Chapter 4: Classification & Prediction

- **4.1 Basic Concepts of Classification and Prediction**
  - 4.1.1 Definition
  - 4.1.2 Classification vs. Prediction
  - 4.1.3 Classification Steps
  - 4.1.4 Issues of Classification and Prediction
- **4.2 Decision Tree Induction**
  - 4.2.1 The Algorithm
  - 4.2.2 Attribute Selection Measures
  - 4.2.3 Tree Pruning
  - 4.2.4 Scalability and Decision Tree Induction
- **4.3 Bayes Classification Methods**
- **4.4 Rule Based Classification**
- **4.5 Lazy Learners**
- **4.6 Prediction**
- **4.7 How to Evaluate and Improve Classification**

4.1.1 Definition

- Classification is also called **Supervised Learning**
- **Supervision**
  - → The training data (observations, measurements, etc) are used to learn a classifier
  - → The training data are **labeled** data
  - → New data (**unlabeled**) are classified

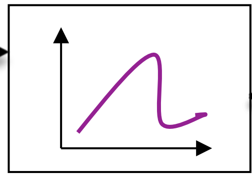
    Using the training data

Training data

Age	Income	Class label
27	28K	Budget-Spenders
35	36K	Big-Spenders
65	45K	Budget-Spenders

Unlabeled data

Age	Income
29	25K



Classifier

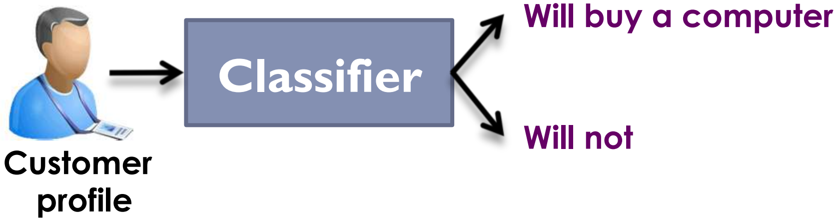
Class label [Budget Spender]

Numeric value [Budget Spender (0.8)]

- **Principle**
  - → Construct models (functions) based on some training examples
  - → Describe and distinguish classes or concepts for future prediction
  - → Predict some unknown class labels

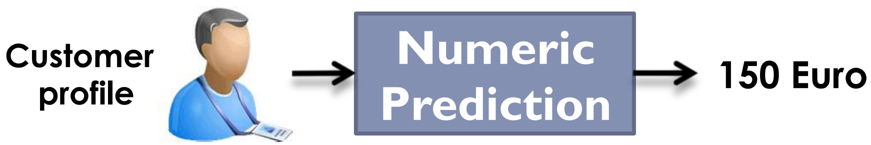
4.1.2 Classification vs. Prediction

- **Classification**
  - → Predicts categorical class labels (discrete or nominal)
  - → Use labels of the training data to classify new data
- **Example**



- A model or classifier is contsructed to predict **categorical labels** such as “safe” or “risky” for a loan application data.

- **Prediction**
  - → Models continuous-valued functions, i.e., predicts unknown or missing values
- **Example**
  - → A marketing manager would like to predict how much a given costumer will spend during a sale



  - → Unlike classification, it provides ordered values
  - → **Regression** analysis is used for prediction
  - → Prediction is a short name for **numeric prediction**

4.1.3 Classification Steps (1/2)

There are two main steps in classification

- **Step1: Model Construction (learning step, or training step)**
  - → Construct a classification model based on **training data**
  - → **Training data**
    - A set of tuples
    - Each tuple is assumed to belong to a predefined class
    - Labeled data (ground truth)
  - → **How a classification model looks like?**

    A classification model can be represented by one of the following forms:
    - Classification rules
    - Decision trees
    - Mathematical formulae

4.1.3 Classification Steps (2/2)

- **Step2: Model Usage**

  Before using the model, we first need to test its accuracy

  → **Measuring model accuracy**
  - To measure the accuracy of a model we need **test data**
  - Test data is similar in its structure to training data (labeled data)
  - **How to test?**

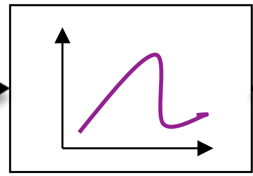
    The known label of test sample is compared with the classified result from the model

Test data

Age	Income	Class label
25	30K	Budget-Spenders
40	50k	Big-Spender

Tuple

Age	Income
25	30K



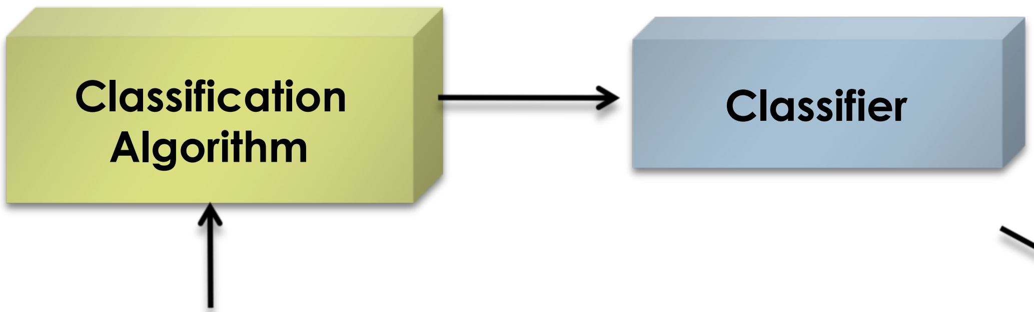
Classifier

Budget-Spender

  - **Accuracy rate** is the percentage of test set samples that are correctly classified by the model
  - **Important**: test data should be independent of training set, otherwise over-fitting will occur

  → **Using the model:** If the accuracy is acceptable, use the model to **classify data** tuples whose class labels are not known

Model Construction



If age <30 & income <30k
Then, Budget-Spender

If age <30 & income > 30k
Then, Big-Spender

If 30<age<60 and income >30k
Then Big-spender

If 30<age<60 and income <30k
Then Budget-spender

If age >60, Then Budget-spender

Training data

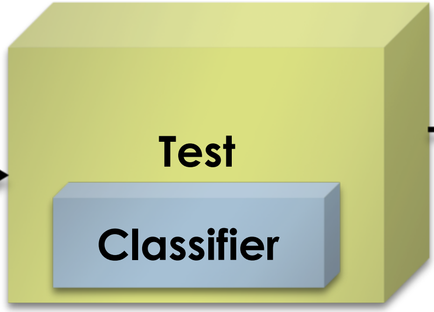
Age	Income	Class label
27	28K	Budget-Spender
35	36K	Big-Spender
38	28K	Budget-Spender
65	45K	Budget-Spender
20	18k	Budget-Spender
75	40k	Budget-Spender
28	50k	Big-Spender
40	60k	Big-Spender
60	65k	Big-Spender

Model Usage

1-Test the classifier

Test Data

Age	Income	Class label
27	28K	Budget-Spenders
25	36K	Big-Spenders
70	45K	Budget-Spenders
40	35k	Big-Spender



Accuracy

2-If acceptable accuracy

Unlabeled data

Age	Income
18	28K
37	40K
60	45K
40	36k

Classified data

Age	Income	Class label
18	28K	Budget-Spenders
37	40K	Big-Spenders
60	45K	Budget-Spenders
40	36k	Budget-Spenders

4.1.4 Issues of Classification & Prediction

Data Preparation

- **Data cleaning**
  - Perform a preprocessing step to reduce noise and handle missing values
  - How to handle missing values?
    - E.g., replacing a missing value with the most commonly occurring value for that attribute, the most probable value based on statistics (prediction) (example)
- **Relevance analysis** (feature selection)
  - Remove irrelevant or redundant attributes
- **Data transformation and reduction**
  - Generalize data to (higher concepts, discretization)
  - Normalizing attribute values (income vs. binary attributes)
  - Reduce the dimensionality of the data

4.1.4 Issues of Classification & Prediction

Evaluating Classification Methods

- **Accuracy**
  - classifier accuracy: the ability of a classifier to predict class labels
  - predictor accuracy: how close is the predicted value from true one.
- **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**
  - Level of understanding and insight provided by the model

Summary of section 4.1.1

- **Classification** predicts class labels
- Numeric **prediction** models continued-valued functions
- Two steps of classification: **1) Training**
  2) Testing and using
- Data cleaning and Evaluation are the main issues of classification and prediction

