

# **DI 501 Introduction to Data Informatics**

## **Lecture 5**

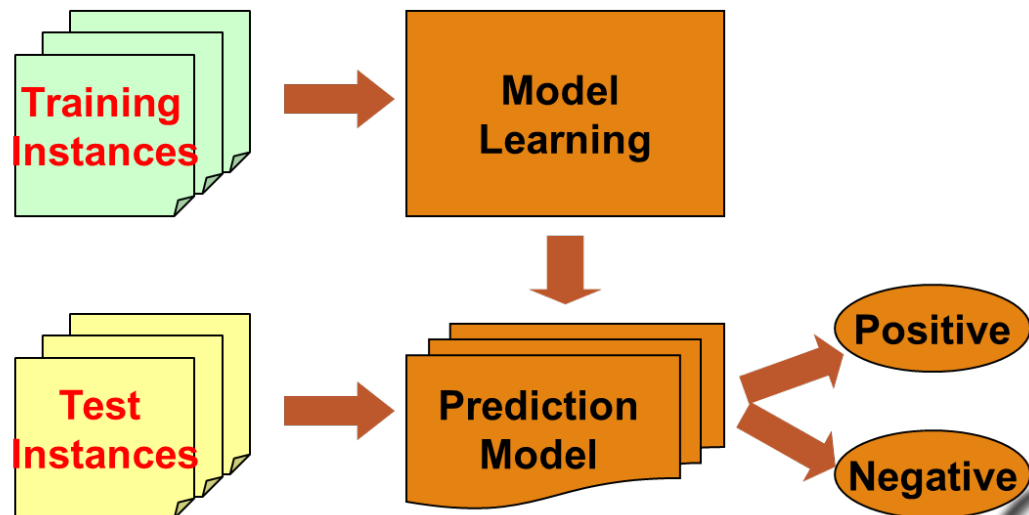
### **Classification- Part1**

# Supervised Learning for Classification

- **Supervision:** The training data such as observations or measurements are accompanied by labels indicating the classes which they belong to.
- New data is classified based on the models built from the training set.

Training Data with class label:

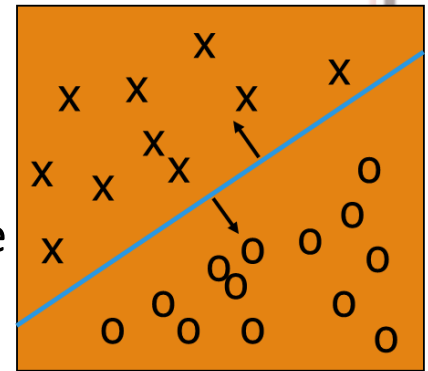
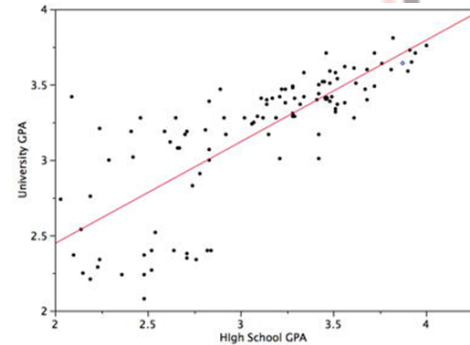
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes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



# Prediction Problems:

## Regression vs. Classification

- **Regression** - Numeric Prediction: Model continuous-valued functions. (i.e., predict unknown or missing values).
- **Classification**: Predict categorical class labels (discrete or nominal).
  - Construct a model based on the training set and the class labels (the values in a classifying attribute) and use it in classifying new instances.
- Typical applications of classification:
  - Credit/loan approval.
  - Medical diagnosis: if a tumor is cancerous or benign.
  - Fraud detection: if a transaction is fraudulent.
  - Web page categorization: which category it is.



# Classification:

## Model Construction, Validation and Testing

- **Model construction** (aka induction): the creation of models from data.
  - Each instance is assumed to belong to a predefined class shown by the label.
  - We generalize from specific cases to general rules.
  - The set of instances used for model construction is **training set**.
  - Our models are general rules in a statistical sense because they usually do not hold 100% of the time.
- **Model Testing**: Estimate the performance (e.g., accuracy) of the model.
  - The known label of test instance is compared with the result from the model.
  - Accuracy: % of test set instances that are correctly classified by the model.
  - Test set is independent of training set .
- **Model Validation**: select or refine models by using a separate data set called **validation set**.
- **Model Deployment**: If the performance is acceptable, use the model to classify new instances.

# Models

- A **model** could be a mathematical formula, a logical statement such as a rule, or hybrid of these.
- A model is a simplified representation of reality created to serve a purpose.
  - It is simplified based on some assumptions about what is and is not important for the specific purpose.
- **Predictive models** are used for estimating the unknown value of interest (target).
- **Descriptive models** are used to gain insight into the underlying phenomenon or process.

# Classification:

## Binary vs. Multi-class

- There are basically two kinds of classification problems:
  - **Binary classification:** Absence or presence of something,  $\{0,1\}$  or  $\{\text{true}, \text{false}\}$ 
    - Example: e-mail content --> spam or not-spam
  - **Multi-class classification:** Three or more classes,  $\{0,1,2,3,4,5\}$  or  $\{a,b,c,d,e,f\}$ 
    - Example: Symptoms --> disease

# Supervised Learning for Classification

## Lazy vs. Eager Learning

- Lazy vs. eager learning
  - **Lazy learning** (e.g., instance-based learning, case-based learning):
    - Just store data set **without** learning from it
    - Start classifying data when it receive test data
    - So it takes less time learning and more time classifying data
  - **Eager learning** (e.g. decision trees, neural networks, Naïve Bayes):
    - When it receives data set, it starts classifying (learning)
    - Then it does not wait for test data to learn
    - So it takes long time learning and less time classifying data



# Supervised Learning for Classification

## Lazy vs. Eager Learning

- Lazy: less time in training but more time in predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form an implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space





# Lazy Learning

# Instance-based representation

- Simplest form of learning: *rote learning*
  - Training instances are searched for instance that most closely resembles new instance
  - The instances themselves represent the knowledge
  - Also called *instance-based* learning
- Similarity function defines what's “learned”
- Instance-based learning is *lazy* learning
- Methods:
  - *nearest-neighbor*
  - *k-nearest-neighbor*
  - ...

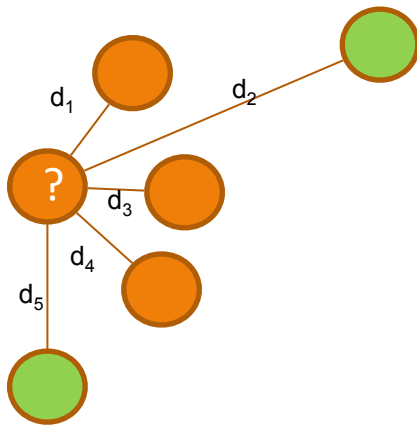
# Lazy Learner: Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing (“lazy evaluation”) until a new instance must be classified
- Typical approaches
  - $k$ -nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference



# The $k$ -Nearest Neighbor Algorithm

- All instances correspond to points in the  $n$ -D space
- The nearest neighbor are defined in terms of Euclidean distance,  $\text{dist}(\mathbf{X}_1, \mathbf{X}_2)$
- Target function could be discrete- or real- valued
- For discrete-valued,  $k$ -NN returns the most common value among the  $k$  training examples nearest to  $X_q$



For  $k=3$ , ? will be categorized as orange  
Three orange circles are the closest to ? circle.

# The distance function

- Simplest case: one numeric attribute
  - Distance is the difference between the two attribute values involved (or a function thereof)
- Several numeric attributes: normally, Euclidean distance is used and attributes are normalized
- Nominal attributes: distance is set to 1 if values are different, 0 if they are equal
- Are all attributes equally important?
  - Weighting the attributes might be necessary

# Instance-based learning

- Distance function defines what's learned
- Most instance-based schemes use *Euclidean distance*:

$$\sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + \dots + (a_k^{(1)} - a_k^{(2)})^2}$$

$\mathbf{a}^{(1)}$  and  $\mathbf{a}^{(2)}$ : two instances with  $k$  attributes

- Taking the square root is not required when comparing distances
- Other popular metric: *city-block (Manhattan) metric*
  - Adds differences without squaring them



# Normalization and other issues

- Different attributes are measured on different scales  $\Rightarrow$  need to be *normalized*:

$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i} \quad \text{or} \quad a_i = \frac{v_i - \text{Avg}(v_i)}{\text{StDev}(v_i)}$$

$v_i$ : the actual value of attribute  $i$

- Nominal attributes: distance either 0 or 1
- Common policy for missing values: assumed to be maximally distant (given normalized attributes)

# Model Selection

- Select  $k$  values within an interval:
  - Try odd numbers.
- Divide your dataset into training, and testing.
- For each instance in your training dataset, find the closest  $k$  points and predict the label based on the neighbors.
- Compare the predicted value with the original one. Compute the error.
- Choose the  $k$  which resulted in minimum error.





# Example

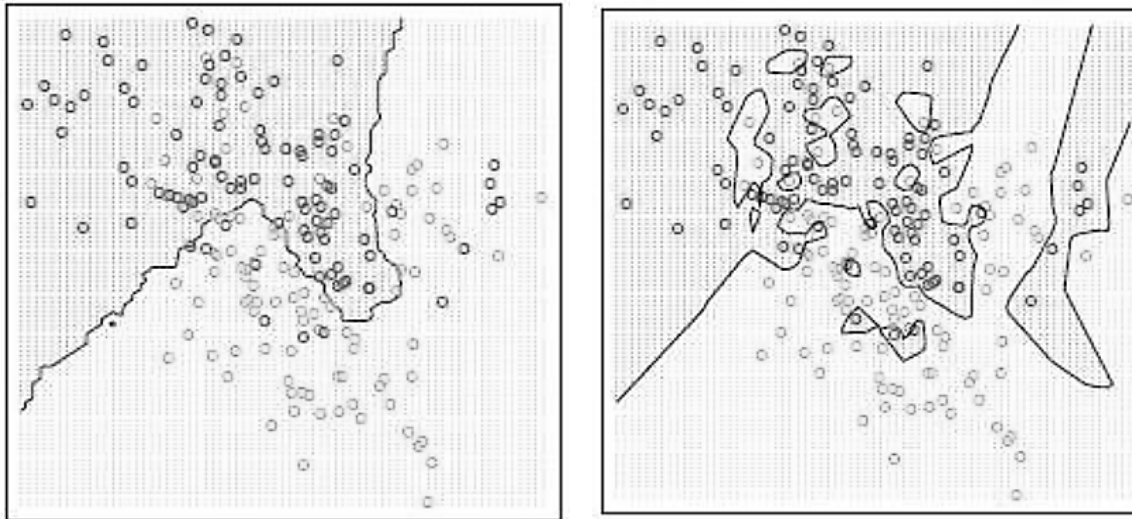


Figure (a) and (b) show the same data sets classified using different k-NN parameters. There are two class variables: dark circles and light color ones. The lines indicate the decision boundaries corresponding to 2 different k-NN models separately. Which one could be produced by 1-nn and 15-nn respectively? Give your reasons.

