

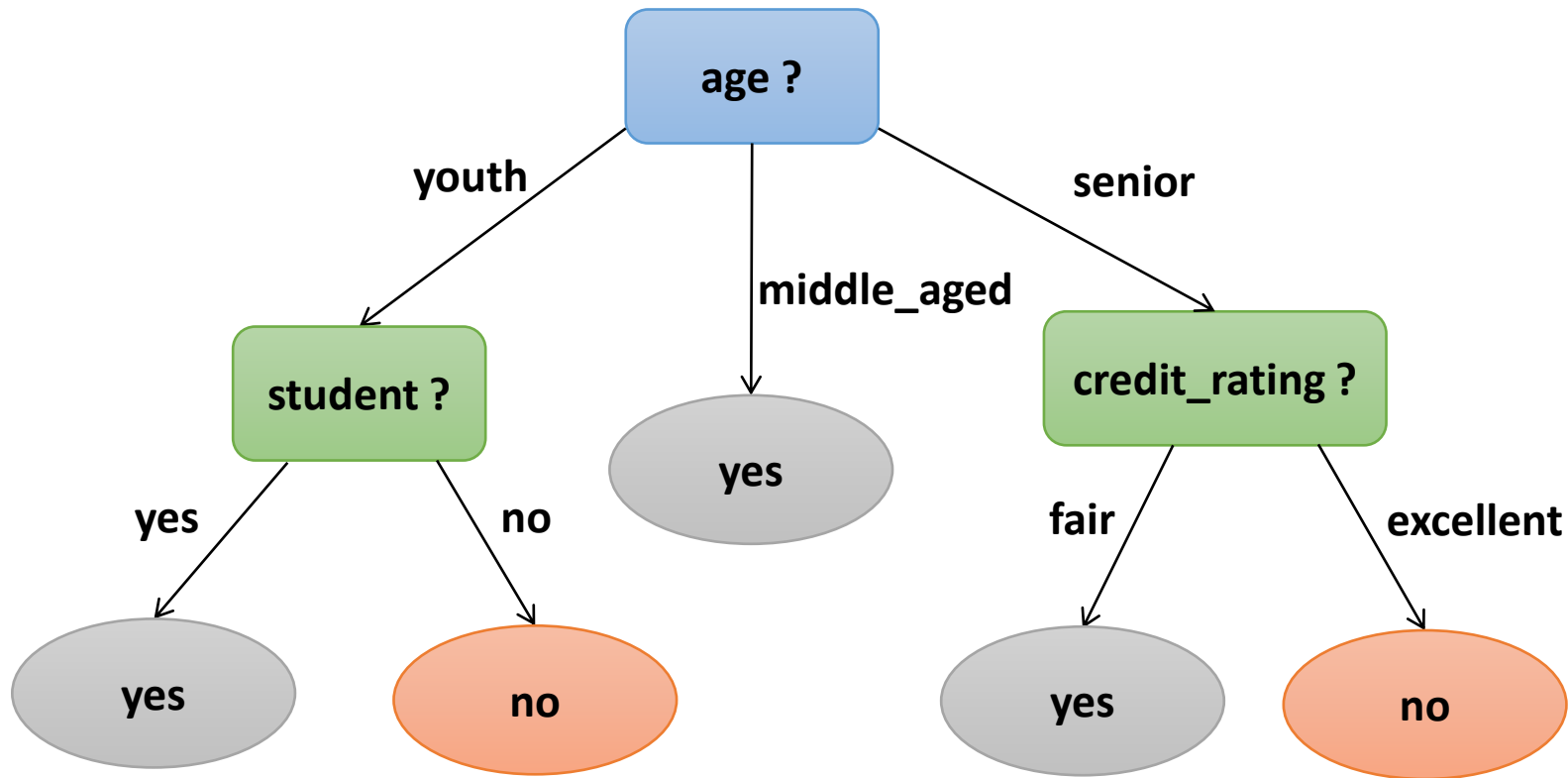
Decision Tree Induction

What is Decision Tree?

- Decision Tree is a type of data mining technique that is used to ***build Classification Models***. It builds classification models in the form of a tree-like structure, just like its name.
- A decision tree is a ***flowchart-like*** tree structure, where each internal node (non-leaf node) denotes a ***test on an attribute***, each branch represents an ***outcome*** of the test, and each leaf node (or terminal node) holds a ***class label***.
- The topmost node in a tree is the root node. It represents the ***best attribute*** selected for classification.

