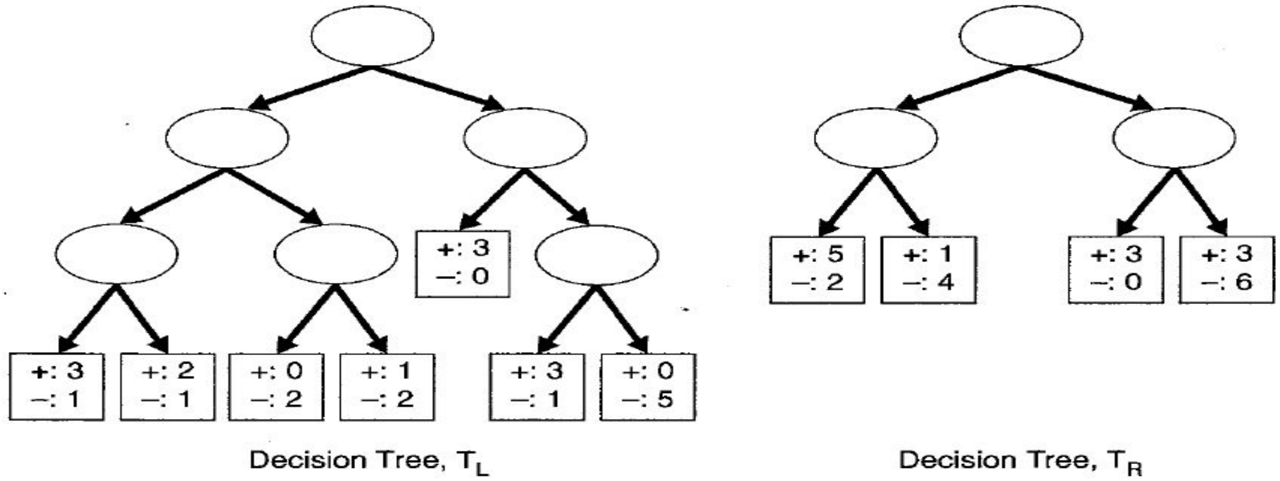


Decision Tree, T_L



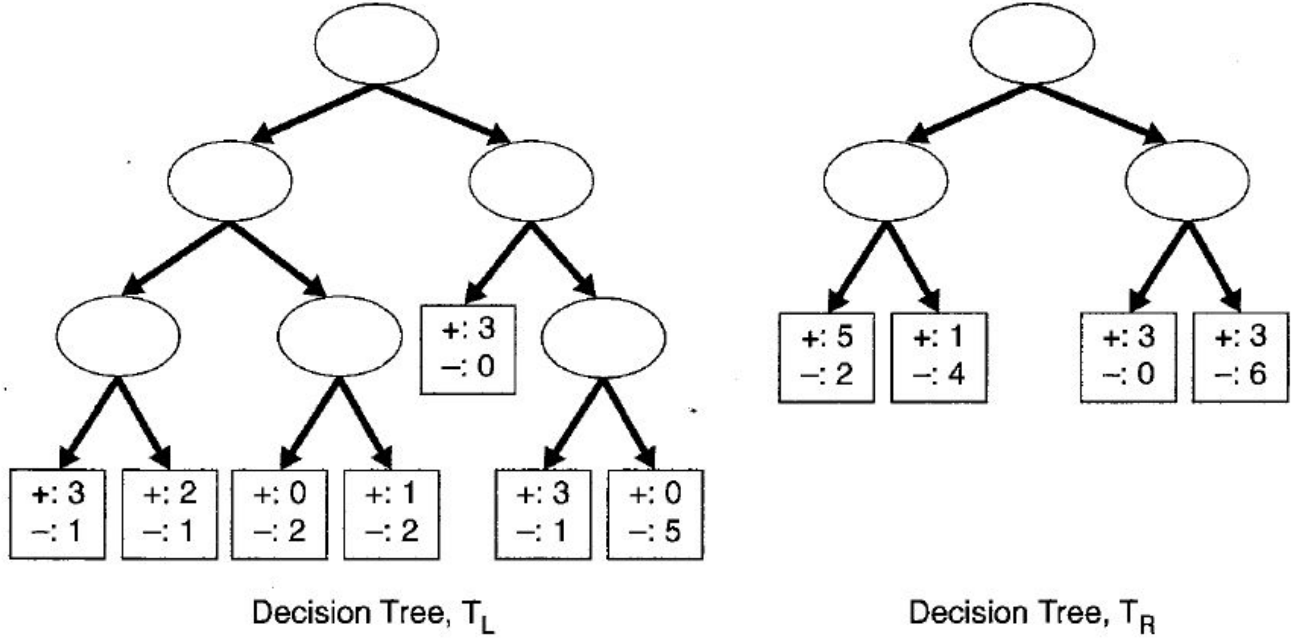
Decision Tree, T_R

Estimating Generalization Errors



$E`(T_L) = (4+7*0.5)/24 = 7.5 / 24 = 0.3125$

Estimating Generalization Errors



$E`(T_L) = (4+7*0.5)/24 = 7.5 / 24 = 0.3125$

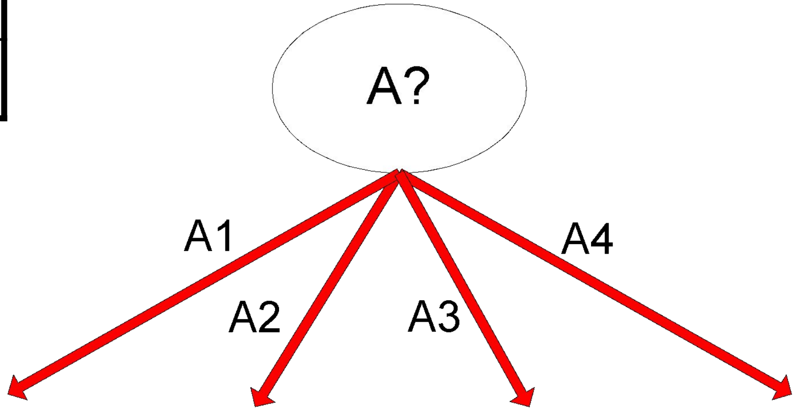
$E`(T_R) = (6+4*0.5)/24 = 8 / 24 = 0.3333$

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

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



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

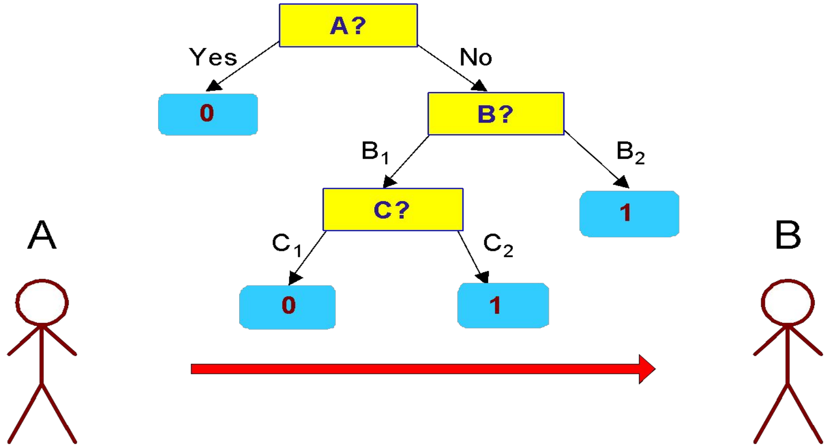
Class = Yes	5
Class = No	1

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

Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1

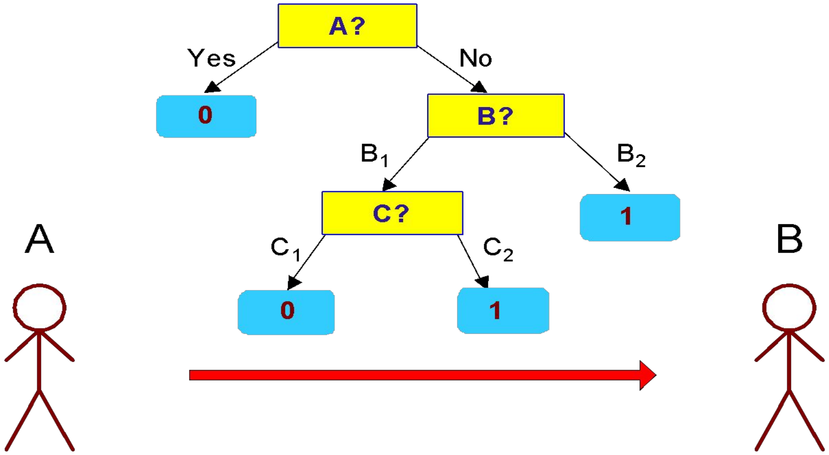


X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- Cost(Model,Data) = Cost(Data|Model) + Cost(Model)
  - Cost is the number of bits needed for encoding.
  - Search for the least costly 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(Data|Model) encodes the misclassification errors.
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding.

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
- Needs entire data to fit in memory.
- Unsuitable for Large Datasets.
- You can download the software from: http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz