Knowledge Discovery and Data Mining

Spring 2021

Chap 8. Classification: Basic Concepts

Jiawei Han, Micheline Kamber and Jian Pei, Data Mining: Concepts and Techniques, 3rd ed., The Morgan Kaufmann Series in Data Management Systems Morgan Kaufmann Publishers, July 2011. ISBN 978-0123814791

Tuong Le, PhD

Outline

What Is Classification?

❑ Classification (Supervised learning)

- Predicts categorical class labels (discrete or nominal)
- Classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

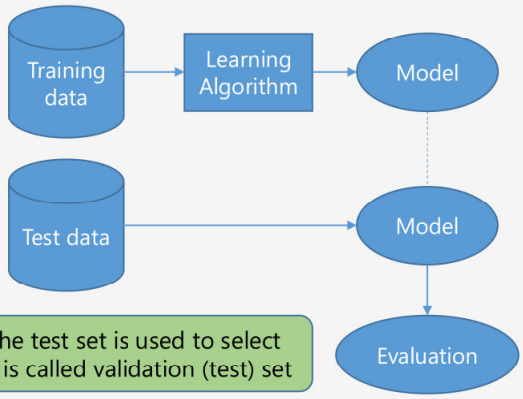
❑ Typical applications

- Credit/loan approval: A bank loans officer needs analysis of her data to learn which loan applicants are "safe" and which are "risky" for the bank.
- Medical diagnosis: if a tumor is cancerous or benign
- Fraud detection: if a transaction is fraudulent
- Web page categorization: which category it is

Supervised learning process: Two steps

❑ Model construction: describing a set of predetermined classes

- Each tuple/sample, X = $(x_1, x_2, ..., x_n)$, is assumed to belong to a predefined class, as determined by the class label attribute
- The set of tuples used for model construction is training set
- **The model** is represented as classification rules, decision trees, or mathematical formulae



❑ Model usage: for classifying future or unknown objects

- Estimate accuracy of the model
  - The known label of test sample is compared with the classified result from the model
  - Accuracy rate is the percentage of test set samples that are correctly classified by the model
  - Test set is independent of training set (otherwise overfitting)
- If the accuracy is acceptable, use the model to classify new data

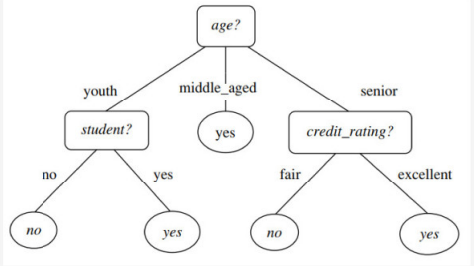
Outline

1. Classification: Basic Concepts
2. **Decision Tree Induction**
   - ❑ Decision Tree Induction Introduction
   - ❑ Decision Tree Induction Pseudocode
   - ❑ Attribute Selection Measure
   - ❑ Tree Pruning
3. Rule-Based Classification
4. Model Evaluation and Selection
5. Summary

Decision Tree Induction Introduction

- ❑ Decision tree induction is the learning of decision trees from class-labeled training tuples
- ❑ A **decision tree** is a flowchart-like tree structure, where each **internal node** (nonleaf node) denotes a test on an attribute, each **branch** represents an outcome of the test, and each **leaf node** (or *terminal node*) holds a class label
- ❑ The topmost node in a tree is the root node
- ❑ "Why are decision tree classifiers so popular?"
  - The construction of decision tree classifiers does not require any domain knowledge or parameter setting
  - Decision trees can handle multidimensional data
  - The learning and classification steps of decision tree induction are simple and fast
- ❑ Decision tree induction algorithms have been used for classification in many application areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology.

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no



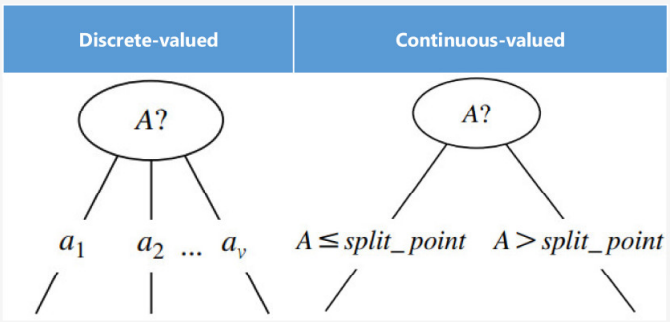
Decision Tree Induction Pseudocode

Generate_decision_tree(D, *attribute_list, Attribute_selection_method*) returns a tree

