Classification

IT 326: Data Mining

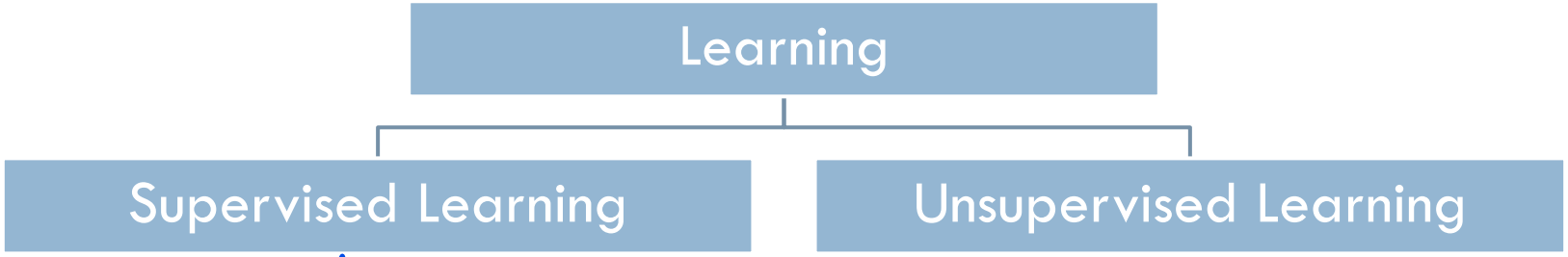
First semester 2021

بسم الله الرحمن الرحيم

Outline

- Classification: Basic Concepts
- Decision Tree Induction
- Prediction using classification model
- Model Evaluation and Selection
- Summary

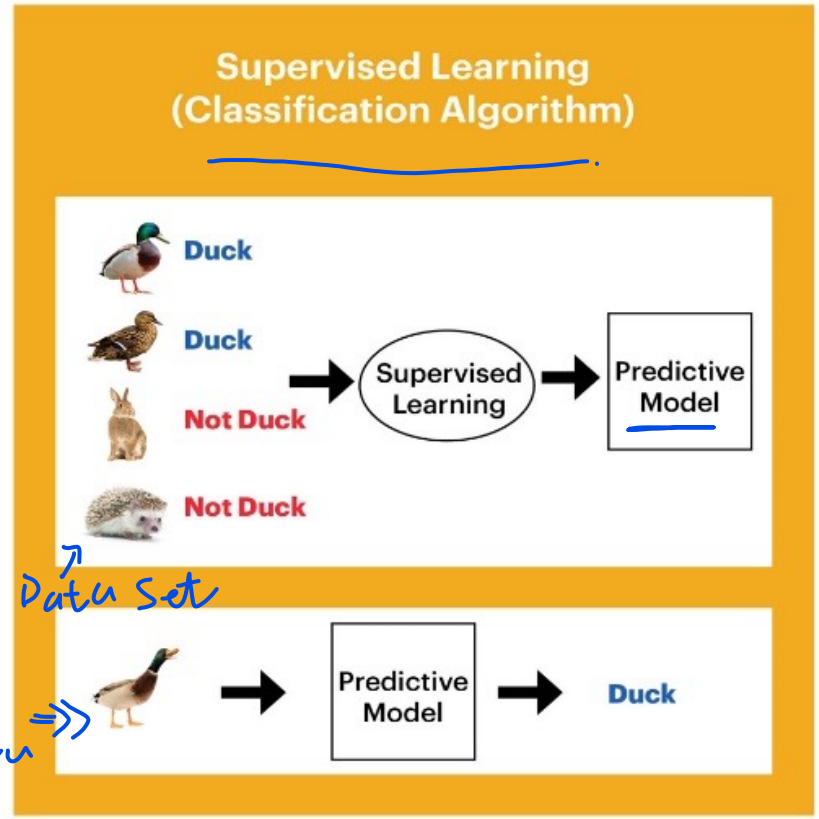
Supervised vs. Unsupervised Learning



* Example

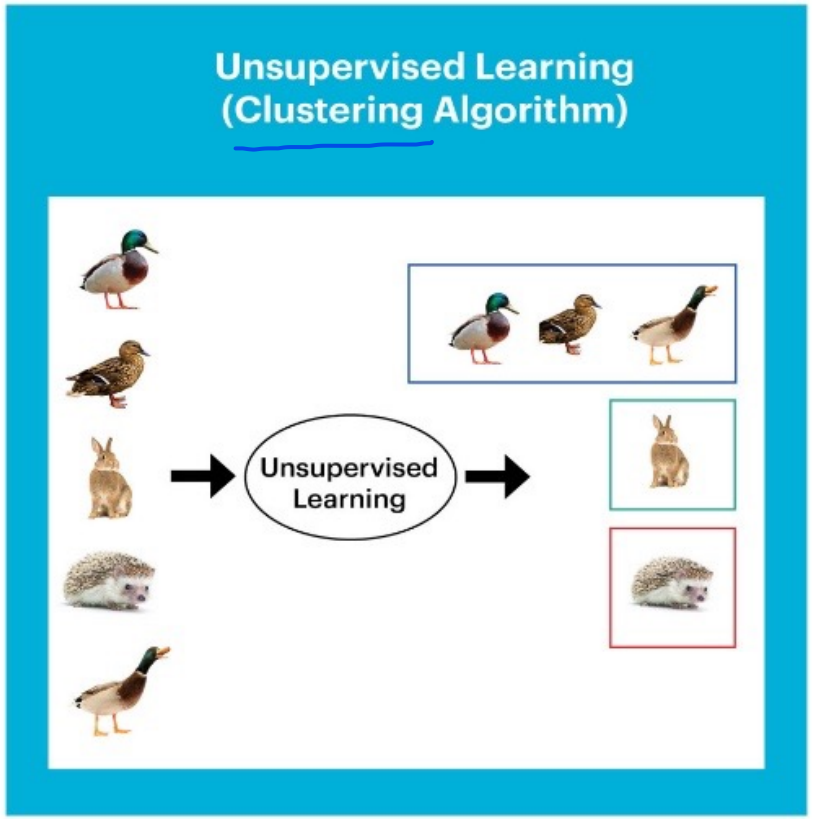
- Supervised learning (classification)
  - The class labels and the number of classes of training data is known.
  - New data is classified based on the training set.
- Unsupervised learning (clustering)
  - The class labels of training data is unknown.
  - Given a set of measurements, observations, etc.… with the aim of establishing the existence of classes or clusters in the data.

Example: Supervised vs. Unsupervised



Class labels √

Number of classes √



Class labels x ⇒ unknown

Number of classes √ or x maybe known or unknown

√ : Known in advance X : Unknown in advance

Prediction Problems: Classification vs. Regression

- Classification and Regression are two major types of prediction problems.
- Classification:
  - 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.
- Regression:
  - Models continuous-valued functions, i.e., predicts unknown or missing values.

Activity: (classification or regression?)

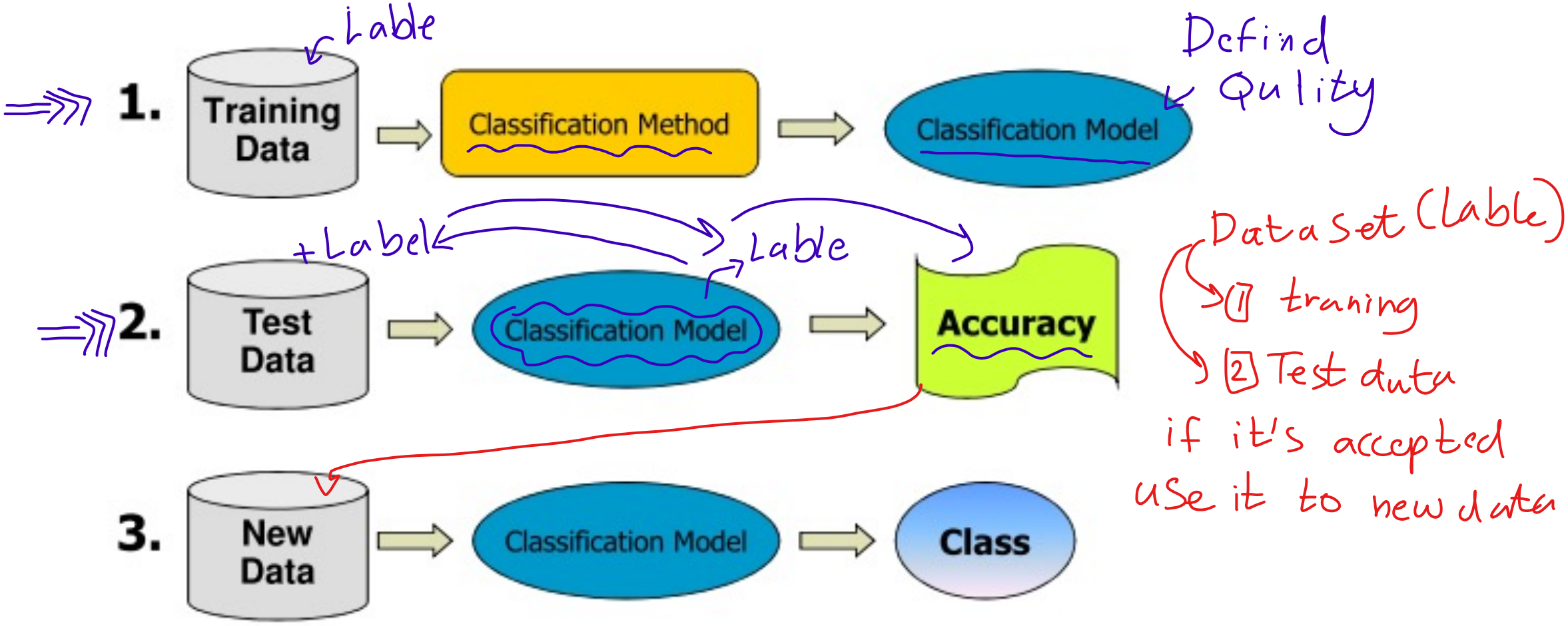
For some loan application data → predict "safe" or "risky".