# Discussion of k-NN

- Often very accurate
- ... but slow:
  - simple version scans entire training data to derive a prediction
- Assumes all attributes are equally important
  - Remedy: attribute selection or weights
- Possible remedies against noisy instances:
  - Take a majority vote over the  $k$  nearest neighbors
  - Removing noisy instances from dataset (difficult!)
- Statisticians have used  $k$ -NN since early 1950s
- $k$  should be chosen as an odd number since with even numbers, there might be problems with predicted classes at par.
- Not effective with very high dimensional cases
  - Euclidean distance does not make sense with curse of dimensionality



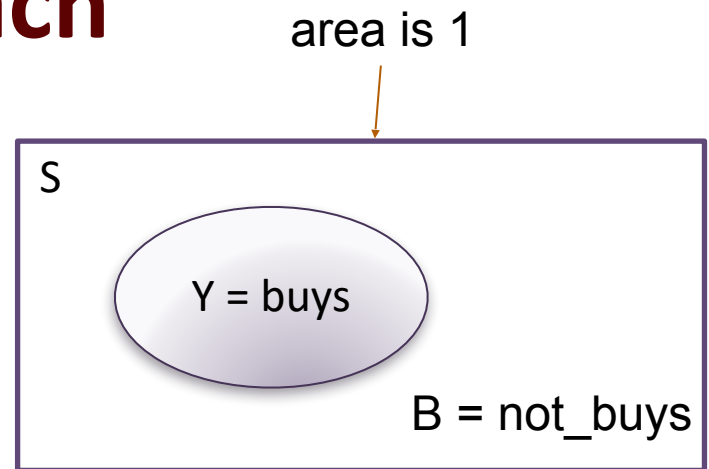
# Eager Learning

# Classification

- We want to segment the population with respect to something that we would like to predict or estimate.
- Target can be something that
  - we would like to avoid such as a given customer is likely not to pay off his/her account balance, or
  - we want to achieve such as a given customer is likely to respond to a special offer.
- The problem may be supervised if we have some training data where we know the value of the target attribute.
- There is an uncertainty about the value of the target
  - Are there any features that reduce the uncertainty?

# A Probabilistic Approach

- Example:
  - Y: Customer buys a computer
  - Two classes:
    - buys: customer buys a computer
    - not\_buys: customer does not buy a computer
- S: Sample space: All customers



- $P(Y = \text{buys})$ : shaded area
- We can use the available data to estimate  $P(Y = \text{buys})$  as:

$$P(\text{buys}) = \frac{\# \text{ of customers buying a computer}}{\# \text{ of customers}}$$

# A Probabilistic Approach

- Other features of a customer may be considered to improve prediction
- e.g.,  $X$ : the customer is a student
- $X \in \{student, not\_student\}$
- $Y \in \{buys, not\_buy\}$

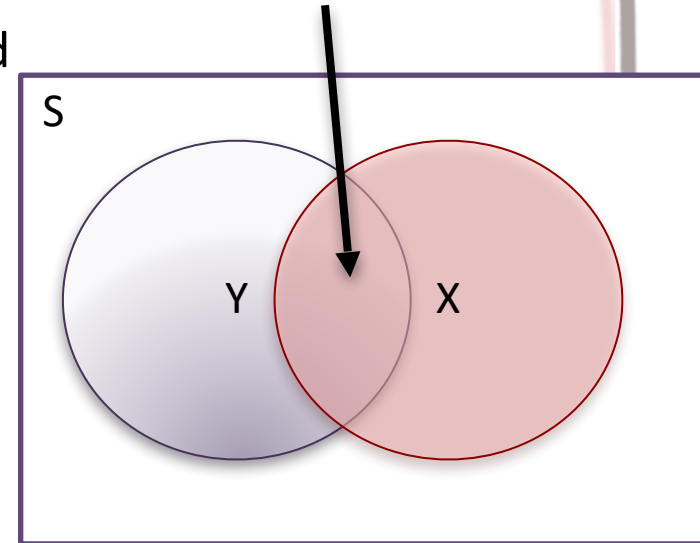
$$P(buys | student) = \frac{P(buys \cap student)}{P(student)}$$

- Which can be estimated as:

$$P(buys | student) = \frac{\# \text{ of student customers buying computer}}{\# \text{ of student customers}}$$

- If we know that the customer is a student, the probability of the customer buys a computer may be smaller or larger than  $P(Y = buys)$

$$P(Y \cap X)$$



$$P(Y | X) = \frac{P(Y \cap X)}{P(X)}$$

# Dependent and Independent Events

- If knowing something about  $X$  happens gives us information about whether  $Y$  happens (and vice versa), events  $X$  and  $Y$  are **dependent**.

- When two events are **independent**, then by definition we have:

$$P(X \cap Y) = P(X)P(Y)$$

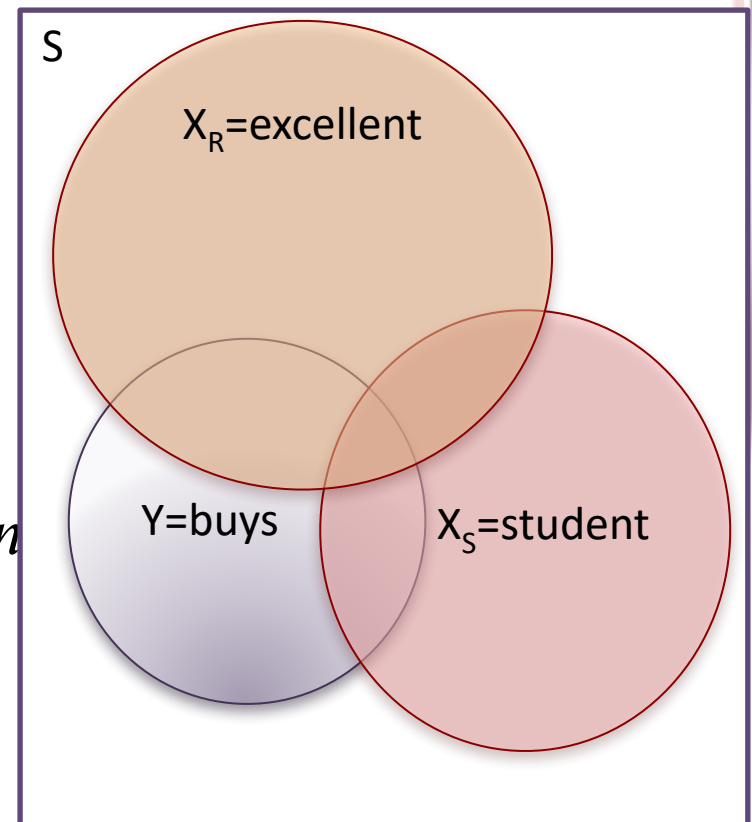
- Hence, if  $X$  and  $Y$  are independent:

$$\begin{aligned} P(Y|X) &= \frac{P(Y \cap X)}{P(X)} \\ &= \frac{P(Y)P(X)}{P(X)} \\ &= P(Y) \end{aligned}$$

# The Joint Distribution

- There may be two or more relevant features of the customers, such as:
  - $X_R$ : Credit rating
    - $X_R \in \{fair, excellent\}$
  - $X_T$ : Student
    - $X_T \in \{student, not\_student\}$
- Hence,

$$P(Y|X_T \cap X_R) = \frac{P(Y \cap X_T \cap X_R)}{P(X_T \cap X_R)}$$



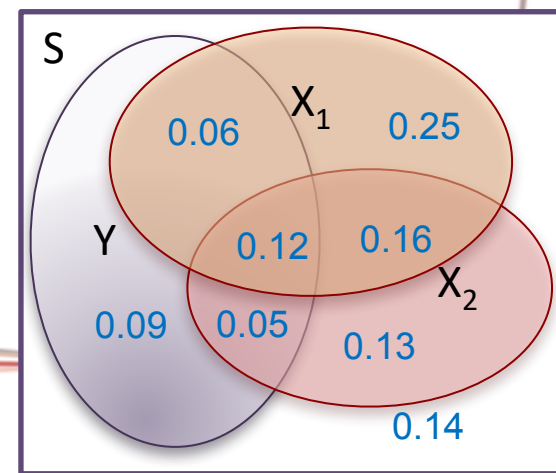


# The Joint Distribution

- If we have  $n$  variables, we can estimate the joint distribution and use it for predictions
- Suppose, we have 3 Boolean variables  $X_1$ ,  $X_2$ , and  $Y$ .
- We can construct a truth table for all possibilities.
  - For  $k$  Boolean variables, we will have  $2^k$  rows
  - Sum of all probabilities should be 1.
  - We have to estimate  $2^k - 1$  probabilities!

Y	$X_1$	$X_2$	$P(Y, X_1, X_2)$
0	0	0	0.14
0	0	1	0.13
0	1	0	0.25
0	1	1	0.16
1	0	0	0.09
1	0	1	0.05
1	1	0	0.06
1	1	1	0.12

$$P(Y = 1 \mid X_1 = 1, X_2 = 0) = ?$$



# The Joint Distribution

- If we have  $n$  variables, we can estimate the joint distribution and use it for predictions
- Suppose, we have 3 Boolean variables  $X_1$ ,  $X_2$ , and  $Y$ .
- We can construct a truth table for all possibilities.
  - For  $k$  Boolean variables, we will have  $2^k$  rows
  - Sum of all probabilities should be 1.
  - We have to estimate  $2^k - 1$  probabilities!

Y	X <sub>1</sub>	X <sub>2</sub>	P(Y,X <sub>1</sub> ,X <sub>2</sub> )
0	0	0	0.14
0	0	1	0.13
0	1	0	0.25
0	1	1	0.16
1	0	0	0.09
1	0	1	0.05
1	1	0	0.06
1	1	1	0.12

$$P(Y = 1 \mid X_1 = 1, X_2 = 0) = P(Y=1, X_1=1, X_2=0) / P(X_1=1, X_2=0) = 0.06 / (0.25 + 0.06) = 6/31$$

# The Joint Distribution

## Example: Who Buys Computer?

- We may consider relevant features of customers, such as:
  - $X_S$ : Age
    - $A \in \{ \leq 30, 31..40, >40 \}$
  - $X_S$ : Student
    - $X_S \in \{ \text{yes, no} \}$
  - $X_I$ : Income:
    - $X_I \in \{ \text{low, medium, high} \}$
  - $X_R$ : Credit Rating:
    - $X_R \in \{ \text{fair, excellent} \}$
- We want to predict:
  - $Y$ : customer buys computer
    - $Y \in \{ \text{buys, not\_buys} \}$
- How many parameters to estimate?
$$= |X_A| * |X_S| * |X_I| * |X_R| * |Y| - 1$$
$$= 3 * 2 * 3 * 2 * 2 - 1 = 71$$
- If we had 50 binary variables??
  - $250 - 1 \cong 10^{15}!!$

# Bayes' Theorem: Basics

- Bayes Theorem:  $P(H_y | X) = \frac{P(X | H_y)P(H_y)}{P(X)}$
- Total Probability Theorem:  $P(X) = \sum_{j=1}^{n_Y} P(X | Y_j)P(Y_j)$
- Let  $X$  be a data instance (“evidence”): class label is known
  - $X_S$ : Age
    - $A \in \{<=30, 31..40, >40\}$
- Let  $H_y$  be a hypothesis that  $X$  belongs to class  $y$  (i.e., posterior probability): the probability that the hypothesis holds given the observed instance  $X$ 
  - We may have  $n_Y$  (i.e., number of distinct classes) such hypotheses.
  - $Y$ : customer buys computer
    - $Y \in \{\text{buys}, \text{not\_buys}\}$
  - Given the  $X_S=23$ , what is the probability of  $H_y=\text{buys}$ ?