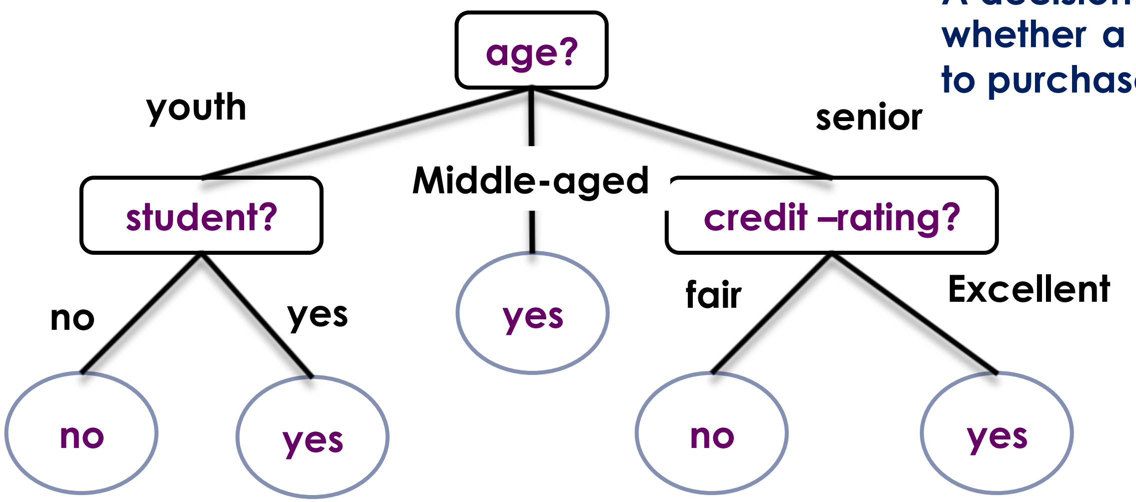
Chapter 4: Classification & Prediction

- **4.1 Basic Concepts of Classification and Prediction**
  - 4.1.1 Definition
  - 4.1.2 Classification vs. Prediction
  - 4.1.3 Classification Steps
  - 4.1.4 Issues of Classification and Prediction
- **4.2 Decision Tree Induction**
  - 4.2.1 The Algorithm
  - 4.2.2 Attribute Selection Measures
  - 4.2.3 Tree Pruning
  - 4.2.4 Scalability and Decision Tree Induction
- **4.3 Bayes Classification Methods**
- **4.4 Rule Based Classification**
- **4.5 Lazy Learners**
- **4.6 Prediction**
- **4.7 How to Evaluate and Improve Classification**

4.2 Decision Tree Induction

- **Decision tree induction** is the learning of decision trees from class-labeled training tuples
- A decision tree is a flowchart-like tree structure
  - → **Internal nodes** (non leaf node) denotes a test on an attribute
  - → **Branches** represent outcomes of tests
  - → **Leaf nodes** (terminal nodes) hold class labels
  - → **Root node** is the topmost node

A decision tree indicating whether a customer is likely to purchase a computer



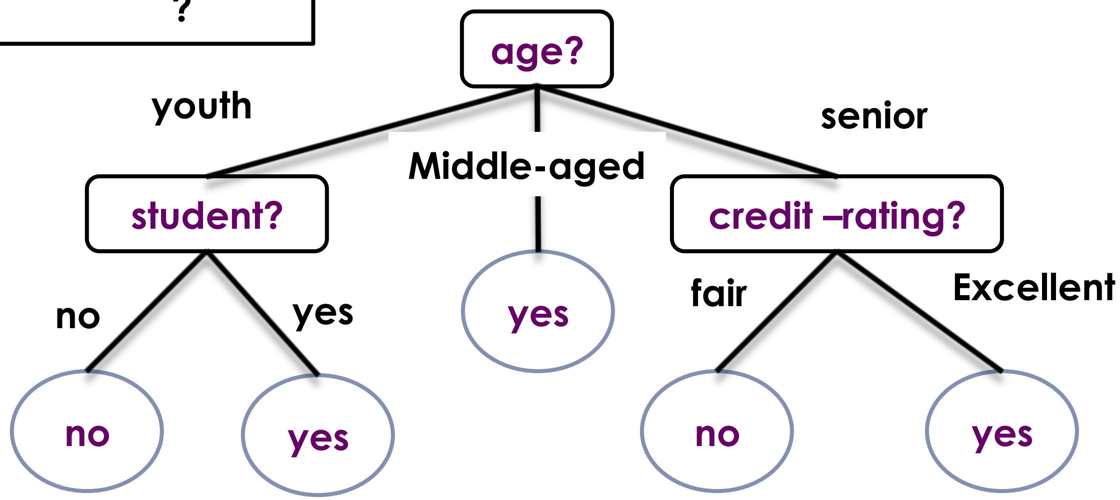
Class-label Yes: The customer is likely to buy a computer
Class-label no: The customer is unlikely to buy a computer

4.2 Decision Tree Induction

- **How are decision trees used for classification?**
  - The attributes of a tuple are tested against the decision tree
  - A path is traced from the root to a leaf node which holds the prediction for that tuple
- **Example**

RID	age	income	student	credit-rating	Class
1	youth	high	no	fair	?

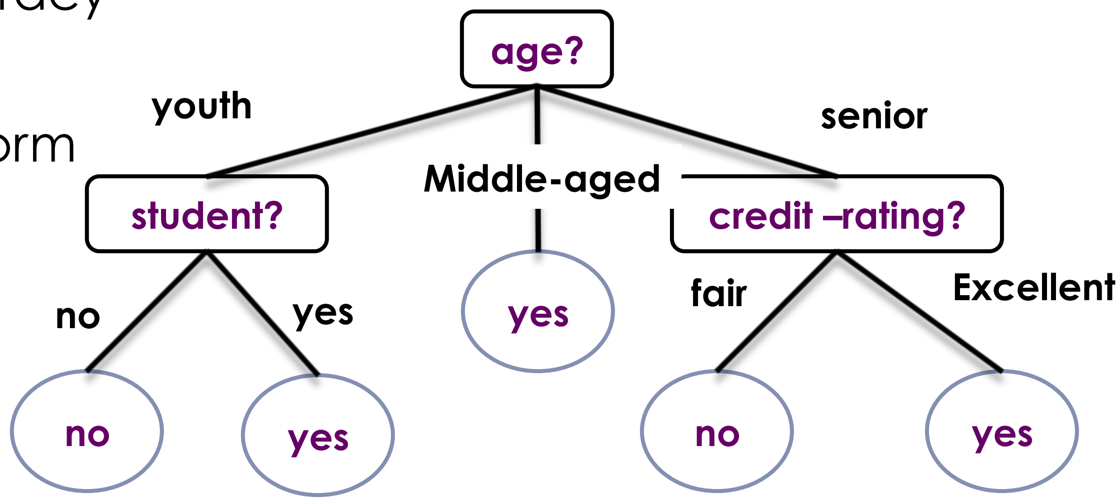
  - Test on age: youth
  - Test of student: no
  - Reach leaf node
  - **Class NO:** the customer Is Unlikely to buy a computer