For a marketing manager → predict how much a given customer will spend during a sale.

Typical applications of classification

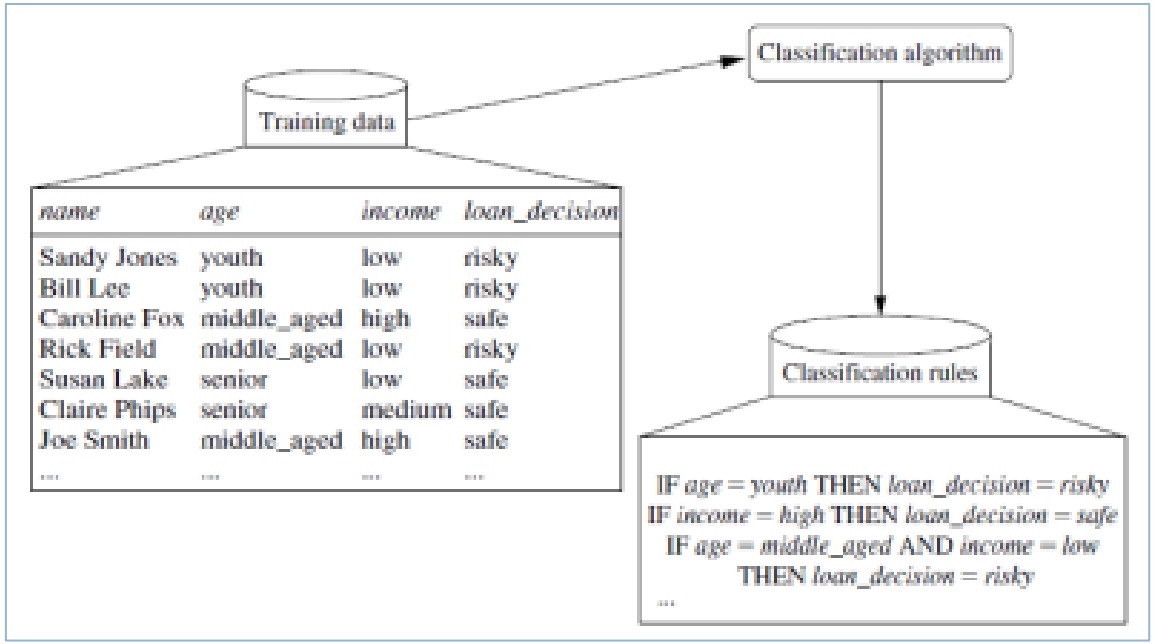
- Credit/loan approval: if a customer should be approved or not.
  * save. risky ⇒ data set + class lable ⟼ model new costemer → model → Risky / Save
  in classfication need to dataset
- Medical diagnosis: if a tumor is malignant or benign.
  DataSet ⇒ model
- Fraud detection: if a transaction is fraudulent.
  transaction + normal / fraud ⇒ model
  new → model

Classification Process: the 3 steps



Step 1: Model construction (Learning)

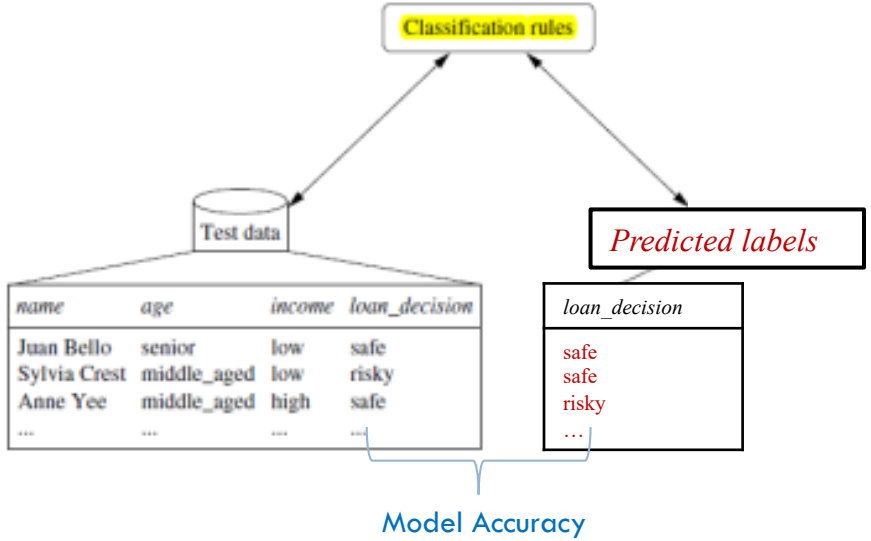
- Each tuple is assumed to belong to a predefined class, as determined by one of the attributes, called the class label.
- The set of all tuples used for construction of the model is called training set.
- The model is represented in the following forms:
  - Classification rules, (IF-THEN statements).
  - Decision tree.
  - Mathematical formula.



Step 2: Model Evaluation

Dataset < training / test

- Estimate the accuracy rate of the model based on a test set.
  - 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 over-fitting will occur.
- If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known.



Step 3: Using the Model in Prediction

- The model is used to classify unseen objects (new data).
  - Give a class label to a new tuple.
  - Predict the value of an actual attribute (class label).