# Bayes' Theorem: Basics

- **Bayes Theorem:**  $P(H_y | X) = \frac{P(X | H_y)P(H_y)}{P(X)}$
- **Total Probability Theorem:**  $P(X) = \sum_{j=1}^{n_Y} P(X | Y_j)P(Y_j)$
- $P(H_y)$  (prior probability): the initial probability
  - e.g.,  $X$  will buy computer, regardless of age, income, ...
- $P(X)$ : probability that sample data is observed
- $P(X | H_y)$  (likelihood): the probability of observing the instance  $X$ , given that the hypothesis holds
  - e.g., Given that  $X$  will buy computer, the prob. that  $X$  is 31..40, medium income

# Prediction Based on Bayes Theorem

- $P(Y = y_j | X = x_i) = \frac{P(X = x_i | Y = y_j)P(Y = y_j)}{P(X = x_i)}$
- $P(Y = y_j | X = x_i)$ : Posteriori Probability
  - We should choose  $y_j$  that gives the largest  $P(Y = y_j | X = x_i)$
  - Note that  $P(X = x_i)$  is constant for different  $y_j$ 's.
  - So we can choose  $y_j$  with the largest  $P(X = x_i | Y = y_j)P(Y = y_j)$
- $P(X = x_i | Y = y_j)$ : Likelihood
  - Likelihood of what we just see if class label is  $y_j$
- $P(Y = y_j)$ : Prior Probability
  - What we knew previously

# Naïve Bayes

- $\mathbf{X} = [X_1, X_2, \dots, X_m]$  is usually a vector of random variables.
- Therefore,

$$P(\mathbf{X} = \mathbf{x}_i | Y = y_j) = P(X_1 = x_{i1}, X_2 = x_{i2}, \dots, X_{im} = x_{im} | Y = y_j)$$

- Naïve Bayes Assumption:  $X_a$  and  $X_b$  are conditionally independent given  $Y$  for all  $a \neq b$
- With this assumption:

$$P(Y = y_j | \mathbf{X} = \mathbf{x}_i) = \frac{\prod_{k=1}^m P(X_k = x_{ik} | Y = y_j) * P(Y = y_j)}{P(\mathbf{X} = \mathbf{x}_i)}$$

- Hence, it is sufficient to find  $y_j$  with largest

$$P(Y = y_j | \mathbf{X} = \mathbf{x}_i) \propto \prod_{k=1}^m P(X_k = x_{ik} | Y = y_j) * P(Y = y_j)$$

# Naïve Bayes:

## Categorical vs. Continuous Valued Variables

- If variable  $X_k$  is categorical,  $P(X_k = x_{ik} | Y = y_j)$  can be estimated as

$$P(X_k = x_{ik} | Y = y_j) = \frac{\text{\# of instances with } Y = y_j \text{ and } X_k = x_{ik}}{\text{\# of instances with } Y = y_j}$$

- If variable  $X_k$  is continuous-valued,  $P(X_k = x_{ik} | Y = y_j)$  is usually computed based on normal distribution with an estimated mean  $\mu_{kj}$  and an standard deviation  $\sigma_{kj}$  of variable  $X_k$  for the instances with  $Y = y_j$ .

$$P(X_k = x_{ik} | Y = y_j) = \frac{1}{\sqrt{2\pi\sigma_{kj}^2}} e^{-\frac{(x_{ik} - \mu_{kj})^2}{2\sigma_{kj}^2}}$$



# Naïve Bayes:

## Number of Parameters to Estimate

- Suppose  $Y$  is a binary variable and  $\mathbf{X} = [X_1, X_2, \dots, X_m]$  where  $X_k$ 's are binary variables
- Without conditional independence assumption:
  - $P(Y = y_j)$  and  $P(X_1 = x_{i1}, X_2 = x_{i2}, \dots, X_m = x_{im} \mid Y = y_j)$  should be estimated for  $y_j = 1$  and  $y_j = 0$
  - Hence,  $2 * (2^m - 1) + 1$  variables should be estimated.
- With conditional independence assumption:
  - $P(Y = y_j)$ ,  $P(X_1 = x_{i1} \mid Y = y_j)$ ,  $\dots$ ,  $P(X_m = x_{im} \mid Y = y_j)$  will be estimated for  $y_j = 1$ ,  $y_j = 0$ ,  $x_{ik} = 1$ , and  $x_{ik} = 0$
  - Hence, only  $2m + 1$  parameters should be estimated!

# Naïve Bayes:

## Example

- Class:  $Y = \text{buys\_computer}$ 
  - $Y \in \{\text{yes}, \text{no}\}$
- Data to be classified:
  - $X = (\text{age} \leq 30,$   
income = medium,  
student = yes,  
credit\_rating = fair)

age	income	student	credit_rating	buys_computer
$\leq 30$	high	no	fair	no
$\leq 30$	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
$\leq 30$	medium	no	fair	no
$\leq 30$	low	yes	fair	yes
$> 40$	medium	yes	fair	yes
$\leq 30$	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
$> 40$	medium	no	excellent	no

# Naïve Bayes:

## Example

- $P(Y_i)$ :  $P(Y = \text{"yes"}) = 9/14 = 0.643$

$$P(Y = \text{"no"}) = 5/14 = 0.357$$

- Compute  $P(X|Y_i)$  for each class

$$P(\text{age} = \leq 30 | Y = \text{"yes"}) = 2/9 = 0.222$$

$$P(\text{age} = \leq 30 | Y = \text{"no"}) = 3/5 = 0.6$$

$$P(\text{income} = \text{"medium"} | Y = \text{"yes"}) = 4/9 = 0.444$$

$$P(\text{income} = \text{"medium"} | Y = \text{"no"}) = 2/5 = 0.4$$

$$P(\text{student} = \text{"yes"} | Y = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{student} = \text{"yes"} | Y = \text{"no"}) = 1/5 = 0.2$$

$$P(\text{credit\_rating} = \text{"fair"} | Y = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{credit\_rating} = \text{"fair"} | Y = \text{"no"}) = 2/5 = 0.4$$

- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair})$

$$P(X|Y_i) : P(X|Y = \text{"yes"}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$$

$$P(X|Y = \text{"no"}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$$

$$P(X|Y_i) * P(Y_i) : P(X|Y = \text{"yes"}) * P(Y = \text{"yes"}) = 0.028$$

$$P(X|Y = \text{"no"}) * P(Y = \text{"no"}) = 0.007$$

Therefore, **X belongs to the class ("buys\_computer = yes")**

age	income	student	credit_rating	buys_computer
$\leq 30$	high	no	fair	no
$\leq 30$	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
$\leq 30$	medium	no	fair	no
$\leq 30$	low	yes	fair	yes
$> 40$	medium	yes	fair	yes
$\leq 30$	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
$> 40$	medium	no	excellent	no

# Naïve Bayes:

## Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional probability be non-zero.
  - Otherwise, the predicted probability will be zero as we multiply all probabilities.

$$\prod_{k=1}^m P(X_k = x_{ik} \mid Y = y_j) * P(Y = y_j)$$

- Example: Suppose a data set with 1000 instances:
  - income = low (0), income = medium (990), and income = high (10)
- Solution: We can use **Laplacian correction** (or Laplacian estimator)
  - Adding 1 to each case
    - $P(\text{income} = \text{low}) = 1/(1000 + 3)$
    - $P(\text{income} = \text{medium}) = (990 + 1)/(1000 + 3)$
    - $P(\text{income} = \text{high}) = (10 + 1)/(1000 + 3)$
- The “corrected” probability estimates are close to their “uncorrected” counterparts

# Naïve Bayes Classifier:

## Strengths vs. Weaknesses

- Strengths

- Easy to implement
- Good results obtained in most of the cases

- Weakness

- Assumption: attributes are conditionally independent
  - Causes loss of accuracy
  - We can use **Bayesian Belief Networks** if the assumption does not hold.
- Practically, dependencies exist among variables
  - E.g., Patients: Profile: age, family history, etc.  
Symptoms: fever, cough etc.  
Disease: lung cancer, diabetes, etc.
  - Dependencies among these cannot be modeled by Naïve Bayes Classifier

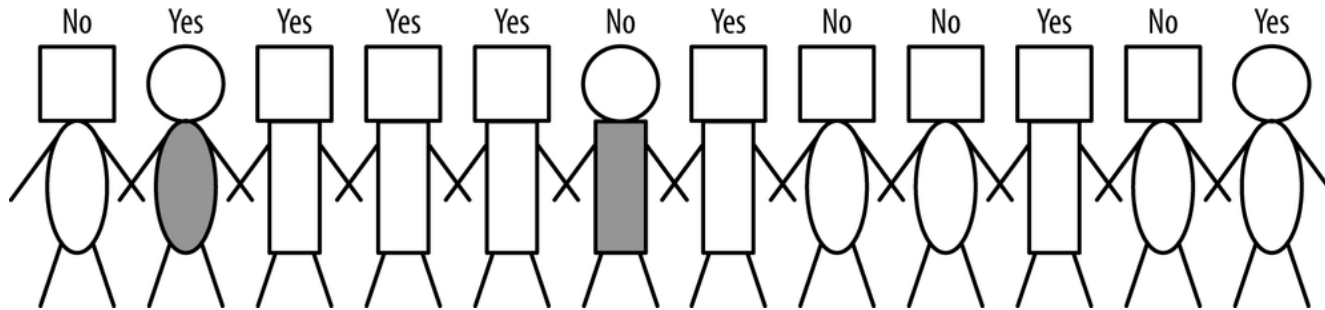
# Classification:

## Informative Attributes

- Are all attributes equally important and useful for classification?
- We can find or select important, **informative attributes** of the entities described by the data.
  - Some attributes may help us to partition the data so that each partition is **as pure as possible**.
    - A partition is pure if all the entities in it belong to the same class.
  - Hence, we can reduce the uncertainty about entity classes.
- Therefore, we can segment the data by progressive attribute selection.