A decision tree indicating whether a customer is likely to purchase a computer

4.2 Decision Tree Induction

- **Why decision trees classifiers are so popular?**
  - The construction of a decision tree does not require any domain knowledge or parameter setting
  - They can handle high dimensional data
  - Intuitive representation that is easily understood by humans
  - Learning and classification are simple and fast
  - They have a good accuracy
- **Note**
  - Decision trees may perform differently depending on The data set
- **Applications**
  - medicine, astronomy
  - financial analysis, manufacturing
  - and many other applications



A decision tree indicating whether a customer is likely to purchase a computer

4.2.1 The Algorithm

Principle

- Basic algorithm (adopted by ID3, C4.5 and CART): a **greedy algorithm**
- Tree is constructed in a top-down recursive divide-and-conquer manner

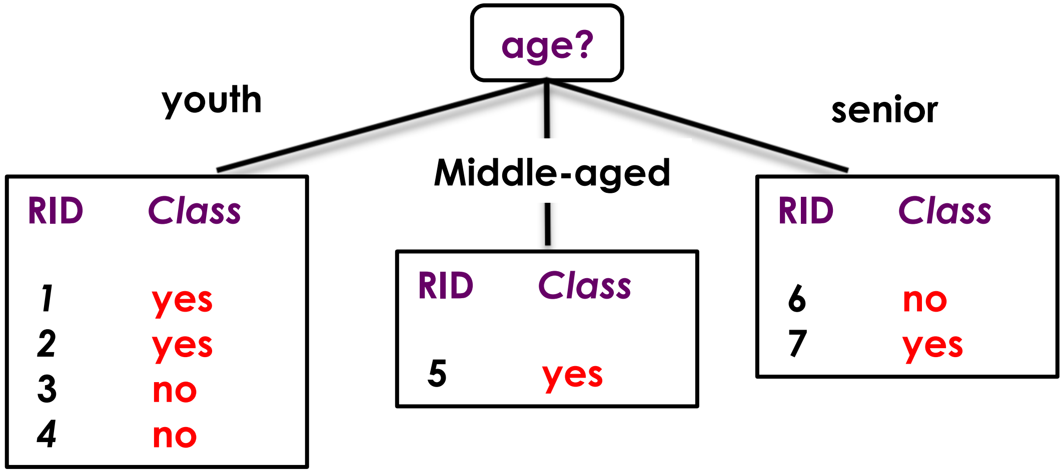
Iterations

- At start, all the training tuples are at the root
- Tuples are partitioned recursively based on selected attributes
- Test attributes are selected on the basis of a heuristic or statistical measure (e.g., **information gain**)

Stopping conditions

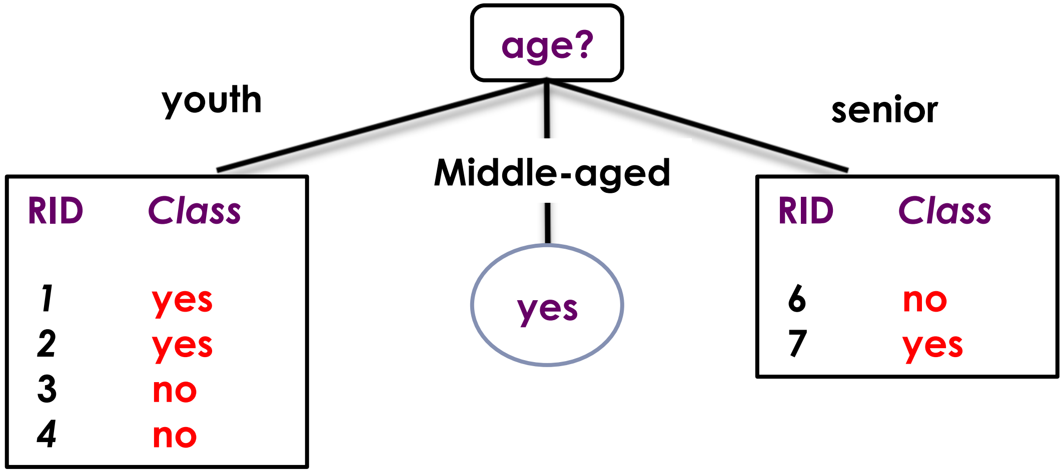
- All samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – **majority voting** is employed for classifying the leaf
- There are no samples left

