# Practical Issues of Classification

Dr. Faisal Kamiran

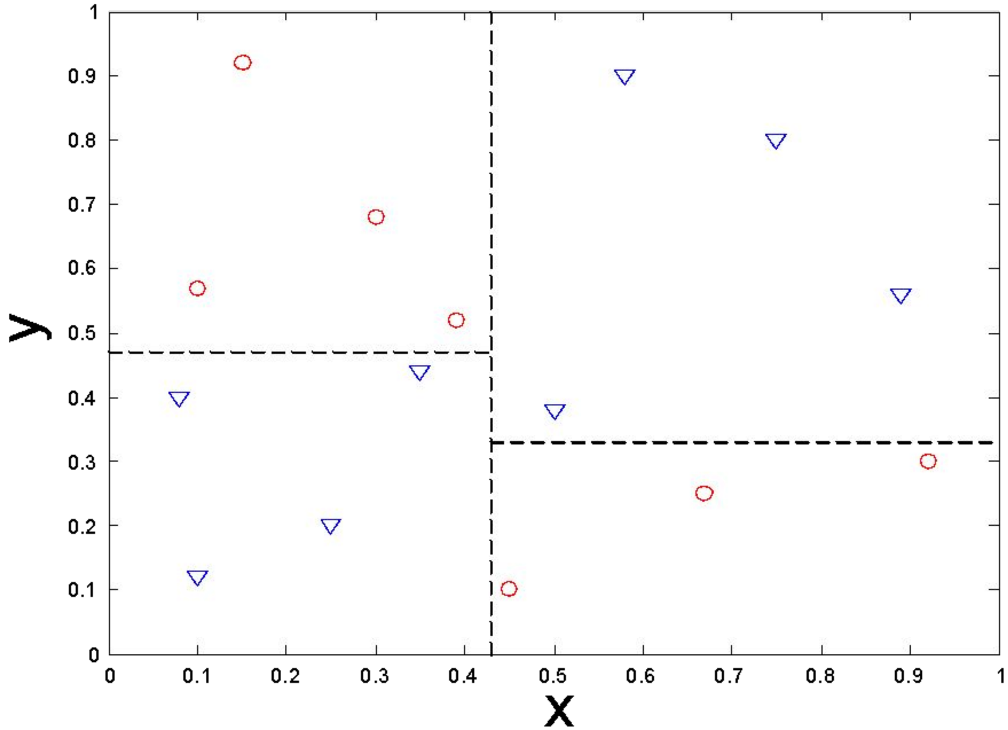
# Practical Issues of Classification

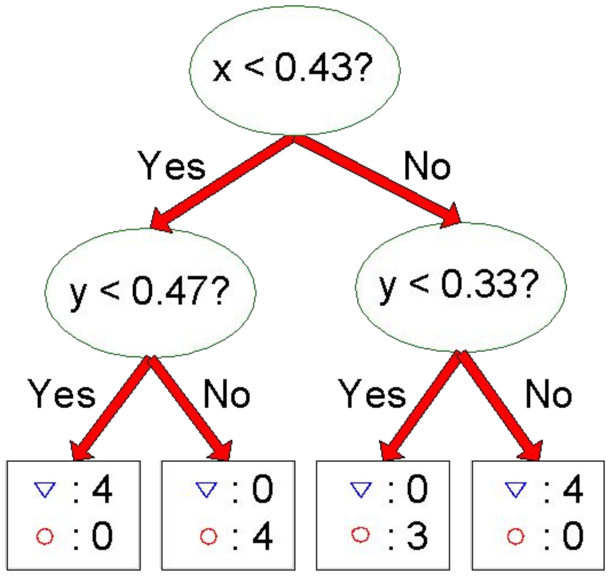
- Underfitting and Overfitting
- Missing Values
- Data Fragmentation

# Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Data Fragmentation

# Decision Boundary



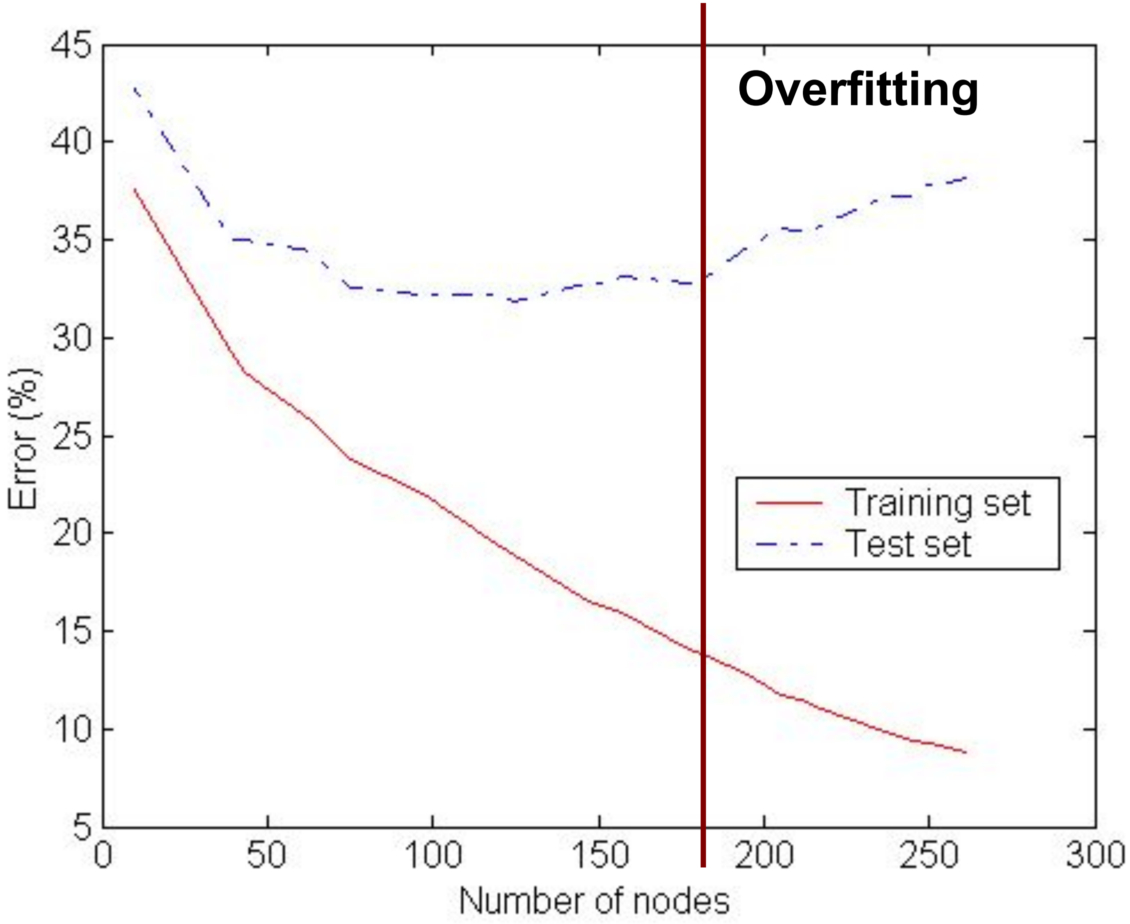

- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

# 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

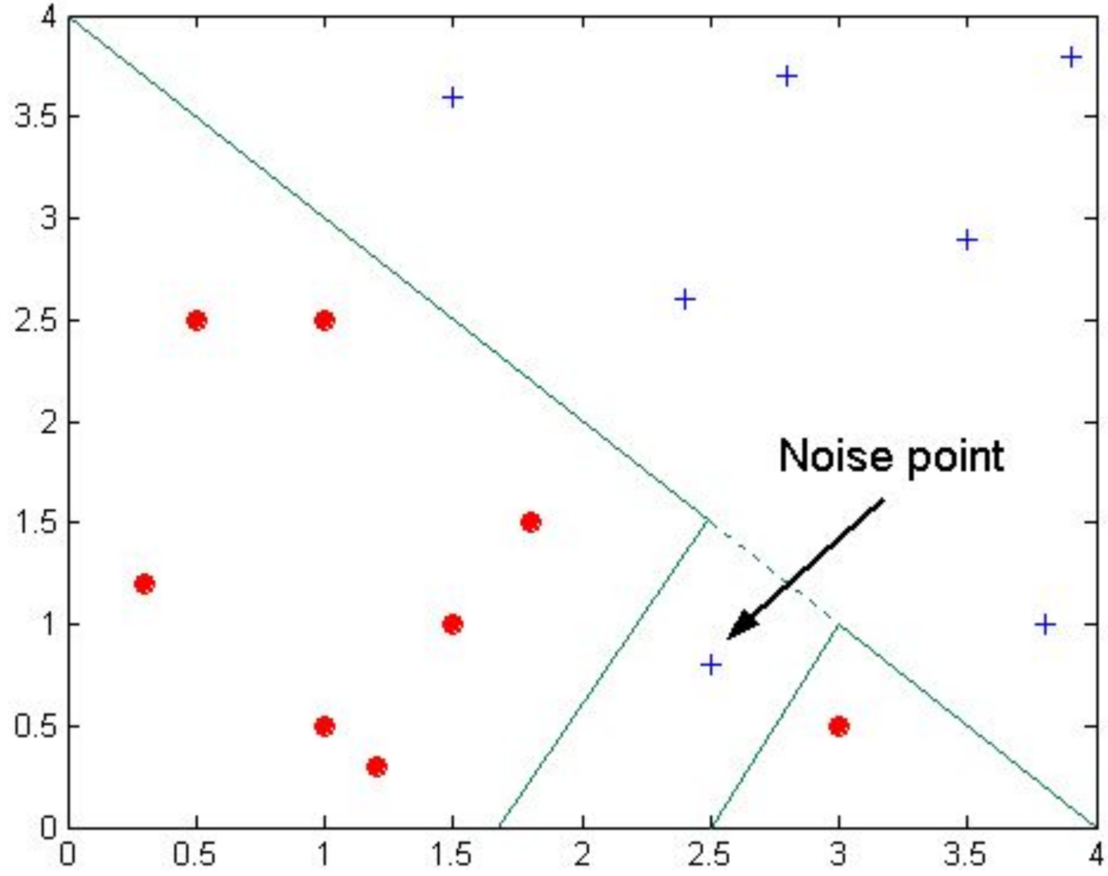
- **Underfitting:** when model is too simple, both training and test errors are large