# Selecting Informative Attributes:

## An Example



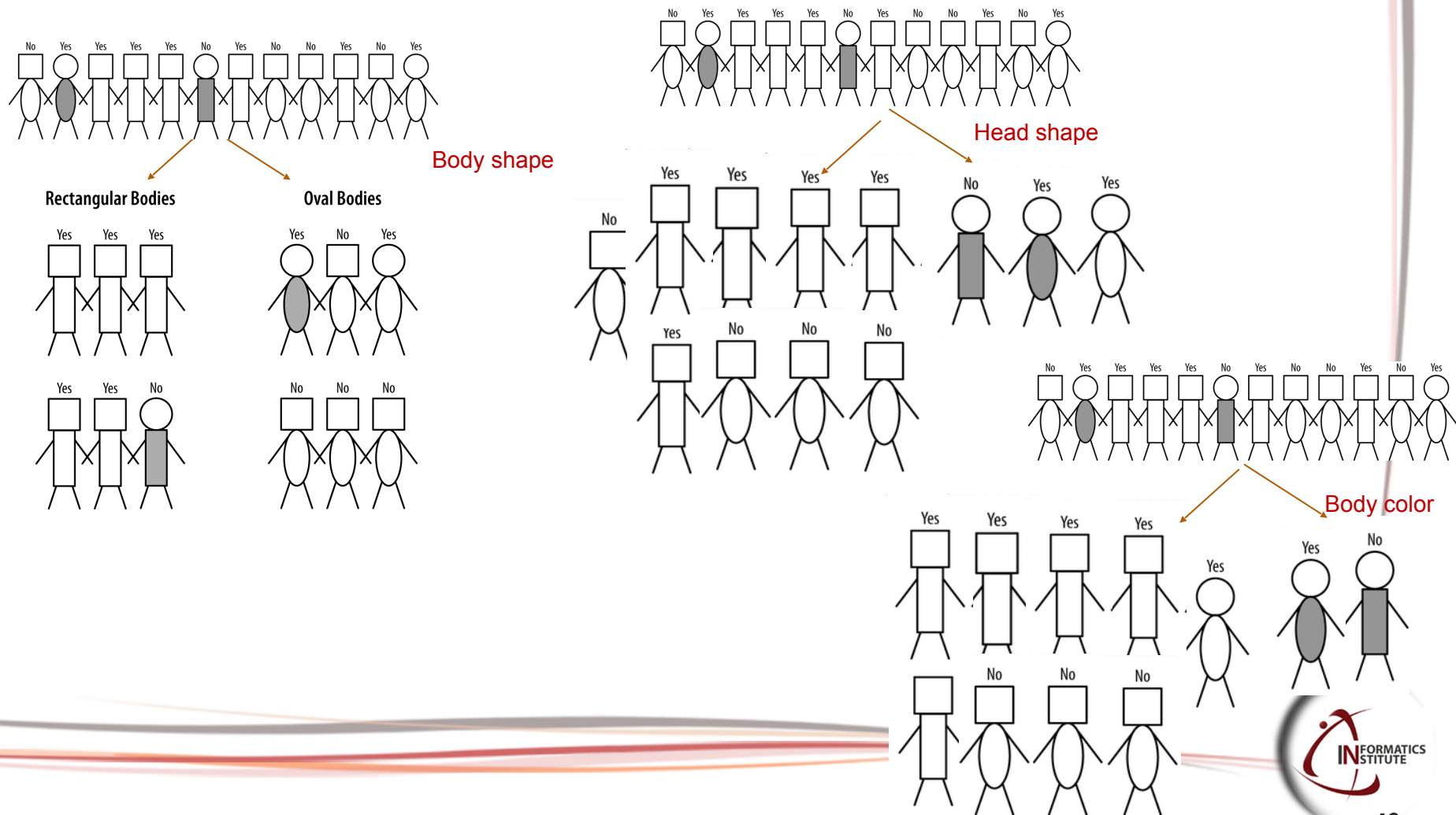
- A loan write-off example.
- Target variable:
  - write-off: *Yes* or *No* (label over each head)
- Attributes:
  - Head-shape: square, circular
  - Body-shape: rectangular, oval
  - Body-color: gray, white

Which of these would be best to segment these people into groups, in a way that will distinguish write-offs from non-write-offs?

# Selecting Informative Attributes:

## An Example: Motivation

- Which attribute best segments these people? Why?

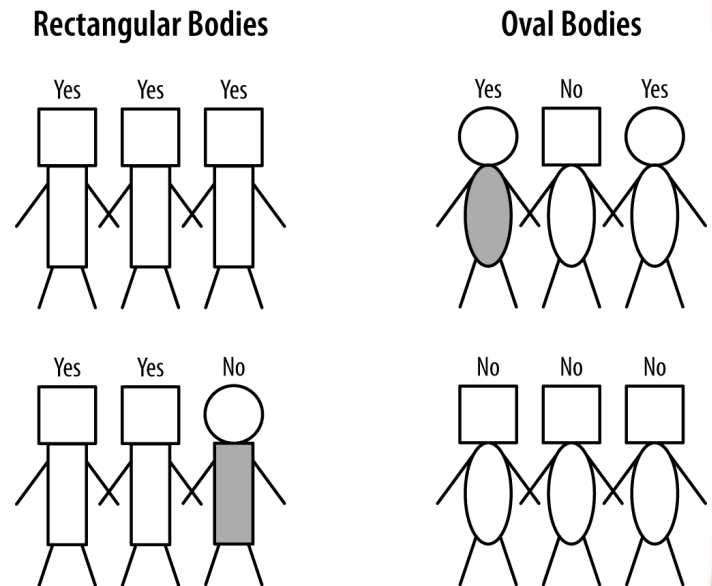




# Selecting Informative Attributes:

## An Example

- Which attribute best segments these people?
- It is seldom possible to find an attribute making segments pure but we can reduce the impurity substantially.
  - Hence, we can predict that instances in one segment has higher or lower write-off rates than those in another segment.
- We need a **purity measure** that can be used to split instances with respect to a chosen target variable.
- The most common splitting criterion is **information gain** which is based on a purity measure called **entropy**.



Attributes:

- Head-shape: square, circular
- Body-shape: rectangular, oval
- Body-color: gray, white

# Calculating Impurity:

## Entropy

- **Entropy** (Information Theory): A measure of uncertainty associated with a random number.
- Suppose a discrete random variable  $Y$  takes  $n$  distinct values  $\{y_1, y_2, \dots, y_n\}$

$$H(Y) = - \sum_{i=1}^n p_i \log_2(p_i) \quad \text{where } p_i = P(Y = y_i)$$

$p_i$  is the proportion of class  $i$  in the data.

- $H(Y)$  is the expected number of bits needed to encode a randomly drawn value of  $Y$  (with the most efficient coding)
  - Higher entropy --> higher uncertainty.
  - Lower entropy --> lower uncertainty.

# Entropy:

## Encoding

- Suppose we have 4 different symbols  $S \in \{a,b,c,d\}$  and the probabilities of occurrences are:
  - $P(S=a)=0.5$
  - $P(S=b)=0.25$
  - $P(S=c)=0.125$
  - $P(S=d)=0.125$
- We have 4 symbols, so a simple coding scheme may assign a 2 bit pattern to each symbol.
- What can be the most efficient coding?

$$\begin{aligned} H(S) &= -0.5 \log_2(0.5) - 0.25 \log_2(0.25) \\ &\quad - 0.125 \log_2(0.125) - 0.125 \log_2(0.125) \\ &= 0.5 * 1 + 0.25 * 2 + 0.125 * 3 + 0.125 * 3 \\ &= 1.75 \text{ bits} \end{aligned}$$

- Consider the following code:
  - $a=0$
  - $b=10$
  - $c=110$
  - $d=111$
- We can uniquely identify each symbol in a sequence of bits!
  - e.g.,  $aabacdb \rightarrow 0010011011110$
- Hence, the expected value of the number of bits for a symbol is:

$$\begin{aligned} \sum_{x \in \{a,b,c,d\}} P(S=x) (\# \text{ of bits to represent } x) \\ &= 0.5 * 1 + 0.25 * 2 + 0.125 * 3 + 0.125 * 3 \\ &= 1.75 \text{ bits} \end{aligned}$$

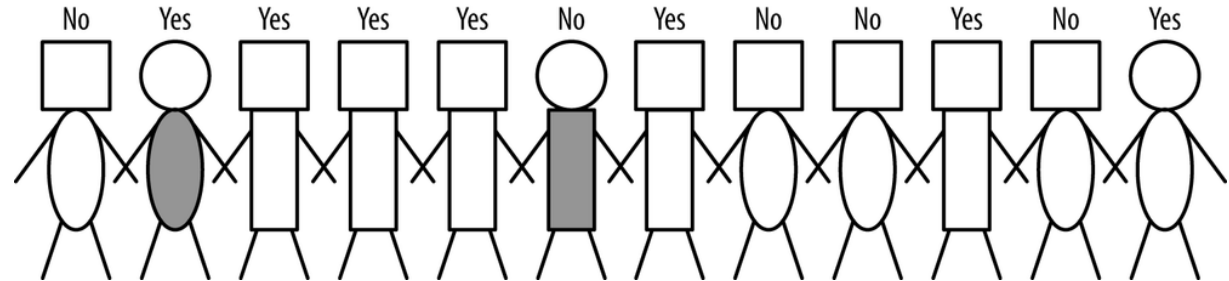
# Entropy:

## Encoding: Question

- Suppose we have 4 different symbols  $S \in \{a,b,c,d\}$  and the probabilities of occurrences are:
  - $P(S=a)=1$
  - $P(S=b)=0$
  - $P(S=c)=0$
  - $P(S=d)=0$
- How many bits do we need to have?

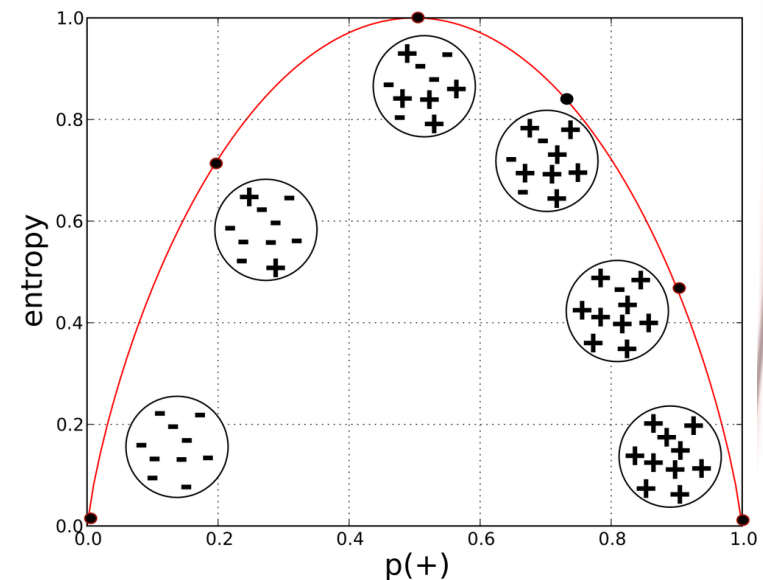
# Calculating Impurity:

## Entropy



- There are 12 instances
  - 7 cases of class “Yes”;  $p_{yes} = \frac{7}{12}$
  - 5 cases of class “No”;  $p_{no} = \frac{5}{12}$
- Entropy of the entire population of instances:

$$H(Y) = -\frac{7}{12}\log_2\left(\frac{7}{12}\right) - \frac{5}{12}\log_2\left(\frac{5}{12}\right)$$



$$H(Y) \cong 0.98$$

What if all were “Yes”?