1. Create a node N;
2. **if** tuples in D are all of the same class, C, **then**
3. return N as a leaf node labeled with the class C;
4. **if** attribute list is empty **then**
5. return N as a leaf node labeled with the majority class in D; // majority voting
6. apply ***Attribute_selection_method***(D, *attribute_list*) to find the "best" *splitting_criterion*;
7. label node N with *splitting_criterion*;
8. **if** *splitting attribute* is discrete-valued **and** multiway splits allowed **then** // not restricted to binary trees
9. *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
10. **for each** outcome j of splitting criterion // partition the tuples and grow subtrees for each partition
11. let D_j be the set of data tuples in D satisfying outcome j; // a partition
12. **if** D_j is empty **then**
13. attach a leaf labeled with the majority class in D to node N;
14. **else** attach the node returned by **Generate_decision_tree**(D_j, attribute_list, *Attribute_selection_method*) to node N;
endfor
15. return N;

Pseudocode Explanation

- ***Discrete-valued***: A branch is created for each known value, a_j, of A and labeled with that value. Partition D_j is the subset of class-labeled tuples in D having value a_j of A. **Removed from attribute list.**
- ***Continuous-valued***: the test at node N has two possible outcomes, corresponding to the conditions A ≤ *split_point* and A > *split_point*, respectively, where split point is the split-point returned by Attribute selection method as part of the splitting criterion.



- **The recursive partitioning stops** only when any one of the following terminating conditions is true:
  - All the tuples in partition D (represented at node N) belong to the same class (step 2)
  - There are no remaining attributes on which the tuples may be further partitioned (step 4)
  - There are no tuples for a given branch, that is, a partition Dj is empty (step 12).

Attribute Selection Measure

- Top-down divide-and-conquer method does a greedy search for a simple tree but does not guarantee to find the smallest.
- The goal is to pick a feature that creates subsets of examples that are relatively "pure" in a single class so they are "closer" to being leaf nodes.
- An attribute selection measure is a heuristic for selecting the splitting criterion that "best" separates a given data partition, D, of class-labeled training tuples into individual classes. This section describes three popular attribute selection measures:
  - **Information gain**,
  - **Gain ratio**,
  - **Gini index**.

Information Gain

- Select the attribute with the highest information gain.
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info_A(D) - Info(D)$$

Information Gain: Example

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- We have two class, nine tuples of class yes and five tuples of class no
- The entropy of *D:*

$$Info(D) = I(9,5) = -\frac{9}{14}log_2\left(\frac{9}{14}\right) - \frac{5}{14}log_2\left(\frac{5}{14}\right) = 0.94$$

- Next, we need to compute the expected information requirement for each attribute

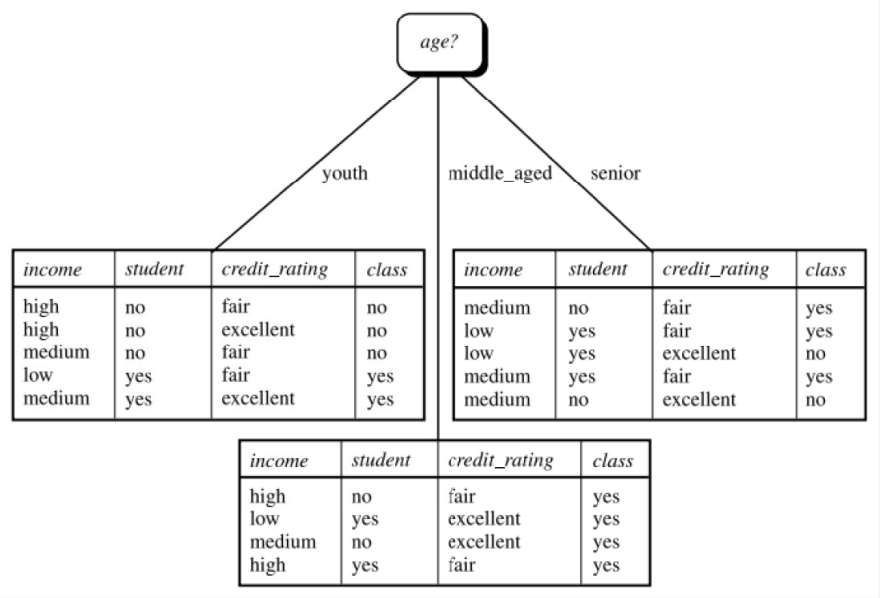
$$Info_{age}(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j) = \frac{5}{14} \times I(2,3) + \frac{4}{14} \times I(4,0) + \frac{5}{14} \times I(3,2) = 0.694$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

- Similarly, $Gain(income)$=0.029, $Gain(student) = 0.151$, and $Gain(credit\_rating)$=0.048. Age is selected.