# Underfitting and Overfitting



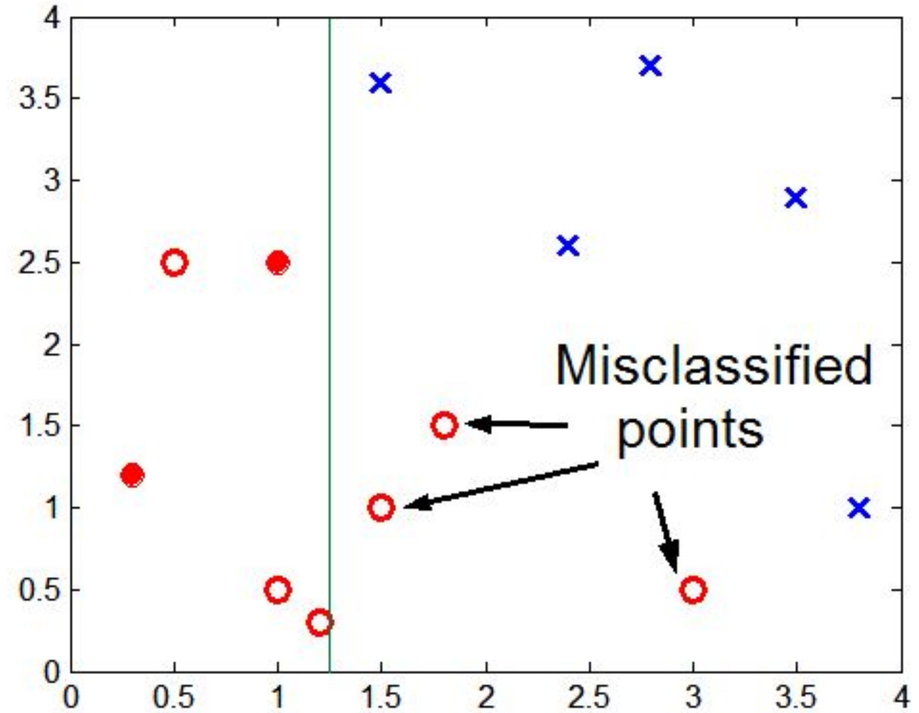

**Underfitting**: when model is too simple, both training and test errors are large

# 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

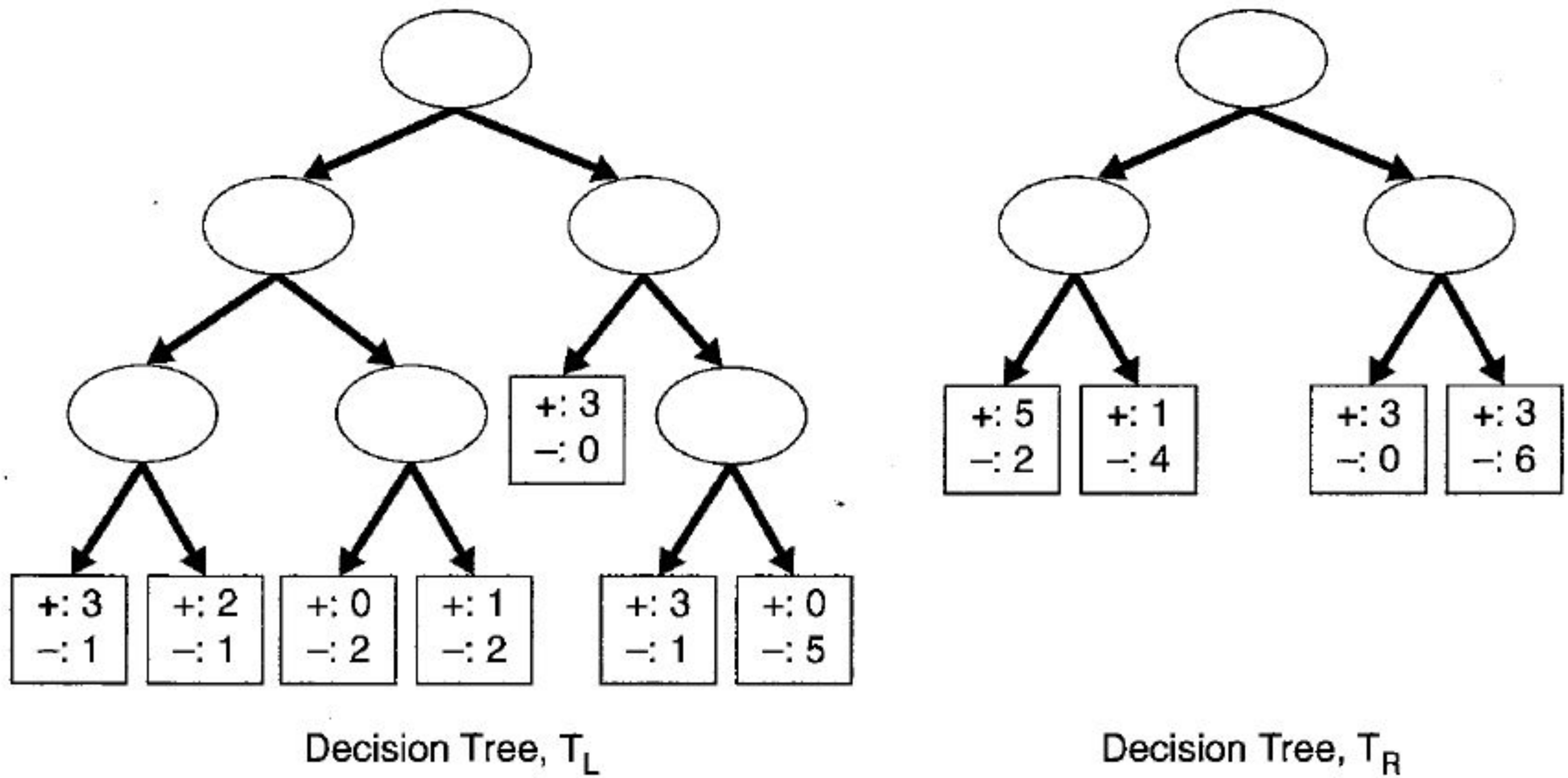
# Estimating Generalization Errors

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

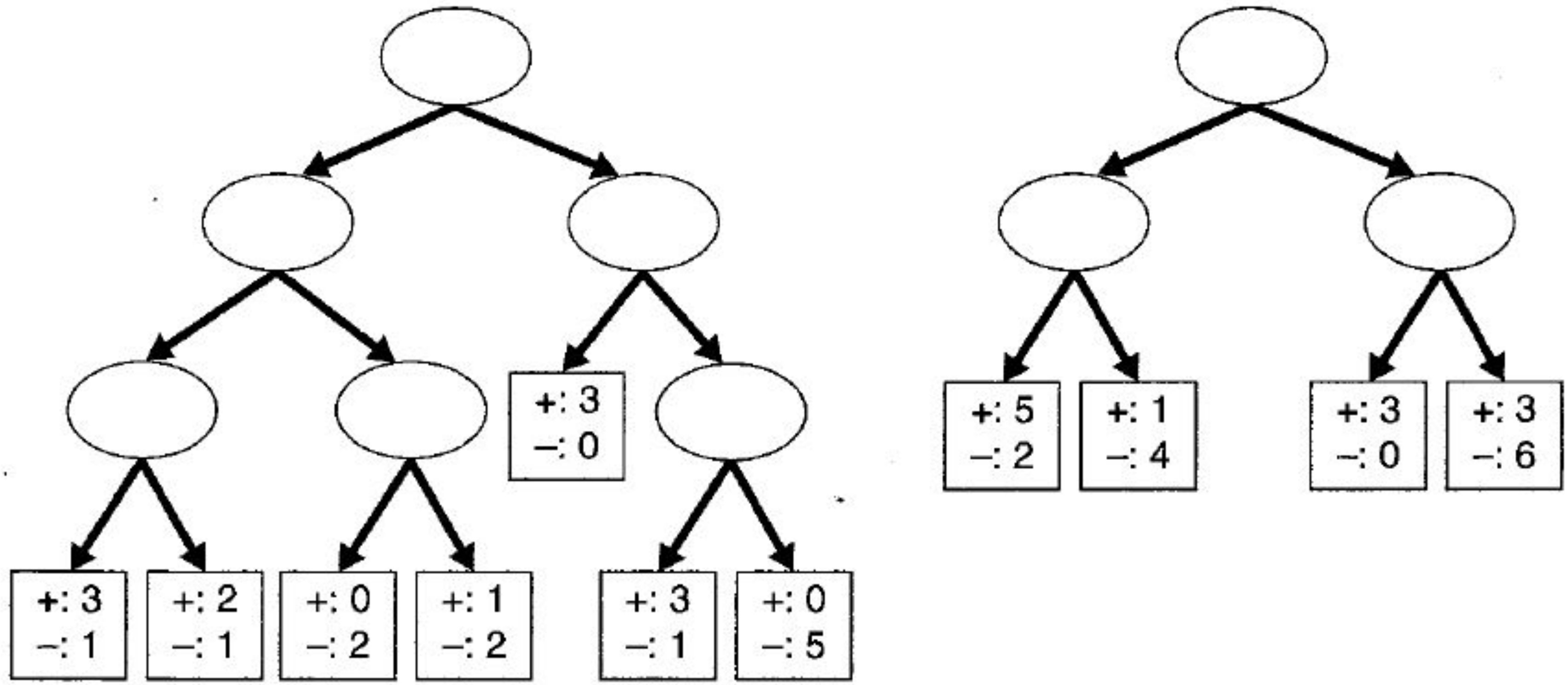
      Training error = 10/1000 = 1%

      Generalization error = (10 + 30×0.5)/1000 = 2.5%
    - Reduced error pruning (REP):