# Conditional Entropy and Information Gain

- Entropy  $H(Y)$  of a random variable  $Y$  is:

$$H(Y) = \sum_{i=1}^{n_c} P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y) = -\frac{7}{12} \log_2 \left( \frac{7}{12} \right) - \frac{5}{12} \log_2 \left( \frac{5}{12} \right) \text{ where } n_c=2 \{ \text{Yes, No} \}$$

- Conditional entropy  $H(Y | X = x)$  of  $Y$  given  $X = x$  is:

$$H(Y | X = x) = \sum_{i=1}^{n_c} P(Y = y_i | X = x) \log_2 P(Y = y_i | X = x)$$

Example:

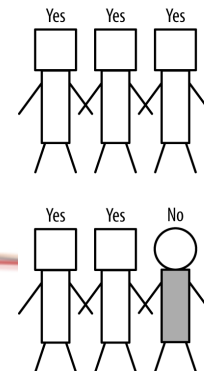
$X = \{ \text{head\_shape} = \text{square}, \text{body\_shape} = \text{oval}, \text{body\_color} = \text{white} \}$

$$H(Y | X = x) = \sum_{i=1}^{n_c} P(Y = y_i | X = x) \log_2 P(Y = y_i | X = x)$$

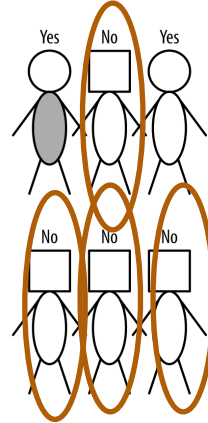
$= 0 + 1 \cdot \log(1) = 0$

We have zero  $Y = \text{yes}$  and four  $Y = \text{no}$ .

Rectangular Bodies



Oval Bodies

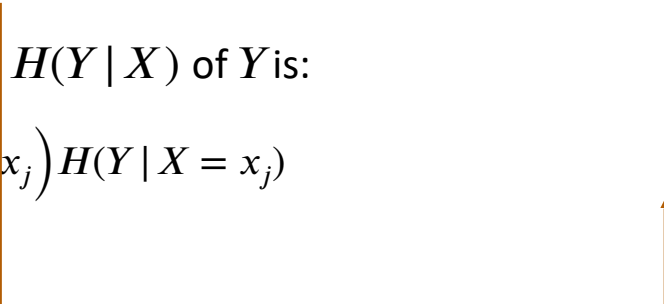


# Conditional Entropy and Information Gain

- Conditional entropy  $H(Y | X = x)$  of  $Y$  given  $X = x$  is:

$$H(Y | X = x) = \sum_{i=1}^{n_c} P(Y = y_i | X = x) \log_2 P(Y = y_i | X = x)$$

- Conditional entropy  $H(Y | X)$  of  $Y$  is:

$$H(Y | X) = \sum_{j=1}^{n_x} P(X = x_j) H(Y | X = x_j)$$


Ex:  $H(Y | head\_shape) = P(head\_shape=square)H(Y = yes | head\_shape = square) + P(head\_shape=square)H(Y = no | head\_shape = square) + P(head\_shape=circular)H(Y = yes | head\_shape = circular) + P(head\_shape=circular)H(Y = no | head\_shape = circular)$

- Information gain is the mutual information between  $Y$  and  $X$ .

- Information Gain of  $Y$  and  $X$  is:

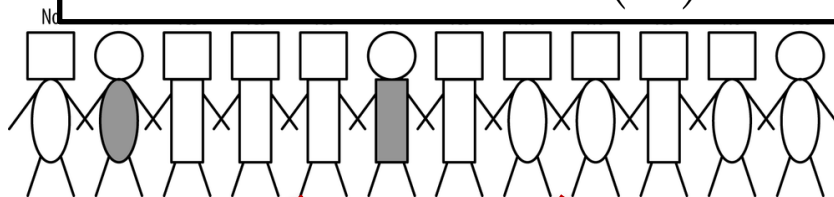
$$I(Y, X) = H(Y) - H(Y | X) = H(X) - H(X | Y)$$

- Information gain measures the expected reduction in entropy of  $Y$  by knowing  $X$ .

# Information Gain:

## Example

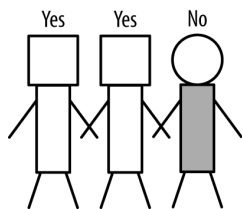
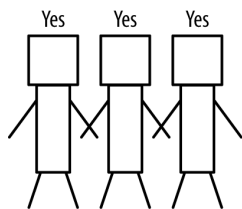
$$H(Y) = -\frac{7}{12}\log_2\left(\frac{7}{12}\right) - \frac{5}{12}\log_2\left(\frac{5}{12}\right) \cong 0.98$$



body=rect.

body=oval

Rectangular Bodies

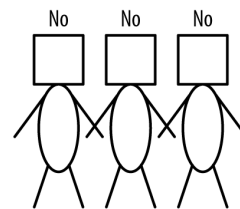
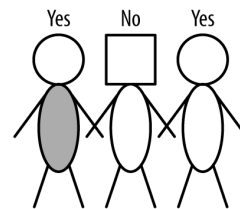


$$H(Y|body) = \frac{6}{12} * 0.65 + \frac{6}{12} * 0.92 \cong 0.78$$

$$I(Y, body) = H(Y) - H(Y|body)$$

$$I(Y, body) = 0.98 - 0.78 = 0.20$$

Oval Bodies



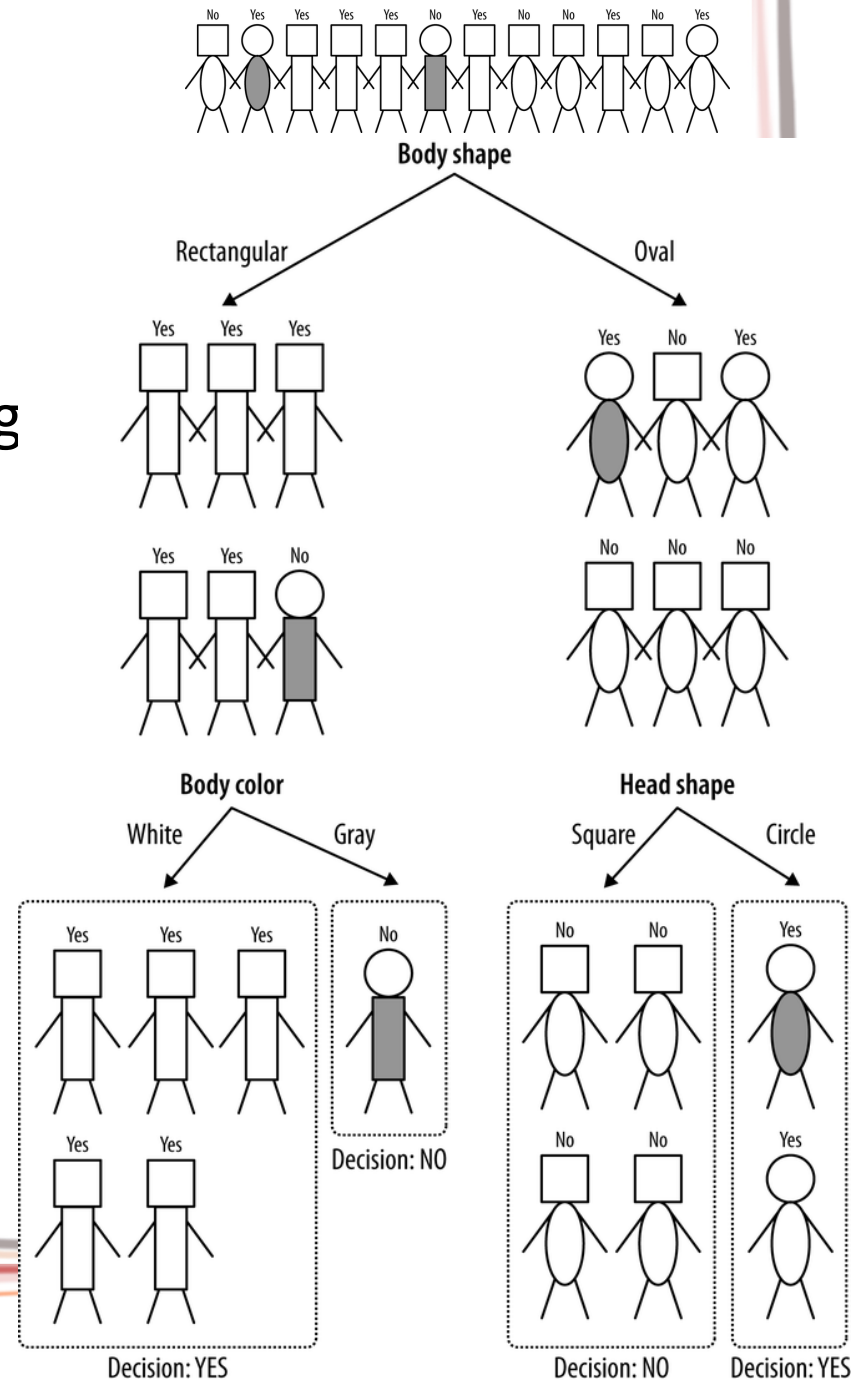
$$H(Y|body = rect) = -\frac{5}{6}\log_2\left(\frac{5}{6}\right) - \frac{1}{6}\log_2\left(\frac{1}{6}\right) \cong 0.65$$

$$H(Y|body = oval) = -\frac{2}{6}\log_2\left(\frac{2}{6}\right) - \frac{4}{6}\log_2\left(\frac{4}{6}\right) \cong 0.92$$