Information Gain: Example

- Node N is labeled with age, and branches are grown for each of the attribute's values.
- Notice that the tuples falling into the partition for age D middle aged all belong to the same class. Because they all belong to class "yes," a leaf should therefore be created at the end of this branch and labeled "yes."



income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit_rating	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Determining the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $\frac{a_i+a_{i+1}}{2}$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for *A* is selected as the *split_point* for *A*.
- Split D into D_1 and D_2:
  - D_1 is the set of tuples satisfying A ≤ *split_point*,
  - D_2 is the set of tuples satisfying A > *split_point*.

Gain Ratio

- Information gain measure is biased towards attributes with a large number of values
- Gain ratio (an extension to information gain) to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log_2\left(\frac{|D_j|}{|D|}\right)$$

- The gain ratio is defined as:

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

- The attribute with the maximum gain ratio is selected as the splitting attribute.

Gain Ratio: Example

- **Computation of gain ratio for the attribute income**: There are three partitions, namely low, medium, and high, containing four, six, and four tuples, respectively.

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- Split Info of **the attribute income**

$$SplitInfo_{inconme}(D)$$

$$= -\frac{4}{14} \times log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times log_2\left(\frac{4}{14}\right)$$

$$= 1.557$$

- We have $Gain(income) = 0.029$. Therefore, $GainRatio(income) = \frac{Gain(A)}{SplitInfo_A(D)} = \frac{0.029}{1.557} = 0.019$.

Gini Index

- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- The **Gini index** measures the impurity of D, a data partition or set of training tuples, as

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

- Note the Gini index considers a binary split for each attribute.
- **The first case:** A is a discrete-valued attribute having v distinct values $\{a_1, a_2, ..., a_v\}$ which provide 2^v possible subsets. For example, if income has three possible values, namely {low, medium, high}, then the possible subsets are ***{low, medium, high}***, {low, medium}, {low, high}, {medium, high}, {low}, {medium}, {high}, and ***{}***.
- There are $2^v - 2$ possible ways to form two partitions of the data, D to create a binary test for attribute A of the form "$A \in S_A$?". If a data set D is split on A into two subsets D1 and D2, the gini index $gini(D)$ is defined as

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2)$$

- The subset that gives the minimum Gini index for that attribute is selected as its splitting subset.

Gini Index (2)

- **The second case** (A is a continuous-valued attributes):
  - All midpoints between each pair of (sorted) adjacent values are considered as the possible split-points.
  - The point giving the minimum Gini index for a given (continuous-valued) attribute is taken as the split-point of that attribute.
- The reduction in impurity that would be incurred by a binary split on a discrete- or continuous-valued attribute A is

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

- The attribute that maximizes the reduction in impurity (or, equivalently, has the minimum Gini index) is selected as the splitting attribute.
- This attribute and either its splitting subset (for a discrete-valued splitting attribute) or split-point (for a continuous-valued splitting attribute) together form the **splitting criterion**.

Gini Index: Example

- The Gini index to compute the impurity of D:

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

- Let's start with the attribute income and consider each of the possible splitting subsets.
- Consider the subset {low, medium}. This would result in 10 tuples in partition D1 satisfying the condition "income $\in$ {low, medium}". The remaining four tuples of D would be assigned to partition D2. We have

$$Gini_{income \in \{low,medium\}}(D) = \frac{10}{14}Gini(D_1) + \frac{4}{14}Gini(D_2) = \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) = 0.443$$

- $Gini_{income \in \{low,high\}}(D)$ is 0.458; $Gini_{income \in \{medium,high\}}(D)$ is 0.450. Thus, split on the $\{low,\ medium\}$ (and {high}) since it has the lowest Gini index.
- Evaluating age, we obtain {youth, senior} as the best split for age with a Gini index of 0.375; the attributes student and credit rating are both binary, with Gini index values of 0.367 and 0.429, respectively.
- Choose the attribute age and splitting subset {youth, senior} which give the minimum Gini index overall, with a reduction in impurity of 0.459 – 0.357 = 0.102.