    - uses validation 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



# Estimating Generalization Errors

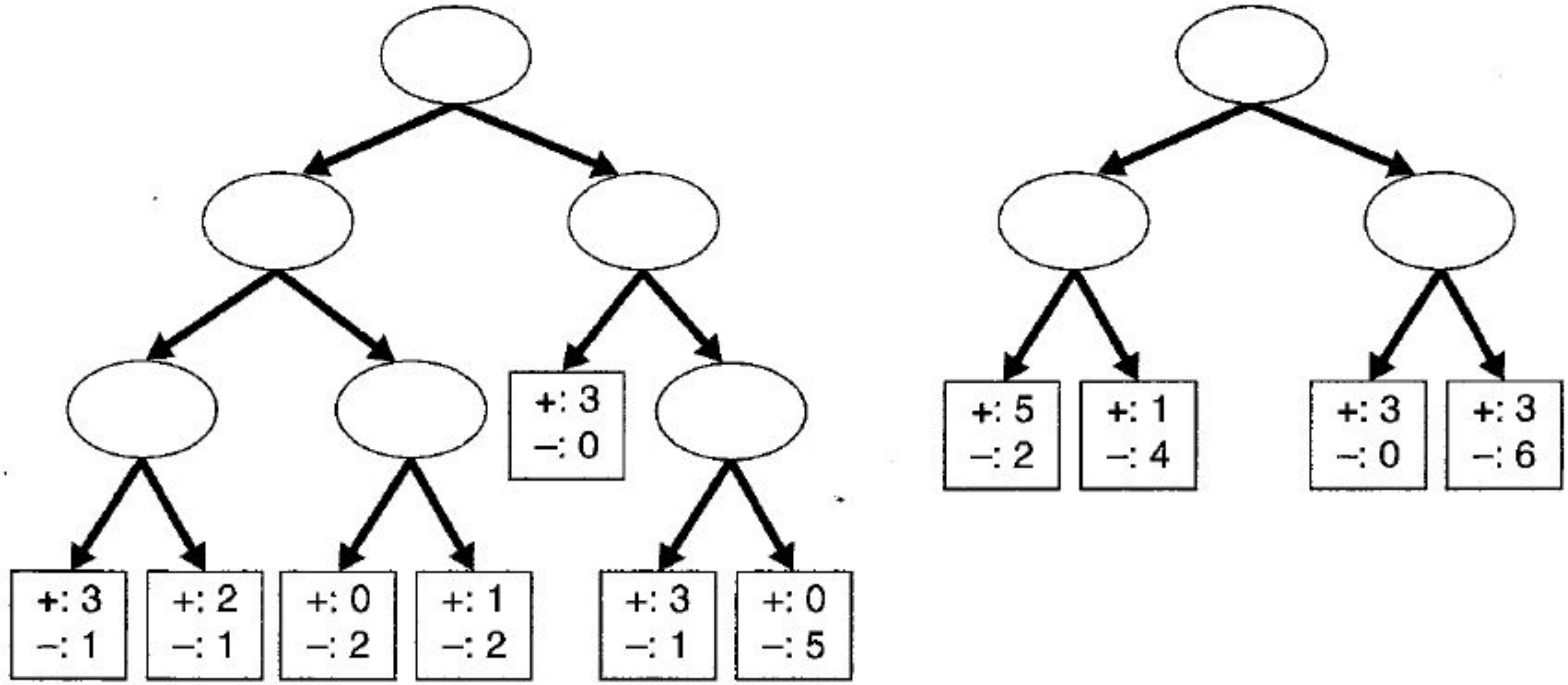



Decision Tree, $T_L$

Decision Tree, $T_R$

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

# Estimating Generalization Errors




Decision Tree, $T_L$ Decision Tree, $T_R$

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

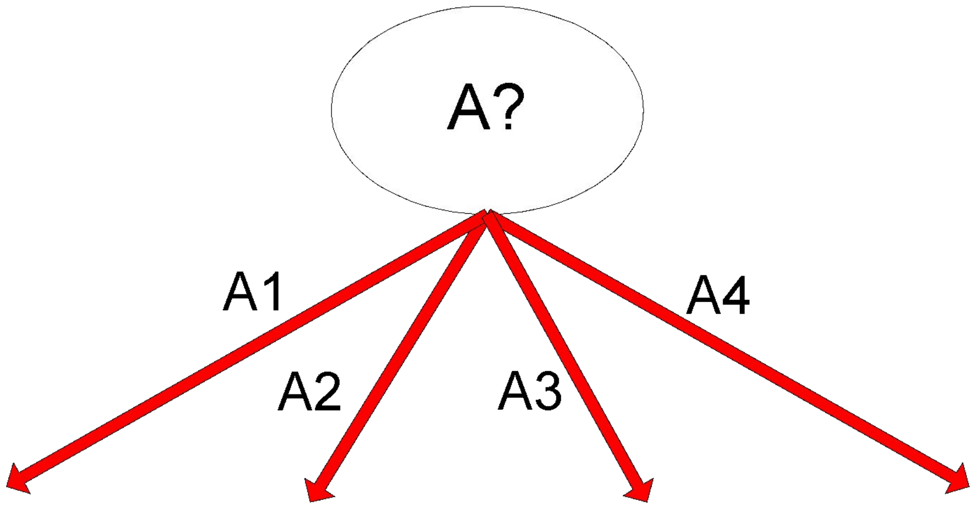
**$E`(T_R) = (6+4*0.5)/24 = 8 / 24 = 0.3333$**

