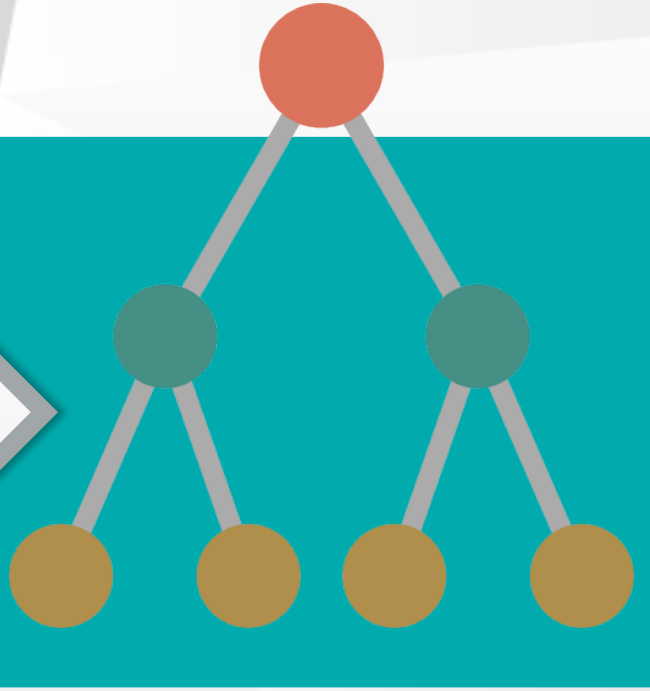
# Unit-6

## Classification and Prediction



# Outline

- Introduction to classification
- Classification & prediction issues
- Classification methods
- Prediction methods

# Introduction to classification

- Classification is a **supervised learning method.**
- It is a data mining function that **assigns items in a collection to target categories or classes**.
- The **goal of classification** is to **accurately predict the target class** for each case in the data.
- **For example**, a classification model could be used to identify loan applicants as low, medium, or high credit risks.
- In supervised learning, the learner(computer program) is provided with two sets of data, **training data set** and **test data set**.
- The idea is for the learner to “learn” from a set of labeled examples in the training set so that it can identify **unlabeled examples** in the **test set with the highest possible accuracy**.

# Introduction to classification (Cont..)

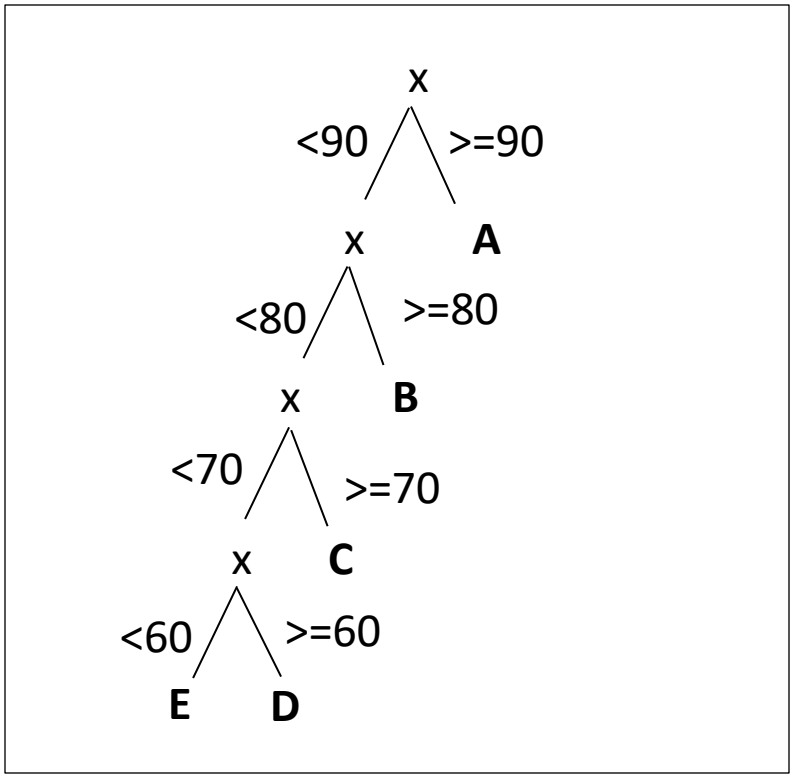
- Suppose a Database D is given as **D = $\{t_1, t_2, .. t_n\}$** and a set of desired classes are **C=$\{C_1, \ldots, C_m\}$.**
- The **classification problem** is to define the mapping m in such a way that which tuple of database **D belongs to which class of C**.
- Actually we divides D into **equivalence classes.**
- **Prediction** is similar, but it viewed as **having infinite number of classes**.

# Classification Example

- Teachers **classify** students grades as **A,B,C,D or E**.
- Identify individuals with **credit risks** (high, low, medium or unknown).
- In **cricket** (batsman, bowler, all-rounder)
- **Websites** (educational, sports, music)

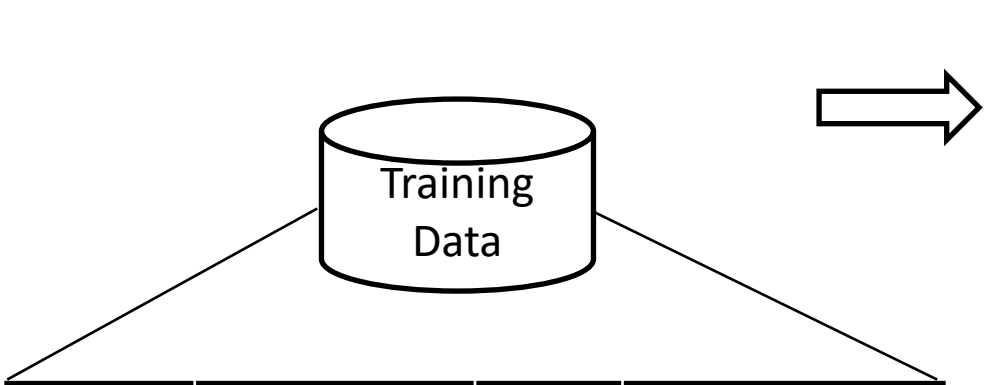
# Classification Example (Cont..)

- How teachers give grades to students based on their obtained marks?
  - If **x >= 90** then **A** grade.
  - If **80 <= x < 90** then **B** grade.
  - If **70 <= x < 80** then **C** grade.
  - If **60 <= x < 70** then **D** grade.
  - If **x < 60** then **E** grade.

