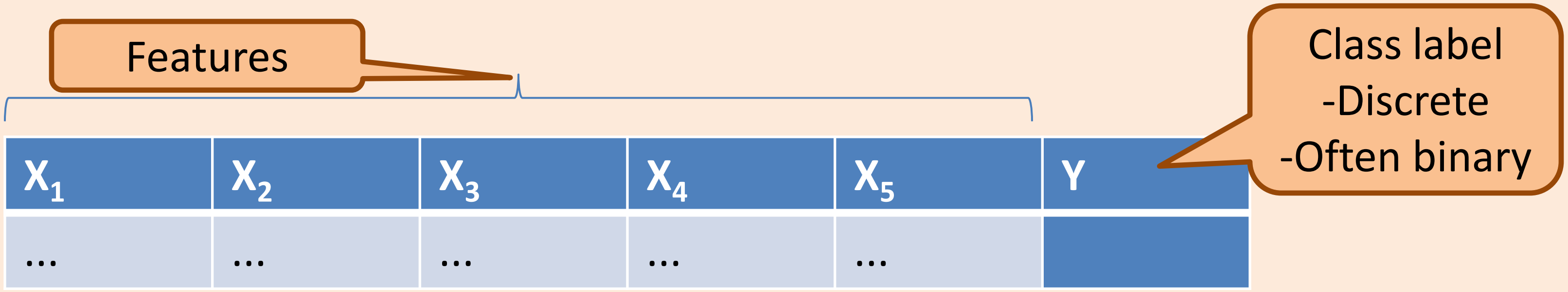
ICS 624 Spring 2013
Data Mining Overview (1)