# Example of Post-Pruning

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

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

| | |
|---|---|
| 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 |

# Example of Post-Pruning

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

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

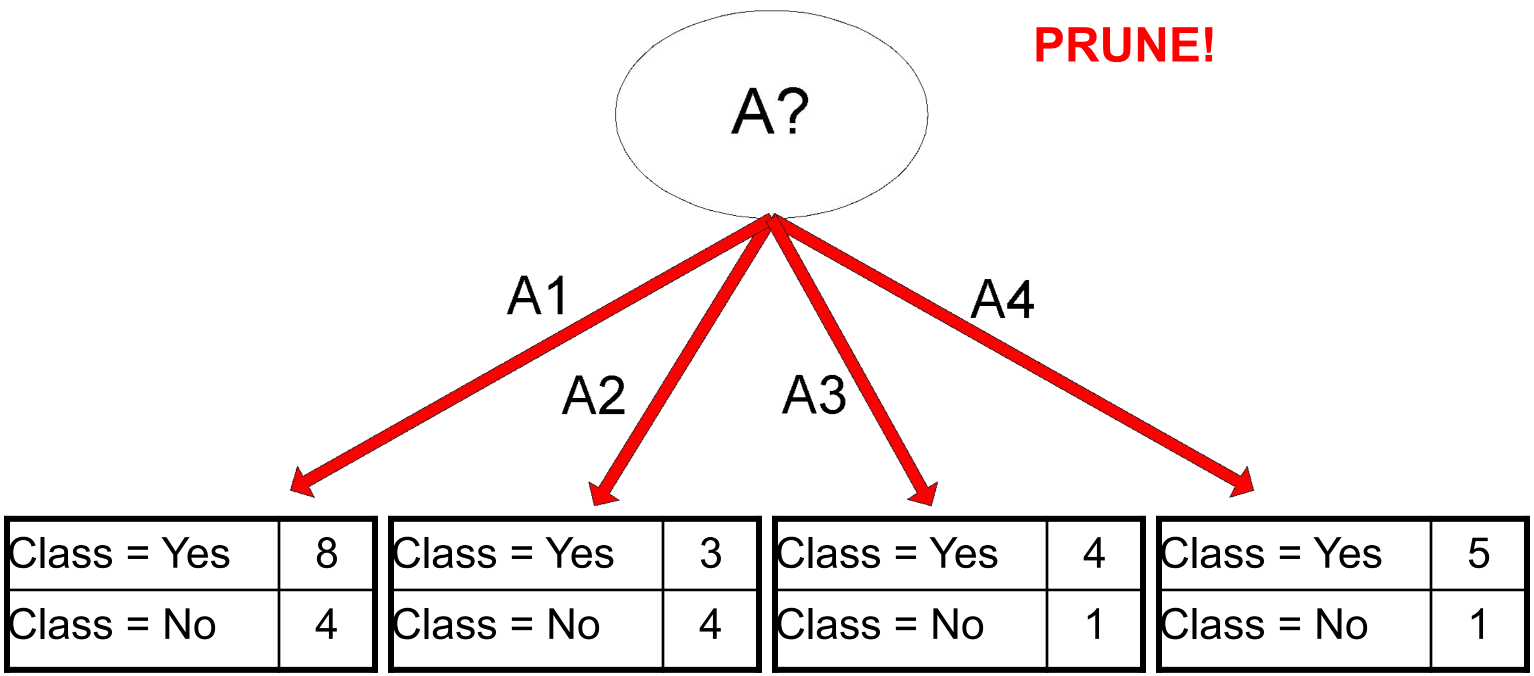
**Pessimistic error = (10 + 0.5)/30 = 10.5/30**

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

**Pessimistic error (After splitting)**

**= (9 + 4 × 0.5)/30 = 11/30**

**PRUNE!**

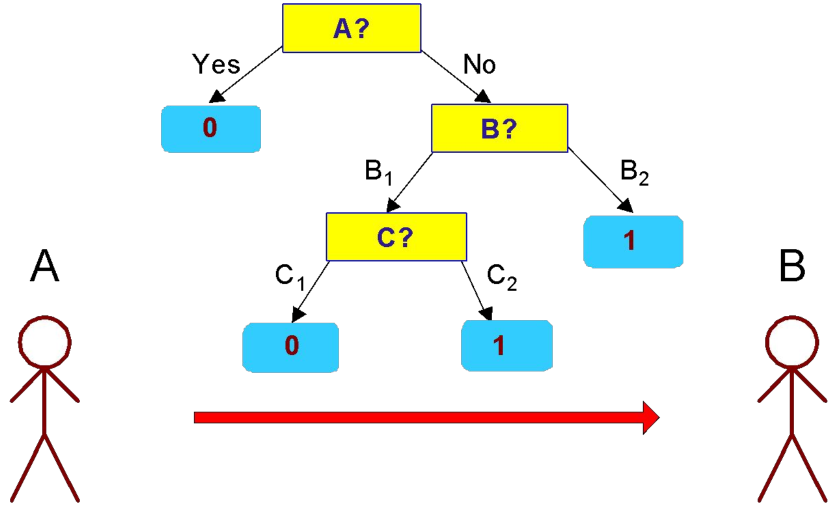


# Occam's Razor

- 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.
- 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:
  Machine Learning/Decision Trees/C4.5 Tutorial (uregina.ca)