Example:

The following is a typical **decision tree**. It represents the concept **buys_computer**, that is, it **predicts** whether a customer is likely to purchase a computer.



In Decision tree,

- **Internal nodes** are denoted by **rectangles**,

- **leaf nodes** are denoted by **ovals**

According to the decision tree provided above, the tuple (**age** = **senior**, **credit_rating** = **excellent**) is associated to the class label (**buys_computer** = **yes**).

Decision Tree Induction

- In 1980, *J. Ross Quinlan*, a researcher, created the **ID3** (***Iterative Dichotomiser***) algorithm, which is a ***decision tree algorithm***. Later on, he introduced **C4.5**, which was the successor of **ID3**.
- In 1984, a group of authors published the book ***Classification and Regression Trees (CART)***, which described the generation of ***binary decision trees***.
- **ID3** and **CART** were invented independently of one another at around the same time, yet follow a ***similar approach*** for creating decision trees from training tuples(Dataset).

Decision Tree Algorithm:

Aim :

Generate a ***decision tree*** from the training data partition(**D**).

Input:

Data partition(**D**) - which is a set of training tuples and their associated class labels;

attribute list: - the set of candidate attributes;

Attribute selection measure - describes the method for selecting the best attribute for discrimination among tuples.

Output:

A decision tree.

Decision Tree Algorithm:

Method :

Step-1: Begin the tree with the ***root node***, says ***R***, which contains the complete dataset ***D***.

Step-2: Find the ***best attribute*** in the dataset using *Attribute Selection Method (ASM)*.

Step-3: Divide the ***dataset(D)*** into subsets that contains possible values for the best attribute.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until reaches leaf nodes.

Example: Suppose there is a person who has a ***job offer*** and wants to decide whether he should ***accept or decline*** the offer. Solve this problem, using the decision tree.

Input:

attribute_list:

Salary, Distance and Cab_facility.

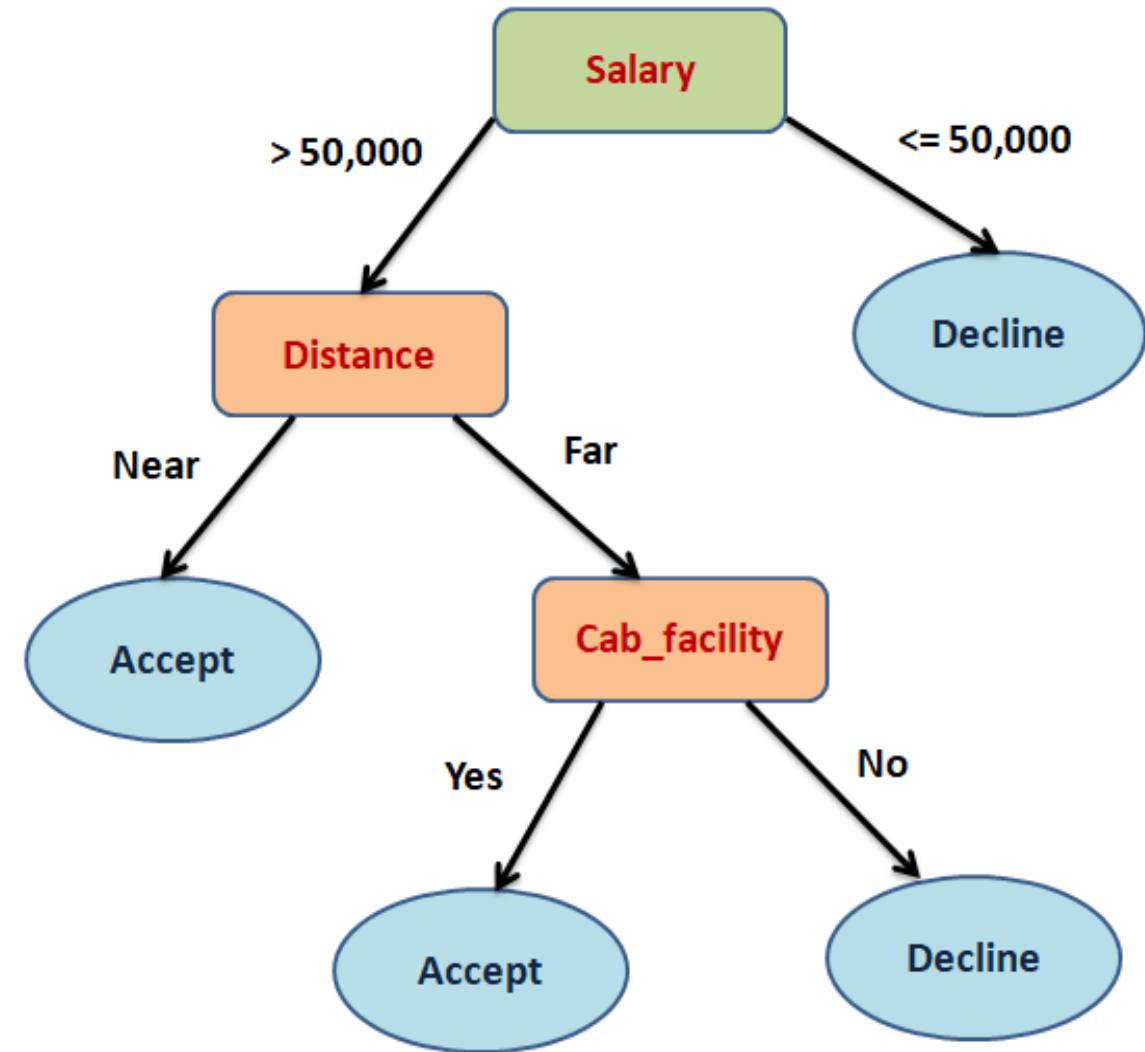
class_labels:

Accept, Decline.

Output:

Method:

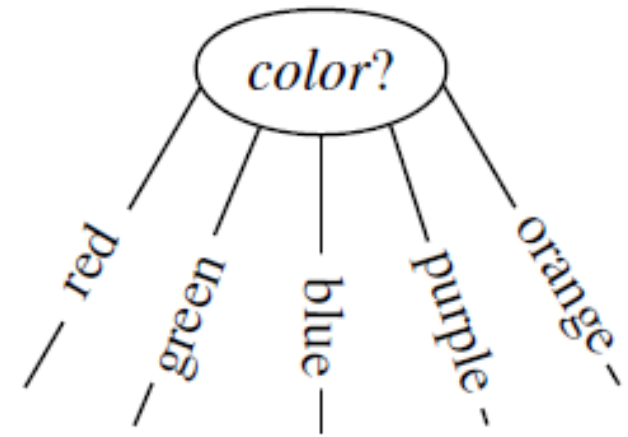
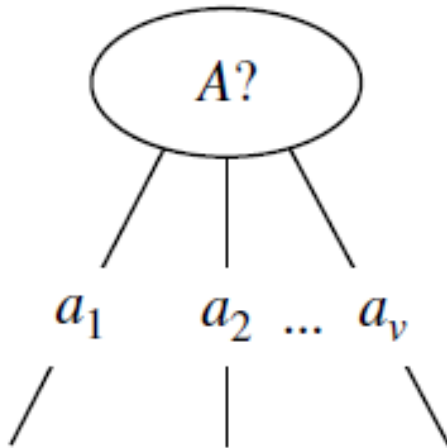
- The decision tree starts with the root node (**Salary**).
- The root node splits further into the next decision node (**Distance** attribute) and one leaf node(**Decline**).
- The next decision node further gets split into one decision node (**Cab_facility** attribute) and one **Accept** .
- Finally, the decision node(**Cab_facility** attribute) splits into two leaf nodes (**Accept** and **Decline**).