Asst. Prof. Lipyeow Lim

Information & Computer Science Department

University of Hawaii at Manoa

Classification Problem



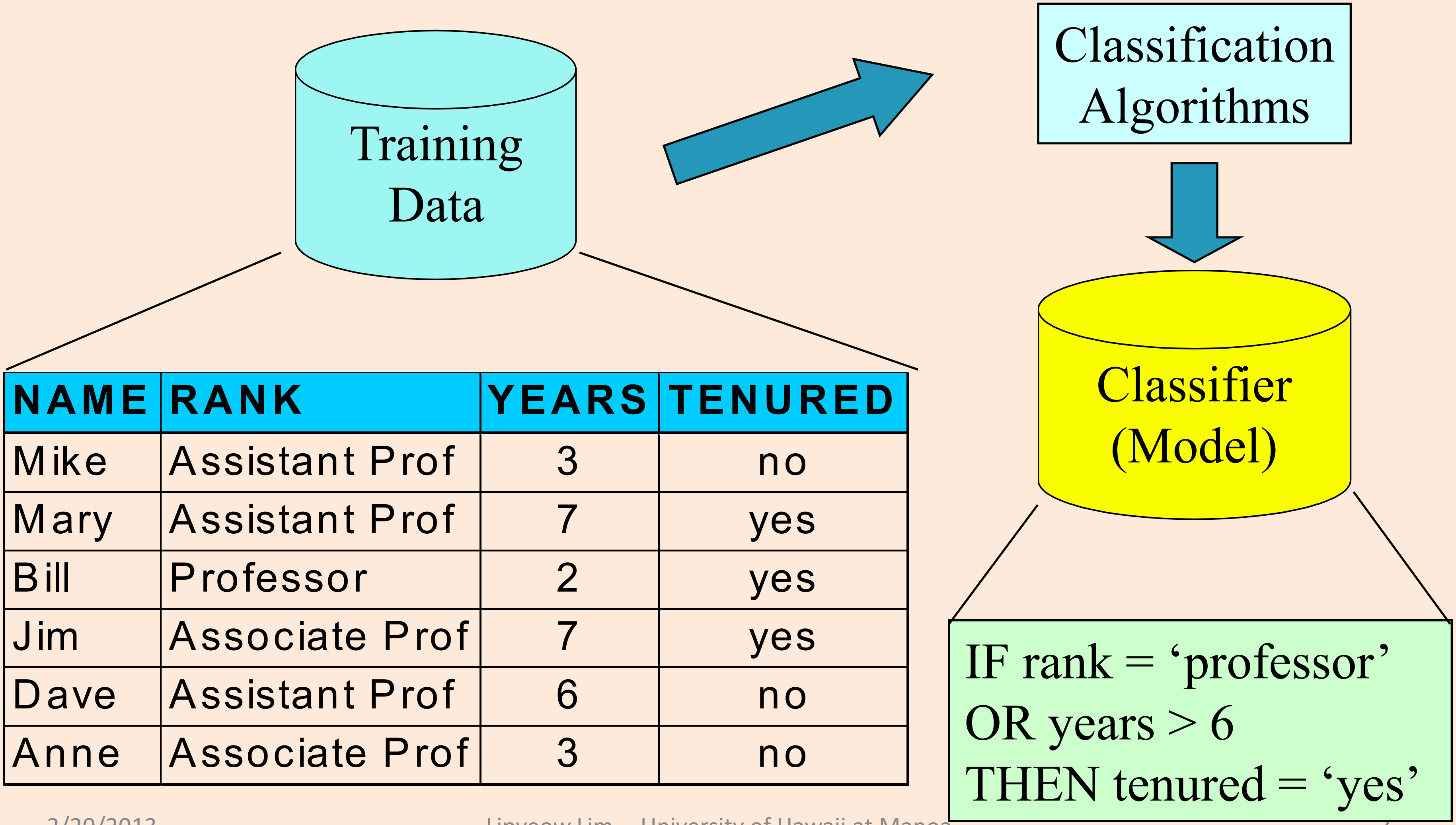
Features

Class label
-Discrete
-Often binary

X_1	X_2	X_3	X_4	X_5	Y
…	…	…	…	…	

- Model construction: describing a set of predetermined classes
  - Each tuple/sample in training set is assumed to belong to a predefined class, as determined by the class label attribute
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model using test set with ground truth
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known
- If class labels are continuous => "Prediction Problem"

Process (1): Model Construction



Training Data

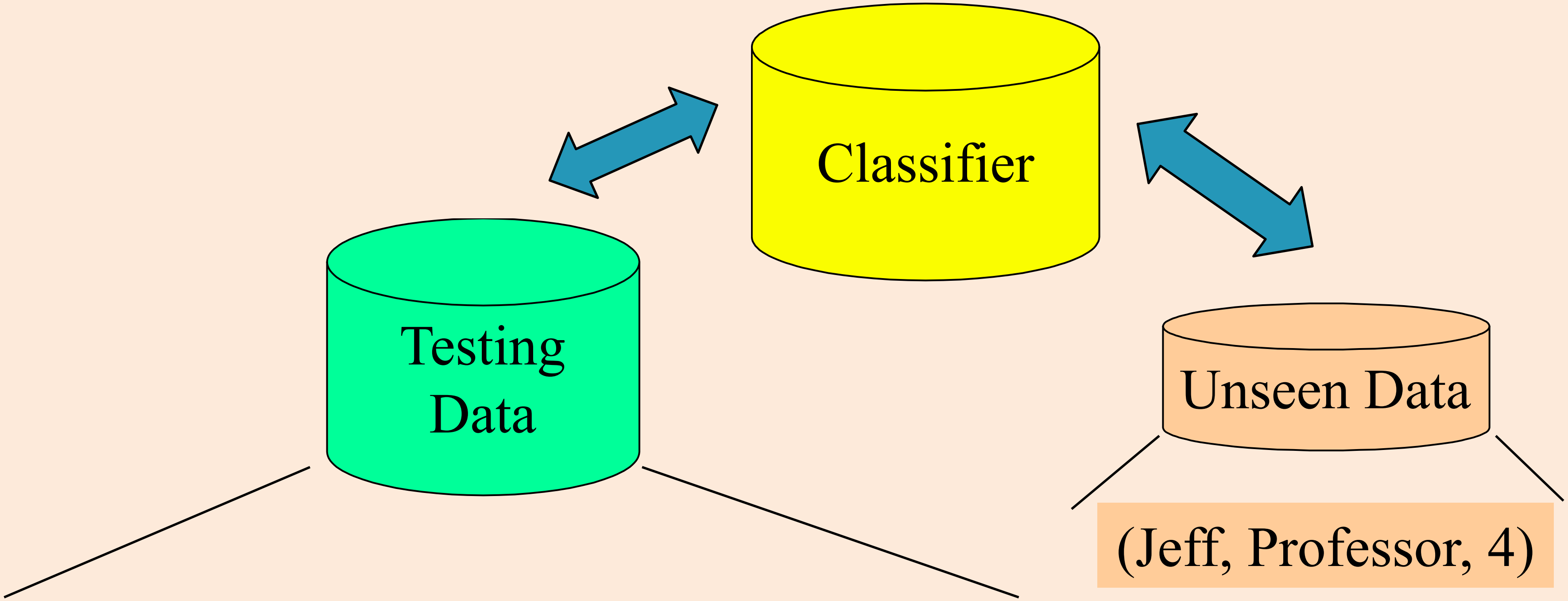
Classification Algorithms

Classifier (Model)

NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Process (2): Using the Model in Prediction



(Jeff, Professor, 4)

NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?