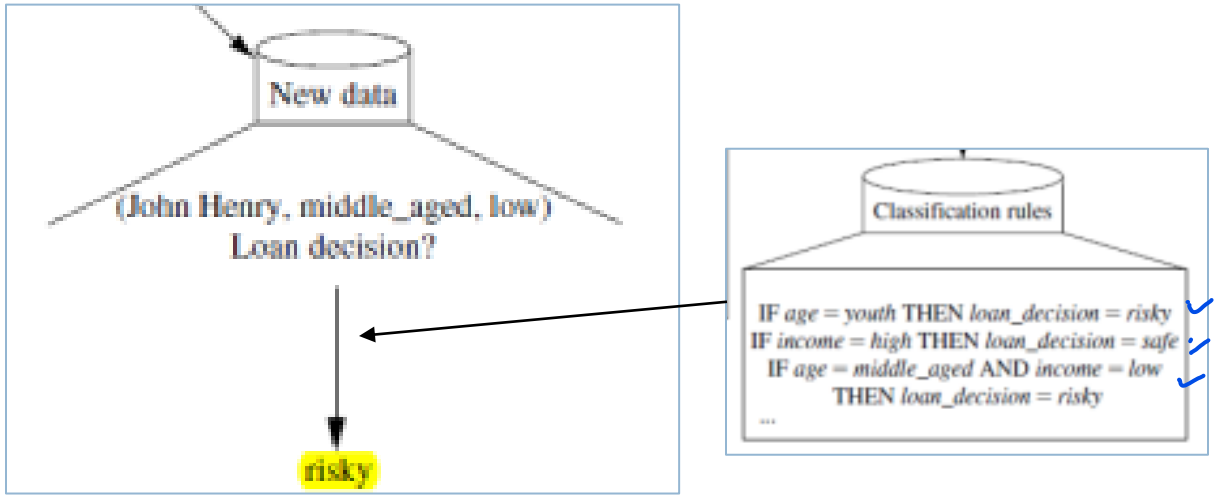
3steps

1 ~~classify~~ construct

2 test

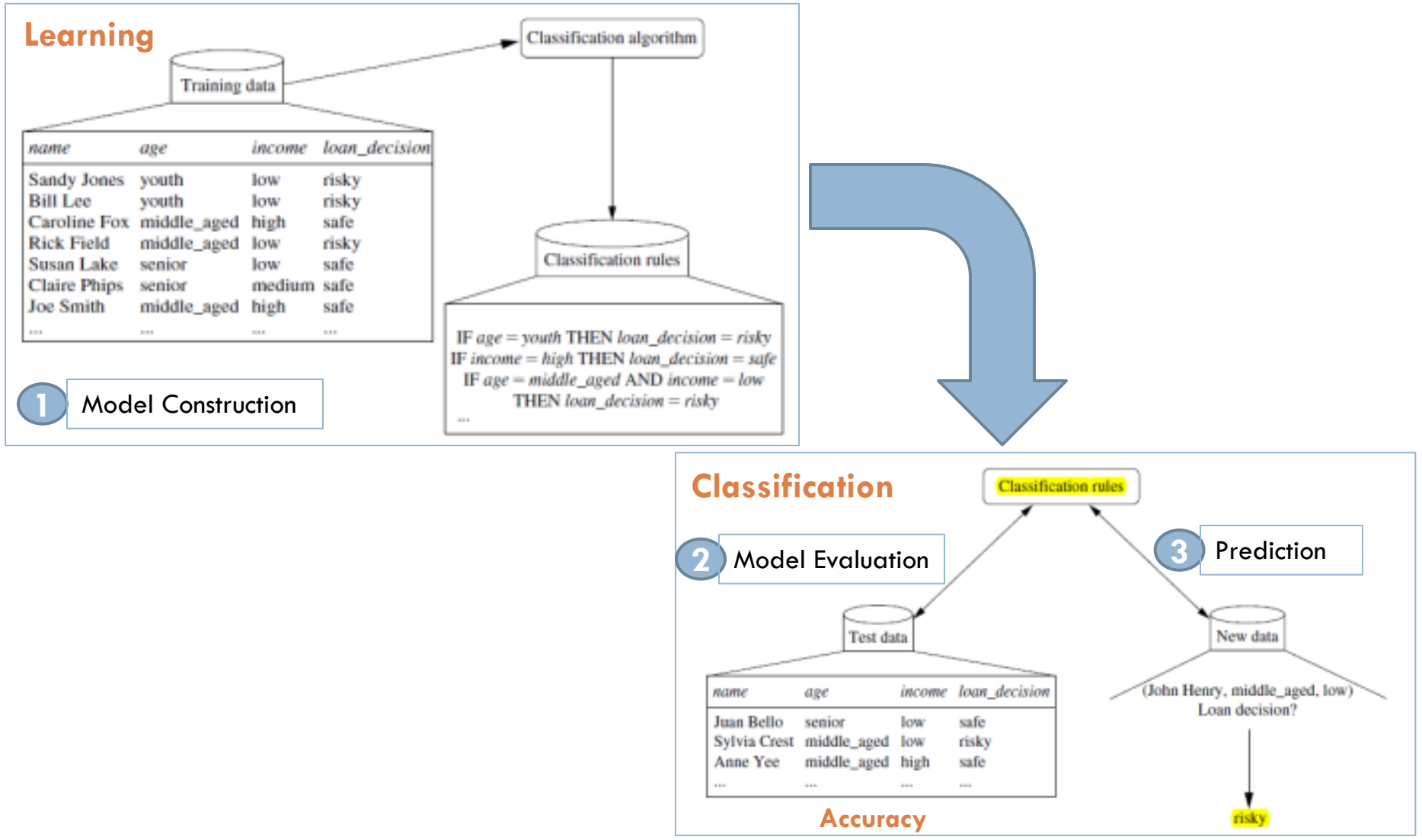
3 predict.

Predict if the new data is (safe) or risky)) ?



نطبق الكلاسفكيشن رولز على البيانات

Classification Process: example



Decision Tree Induction

basic method for classification

Decision Tree Induction:
Algorithm

- **Basic algorithm (a greedy algorithm):**
  - Tree is constructed in a **top-down recursive divide-and-conquer** manner.
  - At start, all the training examples are at the **root**; then it breaks down a dataset into smaller and smaller subsets
  - Examples are **partitioned recursively** based on **selected attributes**.
    - Attributes are categorical (if continuous-valued, they are discretized in advance).
  - The final result is a **tree** with **decision nodes** and **leaf nodes**.
    - A **decision node** is an **attribute**, and the **leaf node** represents **classification** or decision.
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., **information gain**).
- **Conditions for stopping partitioning:**
  - All samples for a given node belong to the same class.
  - There are no remaining attributes for further partitioning – **majority voting** is employed for classifying the leaf.
  - There are no samples left.

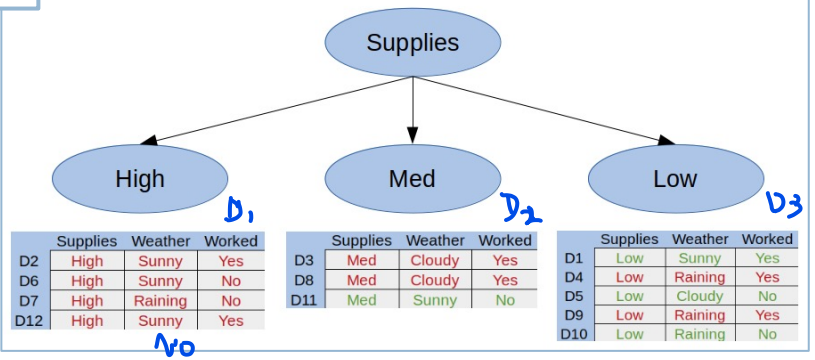
Decision Tree Induction:
Example

Dataset

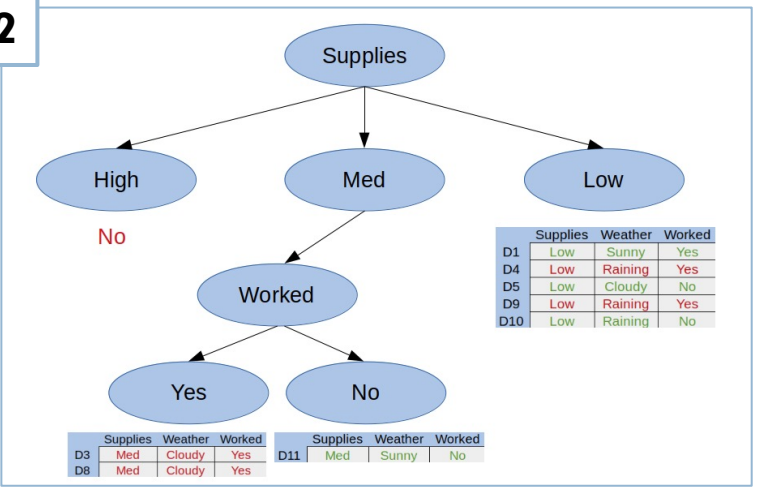
	Supplies	Weather	Worked		Shopped
D1	Low	Sunny	Yes		Yes
D2	High	Sunny	Yes		No
D3	Med	Cloudy	Yes		No
D4	Low	Raining	Yes		No
D5	Low	Cloudy	No		Yes
D6	High	Sunny	No		No
D7	High	Raining	No		No
D8	Med	Cloudy	Yes		No
D9	Low	Raining	Yes		No
D10	Low	Raining	No		Yes
D11	Med	Sunny	No		Yes
D12	High	Sunny	Yes		No