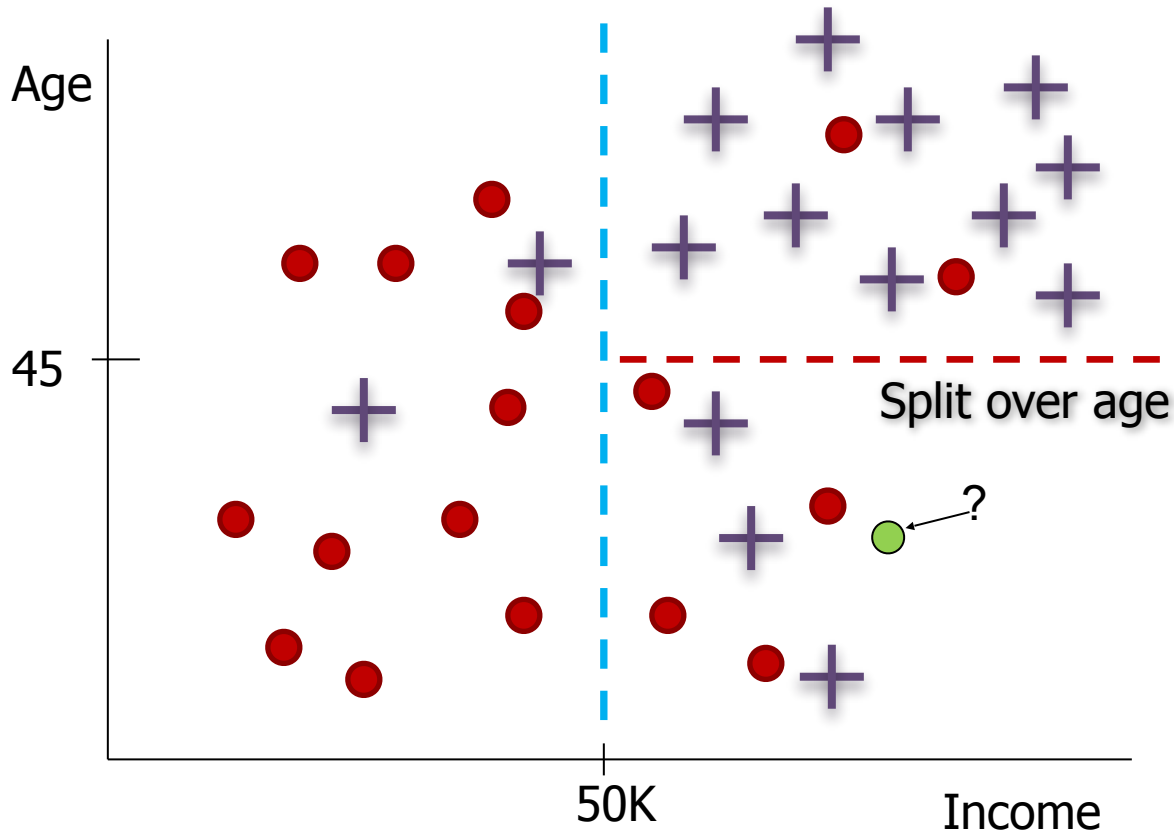
# Classification with Tree-Structured Models

- We can create a simple classification model by selecting a single attribute that gives the most information gain
- How can we use multiple attributes each giving some information gain?
- A hierarchy of rules can be represented by a **tree** diagram.
  - Ideally, the resulting segmentation should be **pure**.



# What are we predicting?

Split over income

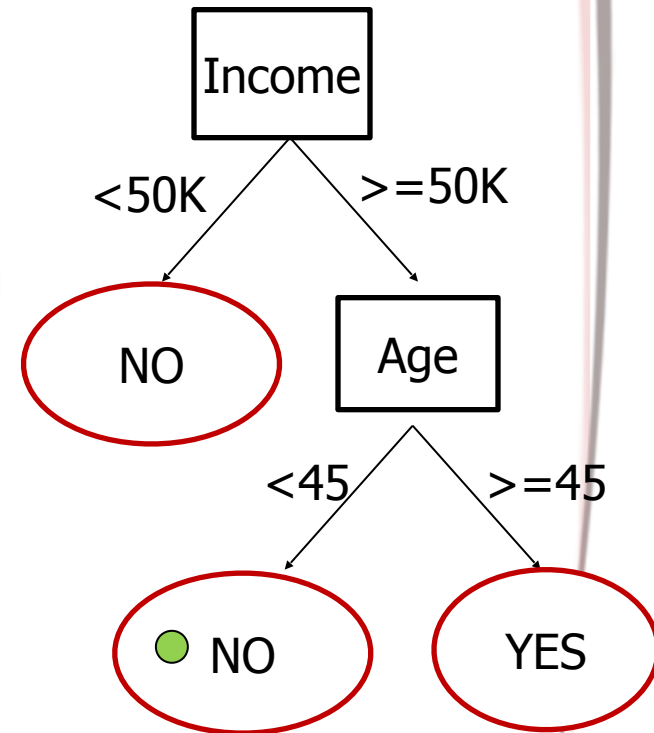


● Did not buy life insurance

+ Bought life insurance

● Interested in LI? NO

Classification tree



# Decision Tree

- We have internal nodes, branches and leaves.
- An internal node is a test on an attribute.
  - E.g., “Income<50” or “income>=50”
- A branch represents an outcome of the test.
  - E.g., “Income<50”
- A leaf node represents a class label (or class label distribution)
  - E.g., “NO”
- At each node, one attribute is chosen to split training instances into distinct classes as much as possible.
- A new case is classified by following a matching path to a leaf node.

# Fitting a Decision Tree Model

- We have a training dataset that contains instances  $\langle \mathbf{x}_i, y_i \rangle$  with feature vectors and labels such as
  - $\mathbf{x}_i = [\text{body\_shape}=\text{oval}, \text{body\_color}=\text{white}, \text{head\_shape}=\text{circle}]$
  - $\mathbf{X}$  is the set of possible instances.
  - $Y$  is the set of classes.
  - $\mathbf{x}_i \in \mathbf{X}, y_i \in Y$
- We want to find the unknown target function  $f: \mathbf{X} \rightarrow Y$
- We have a set of hypotheses  $H = \{h \mid h: \mathbf{X} \rightarrow Y\}$ 
  - Each hypothesis is a decision tree
  - Trees sorts  $\mathbf{x}$  to leaf, which assigns  $y$
- Choose/Find the hypothesis  $h$  that best approximates  $f$

# Decision Tree Induction:

## Algorithm

- Basic algorithm
  - Tree is constructed in a top-down, recursive, divide-and-conquer manner
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - On each node, attributes are selected based on the training examples on that node, and 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
  - There are no samples left
- Prediction
  - Majority voting is employed for classifying the leaf

# Information Gain:

## An Attribute Selection Measure

- Select the attribute with the highest information gain -- used in decision tree induction algorithm **ID3** (Iterative Dichotomiser) and **C4.5**
  - Let  $p_i$  be the probability that an arbitrary tuple in  $D$  belongs to class  $Y_i$ , estimated by

$$p_i = \frac{|Y_{i,D}|}{|D|} = \frac{\text{\# of class } Y_i \text{ instances in } D}{\text{\# of instances in } D}$$

- Expected information (entropy) needed to classify a tuple in  $D$ :

$$Info(D) = H(Y) = \sum_{i=1}^{n_c} p_i \cdot \log_2(p_i)$$

- Information needed classify  $D$  after using feature  $X$  to split  $D$  into  $v$  partitions  $(D_1, \dots, D_v)$ :

$$Info_X(D) = H(Y | X) = \sum_{j=1}^v \frac{|D_j|}{|D|} Info(D_j)$$

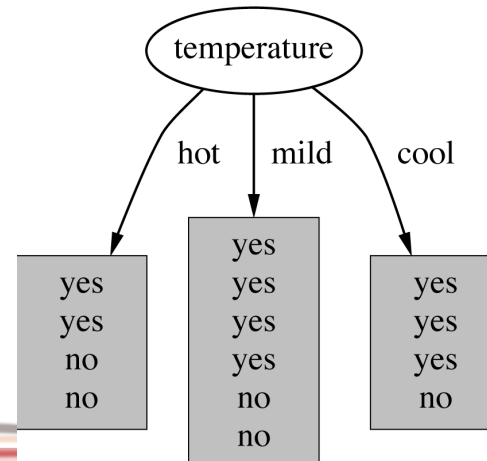
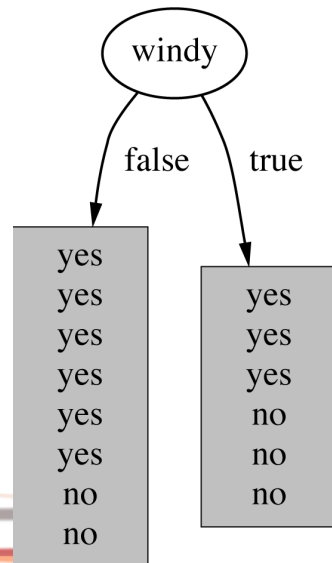
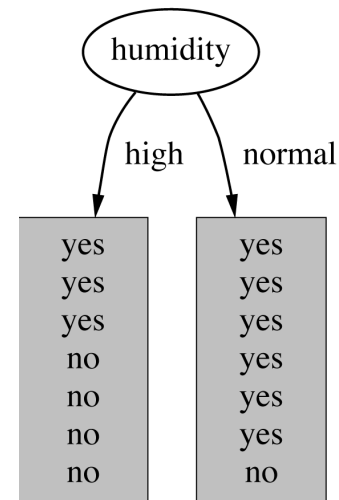
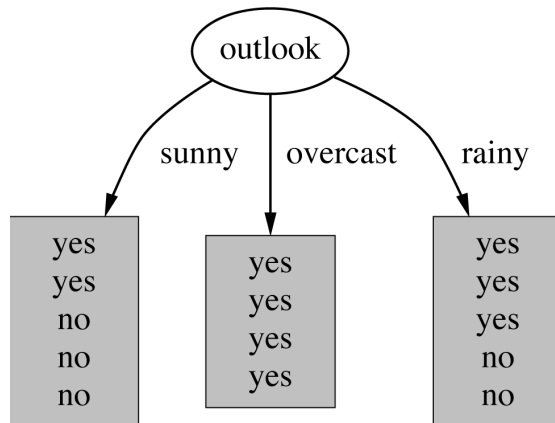
- Information gained by branching on attribute  $X$

$$Gain(X) = I(Y, X) = Info(D) - Info_X(D)$$

# Example: attribute “Outlook”

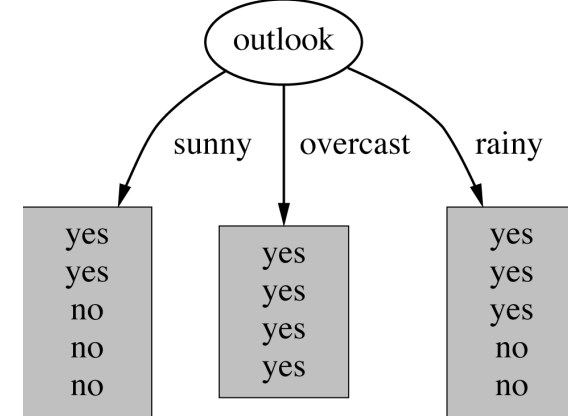
Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	No
sunny	hot	high	true	No
overcast	hot	high	false	Yes
rain	mild	high	false	Yes
rain	cool	normal	false	Yes
rain	cool	normal	true	No
overcast	cool	normal	true	Yes
sunny	mild	high	false	No
sunny	cool	normal	false	Yes
rain	mild	normal	false	Yes
sunny	mild	normal	true	Yes
overcast	mild	high	true	Yes
overcast	hot	normal	false	Yes
rain	mild	high	true	No

# Which attribute to select?





# Example: attribute “Outlook”



- “Outlook” = “Sunny”:

$$\text{info}([2,3]) = \text{entropy}(2/5, 3/5) = -2/5 \log(2/5) - 3/5 \log(3/5) = 0.971 \text{ bits}$$

- “Outlook” = “Overcast”:

$$\text{info}([4,0]) = \text{entropy}(1,0) = -1 \log(1) - 0 \log(0) = 0 \text{ bits}$$