Example



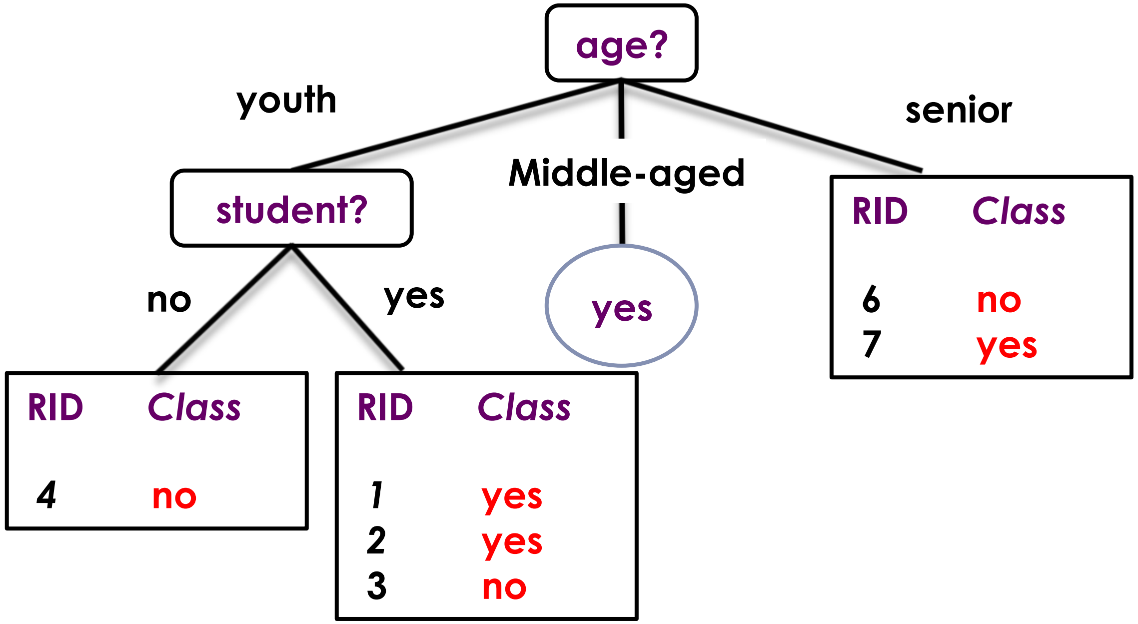
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



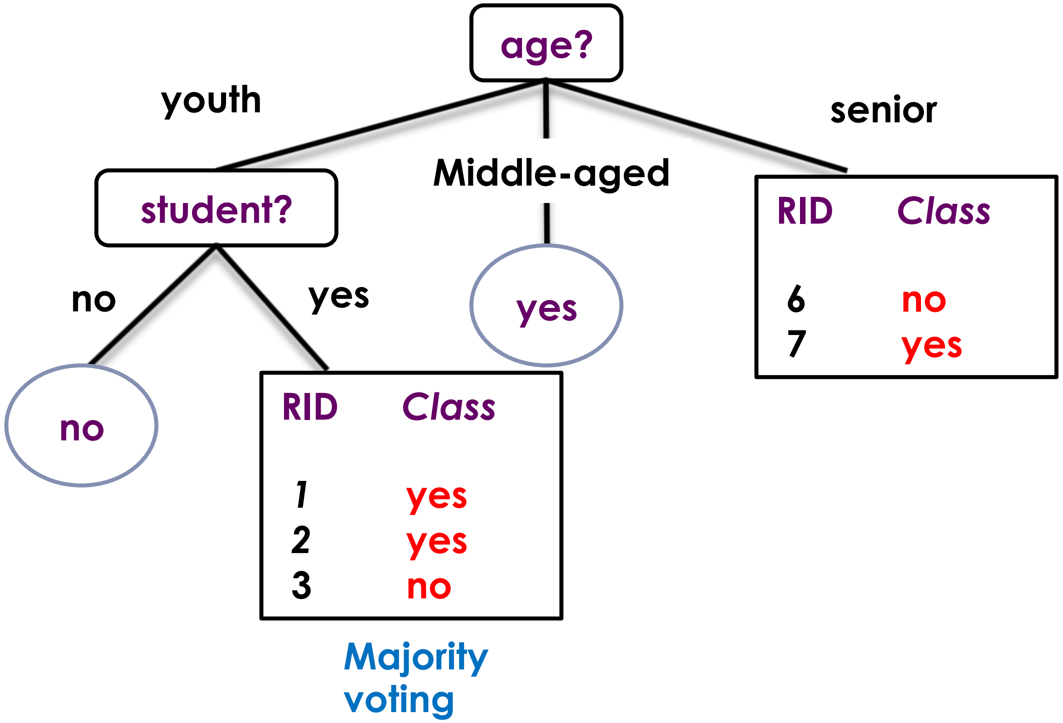
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