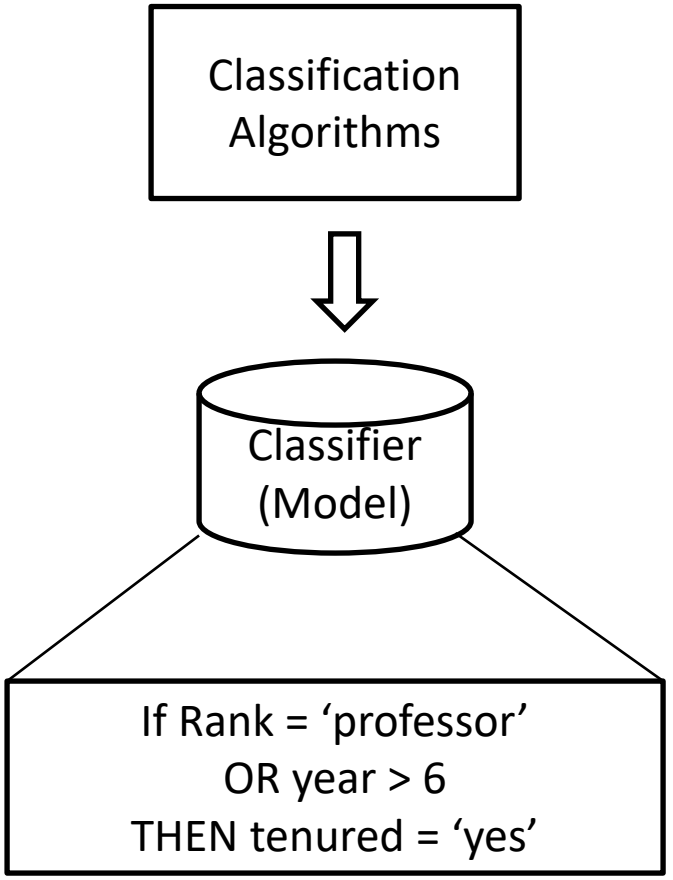


# Classification : a two step process

## 1) Model Construction

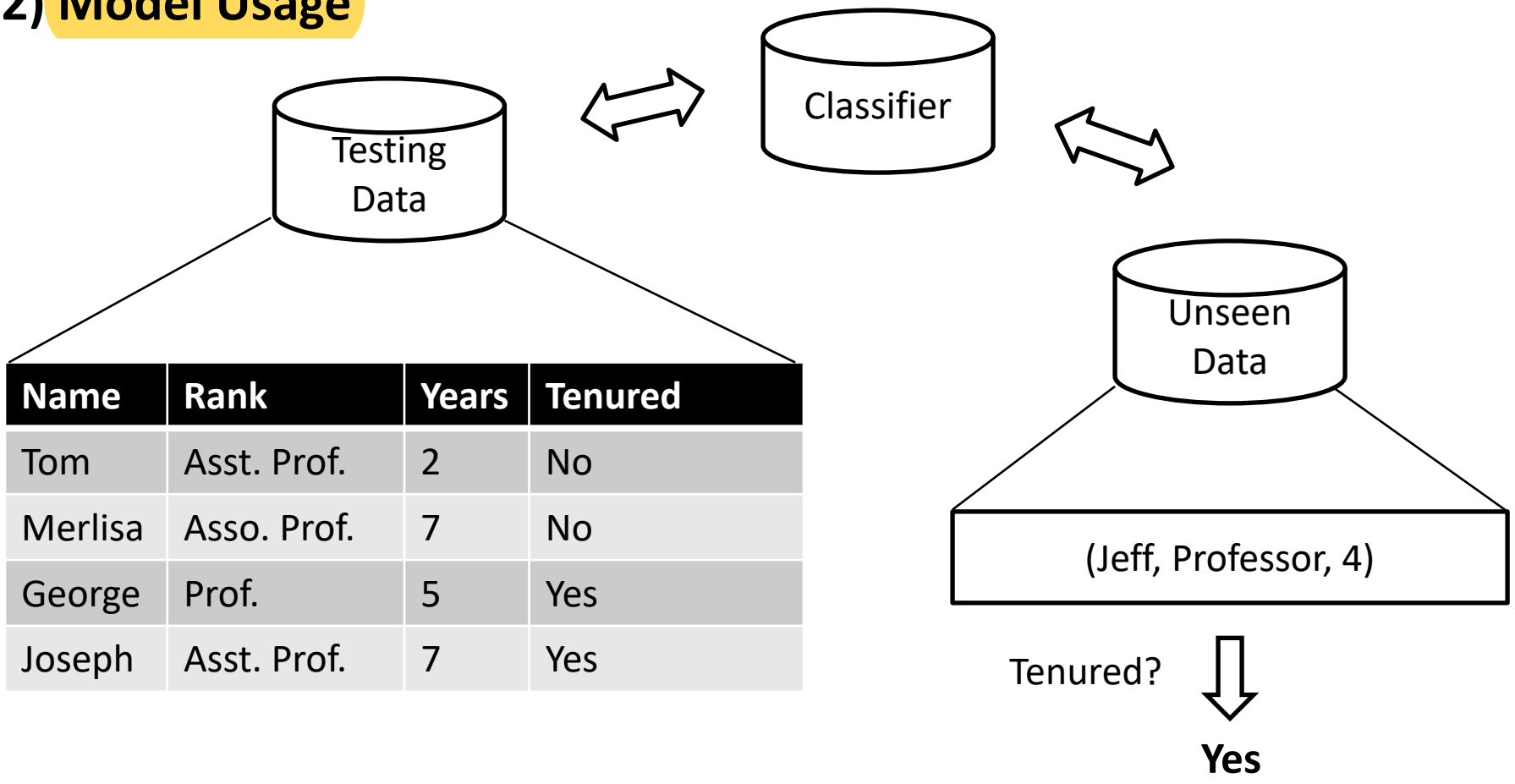



| Name | Rank | Years | Tenured |
|---|---|---|---|
| Mike | Asst. Prof. | 3 | No |
| Mary | Asst. Prof. | 7 | Yes |
| Bill | Prof. | 2 | Yes |
| Jim | Asso. Prof. | 7 | Yes |
| Dave | Asst. Prof. | 6 | No |
| Anne | Asso. Prof. | 3 | No |



# Classification : a two step process (Cont..)

## 2) Model Usage




| Name | Rank | Years | Tenured |
|---|---|---|---|
| Tom | Asst. Prof. | 2 | No |
| Merlisa | Asso. Prof. | 7 | No |
| George | Prof. | 5 | Yes |
| Joseph | Asst. Prof. | 7 | Yes |

# Classification : a two step process (Cont..)

## 1) Model Construction

- Describing a **set of predetermined classes** :
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute.
  - The **set of tuples used for model construction** is called as **training set**.
  - The model is represented as **classification rules, decision trees, or mathematical formulae**.

## 2) Model Usage

- For **classifying future or unknown objects**
  - **Estimate accuracy** of the **model**
  - The known label of test sample is compared with the classified result from the model.
  - Accuracy rate is the percentage of test set samples that are correctly classified by the model.

# Classification & prediction issues

## Data Preparation

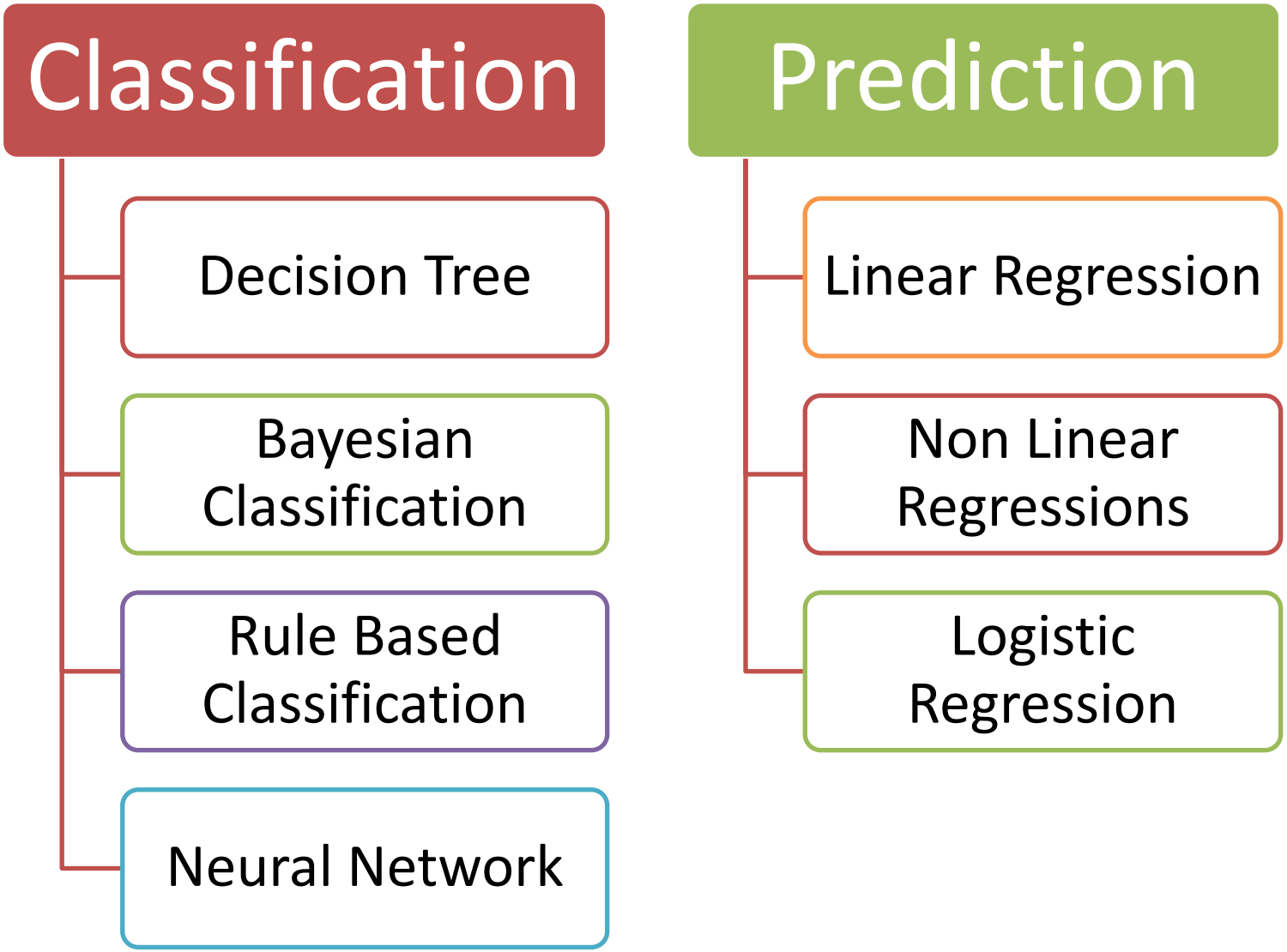
- **Data cleaning**
  - **Pre-process data** in order to **reduce noise and handle missing values**.
- **Relevance analysis** (Feature selection)
  - **Remove** the irrelevant or **redundant attributes**.
- **Data transformation**
  - Generalize the data to higher level concepts using **concept hierarchies** and/or normalize data which involves scaling the values.

# Classification & prediction issues (Cont..)

## Evaluating Classification Methods

- **Predict accuracy**
  - This refers the ability of the model **to correctly predict the class** label of new or previously unseen data.
- **Speed and scalability**
  - Time to **construct** model
  - Time to **use** the model
- **Robustness**
  - Handling noise and missing values
- **Interpretability**
  - Understanding and insight provided by model
- **Goodness of rules**
  - Decision **tree size**
  - Strongest rule or not

# Classification & prediction methods



# Decision tree

- One of the most **common tasks is to build models** for the prediction of the class of an object on the basis of its attributes.
- The objects can be seen as a customer, patient, transaction, e-mail message or even a single character.
- **Attributes of patient object** can be heart rate, blood pressure, weight and gender etc.
- The class of the patient object would most commonly be **positive/negative** for a certain **disease**.