Dataset

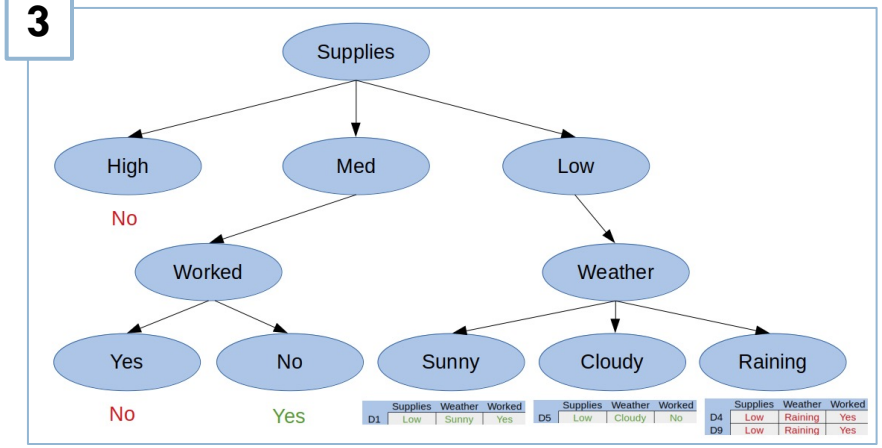
1



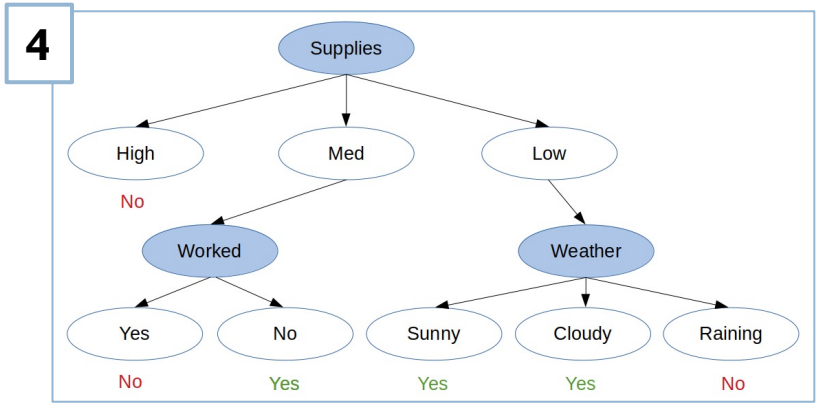
2



3



4



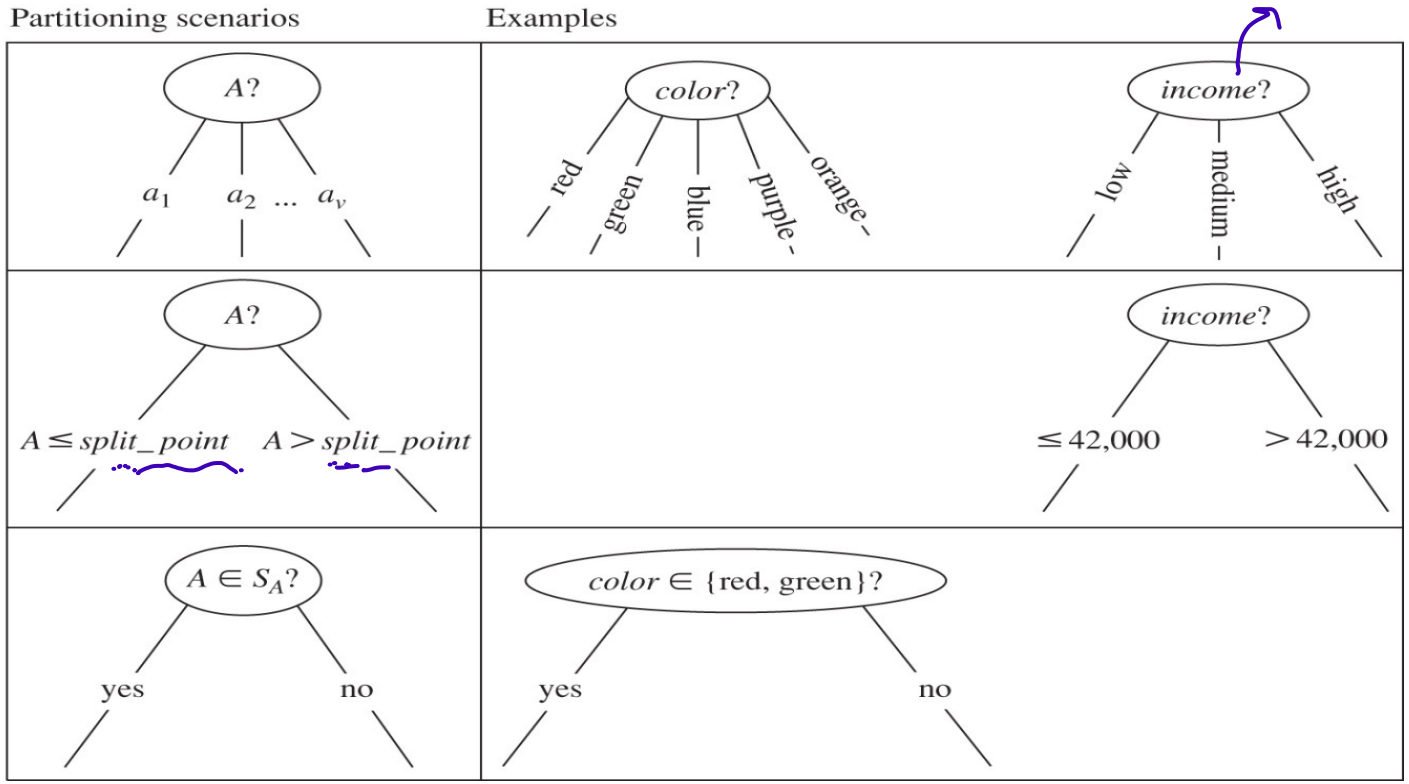
Attribute Partitioning Scenario

- Partitioning is different depends on attribute types and tree type.
  - Binary Tree: 2-way split
  - Non-Binary Tree: Multi-way split

(a) If A is discrete-valued.

(b) If A is continuous-valued.

(c) If A is discrete-valued and a binary tree must be produced.

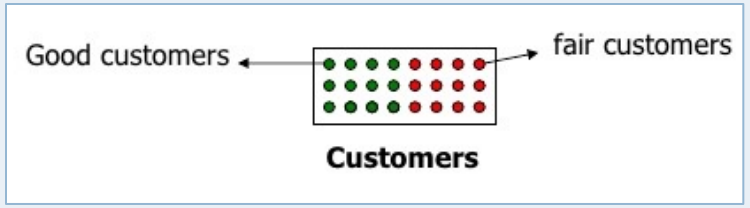


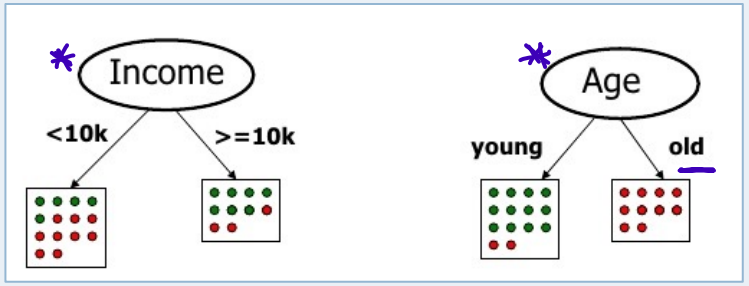
Attribute Selection Measures

تعطي Ranking للاتريبوت

- **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.
- Different measures:
  - Information Gain used in ID3
  - Gain Ratio used in C4.5
  - Gini Index used in CART
- The attribute selection measure provides a ranking for each attribute describing the given training tuples.
- The attribute having the **best score** is chosen as the **splitting attribute**.
- Best score: depending on the measure, either the highest or lowest is chosen as the best.

يعتمد على الـ measure
يفضل الأقل أو الأعلى





The best splitting attribute is .....

Attribute Selection Measure (1):

Information Gain (ID3)

$P_1 = \frac{10}{30}$

$P_2 = \frac{20}{30}$

Best Score = high

Goal: finding the attribute that returns the highest information gain (i.e., the most homogeneous branches).

Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $p_i = \frac{|c_{i,D}|}{|D|}$

D $|C_1| 10$
$|C_2| 20$

D: the data partition
m: # of class labels, C_i (for $i = 1, \dots, m$).
$c_{i,D}$: the set of tuples of class C_i in *D*.
$|D|$: the number of tuples in *D*.
$|c_{i,D}|$: the number of tuples in $c_{i,D}$.

(1) □ Expected information (Entropy E) needed to classify a tuple in D:

قبل التقسيم

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

$-(P_1 \log P_1 + P_2 \log P_2)$

(2) □ Information needed (after using A to split D into v partitions) to classify D (called Conditional Entropy):

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

① length
② $\frac{\text{length}}{D} \times$

(3) □ Information gained by branching on attribute A

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

Note : using the calculator (log is with base 2 → $\log_2$)

Entropy using One Attribute Frequency (Expected information "Entropy")

- **Entropy E** is the measure of the **uncertainty** in the data set D or the average amount of information in each class.
- **ID3** algorithm uses entropy to calculate the homogeneity of a sample.
  - If the sample is completely homogeneous (same class) → entropy is 0
  - if the sample is an equally divided → entropy is 1

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i) \quad , \; p_i = \frac{|c_{i,j}|}{|D|}$$

Play Golf	
Yes	No
9	5

Calculated only for the class attribute

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

total =14

Which change in the data will make this =1?

Class label-attribute

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Entropy using Two Attributes Frequency (Information) : Conditional Entropy

Conditional Entropy: Information needed (after using outlook to split D into 3 partitions) to classify D.

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

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

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

E(3,2) = E($\frac{3}{5}$, $\frac{2}{5}$)

= E(0.6,0.4)

= -((0.6 $\log_2$ 0.6)+ (0.4 $\log_2$ 0.4))

= 0.971

E(x,0)=0

E(x,y)=E(y,x)

Information Gain

- Information Gain is the measure of the difference in entropy before and after split on an attribute A.
  - How much uncertainty in D is reduced after splitting data set D.

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
		9	5	14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)

= 0.94

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

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

Then, we need to calculate Gain the other 3 attributes :
G(PlayGolf,Temp) , G(PlayGolf,Humidity) and G(PlayGolf,Windy)

Constructing Decision Tree: ID3

- Using ID3 → constructing a decision tree is all about finding attribute that returns the **highest** information gain (i.e., the most homogeneous branches).

1. Compute Gain for each attribute.
2. Select attribute with max Gain as the decision node
3. Divide the dataset based on its branches
4. Repeat the same process on every branch

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

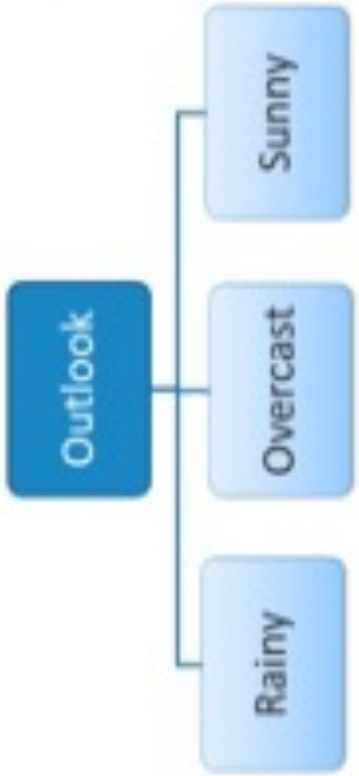
		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

max Gain → Outlook is decision node .