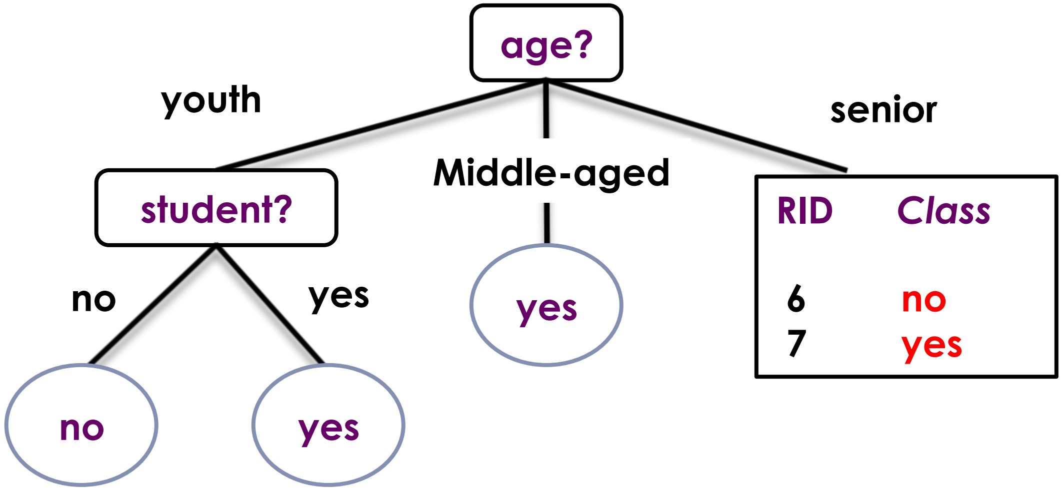
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



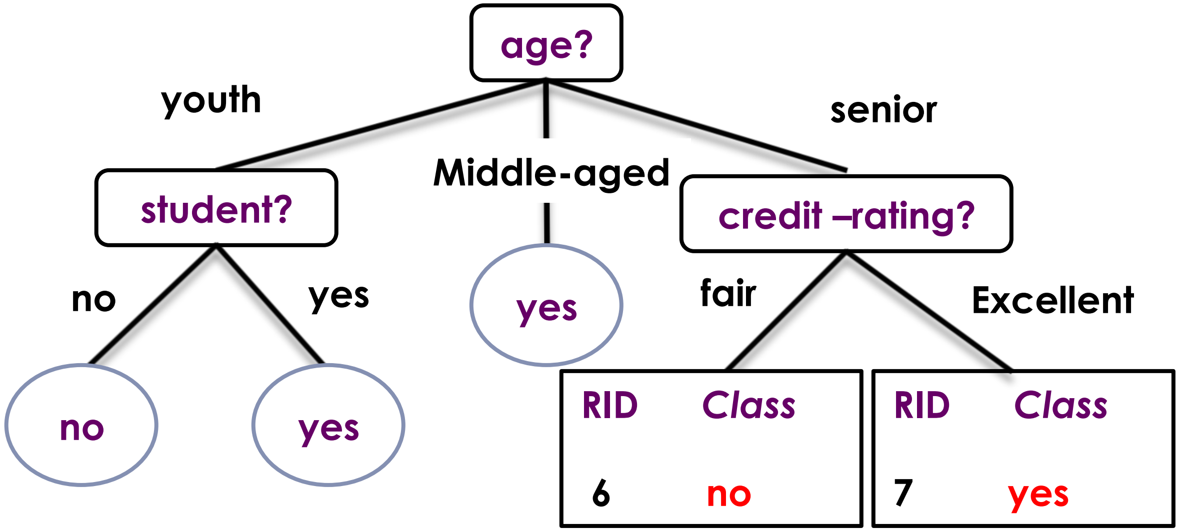
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



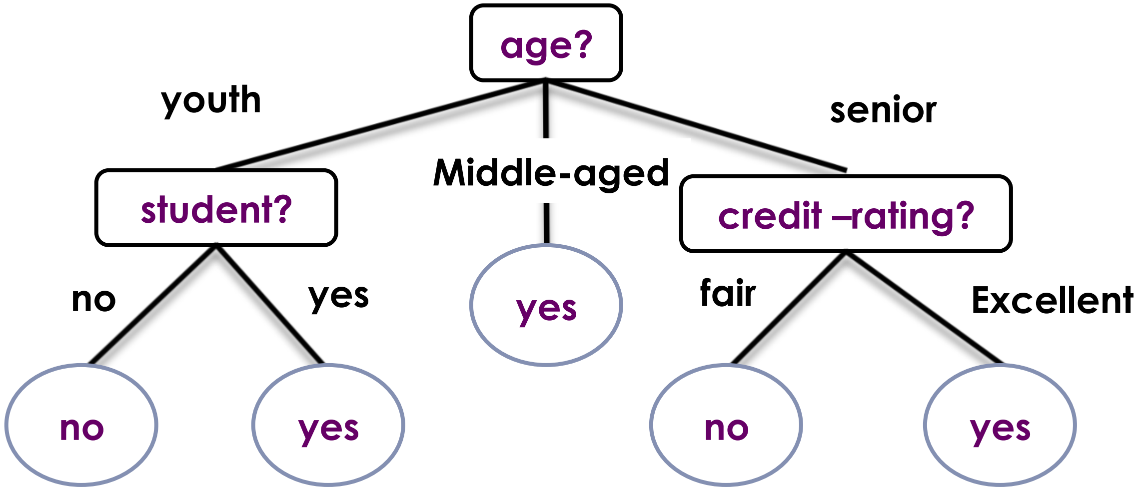
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