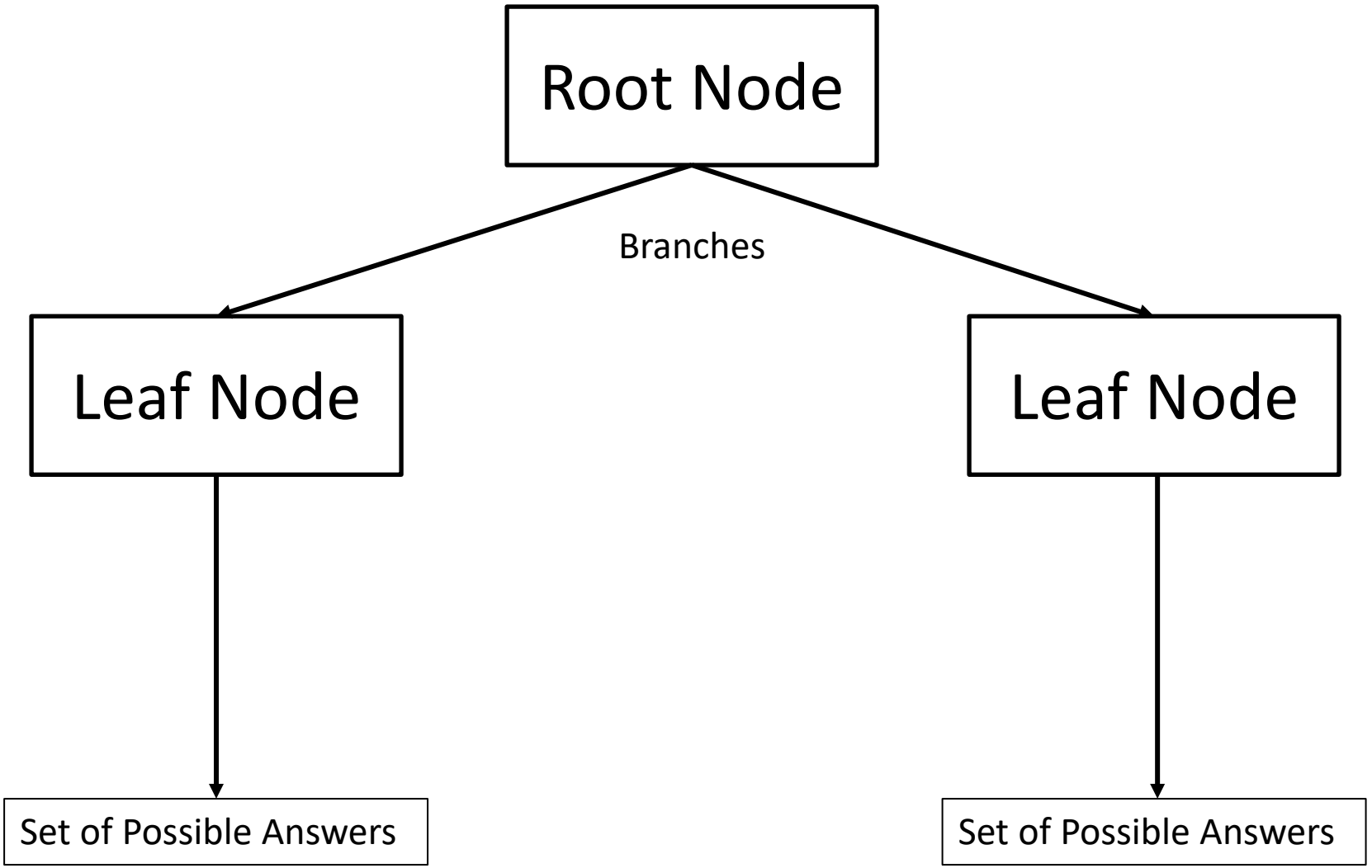
# Decision tree (Cont..)

- In decision tree are represented by a **fixed set of attributes** (e.g. gender) and their values (e.g. male, female) described as **attribute-value pairs**.
- If the attribute has **small** number of **disjoint possible values** (e.g. high, medium, low) or there are only two possible classes (e.g. true, false) then decision tree learning is easy.
- Extension to decision tree algorithm also handles real value attributes (e.g. salary).
- It gives a class label to each instance of dataset.

# Decision tree (Cont..)

- Decision tree is a classifier in the form of a tree structure
  - **Decision node**: Specifies a test on a single attribute
  - **Leaf node**: Indicates the value of the target attribute
  - **Arc/edge**: Split of one attribute
  - **Path**: A disjunction of test to make the final decision

# Decision tree representation- example



# Key requirements for classification

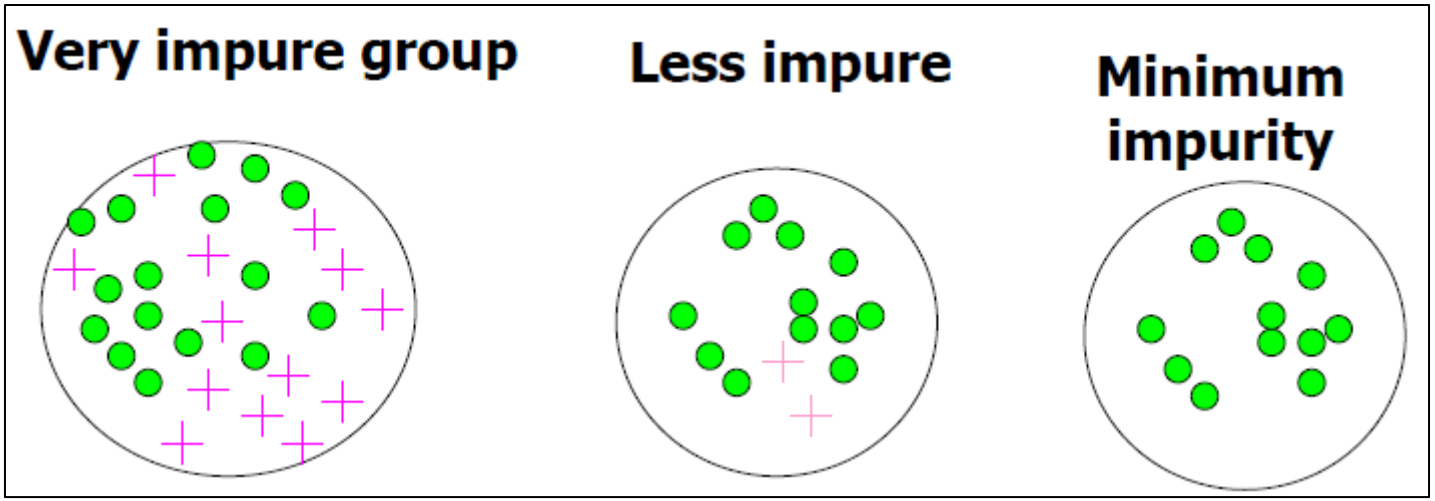
- **Sufficient data:**
  - **Enough training cases** should be provided to learn the model.
- **Attribute-value description:**
  - **Object** or case must be expressible in terms of a **fixed collection of properties or attributes** (e.g., hot, mild, cold).
- **Predefined classes (target values):**
  - The target function has **discrete output values** (boolean or multiclass)

# Important terms for decision tree

- **Entropy**
- **Information Gain**
- **Gini Index**

# Entropy (E)

- It defines the **certainty of a decision**
  - **1** if **completely certain**,
  - **0** if **completely uncertain**,
  - Normally data remains between 0 to 1 as entropy, a probability-based measure used to **calculate the amount of uncertainty**.



# Entropy (E) (Cont..)

- It measures that of **how much information we don't know** (how uncertain we are about the data).
- It can be also **used to measure how much information** we gain from an attribute when the target attribute is revealed to us.
- **Which attribute is best?**
  - ✓ The attribute with the **largest expected reduction in entropy** is the 'best' attribute to use next.
  - ✓ Because if we have a large expected reduction it means taking away that attribute has a big effect, meaning it must be very certain.

# Entropy (E) (Cont..)

- A decision tree is built **top-down** from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous).
- ID3 algorithm uses entropy to calculate the homogeneity of a sample.
- If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.

# Information Gain

- Information gain can be used for **continues-valued** (numeric) **attributes**.
- The attribute which has the **highest information gain** is selected **for split**.
- Assume, that there are two classes P(positive) & N(negative).
- Suppose we have **S** samples, out of these **p** samples belongs to **class P** and **n** samples belongs to **class N**.
- The amount of information, needed to decide split in S belongs to P or N & that is defined as

$$I(p,n) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}$$

# Gini Index

- Assume there exist several possible split values for each attribute.
- We may need other tools, such as clustering, to get the possible split values.
- It can be modified for categorical attributes.
- An alternative method to information gain is called the **Gini Index.**
- Gini is used in CART (Classification and Regression Trees).
- If a dataset T Contains examples from n classes, gini index, gini(T) is defined as

$$\text{Gini (T)} = 1 - \sum_{j=1}^{n} p_j^2$$

  - **n**: the number of classes
  - $\mathbf{p_j}$: the probability that a tuple in D belongs to class Ci

# Gini Index (Cont..)

- After splitting T into two subsets T1 and T2 with sizes N1 and N2, the gini index of the split data is defined as

$$\textbf{Gini}_{\textbf{split}}\ \textbf{(T)} = \frac{N_1}{N}\,\text{gini}\,(T_1) + \frac{N_2}{N}\,\text{gini}\,(T_2)$$

# Example – ID3

| Age | Income | Student | Credit_Rating | Class : 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 |