Constructing Decision Tree: ID3

★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Constructing Decision Tree: ID3

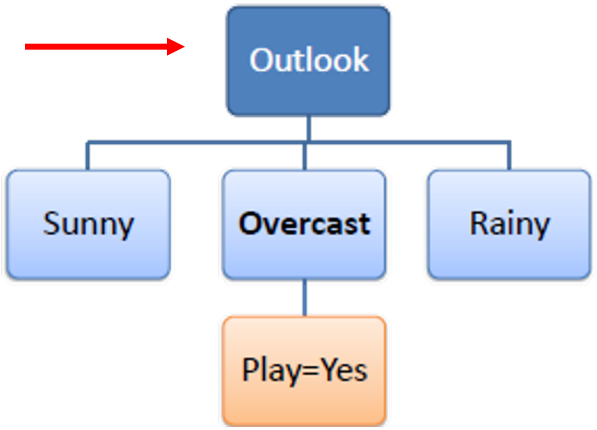
- A branch with entropy = 0 is a leaf node. (**orange** color)
- A branch with entropy > 0 non-leaf needs further splitting. (**dark blue** color)
- The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

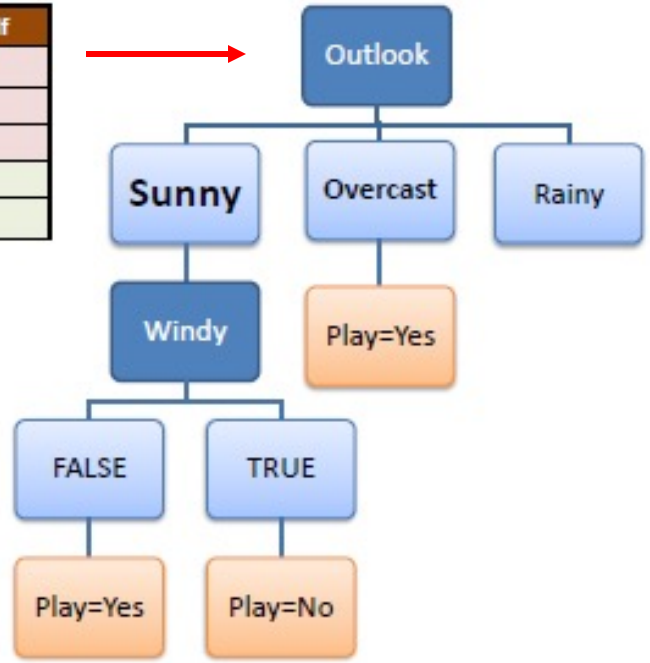
Sunny

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No

Overcast

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes

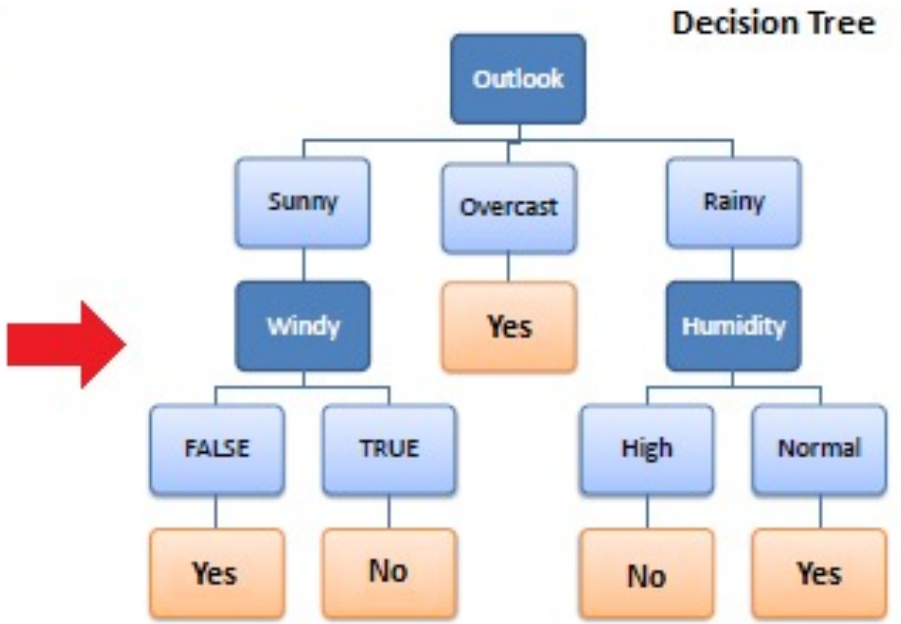




Constructing Decision Tree: ID3

- Final Results..

Predictors				Target
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



Example: Information Gain
Self-study

- Calculate the information gain for each attributes?
  - Gain (age)
  - Gain (income)
  - Gain (student)
  - Gain (credit_rating)
- Which one should be in the root? Why?

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

Example: Information Gain cont.

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

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

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

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) = 0.694 \text{ bits.}$$

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

$$Info(D_j) = -\sum_{i=1}^{m} p_{ji} \log_2(p_{ji})$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

In similar way, we calculate the following:

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

$$Info(age <= 30) = I(2,3) = -\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5}) = 0.971$$

age	P	N	E(P, N)
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

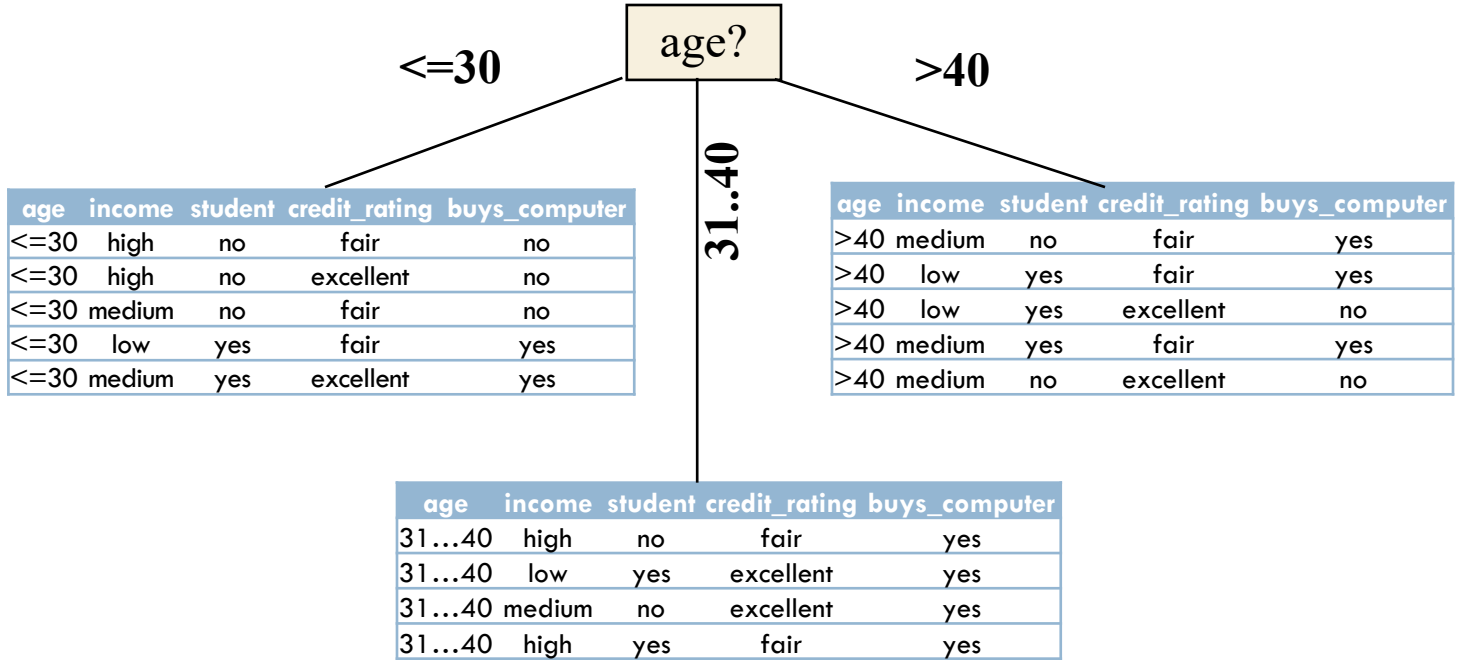
Income	P	N	E(P, N)
High	2	2	1
Medium	4	2	0.918
low	3	1	0.811

Student	P	N	E(P,N)
Yes	6	1	0.592
No	3	4	0.986

Credit Rating	P	N	E(P, N)
Fair	6	2	0.811
Excellent	3	3	1

Example: Information Gain cont.