# Evaluation of Classification Models

Dr. Faisal Kamiran

# 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 | | |
|---|---|---|---|
| | | Class=Yes | Class=No |
| ACTUAL CLASS | 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 | |
|---|---|---|
| | Class=Yes | Class=No |
| ACTUAL CLASS: Class=Yes | a (TP) | b (FN) |
| ACTUAL CLASS: Class=No | c (FP) | d (TN) |

- Most widely-used metric:

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

# Limitation of Accuracy

- 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

# Cost Matrix

| | PREDICTED CLASS | |
|---|---|---|
| | | |

| | C(i\|j) | **Class=Yes** | **Class=No** |
|---|---|---|---|
| ACTUAL CLASS | **Class=Yes** | C(Yes\|Yes) | C(No\|Yes) |
| | **Class=No** | C(Yes\|No) | C(No\|No) |

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

# 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-Sensitive Measures

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

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

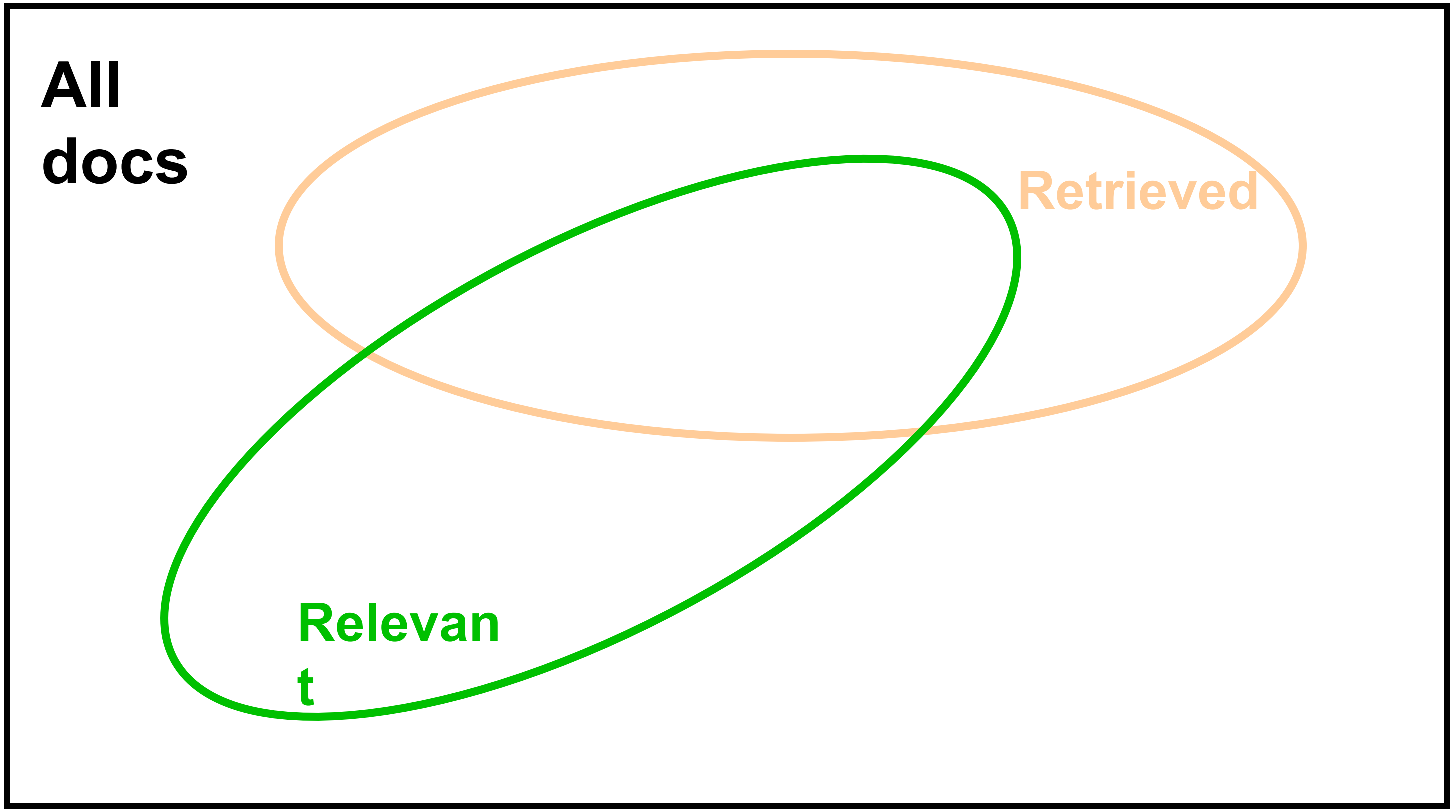
$$\text{F-measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