# Solution – ID3

- **Class P** : buys_computer = “Yes” (9 records)
- **Class N** : buys_computer = “No” (5 records)
- Total number of Records **14**.
- Now, Information Gain = I (p,n)

$$\mathrm{I}(p,n) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}$$

$$\mathrm{I}(9,5) = -\frac{9}{14} log_2 \frac{9}{14} - \frac{5}{14} log_2 \frac{5}{14}$$

$$\boxed{\mathrm{I}(9,5) = 0.940}$$

# Solution – ID3 (Age <=30,31..40,>40)

| 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 | Stu. | Cr_Rating | Buys |
|---|---|---|---|---|
| 31..40 | High | No | Fair | Yes |
| 31..40 | Low | Yes | Excellent | Yes |
| 31..40 | Medium | No | Excellent | Yes |
| 31..40 | High | Yes | Fair | Yes |

| Age | Income | Stu. | Cr_Rating | Buys |
|---|---|---|---|---|
| >40 | Medium | No | Fair | Yes |
| >40 | Low | Yes | Fair | Yes |
| >40 | Low | No | Excellent | No |
| >40 | Medium | Yes | Fair | Yes |
| >40 | Medium | No | Excellent | No |

# Solution – ID3 (Age <=30)

- Compute the information gain & Entropy For Age **<=30**,
  - $P_i$ = Yes class = 2
  - $N_i$ = No class = 3

  So, Information Gain = I (p,n)

$$\mathrm{I}(p,n) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}$$

$$\mathrm{I}(2,3) = -\frac{2}{5} log_2 \frac{2}{5} - \frac{3}{5} log_2 \frac{3}{5}$$

$$\boxed{\mathrm{I}(2,3) = 0.971}$$

# Solution – ID3

| Age | $P_i$ | $N_i$ | I ($P_i$ , $N_i$) |
|---|---|---|---|
| <=30 | 2 | 3 | 0.971 |
| 31..40 | 4 | 0 | 0 |
| >40 | 3 | 2 | 0.971 |

- So the expected information needed to classify a given sample if the samples are partitioned according to age is,
- Calculate entropy using the values from the Table and the formula given below:

$$E(A) = \sum_{i=1}^{v} \frac{P_i+N_i}{p+n} I(P_i, N_i)$$

$$E(\text{Age}) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2)$$

E(Age) = 0.694

# Solution – ID3

$$\text{Gain (Age)} = I(p, n) - E(\text{Age})$$

$$= 0.940 - 0.694$$

$$= 0.246$$

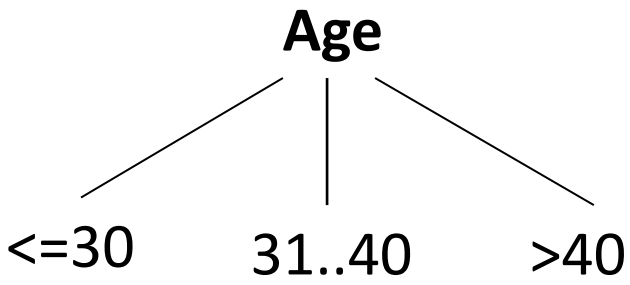
Similarly,

| Gain | value |
|---|---|
| Gain (age) | 0.246 |
| Gain (income) | 0.029 |
| Gain (student) | 0.151 |
| Gain (credit_rating) | 0.048 |

So, here we start decision tree with root node **Age**.

# Solution – ID3

- Now the age has highest information gain among all the attributes, so select age as test attribute and create the node as age and show all possible values of age for further splitting.



# Solution – ID3 (Age <= 30)

| 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 |

# Solution – ID3 (Age <=30)

- Compute Information gain & Entropy for Age with sample $S_{<=30}$.
- For age <=30,
  - Pi = Yes = 2
  - Ni = No = 3

$$\mathrm{I}(p,n) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}$$

$$\mathrm{I}(2,3) = -\frac{2}{5} log_2 \frac{2}{5} - \frac{3}{5} log_2 \frac{3}{5}$$

$$\boxed{\mathrm{I}(3,2) = 0.971}$$

# Solution – ID3 (Age <= 30, Income)

| Income | $P_i$ | $N_i$ | I ($P_i$ , $N_i$) |
|---|---|---|---|
| High | 0 | 2 | 0 |
| Medium | 1 | 1 | 1 |
| Low | 1 | 0 | 0 |

In above table high (0,2) homogeneous so I(0,2) = 0, Medium equal portion so I(1,1) = 1 & Low I(1,0) = 0.

$$E(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{p+n} I(P_i, N_i)$$

$$E(\text{Income}) = \frac{2}{5} I(0,2) + \frac{2}{5} I(1,1) + \frac{1}{5} I(1,0)$$

$$\boxed{E(\text{Income}) = 0.4}$$

Gain ($S_{<=30}$,Income)

= I (p, n) – E (Income)

= 0.971 – 0.4

= 0.571

# Solution – ID3 (Age <= 30, Student)

| student | $P_i$ | $N_i$ | I ($P_i$ , $N_i$) |
|---|---|---|---|
| No | 0 | 3 | 0 |
| Yes | 2 | 0 | 0 |

In above table I (0,3) = 0 & I (2,0) = 0 So E(Student) is 0.

$$E(\text{Student}) = 0$$

---

$$\text{Gain}(S_{<=30}, \text{Student}) = I(p,n) - E(\text{Student})$$
$$= 0.971 - 0$$
$$= 0.971$$

# Solution – ID3 (Age <= 30, credit_rating)

| credit_rating | $P_i$ | $N_i$ | I ($P_i$ , $N_i$) |
|---|---|---|---|
| Fair | 1 | 2 | 0.918 |
| Excellent | 1 | 1 | 1 |

$$E(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{p+n} I(P_i, N_i)$$

$$E(\text{credit\_rating}) = \frac{3}{5} I(1,2) + \frac{2}{5} I(1,1)$$

$$\boxed{E(\text{credit\_rating}) = 0.951}$$

---

Gain ($S_{<=30}$,credit_rating) = I (p, n) – E (credit_rating)

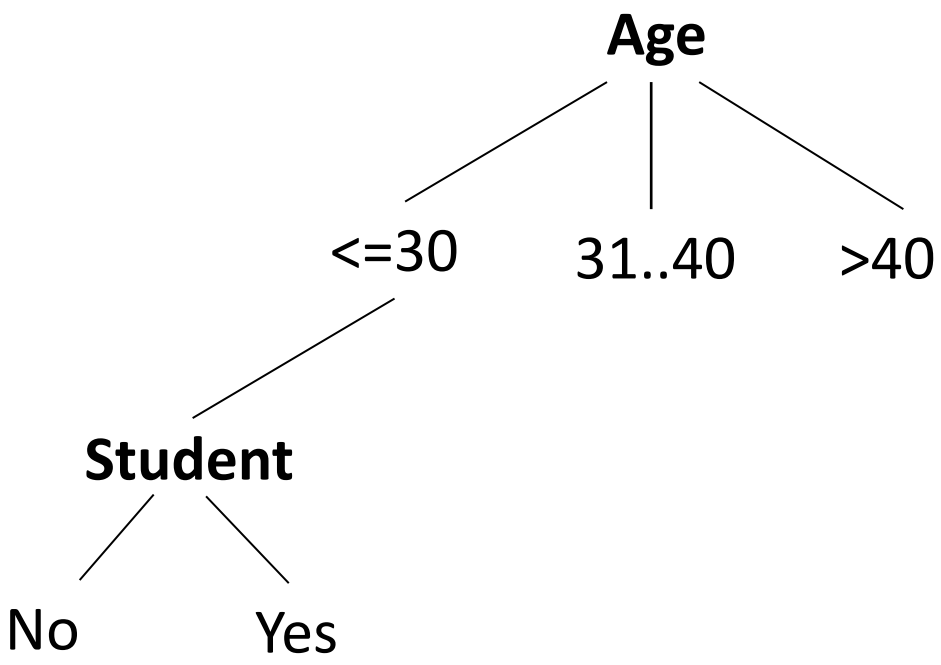
= 0.971 – 0.951

= 0.020

# Solution – ID3 (Age <=30)

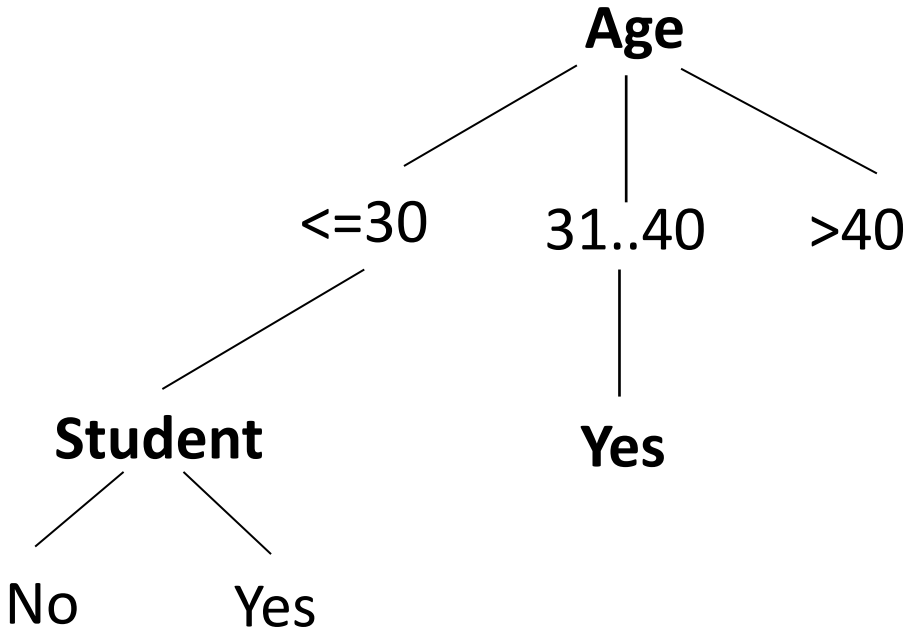
| Gain (Age <= 30) | value |
|---|---|
| Income | 0.571 |
| Student | 0.971 |
| Credit_rating | 0.020 |