Comparing Attribute Selection Measures

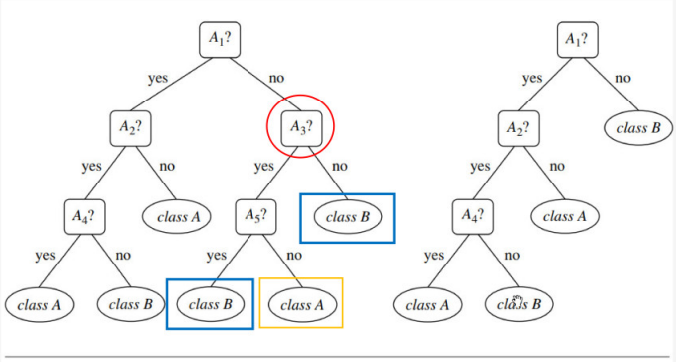
- The three measures, in general, return good results but
  - Information gain:
    - biased towards multivalued attributes
  - Gain ratio:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - biased to multivalued attributes
    - has difficulty when number of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Other Attribute Selection Measures

- **CHAID**: a popular decision tree algorithm, measure based on χ^2 test for independence
- **C-SEP**: performs better than info. gain and gini index in certain cases
- **G-statistic**: has a close approximation to χ^2 distribution
- **MDL (Minimal Description Length) principle** (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - **CART**: finds multivariate splits based on a linear combination of attributes
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

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
  - Pre-pruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold. However, difficult to choose an appropriate threshold.



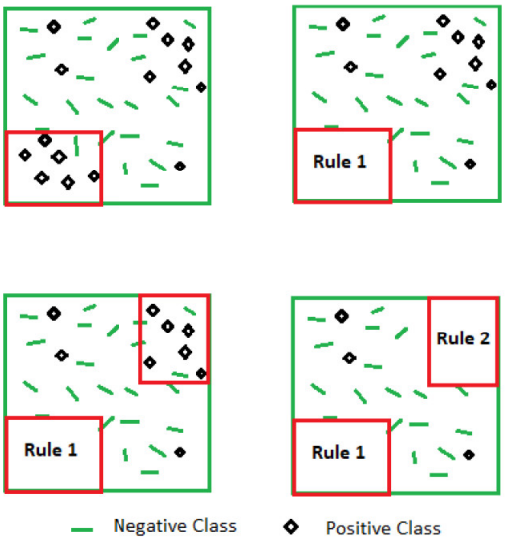
An unpruned decision tree and a pruned version of it.

  - Post-pruning (more common): Removing branches and replacing them with a leaf from a "fully grown" tree.
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Outline

Rule-Based Classification: Introduction

- In rule-based classifiers: The learned model is represented as a set of IF-THEN rules.
- Contents:
  - How IF-THEN rules are used for classification
  - Rule Extraction from a Decision Tree
  - Sequential Covering Algorithm: Rule Extraction directly from the training data



How IF-THEN rules are used for classification

- A rule-based classifier uses a set of IF-THEN rules for classification. An IF-THEN rule is an expression of the form

IF condition **THEN** conclusion

- The "IF" part (or left side) of a rule is known as the rule antecedent or precondition which consists of one or more attribute tests. The "THEN" part (or right side) is the rule consequent which contains a class prediction.
- Assessment of a rule: coverage and accuracy

$$coverage(R) = \frac{n_{cover}}{|D|}, \qquad accuracy(R) = \frac{n_{correct}}{n_{cover}}$$

where n_{covers} = # of tuples covered by R and $n_{correct}$ = # of tuples correctly classified by R.

- That is, a rule's coverage is the percentage of tuples that are covered by the rule (i.e., their attribute values hold true for the rule's antecedent). For a rule's accuracy, we look at the tuples that it covers and see what percentage of them the rule can correctly classify.

How IF-THEN rules are used for classification: Example

❑ Consider R1:

$$R1: (age = youth) \wedge (student = yes) \Rightarrow (buy - computer = yes)$$

which covers 2 of the 14 tuples. It can correctly classify both tuples. Therefore, coverage(R1) = 2/14 = 14.28% and accuracy(R1) = 2/2 = 100%

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

❑ If more than one rule are triggered, need conflict resolution

- Size ordering: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the most attribute tests)
- Class-based ordering: decreasing order of prevalence or misclassification cost per class
- Rule-based ordering (decision list): rules are organized into one long priority list, according to some measure of rule quality or by experts