- The tree root start from Age because it has the highest information gain.
  - The dataset is split accordingly for each value of age.



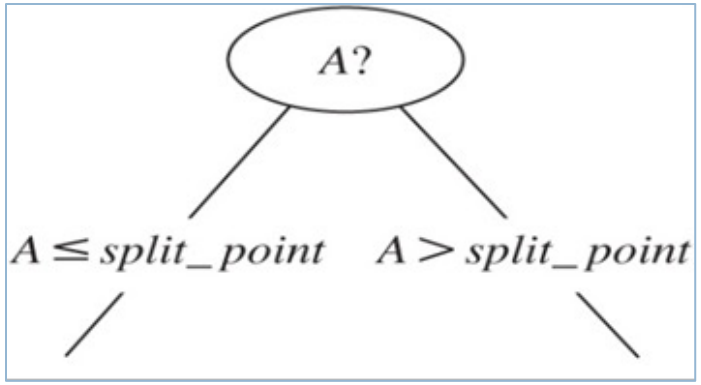
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
<=30	medium	yes	excellent	yes

age	income	student	credit_rating	buys_computer
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
>40	medium	yes	fair	yes
>40	medium	no	excellent	no

age	income	student	credit_rating	buys_computer
31…40	high	no	fair	yes
31…40	low	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes

Computing Information-Gain for 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* ($Info_A(D)$) for A is selected as the split-point for A.
- **Split D into D1 and D2 as:**
  - **D1** is the set of tuples in **D** satisfying **A** ≤ split-point,
  - **D2** is the set of tuples in **D** satisfying **A** > split-point.

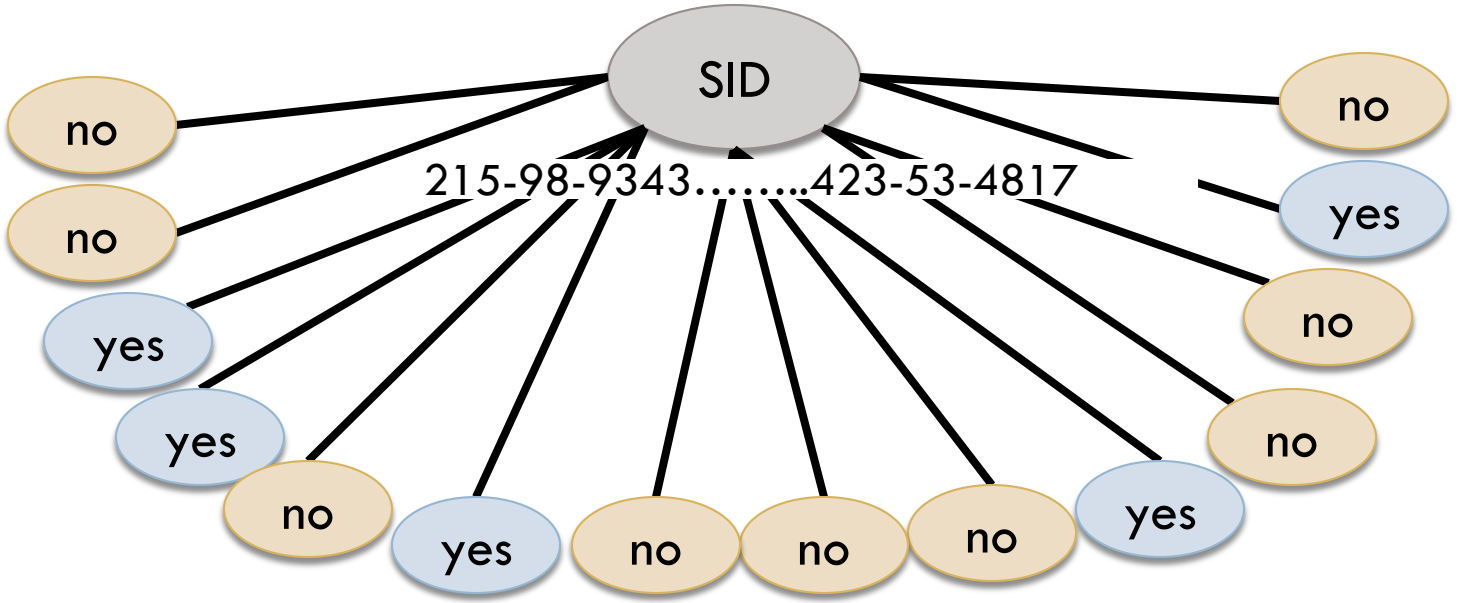


Information Gain Weakness

Added attribute "**social security number**"

SID	age	income	veteran	college_ educated	support_ hillary
215-98-9343	youth	low	no	no	no
238-34-3493	youth	low	yes	no	no
234-28-2434	middle_aged	low	no	no	yes
243-24-2343	senior	low	no	no	yes
634-35-2345	senior	medium	no	yes	no
553-32-2323	senior	medium	yes	no	yes
554-23-4324	middle_aged	medium	no	yes	no
523-43-2343	youth	low	no	yes	no
553-23-1223	youth	low	no	yes	no
344-23-2321	senior	high	no	yes	yes
212-23-1232	youth	low	no	no	no
112-12-4521	middle_aged	high	no	yes	no
423-13-3425	middle_aged	medium	yes	yes	yes
423-53-4817	senior	high	no	yes	no

Information Gain Weakness



- Since : $Info_{SID}(D) = [1/14 * -14[1/1*\log(1/1) + 0/1*\log(0/1)] = 0$
  → maximal Gain will be with SID → root will start with SID.
  - However, it is useless for classification.

Attribute Selection Measure (2): Gain Ratio(C4.5)

- Information gain measure is biased towards attributes with a large number of values.
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain).
  - It applies a kind of normalization to information gain using a "split information" value

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

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

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

Attribute Selection Measure (2): Gain Ratio(C4.5) – example

- Example : using data in the example given in slide#22, find GainRatio for attribute (outlook).

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Step 1 – compute Gain(Outlook)

Previously explained in slides 18-20)

G(PlayGolf, Outlook) = E(PlayGolf) – E(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

Step 2 – compute *SplitInfo*(Outlook)

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

$$SplitInfo_{Outlook}(PlayGolf) = -\left\{\left(\frac{5}{14}\ log_2\frac{5}{14}\right) + \left(\frac{4}{14}\ log_2\frac{4}{14}\right) + \left(\frac{5}{14}\ log_2\frac{5}{14}\right)\right\} = -\ (-\ 1.575) = 1.575$$

Step 3 – compute *GainRatio*(Outlook)

$$GainRatio(outlook) = \frac{Gain(outlook)}{SplitInfo(outlook)} = \frac{0.247}{1.575} = 0.157$$

Summary for outlook

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

Attribute Selection Measure (3):

Gini Index (CART, IBM IntelligentMiner)

- Like Entropy, Gini measures how heterogeneous, mixed or distributed some value is over a set.
- If a data set D contains examples from m classes, Gini index is defined as:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$; where p_i is the probability that a tuple in D belongs to Ci.

$$p_i = \frac{|c_{i,D}|}{|D|}$$

- If a data set D is partitioned based on attribute A into two subsets D_1 and D_2, the Gini index of D given that partitioning is:

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

- Reduction in Impurity incurred by binary split on discrete/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.

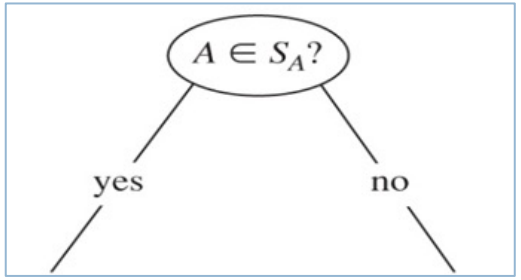
Gini Index: Example

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

- D has in buys_computer 9 tuples = "yes" and 5 = "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 D_1 : {low, medium} and D_2 :{hight}, D_1 has 10 tuples and D_2 has 4.