As shown in table we get maximum gain for student so, select **student** as leaf node for age <=30



# Solution – ID3 (Age 31..40)

| 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** |



# Solution – ID3 (Age > 40)

| Age | Income | Student | Credit_Rating | Buys_Computer |
|---|---|---|---|---|
| >40 | Medium | No | Fair | Yes |
| >40 | Low | Yes | Fair | Yes |
| >40 | Low | No | Excellent | No |
| >40 | Medium | Yes | Fair | Yes |
| >40 | Medium | No | Excellent | No |

# Solution – ID3 (Age > 40)

- Compute Information gain for Age with sample $S_{>40}$.
- For age > 40,
  - Pi = Yes = 3
  - Ni = No = 2

$$\mathrm{I}(p,n) = -\frac{p}{p+n} log_2 \frac{p}{p+n} - \frac{n}{p+n} log_2 \frac{n}{p+n}$$

$$\mathrm{I}(3,2) = -\frac{3}{5} log_2 \frac{3}{5} - \frac{2}{5} log_2 \frac{2}{5}$$

$$\boxed{\mathrm{I}(3,2) = 0.971}$$

# Solution – ID3 (Age > 40, Income)

| Income | $P_i$ | $N_i$ | I ($P_i$ , $N_i$) |
|---|---|---|---|
| High | 0 | 0 | 0 |
| Medium | 2 | 1 | 0.918 |
| Low | 1 | 1 | 1 |

$$E(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{p+n} I(P_i, N_i)$$

$$E(\text{Income}) = \frac{0}{5} I(0,0) + \frac{3}{5} I(2,1) + \frac{2}{5} I(1,1)$$

$$\boxed{E(\text{Income}) = 0.951}$$

Gain ($S_{>40}$,Income)

= I (p, n) – E (Income)

= 0.971 – 0.951

= 0.020

# Solution – ID3 (Age > 40, credit_rating)

| Credit_rating | $P_i$ | $N_i$ | $I(P_i, N_i)$ |
|---|---|---|---|
| Fair | 3 | 0 | 0 |
| Excellent | 0 | 2 | 0 |

$$E(A) = \sum_{i=1}^{v} \frac{P_i + N_i}{p+n} I(P_i, N_i)$$

$$E(\text{credit\_rating}) = \frac{3}{5} I(3,0) + \frac{2}{5} I(0,2)$$

$$\boxed{E(\text{credit\_rating}) = 0}$$

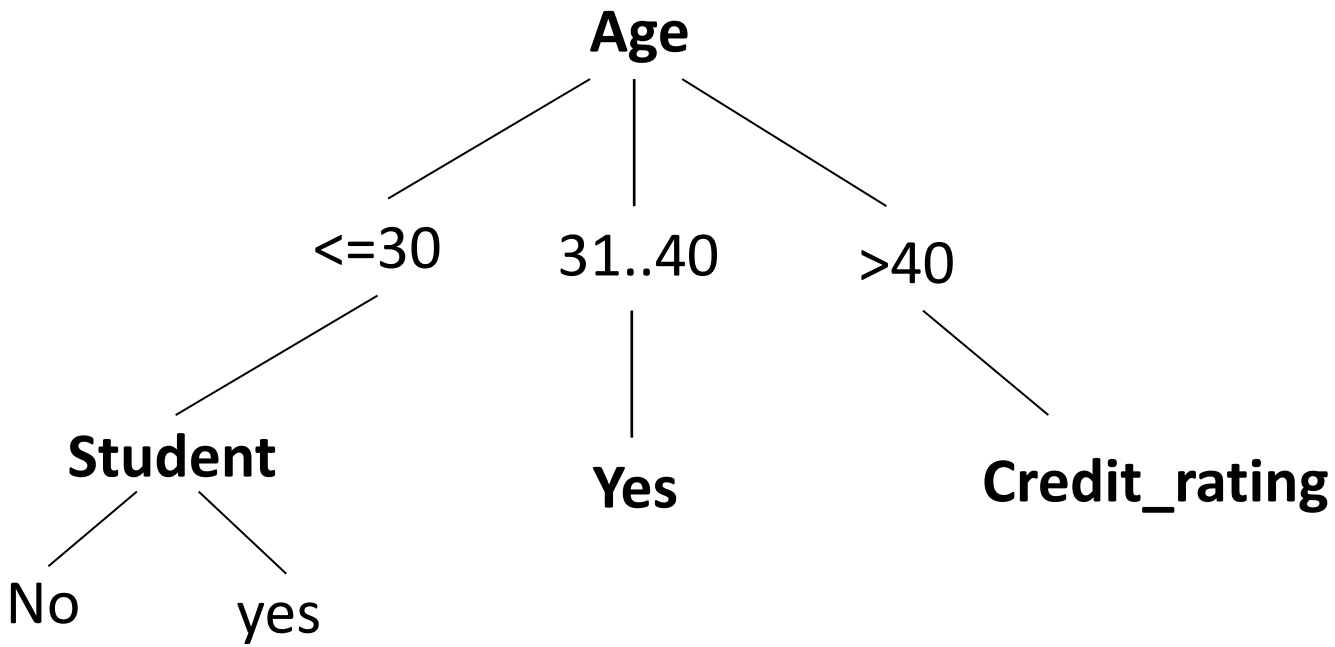
---

$$\text{Gain}(S_{>40}, \text{credit\_rating}) = I(p, n) - E(\text{credit\_rating})$$
$$= 0.971 - 0$$
$$= 0.971$$

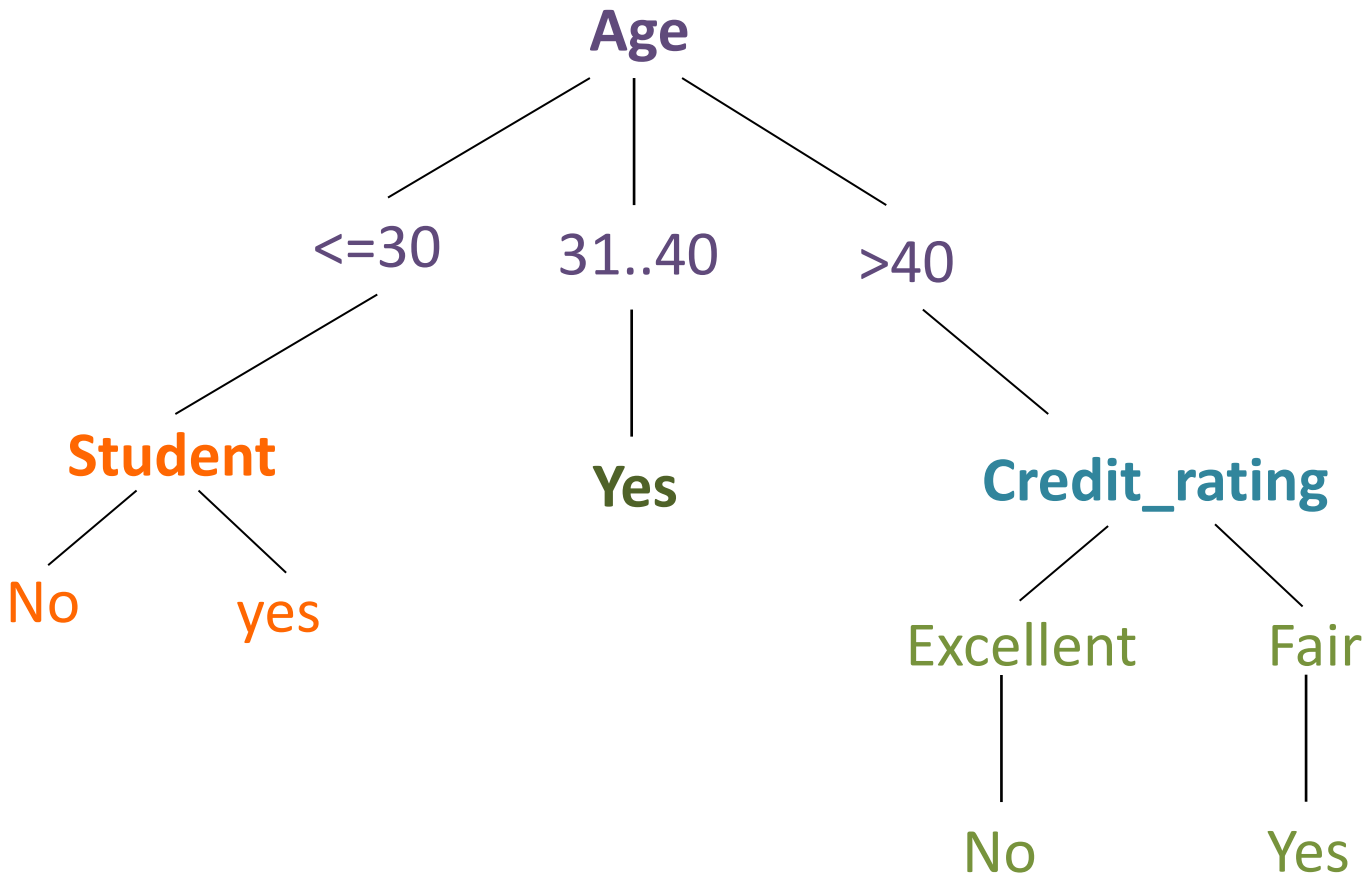
# Solution – ID3 (Age > 40)

| Gain (Age > 40) | value |
|---|---|
| Income | 0.020 |
| Credit_rating | 0.971 |