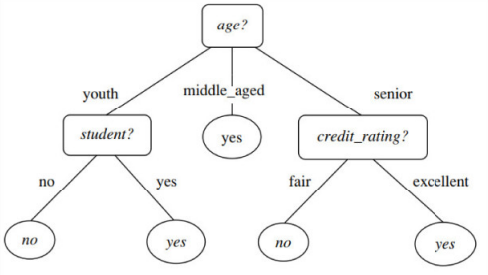
Rule Extraction from a Decision Tree

❑ Rules are *easier to understand* than large trees

❑ One rule is created *for each path* from the root to a leaf

❑ Each attribute-value pair along a path forms a conjunction: the leaf holds the class prediction

❑ Rules are mutually exclusive and exhaustive



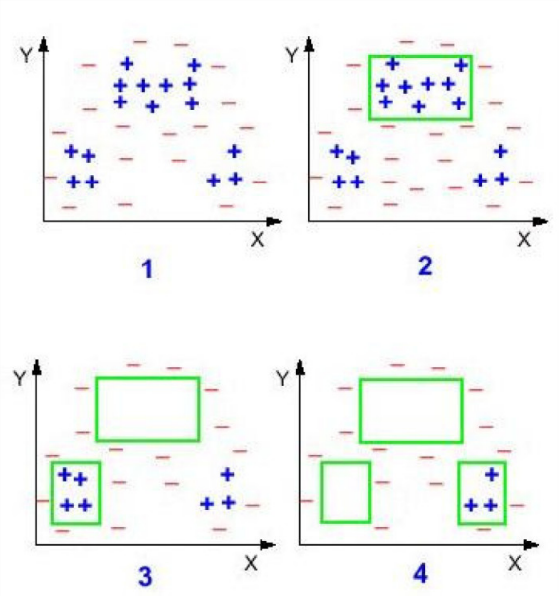
Decision Tree

R1: IF $age = youth$ AND $student = no$ THEN $buys\_computer = no$
R2: IF $age = youth$ AND $student = yes$ THEN $buys\_computer = yes$
R3: IF $age = middle\_aged$ THEN $buys\_computer = yes$
R4: IF $age = senior$ AND $credit\_rating = excellent$ THEN $buys\_computer = yes$
R5: IF $age = senior$ AND $credit\_rating = fair$ THEN $buys\_computer = no$

A rule-based classifier

Sequential Covering Algorithm

- Use them to extract the IF-THEN rules directly from the training data (i.e., without having to generate a decision tree first).
- There are many sequential covering algorithms: AQ, CN2, and RIPPER.
- The general strategy is as follows.
  - Rules are learned one at a time.
  - Each time a rule is learned, the tuples covered by the rule are removed, and
  - The process repeats on the remaining tuples.



Sequential covering algorithm

1. Start with an empty Cover
2. Using Learn-One-Rule to find the best hypothesis.
3. If the Just-Learnt-Rule satisfies the threshold then
   - Put Just-Learnt-Rule to the Cover.
   - Remove examples covered by Just-Learnt-Rule.
   - Go to step 2.
4. Sort the Cover according to its performance over examples.
5. Return: Cover.

Id	Size	Color	Shape	Weight	Expensive
1	Big	Red	Square	Heavy	Yes
2	Small	Blue	Triangle	Light	Yes
3	Small	Blue	Square	Light	No
4	Big	Green	Triangle	Heavy	No
5	Big	Blue	Square	Light	No
6	Big	Green	Square	Heavy	Yes
7	Small	Red	Triangle	Light	Yes

The final set of rules is like follow:

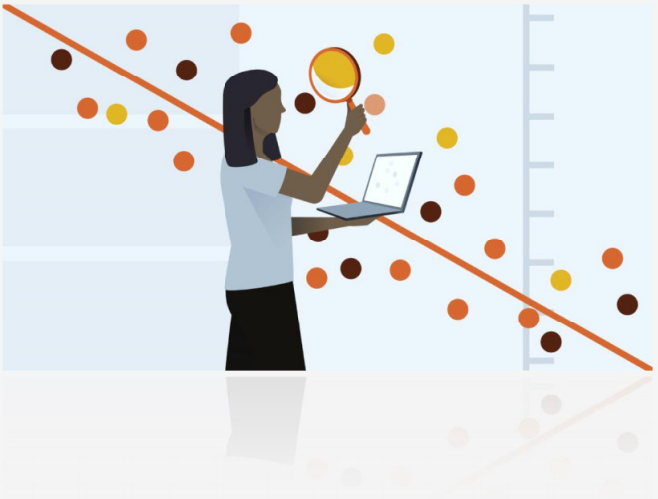
Expensive = Yes if: Color = Red.
Or (Color = Green & Shape = Square).
Or (Color = Blue & Shape = Triangle).