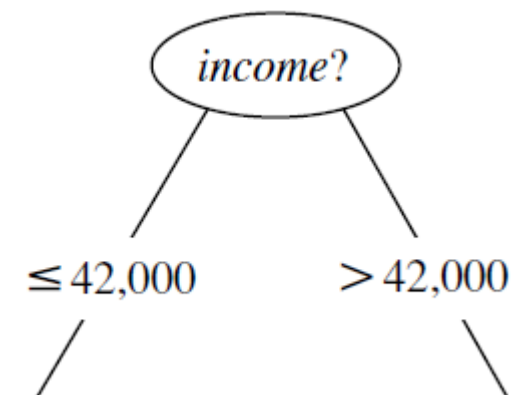
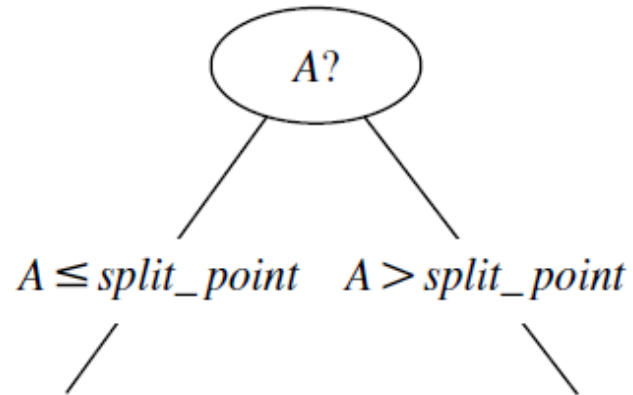
Splitting criterion: The splitting criterion tells us the “**best**” way to separate or partition the tuples in dataset (**D**) into individual classes.

Let **A** be the *splitting attribute*,

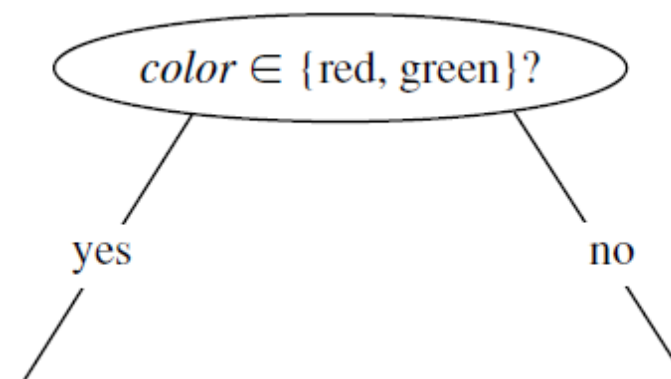
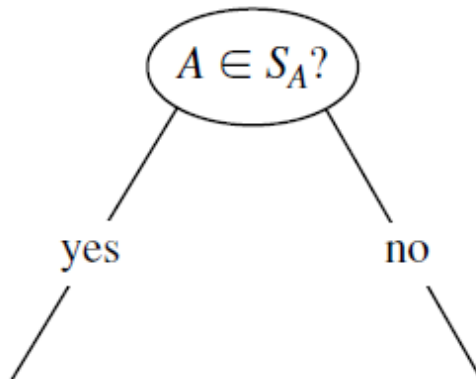
(a) If A is *discrete-valued*, then one branch is grown for each known value of A.



(b) If A is **continuous-valued**, then two branches are grown, corresponding to $A \leq \textit{split_point}$ and $A > \textit{split_point}$.



(c) If A is **discrete-valued** and a **binary tree** must be produced, then the test is of the form $A \in S$, where S is the splitting subset for A .



Attribute Selection Measures:

- When implementing a Decision tree, the primary challenge is to choose the ***most suitable attribute*** for the root node and sub-nodes.
- However, there is a technique known as ***Attribute selection measure*** that can help resolve such issues.
- The following are the most common ***Attribute selection measures***,
 - ***Entropy***
 - ***Information Gain***

Attribute Selection Measures:

Entropy:

- It is the measure of *impurity (or) uncertainty* in the data. It lies between **0** to **1** and is calculated using the below formula.

$$Entropy(D) = \sum_{i=1}^n -p_i * \log_2(p_i)$$

- A *lower entropy* indicates a ***better model***, as it implies that the classes are split more effectively due to *reduced uncertainty*.

For example, let us see the formula for a use case with **2 classes**- **Yes** and **No**.

$$Entropy(D) = -P(Yes)*log_2(P(Yes)) - P(No)*log_2(P(No))$$

Here,

Yes and **No** are two classes ,

D is the total number of samples,

P(Yes) is the probability of Yes,

P(No) is the probability of No.

If there are **many classes** then the formula is

$$Entropy(D) = -P_1*log_2(P_1) - P_2*log_2(P_2) - - P_n*log_2(P_n)$$

Here **n** is the number of classes.