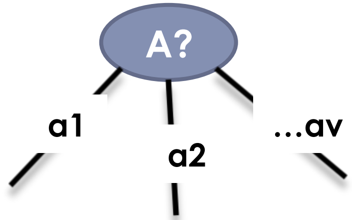
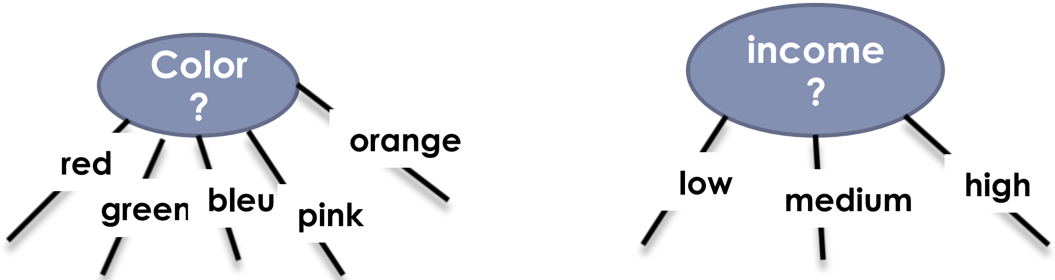
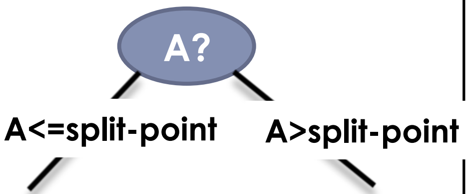
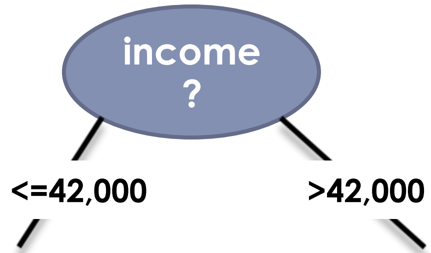
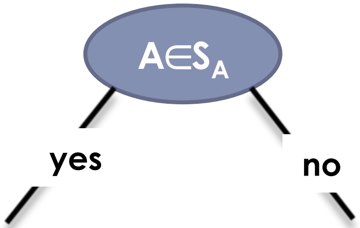
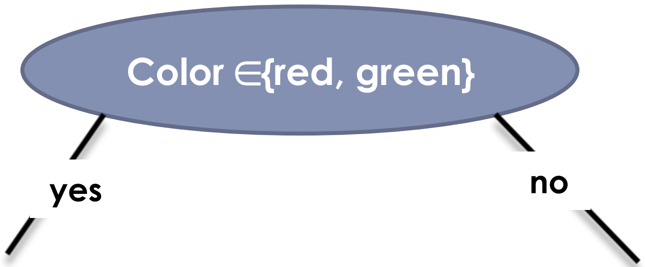
RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	no
7	senior	yes	excellent	yes

Example



RID	age	student	credit-rating	Class: buys_computer
1	youth	yes	fair	yes
2	youth	yes	fair	yes
3	youth	yes	fair	no
4	youth	no	fair	no
5	middle-aged	no	excellent	yes
6	senior	yes	fair	yes
7	senior	yes	excellent	no

Three Possible Partition Scenarios

Partitioning scenarios	Examples
Discrete-valued 	
Continuous-valued 	
Discrete-valued+ binary tree 	

4.2.2 Attribute Selection Measures

- An **attribute selection measure** is a heuristic for selecting the splitting criterion that "best" separates a given data partition **D**

Ideally

- → Each resulting partition would be pure
- → A **pure** partition is a partition containing tuples that all belong to the same class

- Attribute selection measures (splitting rules)
  - → Determine how the tuples at a given node are to be split
  - → Provide ranking for each attribute describing the tuples
  - → The attribute with highest score is chosen
  - → Determine a **split point** or a **splitting subset**
- **Methods**
  - → Information gain
  - → Gain ratio
  - → Gini Index

Before Describing Information Gain

Entropy & Bits

You are watching a set of independent random sample of X

- X has 4 possible values:

P(X=A)=1/4, P(X=B)=1/4, P(X=C)=1/4, P(X=D)=1/4

- You get a string of symbols ACBABBCDADDC…
- To transmit the data over binary link you can encode each symbol with bits (A=00, B=01, C=10, D=11)
- You need 2 bits per symbol

http://www.cs.cmu.edu/~guestrin/Class/10701-S06/Handouts/recitations/recitation-decision_trees-adaboost-02-09-2006.ppt
(Next 4 slides are from this link)