Outline

1. Classification: Basic Concepts
2. Decision Tree Induction
3. Rule-Based Classification
4. Model Evaluation and Selection
5. Summary

Model Evaluation and Selection

- **Evaluation metrics**: How can we measure accuracy?
- Methods for estimating a classifier's accuracy:
  - **Holdout and random subsampling**
  - **Cross-validation**
  - **Bootstrap**
- Model selection:
  - **Tests of statistical significance**
  - **Cost-benefit analysis and ROC Curves**



Evaluation Metrics

- Using training data to derive a classifier and then evaluate this model by testing set. We obtain:
  - **True positives (TP)**: These refer to the positive tuples that were correctly labeled by the classifier. Let TP be the number of true positives.
  - **True negatives (TN):** These are the negative tuples that were correctly labeled by the classifier. Let *TN* be the number of true negatives.
  - **False positives (FP)**: These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class *buys computer=no* for which the classifier predicted *buys computer=yes*). Let *FP* be the number of false positives.
  - **False negatives (FN)**: These are the positive tuples that were mislabeled as negative (e.g., tuples of class *buys computer=yes* for which the classifier predicted *buys computer=no*). Let *FN* be the number of false negatives

Evaluation Metrics

$$Error - rate = \frac{FP + FN}{TP + TN_{FP} + FN}$$

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

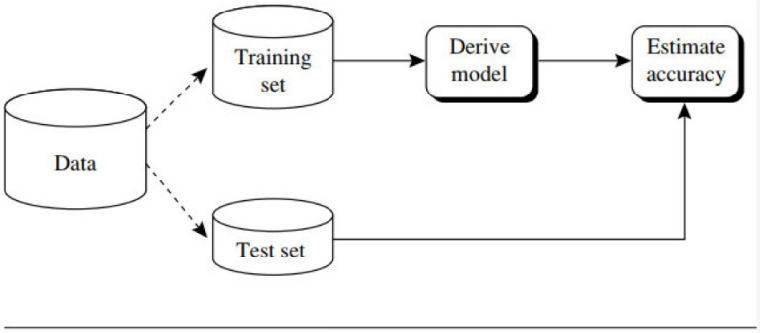
$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

		Predicted Class: Positive	Predicted Class: Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP + FN)}$
Actual Class	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	**Negative Predictive Value** $\frac{TN}{(TN + FN)}$	**Accuracy** $\frac{TP + TN}{(TP + TN + FP + FN)}$

Recall

Holdout Method and Random Subsampling

- 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

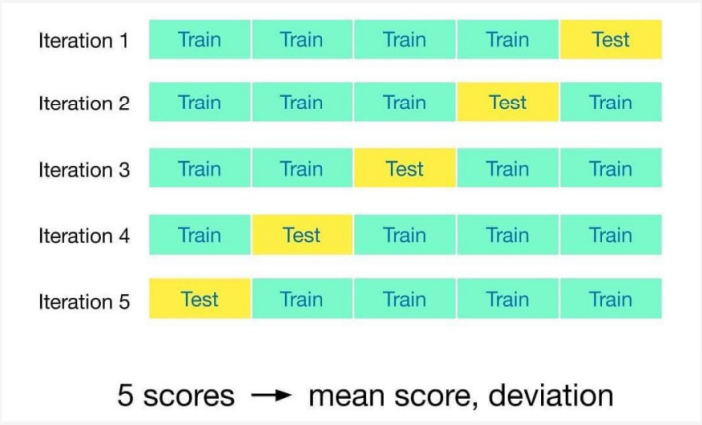


Estimating accuracy with the holdout method.