As shown in table we get maximum gain for credit_rating so, select credit_rating as leaf node for age > 40



# Decision Tree – ID3



# Classification rules from decision tree

- IF age = “<=30” AND student = “no” THEN buys_computer = “no”
- IF age = “<=30” AND student = “yes” THEN buys_computer = “yes”
- IF age = “31..40” THEN buys_computer = “yes”
- IF age = “>40” AND credit_rating = “excellent” THEN buys_computer = “no”
- IF age = “>40” AND credit_rating = “fair” THEN buys_computer = “yes”

# Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - ✓ Too many branches, some may reflect anomalies due to noise or outliers
  - ✓ Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - ✓ Prepruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - ✓ Postpruning: *Remove branches* from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

# Bayesian Classification

- Thomas Bayes, who proposed the Bayes Theorem so, it named Bayesian theorem.
- It is statistical method & supervised learning method for classification.
- **It can solve problems involving both categorical and continuous valued attributes**.
- Bayesian classification is used to find conditional probabilities.

# The Bayes Theorem

- The Bayes Theorem:
  - $P(H|X) = \frac{\mathbf{P(X|H)\ P(H)}}{\mathbf{P(X)}}$
- **P(H|X)** : Probability that the customer will buy a computer given that we know his age, credit rating and income. (Posterior Probability of H)
- **P(H)** : Probability that the customer will buy a computer regardless of age, credit rating, income (Prior Probability of H)
- **P(X|H)** : Probability that the customer is 35 years old, have fair credit rating and earns $40,000, given that he has bought computer (Posterior Probability of X)
- **P(X)** : Probability that a person from our set of customers is 35 years old, have fair credit rating and earns $40,000. (Prior Probability of X)

# Naïve Bayes classifier - Example

| Age | Income | Student | Credit_Rating | Class : 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 |

# Naïve Bayes classifier - Solution

| **Age** | | |
|---|---|---|
| P (<=30 \| **Yes**) = 2/9 | P (<=30 \| **No**) = 3/5 | P (**Yes**) = 9/14<br>P (**No**) = 5/14 |
| P (31..40 \| **Yes**) = 4/9 | P (31..40 \| **No**) = 0/5 | |
| P (> 40 \| **Yes**) = 3/9 | P (> 40 \| **No**) = 2/5 | |
| **Income** | | |
| P (High \| **Yes**) = 2/9 | P (High \| **No**) = 2/5 | |
| P (Medium \| **Yes**) = 4/9 | P (Medium \| **No**) = 2/5 | |
| P (Low \| **Yes**) = 3/9 | P (Low \| **No**) = 1/5 | |
| **Student** | | |
| P (No \| **Yes**) = 3/9 | P (No \| **No**) = 4/5 | |
| P (Yes \| **Yes**) = 6/9 | P (Yes \| **No**) = 1/5 | |
| **Credit_rating** | | |
| P (Fair \| **Yes**) = 6/9 | P (Fair \| **No**) = 2/5 | |
| P (Excellent \| **Yes**) = 3/9 | P (Excellent \| **No**) = 3/5 | |

# Naïve Bayes classifier - Solution

- **An unseen sample Y = (<=30, Low, Yes, Excellent)**
- P(Y|Yes).P(Yes) = P(<=30|Yes). P(Low|Yes). P(Yes|Yes). P(Excellent|Yes) . P(Yes)

  = 2/9 * 3/9 * 6/9 * 3/9 * 9/14

  = 0.010582

---

- P(Y|No).P(No) = P(<=30|No). P(Low|No). P(Yes|No). P(Excellent|No) . P(No)

  = 3/5 * 1/5 * 1/5 * 3/5 * 5/14

  = 0.005142

- Choose the class so that it maximizes this probability, this means that new instance will be classified as **Yes** (Buys_computer)

# Try yourself (Bayesian Classification)

| Car No | Color | Type | Origin | Stolen? |
|---|---|---|---|---|
| 1 | Red | Sports | Domestic | Yes |
| 2 | Red | Sports | Domestic | No |
| 3 | Red | Sports | Domestic | Yes |
| 4 | Yellow | Sports | Domestic | No |
| 5 | Yellow | Sports | Imported | Yes |
| 6 | Yellow | SUV | Imported | No |
| 7 | Yellow | SUV | Imported | Yes |
| 8 | Yellow | SUV | Domestic | No |
| 9 | Red | SUV | Imported | No |
| 10 | Red | Sports | Imported | Yes |

Unseen Data

Y = <Red, Domestic, SUV>

Actual Data

Y = <Red, Sports, Domestic>

0.024,0.072 (Unseen)

0.192, 0.096 (Actual)

# Rule Based Classification

- It is featured by building rules based on an object attributes.
- Rule-based classifier makes use of a set of **IF-THEN rules** for classification.
- We can express a rule in the following from
  - IF condition THEN conclusion
- Let us consider a rule R1,

  R1: **IF age=youth AND student=yes THEN buy_computer=yes**
  - The IF part of the rule is called rule antecedent or precondition.
  - The THEN part of the rule is called rule consequent (conclusion).
  - The antecedent (IF) part the condition consist of one or more attribute tests and these tests are logically ANDed.
  - The consequent (THEN) part consists of class prediction.

# Rule Based Classification (Cont..)

- We can also write rule R1 as follows:

  R1: ((age = youth) ^ (student = yes)) => (buys_computer = yes)

  - If the condition (that is, all of the attribute tests) in a rule antecedent holds true for a given tuple, we say that the rule antecedent is satisfied and that the rule covers the tuple.

- A rule **R can be assessed by its coverage and accuracy**.
- Given a tuple X, from a class labeled data set D, let it covers the number of tuples by R; the number of tuples correctly classified by R; and |D| be the number of tuples in D.
- We can define the coverage and accuracy of R as

$$\text{Coverage (R)} = \frac{n_{covers}}{|D|} \qquad \text{Accuracy (R)} = \frac{n_{correct}}{n_{covers}}$$

# Metrics for Evaluating Classifier Performance

| *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 | $\frac{TP}{TP+FP}$ |
| $F$, $F_1$, $F$-score, harmonic mean of precision and recall | $\frac{2 \times precision \times recall}{precision + recall}$ |
| $F_\beta$, where $\beta$ is a non-negative real number | $\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$ |

***TP,TN,FP,P, N* refer to the number of true positive, true negative, false positive, positive, and negative samples, respectively**

# Metrics for Evaluating Classifier Performance

- True positives (TP): These refer to the positive tuples that were correctly labeled by the classifier. Let TP be the number of true positives.
- True negatives (TN): These are the negative tuples that were correctly labeled by the classifier. Let TN be the number of true negatives.
- False positives (FP): These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class buys computer D no for which the classifier predicted buys computer D yes). Let FP be the number of false positives.
- False negatives (FN): These are the positive tuples that were mislabeled as negative (e.g., tuples of class buys computer D yes for which the classifier predicted buys computer D no). Let FN be the number of false negatives.
- These terms are summarized in the **confusion matrix**

# Metrics for Evaluating Classifier Performance

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

Confusion matrix, shown with totals for positive and negative tuples.

| *Classes* | *buys_computer = yes* | *buys_computer = no* | *Total* | *Recognition (%)* |
|---|---|---|---|---|
| *buys_computer = yes* | **6954** | **46** | 7000 | 99.34 |
| *buys_computer = no* | **412** | **2588** | 3000 | 86.27 |
| Total | 7366 | 2634 | 10,000 | 95.42 |

# Metrics for Evaluating Classifier Performance

- **The accuracy** of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. That is

$$accuracy = \frac{TP + TN}{P + N}.$$

- The error rate or misclassification rate of a classifier, M, which is simply 1- accuracy, where accuracy is the accuracy of M. This also can be computed as,

$$error\ rate = \frac{FP + FN}{P + N}.$$

# Metrics for Evaluating Classifier Performance

- consider the class imbalance problem, where the main class of interest is rare.
- That is, the data set distribution reflects a significant majority of the negative class and a minority positive class.
- For example, in fraud detection applications, the class of interest (or positive class) is “fraud,” which occurs much less frequently than the negative “nonfraudulant” class.

# Metrics for Evaluating Classifier Performance

- The sensitivity and specificity measures can be used, respectively, for this purpose.
- Sensitivity is also referred to as the true positive (recognition) rate (i.e., the proportion of positive tuples that are correctly identified), while specificity is the true negative rate (i.e., the proportion of negative tuples that are correctly identified).

$$sensitivity = \frac{TP}{P}$$

$$specificity = \frac{TN}{N}.$$

# Metrics for Evaluating Classifier Performance

- Precision can be thought of as a measure of exactness (i.e., what percentage of tuples labeled as positive are actually such),
- Recall is a measure of completeness (what percentage of positive tuples are labeled as such).