Yes

Supervised vs. Unsupervised Learning

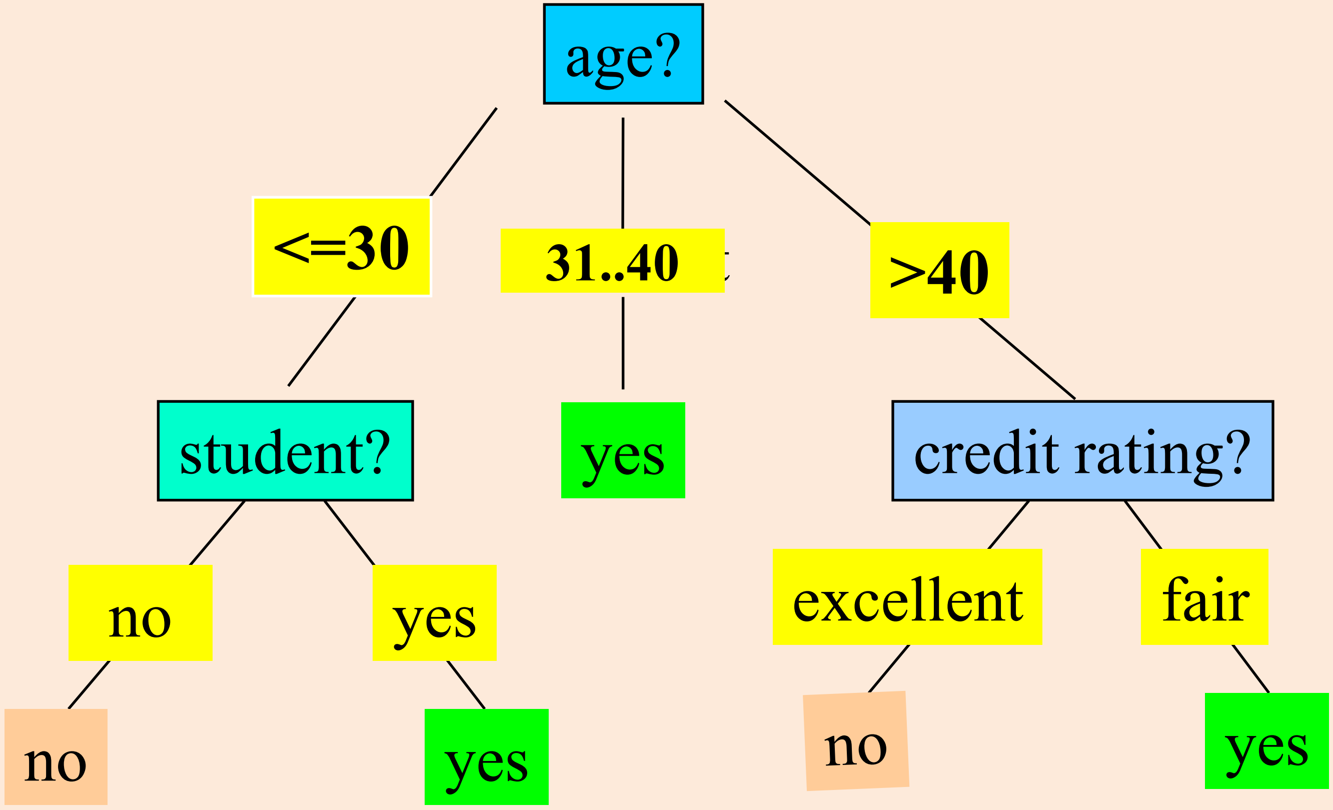
- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Decision Tree Induction: Training Dataset

This follows an example of Quinlan's ID3 (Playing Tennis)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Output: A Decision Tree for “*buys_computer*”



Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

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”

Accuracy Measures

Pred. \ Truth	C_1	C_2
C_1	True positive	False negative
C_2	False positive	True negative