- **Random sampling**: a variation of holdout
  - Repeat holdout k times, accuracy = avg. of the accuracies obtained

Cross-validation

- **k-fold Cross-validation**
  - 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
  - Leave-one-out: *k* folds where *k* = number of tuples, for small sized data



- ***Stratified cross-validation*:** folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

Bootstrap

- 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 bootstrap methods, and a common one is **.632 boostrap**
  - A data set with *d* tuples 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 end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

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

Tests of statistical significance

- Suppose we have 2 classifiers, M_1 and M_2. Applying 10-fold cross-validation to obtain the mean error rates, $\overline{err}(M_1)$ and $\overline{err}(M_2)$, which model is better?
- It may seem intuitive to select the model with the lowest error rate; however, the mean error rates are just estimates of error on the true population of future data cases.
- Although $\overline{err}(M_1)$ and $\overline{err}(M_2)$ may appear different, that difference may not be statistically significant.
- What if the difference between the 2 error rates is just attributed to *chance*?
  - Use a **test of statistical significance**
  - Obtain **confidence limits** for our error estimates

Tests of statistical significance: Null Hypothesis

- Perform 10-fold cross-validation
- Assume samples follow a **t distribution** with *k–1* **degrees of freedom** (here, *k=10*)
- Use **t-test** (or **Student's t-test**)
- **Null Hypothesis**: M_1 & M_2 are the same. In other words, that difference in mean error rate between the two is zero.
- If we can **reject** null hypothesis, then
  - We conclude that the difference between M_1 & M_2 is **statistically significant**
  - Chose model with lower error rate.

Tests of statistical significance: t-test

- If the same test set for both M_1 & M_2: Use **a pairwise comparison** of the two models
  - For i-th round for 10-fold cross-validation, the same cross partitioning is used to obtain $err(M_1)_i$ and $err(M_2)_i$
  - Average over 10 rounds to get $\overline{err}(M_1)$ and $\overline{err}(M_2)$
  - t-test computes t-statistic with k-1 degrees of freedom

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{\frac{var(M_1 - M_2)}{k}}}$$

where $var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))]$

Estimating Confidence Intervals: Statistical Significance

- Are M_1 & M_2 **significantly different**?
  - Compute *t* and Select *significance level* (e.g. *sig = 0.05)*
  - Consult table for t-distribution: Find *t value* corresponding to *k-1 degrees of freedom* (here, 9)
  - t-distribution is symmetric: typically upper % points of distribution shown → look up value for **confidence limit** *z=sig/2* (here, 0.025)
  - **If t > z or t < -z**, then t value lies in rejection region:
    - **Reject null hypothesis** that mean error rates of M_1 & M_2 are same
    - Conclude: statistically significant difference between M_1 & M_2
  - **Otherwise**, conclude that any difference is **chance**

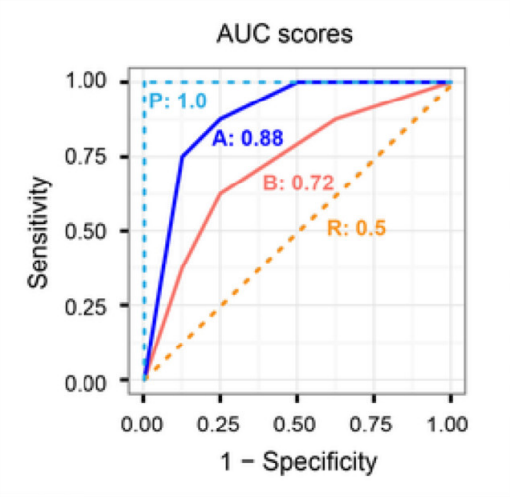
	Level of Significance α								
df	0.200	0.100	0.075	0.050	0.025	0.010	0.005	0.001	0.0005
1	1.642	2.706	3.170	3.841	5.024	6.635	7.879	10.828	12.116
2	3.219	4.605	5.181	5.991	7.378	9.210	10.597	13.816	15.202
3	4.642	6.251	6.905	7.815	9.348	11.345	12.838	16.266	17.731
4	5.989	7.779	8.496	9.488	11.143	13.277	14.860	18.467	19.998
5	7.289	9.236	10.008	11.070	12.833	15.086	16.750	20.516	22.106
6	8.558	10.645	11.466	12.592	14.449	16.812	18.548	22.458	24.104
7	9.803	12.017	12.883	14.067	16.013	18.475	20.278	24.322	26.019
8	11.030	13.362	14.270	15.507	17.535	20.090	21.955	26.125	27.869
9	12.242	14.684	15.631	16.919	19.023	21.666	23.589	27.878	29.667
10	13.442	15.987	16.971	18.307	20.483	23.209	25.188	29.589	31.421
11	14.631	17.275	18.294	19.675	21.920	24.725	26.757	31.265	33.138
12	15.812	18.549	19.602	21.026	23.337	26.217	28.300	32.910	34.822
13	16.985	19.812	20.897	22.362	24.736	27.688	29.820	34.529	36.479
14	18.151	21.064	22.180	23.685	26.119	29.141	31.319	36.124	38.111
15	19.311	22.307	23.452	24.996	27.488	30.578	32.801	37.698	39.720
16	20.465	23.542	24.716	26.296	28.845	32.000	34.267	39.253	41.309
17	21.615	24.769	25.970	27.587	30.191	33.409	35.719	40.791	42.881
18	22.760	25.989	27.218	28.869	31.526	34.805	37.157	42.314	44.435
19	23.900	27.204	28.458	30.144	32.852	36.191	38.582	43.821	45.974
20	25.038	28.412	29.692	31.410	34.170	37.566	39.997	45.315	47.501
21	26.171	29.615	30.920	32.671	35.479	38.932	41.401	46.798	49.013
22	27.301	30.813	32.142	33.924	36.781	40.289	42.796	48.269	50.512
23	28.429	32.007	33.360	35.172	38.076	41.639	44.182	49.729	52.002
24	29.553	33.196	34.572	36.415	39.364	42.980	45.559	51.180	53.480
25	30.675	34.382	35.780	37.653	40.646	44.314	46.928	52.620	54.950
26	31.795	35.563	36.984	38.885	41.923	45.642	48.290	54.053	56.409
27	32.912	36.741	38.184	40.113	43.195	46.963	49.645	55.477	57.860
28	34.027	37.916	39.380	41.337	44.461	48.278	50.994	56.894	59.302
29	35.139	39.087	40.573	42.557	45.722	49.588	52.336	58.302	60.738
30	36.250	40.256	41.762	43.773	46.979	50.892	53.672	59.704	62.164
40	47.269	51.805	53.501	55.759	59.342	63.691	66.766	73.403	76.097
50	58.164	63.167	65.030	67.505	71.420	76.154	79.490	86.662	89.564
60	68.972	74.397	76.411	79.082	83.298	88.380	91.952	99.609	102.698
70	79.715	85.527	87.680	90.531	95.023	100.425	104.215	112.319	115.582
80	90.405	96.578	98.861	101.880	106.629	112.329	116.321	124.842	128.267
90	101.054	107.565	109.969	113.145	118.136	124.117	128.300	137.211	140.789
100	111.667	118.498	121.017	124.342	129.561	135.807	140.170	149.452	153.174