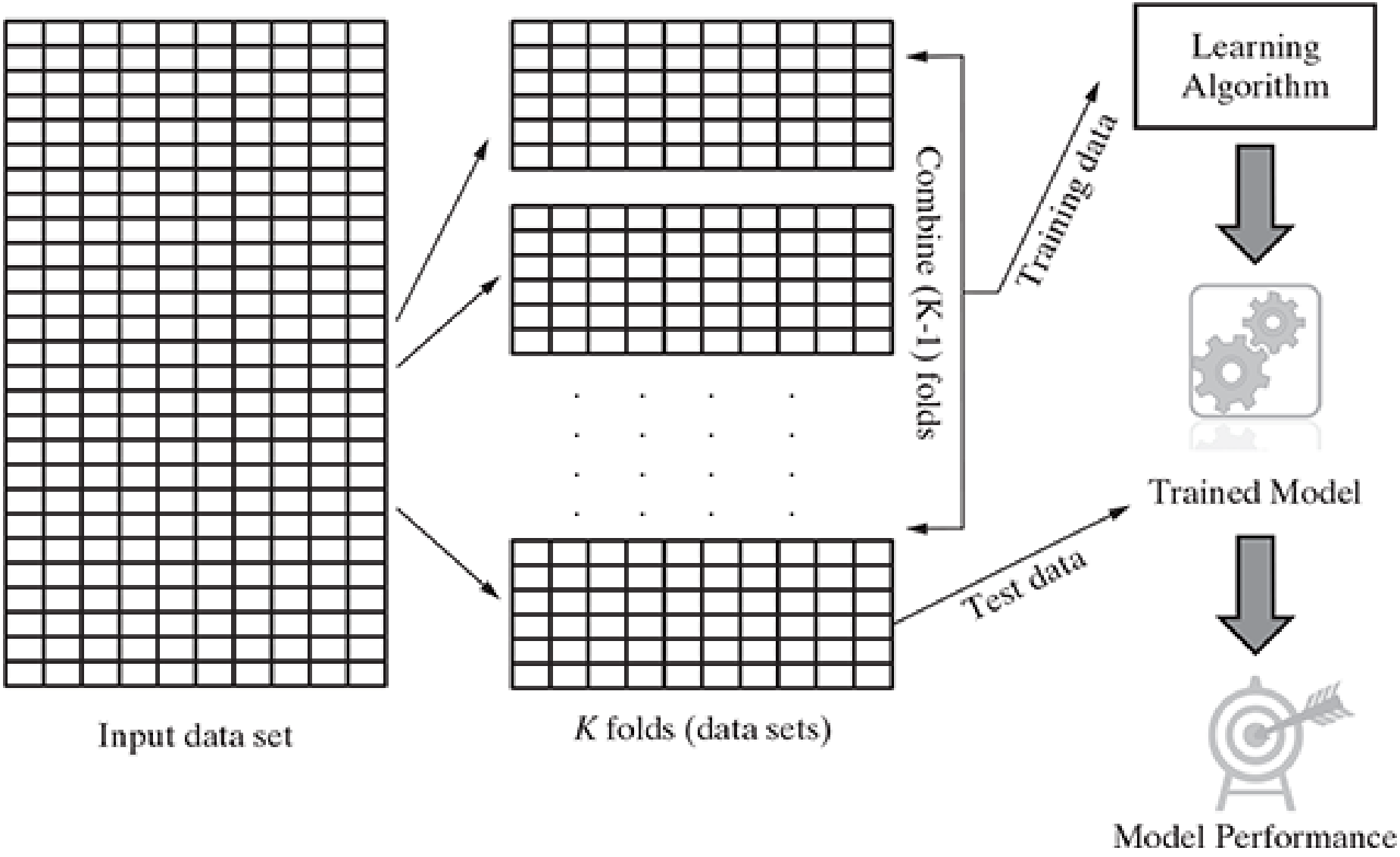
$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}.$$

# Cross-Validation

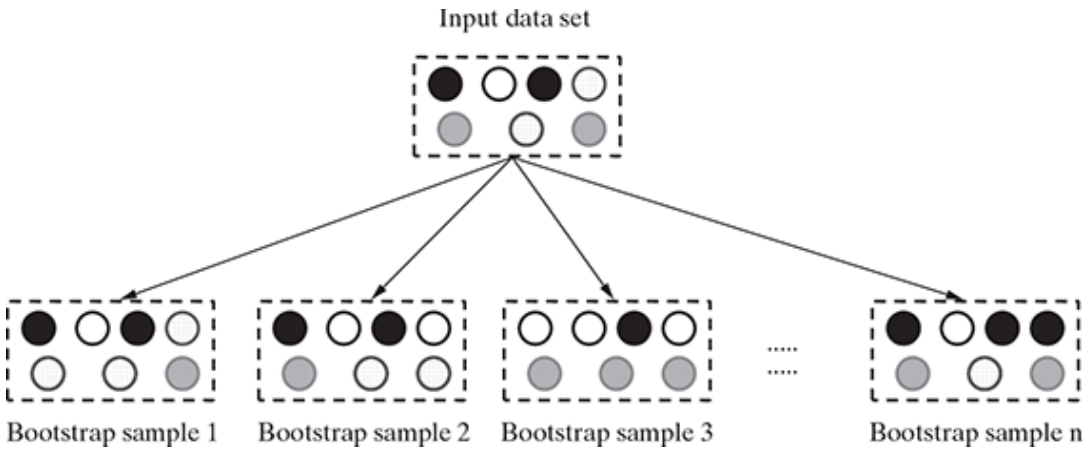
- In k-fold cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or “folds,” D1, D2, … , Dk, each of approximately equal size.
- Training and testing is performed k times.
- In iteration i, partition Di is reserved as the test set, and the remaining partitions are collectively used to train the model.
- That is, in the first iteration, subsets D2, … , Dk collectively serve as the training set to obtain a first model, which is tested on D1; the second iteration is trained on subsets D1, D3, …. , Dk and tested on D2; and so on.

# Cross-Validation



# Bootstrap

- The **bootstrap** method samples the given training tuples uniformly with replacement.
- That is, each time a tuple is selected, it is equally likely to be selected again and re-added to the training set.



# Ensemble method

- It is used for increasing classification accuracy.
- An ensemble for classification is a composite model, made up of a combination of classifiers.
- The individual classifiers vote, and a class label prediction is returned by the ensemble based on the collection of votes.
- Ensembles tend to be more accurate than their component classifiers.

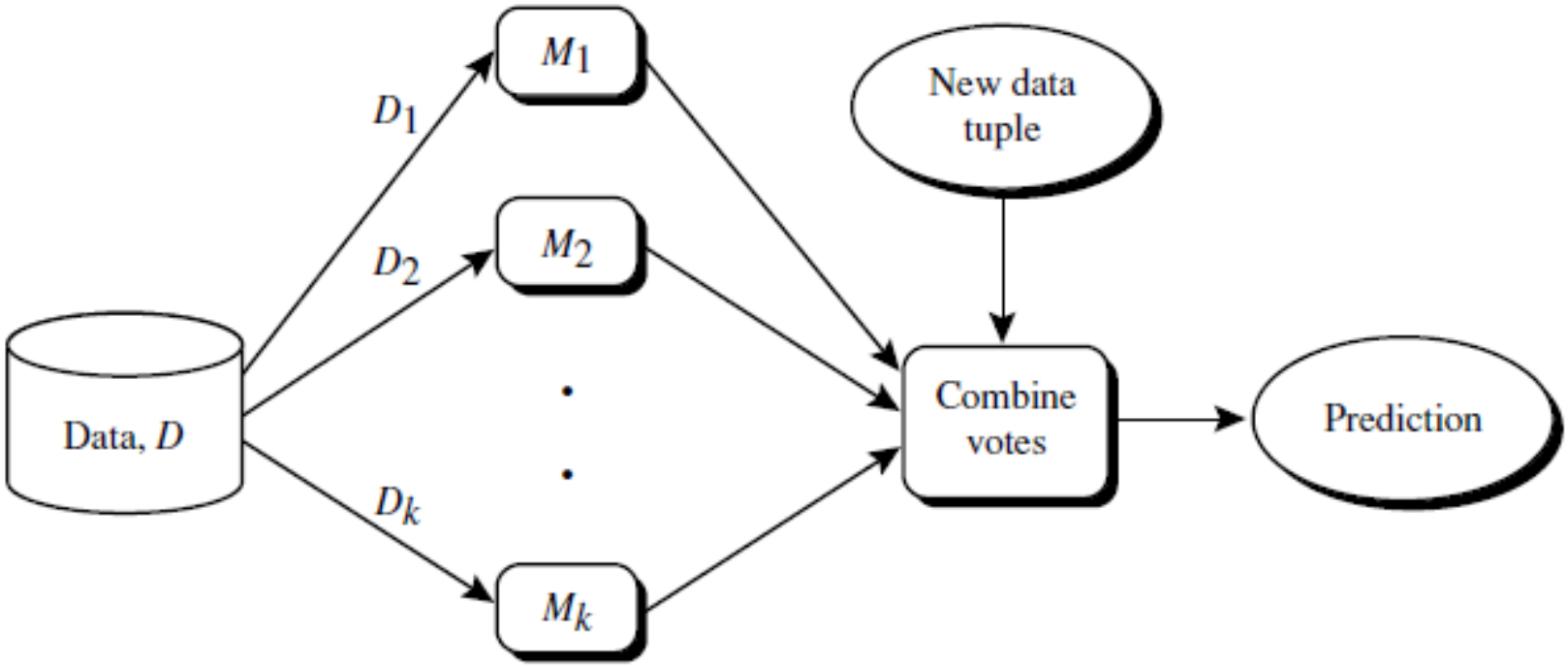
# Ensemble method

- Popular ensemble methods are:
  - ✓ Bagging,
  - ✓ Boosting, and
  - ✓ Random forests

# Ensemble method

- An ensemble combines a series of k learned models (or base classifiers),M1, M2, … , Mk, with the aim of creating an improved composite classification model, M*.
- A given data set, D, is used to create k training sets, D1, D2, . . . , Dk, where Di (1<= i <=k-1) is used to generate classifier Mi .
- Given a new data tuple to classify, the base classifiers each vote by returning a class prediction.
- The ensemble returns a class prediction based on the votes of the base classifiers.