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

# Precision vs. Recall

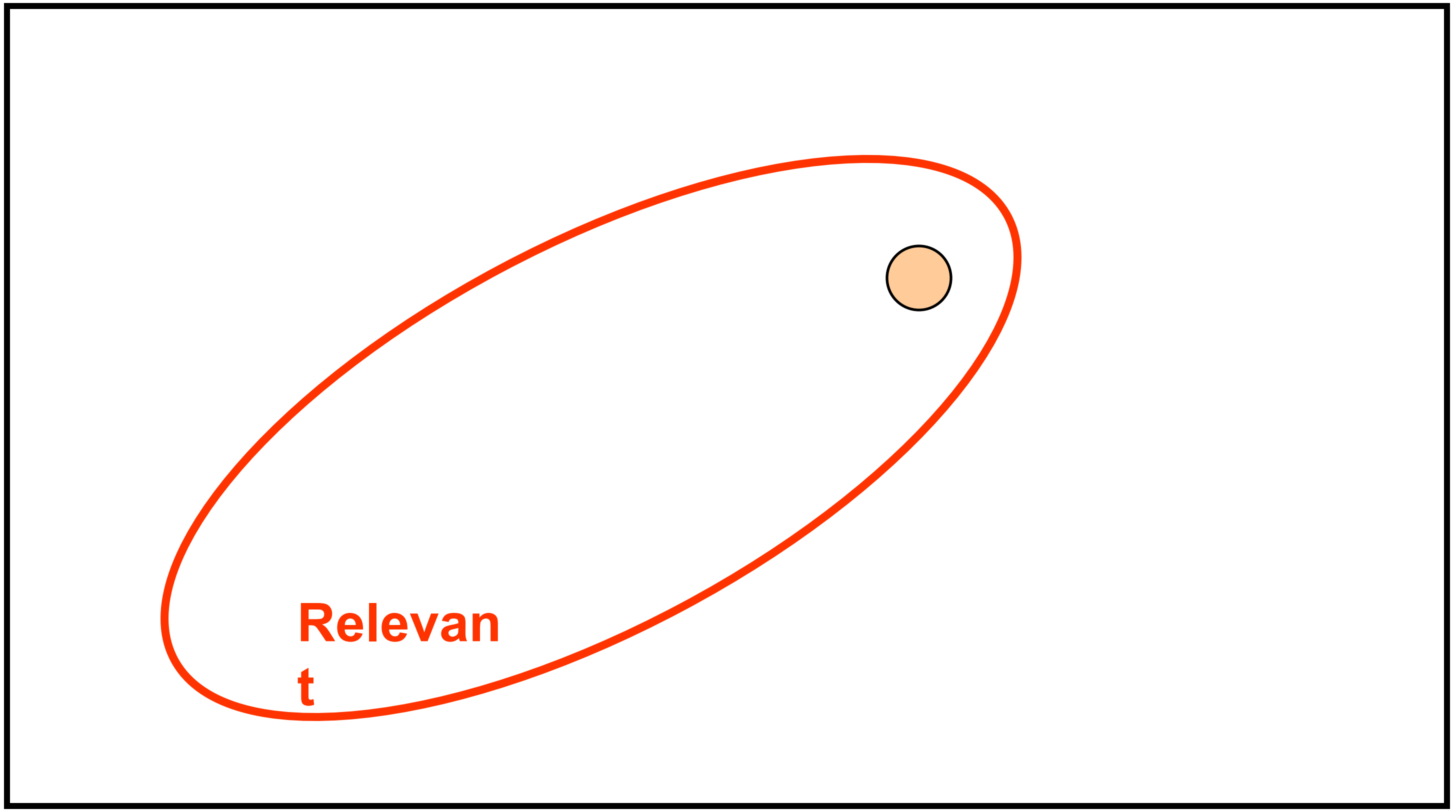
$$\text{Precision} = \frac{|\,\text{RelRetrieved}\,|}{|\,\text{Retrieved}\,|}$$

$$\text{Recall} = \frac{|\,\text{RelRetrieved}\,|}{|\,\text{Rel in Collection}\,|}$$