- Accuracy of a classifier M, acc(M): percentage of test set tuples that are correctly classified by the model M
  - Error rate (misclassification rate) of M = 1 – acc(M)
  - Given *m* classes, $CM_{i,j}$, an entry in a **confusion matrix**, indicates # of tuples in class *i* that are labeled by the classifier as class *j*
- Alternative accuracy measures (e.g., for cancer diagnosis)

  sensitivity = t-pos/pos /* true positive recognition rate */

  specificity = t-neg/neg /* true negative recognition rate */

  precision = t-pos/(t-pos + f-pos)

  accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)

  - This model can also be used for cost-benefit analysis

Evaluating the Accuracy (I)

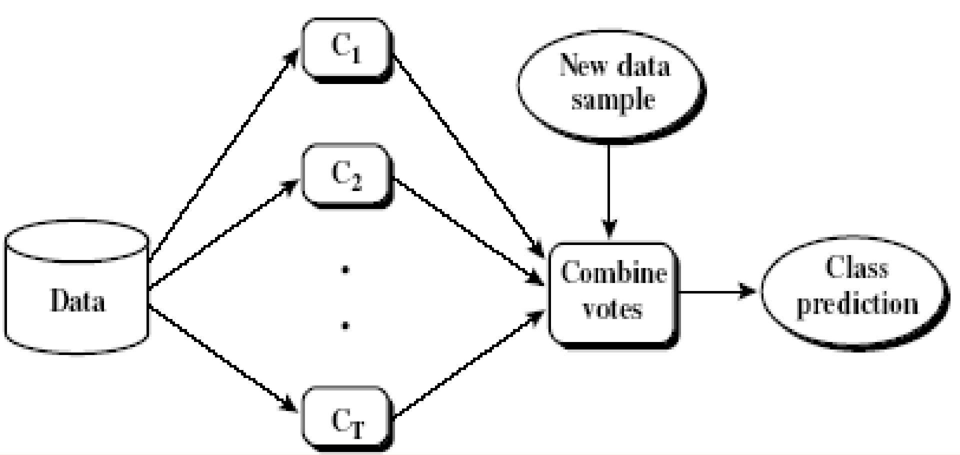
- **<u>Holdout method</u>**
  - – Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - – Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **<u>Cross-validation</u>** (*k*-fold, where k = 10 is most popular)
  - – Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - – At *i*-th iteration, use D_i as test set and others as training set
  - – <u>Leave-one-out</u>: k folds where k = # of tuples, for small sized data
  - – <u>Stratified cross-validation</u>: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

Evaluating the Accuracy (II)

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several boostrap methods, and a common one is **.632 boostrap**
  - Suppose we are given a data set of d tuples. The data set is sampled d times, with replacement, resulting in a training set of d samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data will end up in the bootstrap, and the remaining 36.8% will form the test set (since $(1 – 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedue k times, overall accuracy of the model:

$$acc(M) = \sum_{i=1}^{k} (0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})$$

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors’ majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., boostrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significant better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - Weights are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i
  - The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- The boosting algorithm can be extended for the prediction of continuous values
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclssified, its weight is increased, o.w. it is decreased
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

- The weight of classifier M_i's vote is

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

Random Decision Forests

T.K.Ho. Random Decision Forests. ICDAR 1995.

X_1	X_2	X_3	X_4	X_5	Y
…	…	…	…	…	

- **Idea**: Construct a forest of decision trees by
  - Randomly choosing a subset of features
  - Building a decision tree using the subset of features
  - Combine the class labels from the forest using a discriminant function (see paper for details).
- **Claims**:
  - Increasing the size of the subset of features increases the accuracy
  - Increasing the size of the forest increases accuracy
  - Accuracy doesn't seem to be limited cf. other techniques

One-Class Classification

K. Hempstalk, E. Frank, and I. Witten. One-class classification by combining density and class probability estimation. In Proc. of ECML'08, pages 505–519, 2008

- **Problem**: Training data does not contain examples with negative labels ("2nd class")
- **Idea 1**: Artificially generate negative examples
  - A uniformly random example may be far away from the decision boundary
  - Use a "reference distribution" P(X|A) close to target class
- **Idea 2**: View the problem as density estimation – decision boundary is a threshold
  - Use Bayes' Thm to correct for the reference distribution
    - $P(X|T) = P(X|A) \ * [\ 1-P(T)\ ]\ P(T|X) / \{\ P(T)\ [\ 1-P(T|X)\ ]\ \}$
  - Use traditional classifiers to estimate P(Target|X)
  - Tune the decision threshold to get desired accuracy