*Note:  $\log(0)$  is not defined, but we evaluate  $0 \cdot \log(0)$  as zero*

- “Outlook” = “Rainy”:

$$\text{info}([3,2]) = \text{entropy}(3/5, 2/5) = -3/5 \log(3/5) - 2/5 \log(2/5) = 0.971 \text{ bits}$$

- Expected information for attribute:

$$\begin{aligned} \text{info}([2,3], [4,0], [3,2]) &= (5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.971 \\ &= 0.693 \text{ bits} \end{aligned}$$

# Computing the information gain

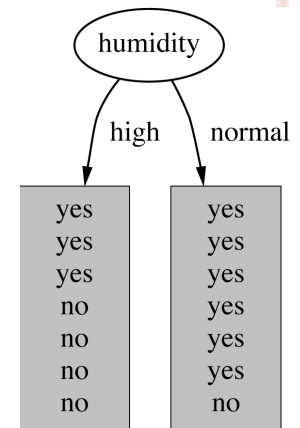
- Information gain:

(information before split) – (information after split)

$$\begin{aligned}\text{gain("Outlook")} &= \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693 \\ &= 0.247 \text{ bits}\end{aligned}$$

- Compute for attribute “Humidity”

# Example: attribute “Humidity”



- “Humidity” = “High”:

$\text{info}([3,4]) = \text{entropy}(3/7, 4/7) = -3/7 \log(3/7) - 4/7 \log(4/7) = 0.985 \text{ bits}$

- “Humidity” = “Normal”:

$\text{info}([6,1]) = \text{entropy}(6/7, 1/7) = -6/7 \log(6/7) - 1/7 \log(1/7) = 0.592 \text{ bits}$

- Expected information for attribute:

$\text{info}([3,4], [6,1]) = (7/14) \times 0.985 + (7/14) \times 0.592 = 0.788 \text{ bits}$

- Information Gain:

$\text{info}([9,5]) - \text{info}([3,4], [6,1]) = 0.940 - 0.788 = 0.152$

# Computing the information gain

- Information gain:

(information before split) – (information after split)

$$\text{gain("Outlook")} = \text{info}([9,5]) - \text{info}([2,3],[4,0],[3,2]) = 0.940 - 0.693$$
$$= 0.247 \text{ bits}$$

- Information gain for attributes from weather data:

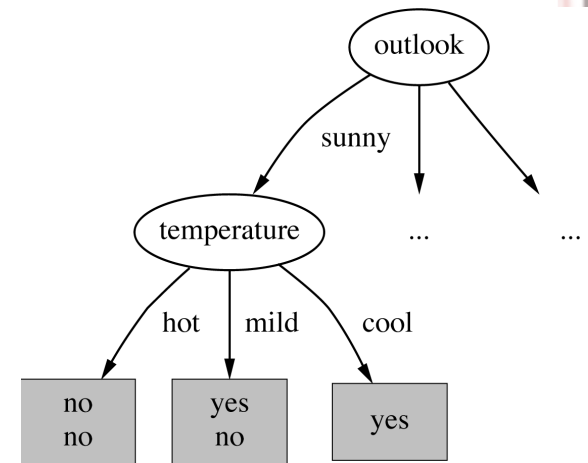
$$\text{gain("Outlook")} = 0.247 \text{ bits}$$

$$\text{gain("Temperature")} = 0.029 \text{ bits}$$

$$\text{gain("Humidity")} = 0.152 \text{ bits}$$

$$\text{gain("Windy")} = 0.048 \text{ bits}$$

# Continuing to split



- “Temperature” = “Hot”:

$$\text{info}([0,2]) = \text{entropy}(0/2,2/2)=0 \text{ bits}$$

- “Temperature” = “Mild”:

$$\text{info}([1,1]) = \text{entropy}(1/2,1/2)= -1/2 \log(1/2) -1/2 \log(1/2) = 1 \text{ bits}$$

- “Temperature” = “Cool”:

$$\text{info}([1,0]) = 0 \text{ bits}$$

- Expected information for attribute:

$$\text{info}([0,2],[1,1],[1,0]) = (2/5) \times 0 + (2/5) \times 1 + (1/5) \times 0 = 0.4$$

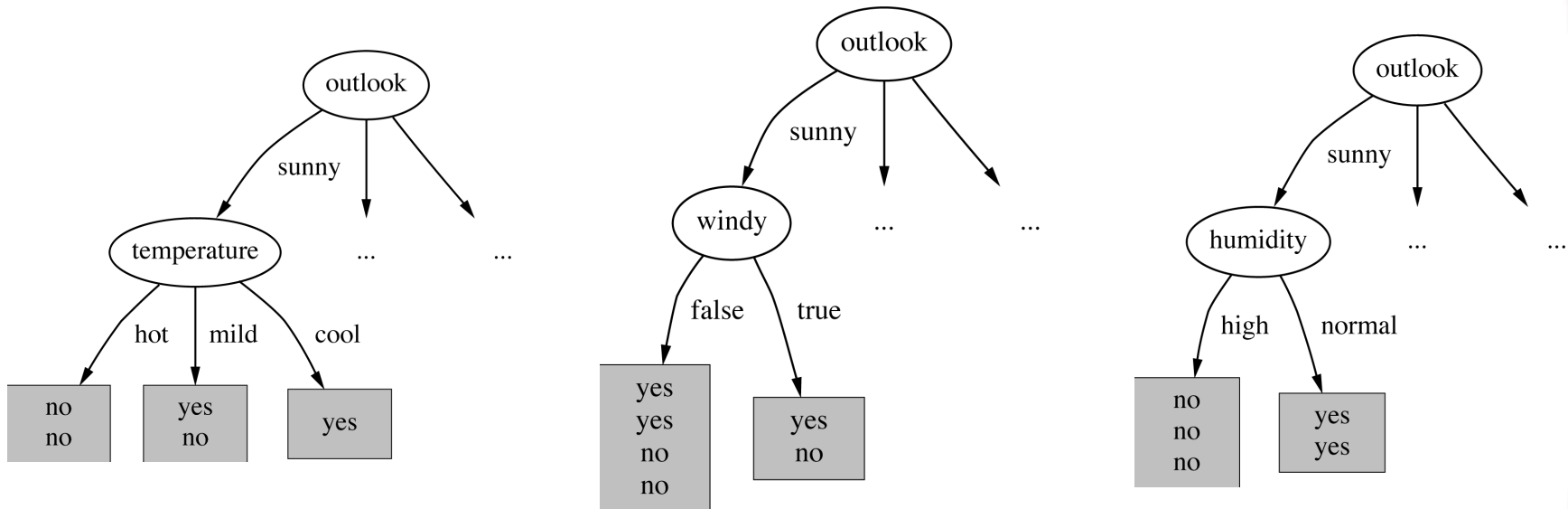
- Information Gain:  $0.971 - 0.4 = 0.571$  bits

Recall:

- “Outlook” = “Sunny”:

$$\text{info}([2,3]) = \text{entropy}(2/5,3/5) = -2/5 \log(2/5) - 3/5 \log(3/5) = 0.971 \text{ bits}$$

# Continuing to split

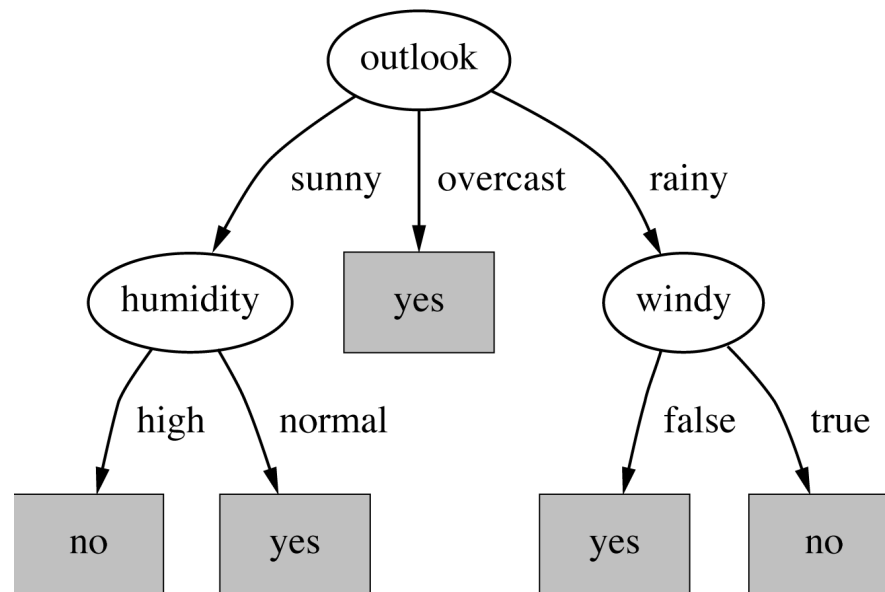


$\text{gain}(\text{"Temperature"}) = 0.571 \text{ bits}$

$\text{gain}(\text{"Humidity"}) = 0.971 \text{ bits}$

$\text{gain}(\text{"Windy"}) = 0.020 \text{ bits}$

# The final decision tree

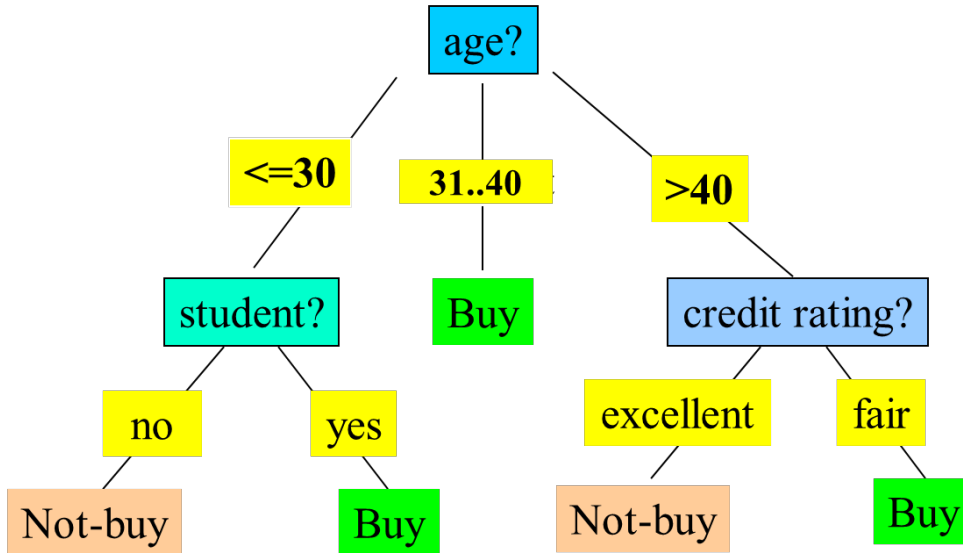


- Note: not all leaves need to be pure; sometimes identical instances have different classes  
⇒ Splitting stops when data can't be split any further

# Decision Tree Induction:

## An Example

- A top-down, recursive, divide-and-conquer process.
- Resulting tree:



Training data set: Who buys computer

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



# Decision Tree Induction: An Example

- Class  $Y_1$ : buys\_computer = "yes"
- Class  $Y_2$ : buys\_computer = "no"

$$H(Y) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940$$

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

age	$Y_1$	$Y_2$	$H(Y X_{age})$
<=30	2	3	0.971
31..40	4	0	0
>40	3	2	0.971

$$H(Y|X_{age}) = \frac{5}{14} 0.971 + \frac{4}{14} 0 + \frac{5}{14} 0.971$$

$$H(Y|X_{age}) = 0.694$$

$$\text{Gain}(X_{age}) = I(Y, X_{age}) = 0.940 - 0.694 = 0.246$$

Similarly,

$$\text{Gain}(X_{income}) = I(Y, X_{income}) = 0.029$$

$$\text{Gain}(X_{student}) = I(Y, X_{student}) = 0.151$$

$$\text{Gain}(X_{credit\_rating}) = I(Y, X_{credit\_rating}) = 0.048$$

Hence, "age" is the attribute with the highest information gain.