Before Describing Information Gain

Fewer Bits – example 1

- Now someone tells you the probabilities are not equal

P(X=A)=1/2, P(X=B)=1/4, P(X=C)=1/8, P(X=D)=1/8

- **Now, it is possible to find coding that uses only 1.75 bits on the average. How?**

→ E.g., Huffman coding

Before Describing Information Gain

Fewer Bits – example 2

- Suppose there are three equally likely values

P(X=A)=1/3, P(X=B)=1/3, P(X=C)=1/3

- Naïve coding: A = 00, B = 01, C=10
- Uses 2 bits per symbol
- **Can you find coding that uses 1.6 bits per symbol?**
- **In theory** it can be done with **1.58496 bits**

Before Describing Information Gain

Entropy – General Case

- Suppose X takes *n* values, V_1, V_2,… V_n, and

$P(X=V1)=p_1$, $P(X=V2)=p_2$, … $P(X=Vn)=p_n$

- What is the smallest number of bits, on average, per symbol, needed to transmit the symbols drawn from distribution of X? It's

$$H(X) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- **H(X)** = the **entropy** of X

Before Describing Information Gain

Entropy is a measure of the average information content one is missing when one does not know the value of the random variable

- **High Entropy**
  - → X is from a **uniform** like distribution
  - → Flat histogram
  - → Values sampled from it are less predictable
- **Low Entropy**
  - → X is from a varied (**peaks and valleys**) distribution
  - → Histogram has many lows and highs
  - → Values sampled from it are more predictable

1st approach: Information Gain Approach

D: the current partition

N: represent the tuples of partition D

- Select the attribute with the highest information gain (based on the work by Shannon on information theory)
- This attribute
  - → minimizes the information needed to classify the tuples in the resulting partitions
  - → reflects the least randomness or "impurity" in these partitions
- **Information gain** approach minimizes the expected number of tests needed to classify a given tuple and guarantees a simple tree

Information Gain Approach

Step1: compute **Expected information** (entropy) of **D -Info(D)-**

- The expected information needed to classify a tuple in **D** is given by:

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

  - **m**: the number of classes
  - $\mathbf{p_i}$: the probability that an arbitrary tuple in **D** belongs to class $\mathbf{C_i}$ estimated by: $|C_{i,D}|/|D|$ (proportion of tuples of each class)
  - A **log** function to the base 2 is used because the information is encoded in bits
- **Info(D)**
  - The average amount of information needed to identify the class label of a tuple in D
  - It is also known as **entropy**

Info(D): Example

RID	age	income	student	credit-rating	class:buy_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

m=2 (the number of classes)
N= 14 (number of tuples)

9 tuples in class yes
5 tuples in class no

The entropy (Info(D)) of the current partition D is:

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}$$

Information Gain

Step2: for each attribute, compute the amount of information needed to arrive at an exact classification after portioning using that attribute

- suppose that we were to partition the tuples in D on some attribute **A $\{a_1,\ldots,a_v\}$**
  - If A discrete: v outcomes

    D split into **v** partitions **$\{D_1,D_2,\ldots D_v\}$**
  - **Ideally** D_i partitions are pure but it is unlikely
  - This amount of information would we still need to arrive at an exact classification is measured by:

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

  - **$|D_j|/|D|$**: the weight of the jth partition
  - **Info(Dj)**: the entropy of partition D_j
  - The smaller the expected information still required, the greater the purity of the partitions

$Info_{age}(D)$: Example

RID	age	income	student	credit-rating	class:buy_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

Using attribute age

1st partition (youth) **D1** has **2 yes** and **3 no** — **I(2,3)= entropy of D1(Info(D1))**
2nd partition (middle-aged) **D2** has **4 yes** and **0 no** — **I(4,0)= entropy of D2(Info(D2))**
3rd partition (senior) **D3** has **3 yes** and **2 no** — **I(3,2)= entropy of D3(Info(D3))**

$Info_{age}$(D) is:

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

Information Gain Approach

- **Step1:** compute **Expected information** (entropy)of the current partition **Info(D)**
- **Step2:** compute **Info$_A$(D)**, the amount of information would we still need to arrive at an exact classification after partitioning using attribute A
- **Step3:** compute information gain:
- Information gain by branching on A is

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

- **Information gain** is the expected reduction in the information requirements caused by knowing the value of A
  - The attribute **A** with the **highest information gain** ,(Gain(A)), is **chosen** as the splitting attribute at node N

$\text{Info}_{age}(D)$: Example

RID	age	income	student	credit-rating	class:buy_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

$$1)\quad Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits}$$

$$2)\quad Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

$$3)\quad Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly, Gain(Income)=0.029, Gain(student)=0.151, Gain(credit_rating)=0.48

Attribute age has the highest gain ⇒ It is chosen as the splitting attribute

Note on Continuous Valued Attributes

- Let attribute A be a continuous-valued attribute

- Must determine 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*
    - $(a_i+a_{i+1})/2$ 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***
  - $\mathbf{D_1}$ is the set of tuples in D satisfying **A ≤ split-point**, and $\mathbf{D_2}$ is the set of tuples in D satisfying **A > split-point**