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

In the **10** tuples : 7 yes and 3 no

If you compute Gini{high} it will be the same exact equation and result because

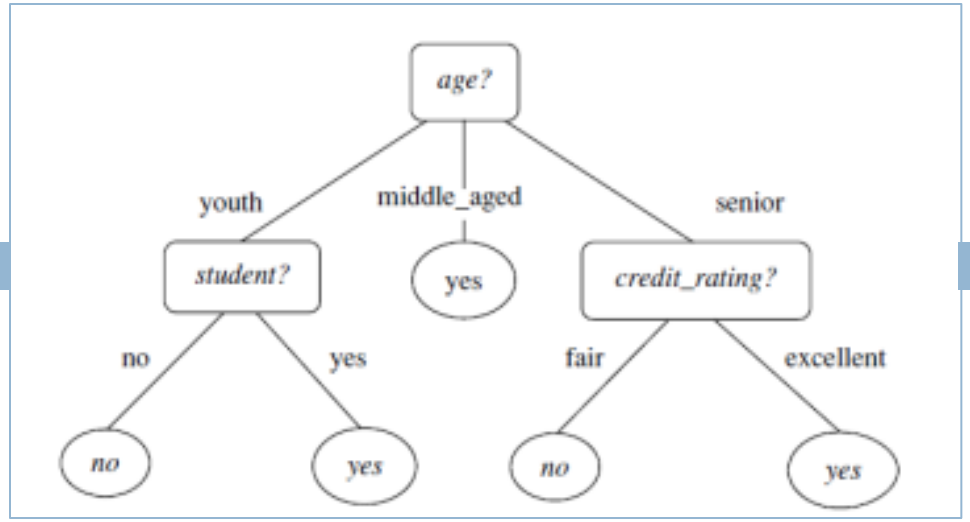
- Gini{low,medium}=Gini{high}
- Gini{low,high}=Gini{medim}
- Gini{medium,high}=Gini{low}

- Same way we compute Gini{low,high} and Gini{medium,high}.
  - Gini{low,high} is 0.458; Gini{medium,high} is 0.450; Gini{low,medium} is 0.443;
- Thus, split on the {low,medium} and {high}) since it has the lowest Gini index

Comparing Attribute Selection Measures

- The three measures, in general, return good results but…
  - Information gain:
    - Uses Entropy.
    - Biased towards multivalued attributes.
  - Gain ratio:
    - Uses Information Gain and Splitinfo.
    - 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 (Binary split) and purity in both partitions.

Rule Extraction



- Rules can be extracted from a decision tree:
  - Rules are easier to understand than large trees.
  - the knowledge is represented in the form of IF-THEN rules.
    - Rule antecedent/precondition vs. rule consequent.
- 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.
- Example: Rule extraction from buys_computer decision-tree:
  - IF age = youth AND student = no THEN buys_computer = no
  - IF age = youth AND student = yes THEN buys_computer = yes
  - IF age = mid-aged THEN buys_computer = yes
  - IF age = senior AND credit_rating = excellent THEN buys_computer = yes
  - IF age = senior AND credit_rating = fair THEN buys_computer = no

Rule Assessment

- A rule **R** can be assessed by its coverage and accuracy.
- Coverage (R): the percentage of tuples that are covered by the rule.
  - their attribute values hold true for the rule's antecedent.

- Accuracy(R): percentage of covered rules that are correctly classified by R.

$$coverage(R) = \frac{n_{covers}}{|D|}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$

n_{covers}= # of tuples covered by R

$n_{correct}$ = # of tuples correctly classified by R

Rule Assessment: Example

- IF age<=30 THEN buys_computer = no
  - $|D| = 14$
  - $n_{covers} = 5$
  - $n_{correct} = 3$

$$coverage(R) = \frac{n_{covers}}{|D|}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}.$$

  - Coverage (R)= 5/14
  - Accuracy (R)= 3/5

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

Exercise: IF age<=30 and student=no THEN buys_computer = no

Prediction using Classification Model

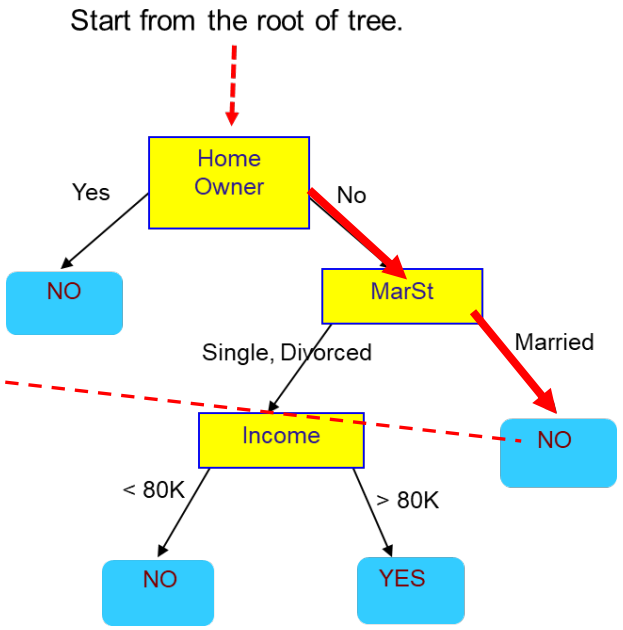
Prediction using Classification Model

- The last step in classification is to use the tested model in prediction.
  - Give a class label to a new tuple.
- Example: Apply given Decision Tree Model to predict class of given data

Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

Assign "No" to Defaulted



Model Evaluation and Selection

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use testing set of class-labeled tuples instead of training set when assessing accuracy.
  - To avoid overfitting problem
- Methods for assessing a classifier's accuracy:
  - Holdout method, random sub-sampling, Cross-validation, bootstrap.
- Comparing classifiers:
  - ROC Curves.

Classifier Evaluation Metrics:
Confusion Matrix

- Given *m* classes, an entry, $CM_{i,j}$ in a **Confusion Matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*.
- May have extra rows/columns to provide totals or recognition rate per class.
- Confusion Matrix:

Actual class\Predicted class	buy_computer =yes	buy_computer = no	Total	Model Accuracy Recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
Total	7366	2634	10000	95.42

Classifier Evaluation Metrics:
Accuracy & Error Rate

Confusion Matrix:

Actual class\Predicted class	C_1	$\sim C_1$
C1	True Positives (TP)	False Negatives (FN)
~C1	False Positives (FP)	True Negatives (TN)

Classifier Accuracy, or **recognition rate**: percentage of test set tuples that are correctly classified,

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Error rate: (misclassification rate)

- error-rate = *1 – accuracy*,

or

- $$error\ rate = \frac{FP + FN}{TP + TN + FP + FN}$$

Classifier Evaluation Metrics:
Accuracy Measures

- Problems:
  - Resubstitution error: use the training set (instead of a test set) to estimate the error rate\accuracy of a model→ accuracy\error estimate is optimistic NOT real
  - Class imbalance problem: main class of rare. That is, the data set distribution reflects a significant majority of the negative class and a minority positive class → Accuracy is misleading .
- Example: class imbalance problem.
  - Consider a 2-class problem:
    - Number of Class 0 examples = 9990
    - Number of Class 1 examples = 10

If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %

Accuracy is misleading because model does not detect any class 1 example

Classifier Evaluation Metrics: Cost-Sensitive Measures

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

- Sensitivity and Specificity
  - Sensitivity (True Positive Rate (TPR))
    - the proportion of positive tuples that are correctly identified.
  - Specificity (True Negative Rate (TNR))
    - the proportion of negative tuples that are correctly identified).
- Precision and Recall
  - Precision "exactness":
    - what % of tuples that the classifier labeled as positive are actually positive
  - Recall "completeness" (True Positive Rate (TPR))

Measure	*Formula*
accuracy, recognition rate	$\frac{TP+TN}{P+N}$
error rate, misclassification rate	$\frac{FP+FN}{P+N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP+FP}$

- Sensitivity = Recall
- an Inverse relationship between precision & recall

False Positive Rate (FPR)

FPR = FP/(FP+TN)

False Negative Rate (FNR)

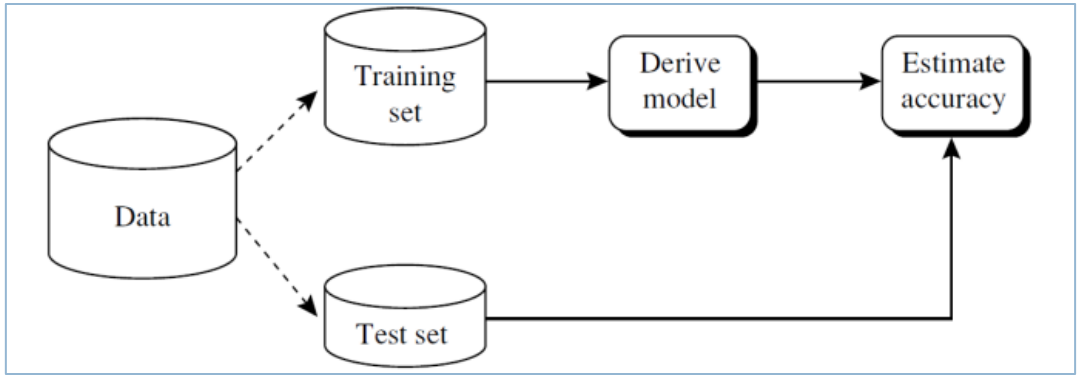
FNR= FN/(FN+TP)

Obtaining reliable classifier accuracy estimates

- Common techniques for assessing accuracy, based on randomly sampled partitions of the given data into testing and training sets are :
  - Holdout and Random Sampling methods
  - Cross-validation methods
  - Bootstrap methods

Evaluating Classifier Accuracy:
Holdout Method and Random Sampling

- **Holdout method:** Unbiased and efficient but require a large number of samples; thus, used for data set with large number of samples.
  - Given data is randomly partitioned into two independent sets:
    - Training set (e.g., 2/3) for model construction.
    - Test set (e.g., 1/3) for accuracy estimation.