The first partitioning is based on "age".

Then we continue to split similarly.

# How to Handle Continuous-Valued Attributes?

- **Method 1:** Discretize continuous values and treat them as categorical values
  - E.g., age: “<20”, “20..30”, “30..40”, “40..50”, “>50”
- **Method 2:** Determine the **best split point** for continuous-valued attribute A
  - Sort the value A in increasing order:, e.g. 15, 18, 21, 22, 24, 25, 29, 31, ...
  - Possible split point: the midpoint between each pair of adjacent values
    - $(a_i + a_{i+1})/2$  is the midpoint between the values of  $a_i$  and  $a_{i+1}$
    - e.g., 16.5, 19.5, 21.5, 23, 24.5, 27, 30, ...
  - The point with the **maximum information gain** for A is selected as the split-point for A
- **Split:** Based on split point P determined
  - The set of tuples in D satisfying  $A \leq P$  vs. those with  $A > P$

# Weather data - numeric

Outlook	Temperature	Humidity	Windy	Play
Sunny	85	85	False	No
Sunny	80	90	True	No
Overcast	83	86	False	Yes
Rainy	75	80	False	Yes
...	...	...	...	...

```
If outlook = sunny and humidity > 83 then play = no
If outlook = rainy and windy = true then play = no
If outlook = overcast then play = yes
If humidity < 85 then play = yes
If none of the above then play = yes
```

# Example

- Split on temperature attribute:

64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

- E.g. temperature  $< 71.5$ : yes/4, no/2  
temperature  $\geq 71.5$ : yes/5, no/3
  - Info([4,2],[5,3])  
=  $6/14 \text{ info}([4,2]) + 8/14 \text{ info}([5,3])$   
= 0.939 bits
- Place split points halfway between values
- Can evaluate all split points in one pass!

# Gain Ratio:

## A Refined Measure for Attribute Selection

- Information gain measure is biased towards attributes with a large number of values.
- Subsets are more likely to be pure if a feature has a large number of unique values.
  - e.g., using serial number as a feature
- This may result in overfitting (selection of an attribute that is non-optimal for prediction)
- In order to overcome the problem, information gain can be normalized by **gain ratio**.

# Gain Ratio

- Gain ratio is used in a popular algorithm C4.5 (a successor to ID3)
- The attribute with the maximum gain ratio is selected as the splitting attribute.
- Split Information is used to normalize information gain for branching on X
  - It corresponds to intrinsic information that is how much info do we need to tell which branch an instance belongs to:

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

- Gain ratio for branching on  $X$  is:

$$GainRatio(X) = \frac{Gain(X)}{SplitInfo_X(D)}$$

- Example:
  - Consider “who buys computer” example,

$$Gain(X_{income}) = I(Y, X_{income}) = 0.029$$

$$SplitInfo_{income}(D) = -\frac{4}{14} \log_2 \left( \frac{4}{14} \right) - \frac{6}{14} \log_2 \left( \frac{6}{14} \right) - \frac{4}{14} \log_2 \left( \frac{4}{14} \right) = 1.557$$

$$\text{Therefore, } GainRatio(X_{income}) = \frac{0.029}{1.557} = 0.019$$

# Gain Ratio

- Gain ratio takes the number and the size of branches into account when choosing an attribute.
  - Split information is large when data is evenly spread.
  - Split information is small when all data belong to one branch.
- However, it may overcompensate
  - May choose an attribute just because its intrinsic information is very low.
  - Standard fix:
    - First, only consider attributes with greater than average information gain.
    - Then, compare them on gain ratio.

# Computing the gain ratio

## Revisiting Weather Example

- Example: intrinsic information for ID code

$$\text{info}([1, 1, \boxed{?}, 1]) = 14 \times (-1/14 \times \log 1/14) = 3.807 \text{ bits}$$

- **Importance of attribute decreases as intrinsic information gets larger**

- Example of gain ratio:

$$\text{gain\_ratio}(\text{"Attribute"}) = \frac{\text{gain}(\text{"Attribute"})}{\text{intrinsic\_info}(\text{"Attribute"})}$$

- Example:  $\text{gain\_ratio}(\text{"ID\_code"}) = \frac{0.940 \text{ bits}}{3.807 \text{ bits}} = 0.246$





# Gain ratios for weather data

For outlook split  $\text{info} = -(5/14) \times \text{LOG}_2(5/14) - (4/14) \times \text{LOG}_2(4/14) - (5/14) \times \text{LOG}_2(5/14)$

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: $0.940 - 0.693$	0.247	Gain: $0.940 - 0.911$	0.029
Split info: $\text{info}([5,4,5])$	1.577	Split info: $\text{info}([4,6,4])$	1.362
Gain ratio: $0.247/1.577$	0.156	Gain ratio: $0.029/1.362$	0.021

Humidity		Windy	
Info:	0.788	Info:	0.892
Gain: $0.940 - 0.788$	0.152	Gain: $0.940 - 0.892$	0.048
Split info: $\text{info}([7,7])$	1.000	Split info: $\text{info}([8,6])$	0.985
Gain ratio: $0.152/1$	0.152	Gain ratio: $0.048/0.985$	0.049

Recall for Outlook attribute:

$$\begin{aligned} \text{info}([3,2],[4,0],[3,2]) &= (5/14) \times 0.971 + (4/14) \times 0 + (5/14) \times 0.971 \\ &= 0.693 \text{ bits} \end{aligned}$$



# More on the gain ratio

- “Outlook” still comes out top
- However: “ID code” has greater gain ratio
  - Standard fix: *ad hoc* test to prevent splitting on that type of attribute
- Problem with gain ratio: it may overcompensate
  - May choose an attribute just because its intrinsic information is very low
  - Standard fix:
    - First, only consider attributes with greater than average information gain
    - Then, compare them on gain ratio

# Another Measure: Gini Index

- Gini index: Used in **CART** (Classification and Regression Trees)
- If a data set  $D$  contains examples from  $n$  classes, gini index,  $gini(D)$  is defined as:

$$gini(D) = 1 - \sum_{j=1}^n p_j^2$$

where  $p_j$  is the relative frequency of class  $j$  in  $D$

- If a data set  $D$  is split on  $X$  into two subsets  $D_1$  and  $D_2$ , the  $gini$  index  $gini(D)$  is defined as:

$$gini_X(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(X) = gini(D) - gini_X(D)$$

- The attribute provides the smallest  $gini_X(D)$  (or the largest reduction in impurity) is chosen to split the node.

# CART Splitting Criteria:

## Gini Index

- The Gini index considers a binary split for each attribute.
  - For each attribute, each of the possible binary splits is considered.
  - We need to enumerate all the possible splitting points for each attribute.
- The attribute provides the smallest  $gini_X(D)$  (i.e., the largest reduction in impurity) is chosen to split the node.

# CART Splitting Criteria:

## Gini Index

- Example:  $X$  is a discrete-valued attribute having  $v$  distinct values,  $\{x_1, x_2, \dots, x_v\}$ , occurring in  $D$ .
  - To determine the best binary split on  $X$ , we examine all the possible subsets that can be formed using known values of  $X$ .
  - If  $X$  has  $v$  possible values, then there are  $2^v$  possible subsets.
  - If income has three possible values  $\{\text{low}, \text{medium}, \text{high}\}$ , then the possible subsets are  $\{\text{low}, \text{medium}, \text{high}\}$ ,  $\{\text{low}, \text{medium}\}$ ,  $\{\text{low}, \text{high}\}$ ,  $\{\text{medium}, \text{high}\}$ ,  $\{\text{low}\}$ ,  $\{\text{medium}\}$ ,  $\{\text{high}\}$ , and  $\{\}$
  - Hence the possible splits are  $\{\text{low}, \text{medium}\} - \{\text{high}\}$ ,  $\{\text{low}, \text{high}\} - \{\text{medium}\}$  and  $\{\text{low}\} - \{\text{medium}, \text{high}\}$

# Gini Index:

## Example

- Example: D has 9 tuples in buys\_computer = “yes” and 5 in “no”

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

- Suppose the attribute income partitions D into 10 in  $D_1$ : {low, medium} and 4 in  $D_2$ : {high}

$$\begin{aligned} gini_{income \in \{low, medium\}}(D) &= \frac{10}{14} gini(D_1) + \frac{4}{14} gini(D_2) \\ &= \frac{10}{14} \left( 1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2 \right) + \frac{4}{14} \left( 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 \right) = 0.443 = Gini_{income \in \{high\}}(D) \end{aligned}$$

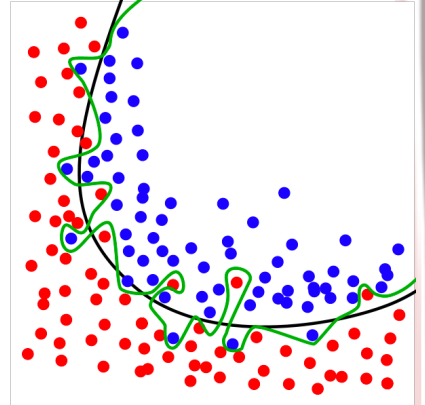
similarly,

- $gini_{income \in \{low, high\}}(D) = 0.458$
- $gini_{income \in \{medium, high\}}(D) = 0.450$
- Therefore, we split on the {low,medium}-{high} as it has the lowest Gini index

# Attribute Selection Measures: Comparison

- 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 # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

# 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 test data that is different from the training data to decide which is the “best pruned tree”.



# Decision Tree Methods

## Strengths

- Tree induction is one of the most popular tools for classification
  - It can also be used for regression
- It is
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap at classification time
- Works remarkably well
  - not the most accurate!
- Most of the data mining packages include tree algorithms.
- Has advantages for model comprehensibility, which is important for:
  - model evaluation
  - communication to non-DM-savvy stakeholders

# Decision Tree Methods

## Weaknesses

- Greedy algorithm: no global optimization
- Error-prone with too many classes: numbers of training examples become smaller quickly in a tree with many levels/branches.
- Expensive to train: sorting, combination of attributes, calculating quality measures, etc.
- Trouble with non-rectangular regions: the rectangular classification boxes that may not correspond well with the actual distribution of records in the decision space.