Cost-benefit analysis and ROC Curves

- The cost associated with a false negative (such as incorrectly predicting that **a cancerous patient** is **not cancerous**) is far greater than those of a false positive (incorrectly yet conservatively labeling a noncancerous patient as cancerous)
- In such cases, we can outweigh one type of error over another by assigning a different cost to each. These costs may consider the danger to the patient, financial costs of resulting therapies, and other hospital costs.
- Up to now, to compute classifier accuracy, we have assumed equal costs and essentially divided the sum of true positives and true negatives by the total number of test tuples.

Cost-benefit analysis and ROC Curves

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- An ROC curve for a given model shows the trade-off between the true positive rate (TPR) and the false positive rate (FPR) using different thresholds.
- The area under the ROC curve (AUC) is a measure of the accuracy of the model.
  - The score is 1.0 for the classifier with the perfect performance level (P)
  - 0.5 for the classifier with the random performance level (R).
  - ROC curves clearly shows classifier A outperforms classifier B, which is also supported by their AUC scores (0.88 and 0.72).



Issues Affecting Model Selection

- **Accuracy:** classifier accuracy: predicting class label
- **Speed**
  - Time to construct the model (training time)
  - Time to use the model (classification/prediction time)
- **Robustness**: handling noise and missing values
- **Scalability**: efficiency in disk-resident databases
- **Interpretability:** understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Outline

1. Classification: Basic Concepts
2. Decision Tree Induction
3. Rule-Based Classification
4. Model Evaluation and Selection
5. Summary

Summary

- **Classification** is a form of data analysis that extracts models describing important data classes.
- **Decision tree induction** is a top-down recursive tree induction algorithm, which uses an attribute selection measure to select the attribute tested for each non-leaf node in the tree.
- **A rule-based classifier** uses a set of IF-THEN rules for classification. Rules can be extracted from a decision tree. Rules may also be generated directly from training data using sequential covering algorithms.
- **Evaluation metrics** include: accuracy, sensitivity, specificity, precision, recall, F measure, and Fß measure.
- Construction and evaluation of a classifier require partitioning labeled data into a training set and a test set. **Holdout**, **random sampling**, **cross-validation**, and **bootstrapping** are typical methods used for such partitioning.
- **Significance tests** and **ROC curves** are useful for model selection.