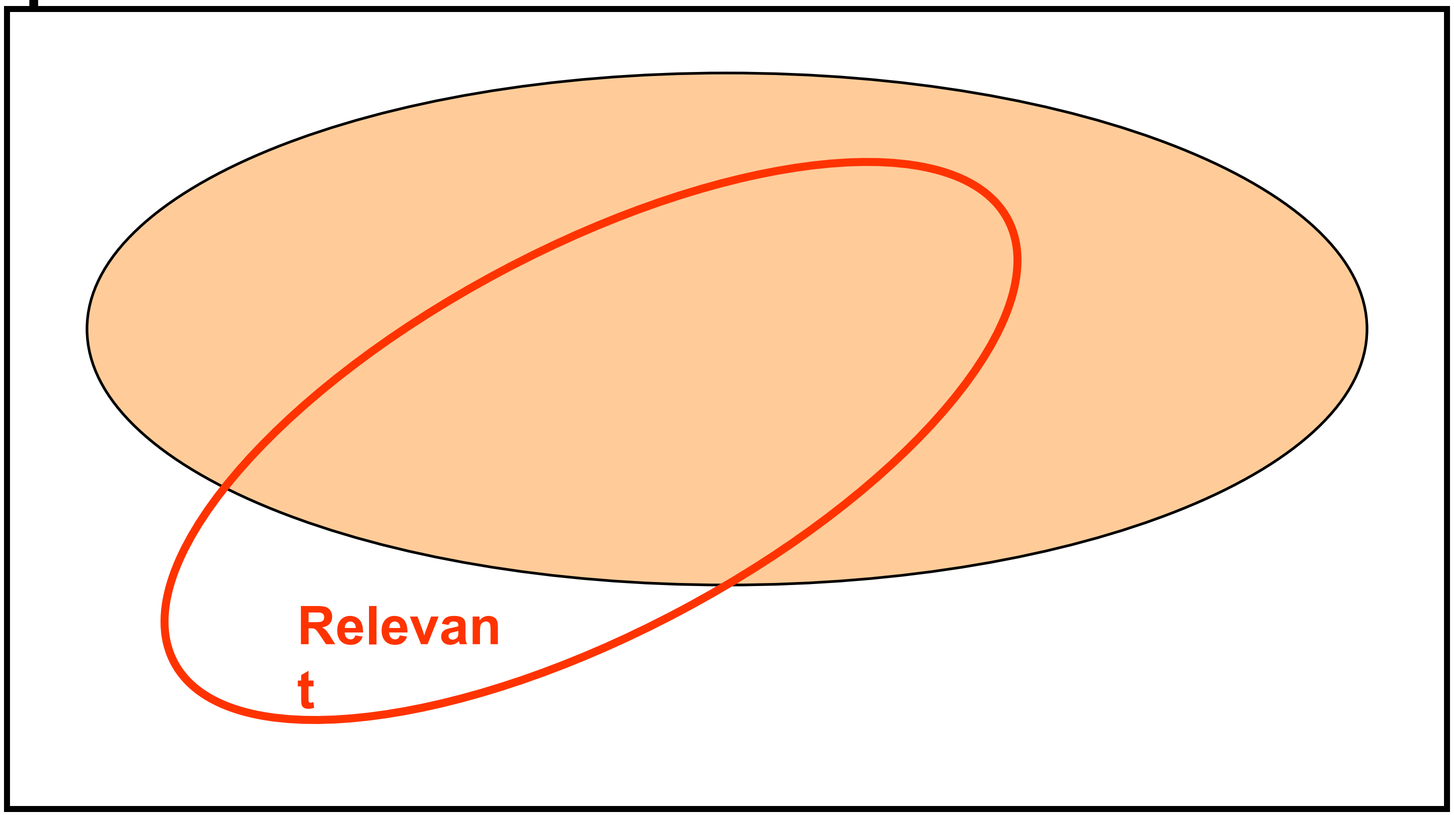
# Retrieved vs. Relevant Documents

**Very high precision, very low recall**



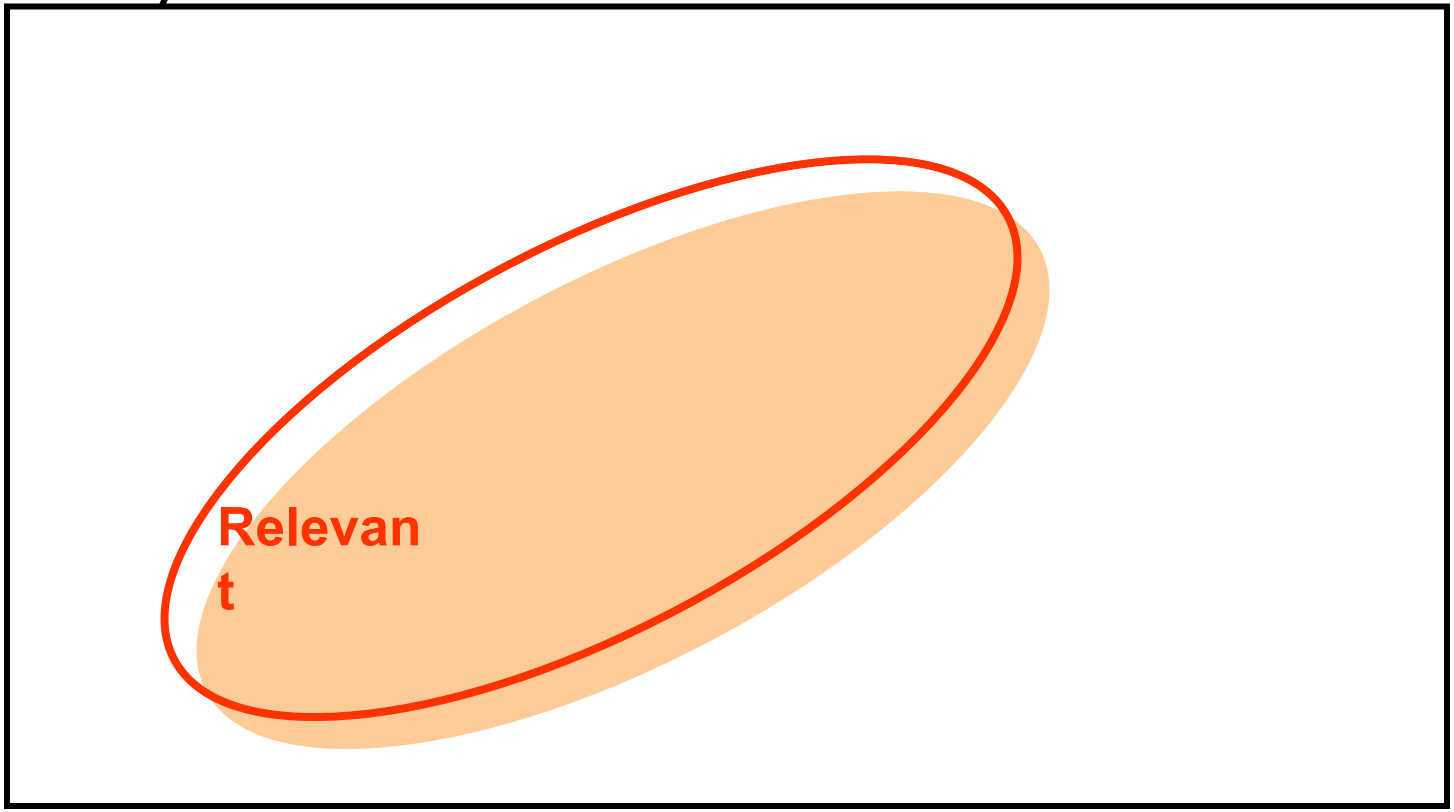
# Retrieved vs. Relevant Documents

**High recall, but low precision**



# Retrieved vs. Relevant Documents

High precision, high recall (at last!)



# Methods of Estimation

- Holdout
  - Reserve 2/3 for training and 1/3 for testing
- Stratified sampling
  - Oversampling vs undersampling

- 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

# 10 Fold Cross Validation (Example)

- What if we don't have enough data to set aside a test dataset?
- Cross-Validation:
  - Each data point is used *both* as train and test data.
- Basic idea:
  - Fit model on 90% of the data; test on other 10%.
  - Now do this on a different 90/10 split.
  - Cycle through all 10 cases.
  - 10 "folds" a common rule of thumb.

# 10 Fold Cross Validation (Example)

- Divide data into 10 equal pieces $P_1 \ldots P_{10}$.
- Fit 10 models, each on 90% of the data.
- Each data point is treated as an out-of-sample data point by exactly one of the models.

| model | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | train | train | train | train | train | train | train | train | train | test |
| 2 | train | train | train | train | train | train | train | train | test | train |
| 3 | train | train | train | train | train | train | train | test | train | train |
| 4 | train | train | train | train | train | train | test | train | train | train |
| 5 | train | train | train | train | train | test | train | train | train | train |
| 6 | train | train | train | train | test | train | train | train | train | train |
| 7 | train | train | train | test | train | train | train | train | train | train |
| 8 | train | train | test | train | train | train | train | train | train | train |
| 9 | train | test | train | train | train | train | train | train | train | train |
| 10 | test | train | train | train | train | train | train | train | train | train |

# 10 Fold Cross Validation (Example)

- Collect the scores from the red diagonal…

| model | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | train | train | train | train | train | train | train | train | train | test |
| 2 | train | train | train | train | train | train | train | train | test | train |
| 3 | train | train | train | train | train | train | train | test | train | train |
| 4 | train | train | train | train | train | train | test | train | train | train |
| 5 | train | train | train | train | train | test | train | train | train | train |
| 6 | train | train | train | train | test | train | train | train | train | train |
| 7 | train | train | train | test | train | train | train | train | train | train |
| 8 | train | train | test | train | train | train | train | train | train | train |
| 9 | train | test | train | train | train | train | train | train | train | train |
| 10 | test | train | train | train | train | train | train | train | train | train |

# 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 TP (on the y-axis) against FP (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point