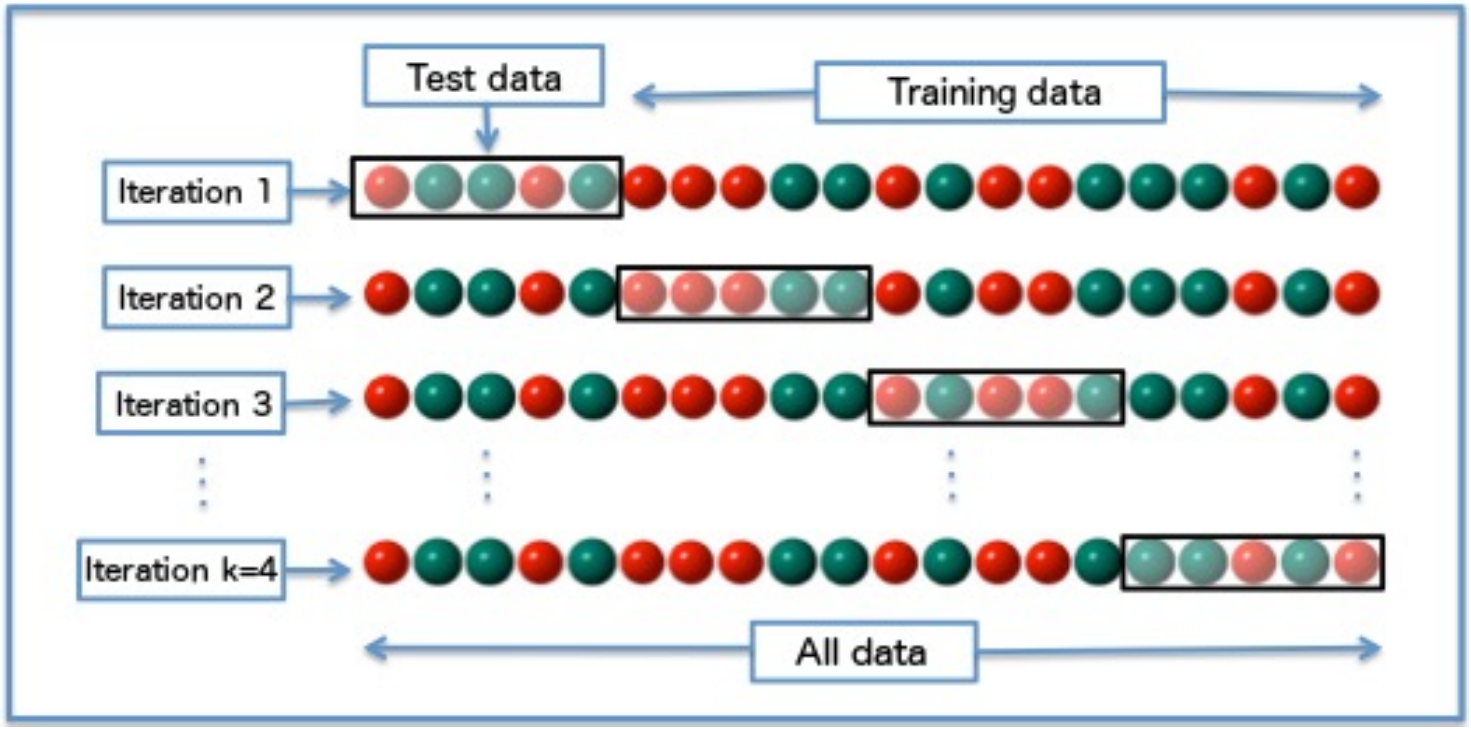
- **Random sampling:** a variation of holdout.
  - Repeat holdout k times, accuracy = the average of the accuracies obtained from each iteration.

Evaluating Classifier Accuracy:
Cross-Validation Method

- **Cross-validation** (k-fold, where k = 10 is the most popular):
  - Randomly partition the data into k mutually exclusive subsets, each approximately equal size.
  - At i-th iteration, use **Di** as test set and others as training set.

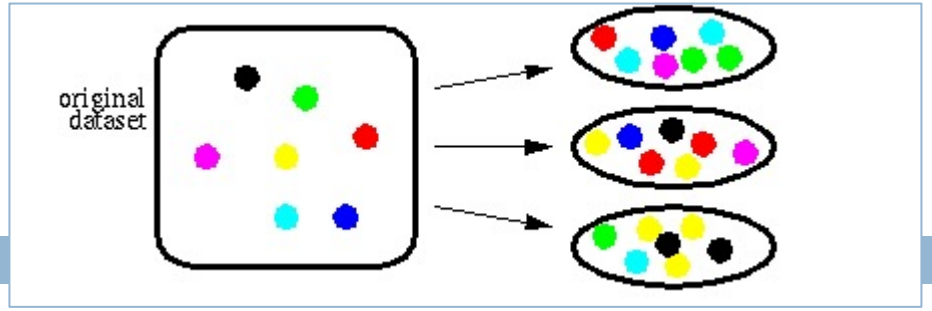
  - Variations: (special types of Cross-validation)
    - **Leave-one-out:** k folds where k = # of tuples, for small sized data.
    - **Stratified cross-validation:** folds are stratified so that class distribution in each fold is approximately the same as that in the initial data.

Evaluating Classifier Accuracy:
Cross-Validation Method

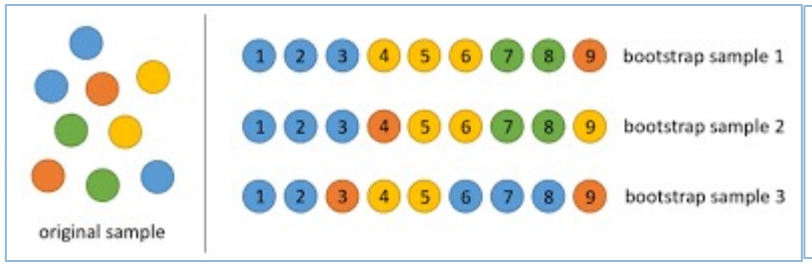


- Cross-validation is nearly unbiased so it is preferred to estimate the quality of learning algorithms in Machine Learning.

Evaluating Classifier Accuracy:
Bootstrap



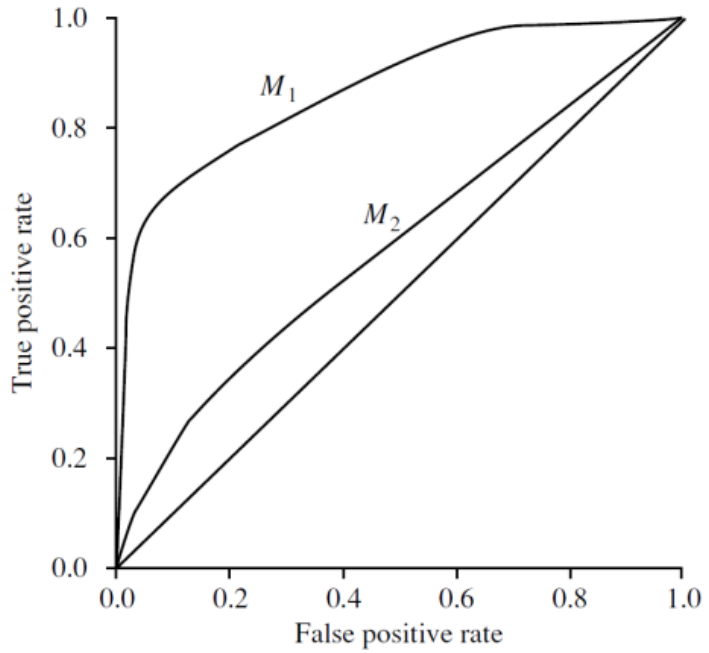
- **Bootstrap:**
  - Works well with small data sets
  - Samples the given training tuples uniformly with replacement
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set.
- Several boostrap methods, and a common one is 0.632 boostrap
  - About 63.2% of the original data will end up in the bootstrap, and the remaining 36.8% will form the test set.
- The sampling procedure can be repeated *k* times, where in each iteration, the current test set is used to obtain an accuracy estimate of the model obtained from the current bootstrap sample.



Given a data set of **d** tuples.
The data set is sampled **d times**, with replacement: a training set **of d samples.**
The data tuples that did not make it into the training set end up forming **the test set.**

Model Selection: ROC Curves

- Receiver Operating Characteristics (ROC) curves:
  - used for visual comparison of classification models.
  - Shows the trade-off between true positive rate and false positive rate.



ROC Graph:

- Vertical axis represents the true positive rate.
- Horizontal axis represents the false positive rate.
- The area under the ROC curve is a measure of the accuracy of the model.
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model.
- A model with perfect accuracy will have an area of 1.0

Summary

- Classification: Supervised learning.
- Decision Tree Induction: Splitting Methods.
- Prediction using classification model.
- Model Evaluation and Selection: Error estimation, Accuracy, precision, ROC curves for model selection.