# Ensemble method



# Random Forests

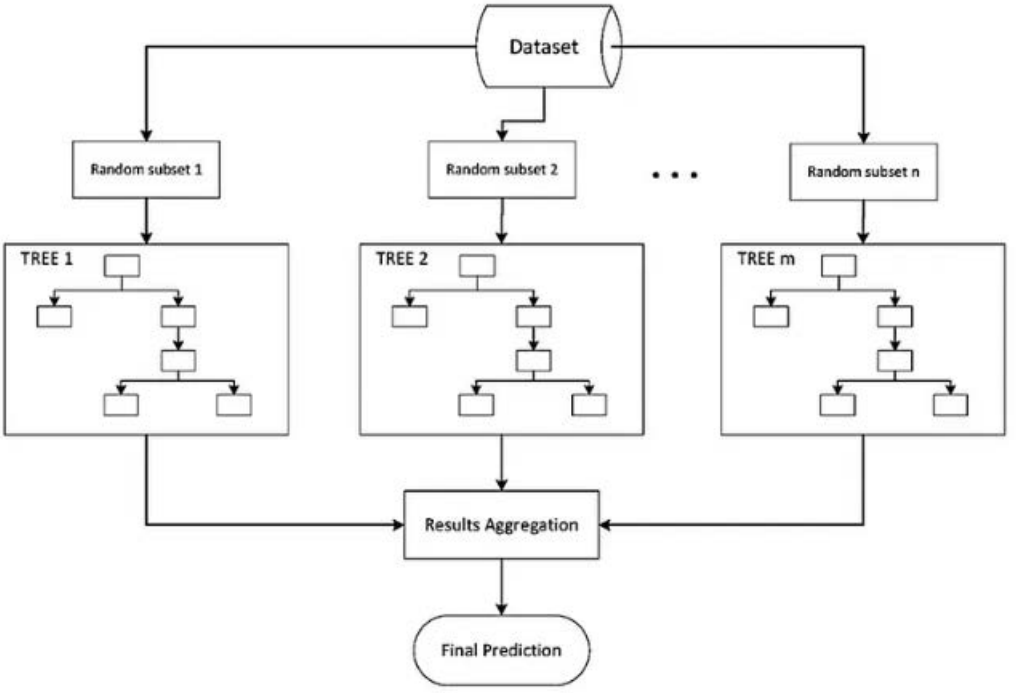
- Imagine that each of the classifiers in the ensemble is a *decision tree* classifier so that the collection of classifiers is a "forest."
- The individual decision trees are generated using a random selection of attributes at each node to determine the split.
- More formally, each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.
- During classification, each tree votes and the most popular class is returned.
- Random forests can be built using bagging.
- A training set, *D*, of *d* tuples is given.
- The general procedure to generate *k* decision trees for the ensemble is as follows.
- For each iteration, i (i = 1, 2,. . . , k), a training set, *Di* , of *d* tuples is sampled with replacement from *D*.
- Each Di is a bootstrap sample of D, so that some tuples may occur more than once in Di , while others may be excluded.

# Random Forests

- Let F be the number of attributes to be used to determine the split at each node, where F is much smaller than the number of available attributes.
- To construct a decision tree classifier, Mi , randomly select, at each node, F attributes as candidates for the split at the node.
- The CART methodology is used to grow the trees. The trees are grown to maximum size and are not pruned.
- Random forests formed this way, with random input selection, are called Forest-RI.

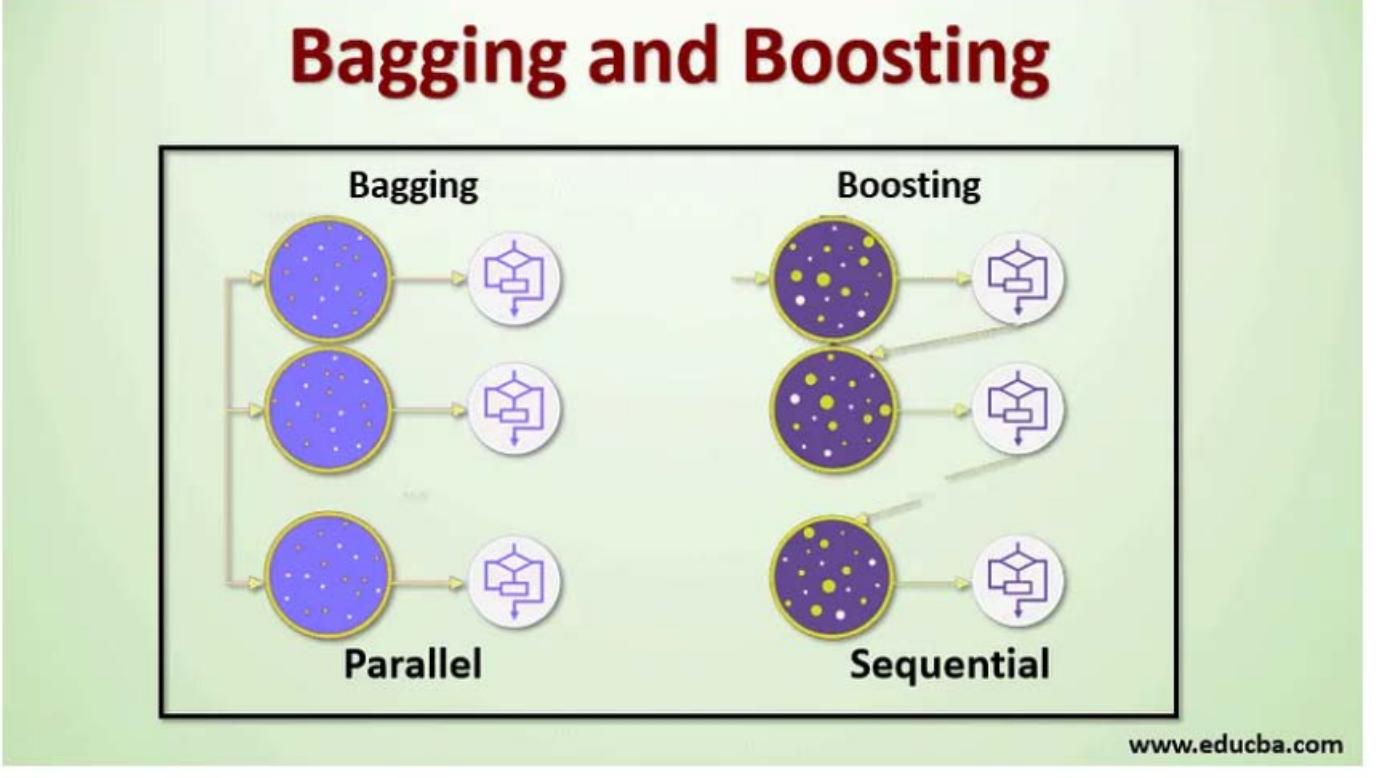
## Types of Ensemble Methods

1. *BAGG*ing, or *B*ootstrap *AGG*regating. BAGGing gets its name because it combines *B*ootstrapping and *Agg*regation to form one ensemble model. Given a sample of data, multiple bootstrapped subsamples are pulled. A Decision Tree is formed on each of the bootstrapped subsamples. After each subsample Decision Tree has been formed, an algorithm is used to aggregate over the Decision Trees to form the most efficient predictor. The image below will help explain:

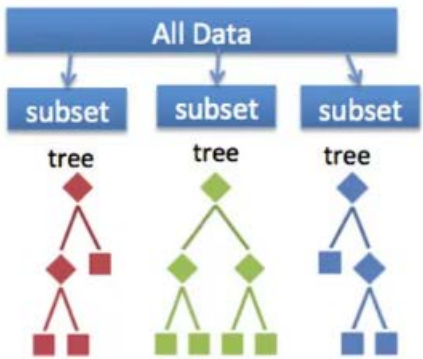



Given a Dataset, bootstrapped subsamples are pulled. A Decision Tree is formed on each bootstrapped sample. The results of each tree are aggregated to yield the strongest, most accurate predictor.

**Definition:** — The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners. Boosting is an ensemble method for improving the model predictions of any given learning algorithm. The idea of boosting is to train weak learners sequentially, each trying to correct its predecessor. The weak learners are sequentially corrected by their predecessors and, in the process, they are converted into strong learners.



2. **Random Forest** Models. Random Forest Models can be thought of as **BAGG**ing, with a slight tweak. When deciding where to split and how to make decisions, BAGGed Decision Trees have the full disposal of features to choose from. Therefore, although the bootstrapped samples may be slightly different, the data is largely going to break off at the same features throughout each model. In contrary, Random Forest models decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree will split based on different features. This level of differentiation provides a greater ensemble to aggregate over, ergo producing a more accurate predictor. Refer to the image for a better understanding.




A random forest takes a random subset of features from the data, and creates n random trees from each subset. Trees are aggregated together at end.