Attribute Selection Measures:

Information Gain:

- This method is the main method that is used to ***build decision trees***.
- It reduces the ***number of tests*** that are needed to classify the given tuple.
- The attribute with the ***highest information gain*** is ***selected***.
- Information gain is basically measuring the ***change in entropy***.

- Information gain is defined as the ***difference*** between the *original information requirement* and the *new requirement* (i.e., obtained after portioning on attribute **A**)

$$Gain(A) = Entropy(D) - Entropy_A(D)$$

- The attribute **A** with the highest information gain, $Gain(A)$, is chosen as the splitting attribute at node N.

Example:

Let us illustrate the *ID3 algorithm* with a simple example of classifying whether customer *buys a computer* based on some conditions.

(or)

Construct a *decision tree* to classify “*buys_computer*.”

Consider the following dataset:

age	income	student	credit_rating	buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Table-1: customer dataset

- In the given customer dataset, we have 5 attributes/columns, those are *age, income, student, credit_rating*, and *buys_computer*.
- In which *age, income, student, credit_rating* attributes are categorized as **input features** and where as *buys_computer* categorized as **output/target feature**.
- Here, we are going to classify whether customer *buys a computer* or *not* based on the *input features*. And we will be going to use *ID3 algorithm* for building *decision tree*.

Solution:

- To create a *decision tree*, First we need to select **best attribute** for *root* node.
- To select best attribute , we will use Attribute Selection Measures, those are *Entropy* and *Information Gain* measures.
- First, we need to **count** the number of “yes” and “no” for output *column / feature* called “*buys_computer.*”
- There are “no” = 5 and “yes” = 9 (i.e., Total number of tuples = 14)

First, we will calculate **entropy** for **output/target** variable.

The formula for **entropy** is

$$Entropy(D) = \sum_{i=1}^n -p_i * \log_2(p_i)$$

The given dataset with **2 classes- yes** and **no**. Then the formula of **Entropy** for **buys_computer** is

$$Entropy(\mathbf{buys_computer}) = -P(\mathbf{yes}) * \log_2(P(\mathbf{yes})) - P(\mathbf{no}) * \log_2(P(\mathbf{no}))$$

$$\begin{aligned} P(\mathbf{yes}) &= p(\mathbf{yes}) / (p(\mathbf{yes}) + p(\mathbf{no})) \\ &= 9 / (9 + 5) \\ &= 9/14 \end{aligned}$$

for “**yes**” component of **buys_computer**.

$$\begin{aligned} P(\mathbf{no}) &= p(\mathbf{no}) / (p(\mathbf{yes}) + p(\mathbf{no})) \\ &= 5 / (9 + 5) \\ &= 5/14 \end{aligned}$$

for “**no**” component of **buys_computer**.

Let us substitute the values in the ***entropy formula***,

$$\begin{aligned}\text{Entropy}(\textit{buys_computer}) &= - (9/14) * \log_2(9/14) - (5/14) * \log_2(5/14) \\ &= - (0.64) * \log_2(0.64) - (0.36) * \log_2(0.36) \\ &= 0.412 + 0.53 \\ &= \mathbf{0.94} \text{ bits.}\end{aligned}$$

$$\text{Entropy}(\textit{buys_computer}) = \mathbf{0.94} \text{ bits}$$

Once we are ready with the *Entropy of target variable*, we will find out the Entropy for ***each column/variable/attribute*** with respect to ***target variable***.

Let's start with the ***age column***. Extract the ***age column*** from given dataset

		buys_computer		
		yes	no	
age	youth	2	3	5
	middle_aged	4	0	4
	senior	3	2	5
				14

So, here we can see that

- *youth* has **2 yes**, and **3 no**
- *middle_aged* has **4 yes** and **0 no**
- *senior* has **3 yes** and **2 no**.

Now we can calculate the **Entropy** for **each value** of **age** column.

$$\begin{aligned} \text{Entropy}(\text{youth}) &= - (2/5) * \log_2(2/5) - (3/5) * \log_2(3/5) \\ &= \mathbf{0.97} \end{aligned}$$

$$\begin{aligned} \text{Entropy}(\text{senior}) &= - (3/5) * \log_2(3/5) - (2/5) * \log_2(2/5) \\ &= \mathbf{0.97} \end{aligned}$$

$$\begin{aligned} \text{Entropy}(\text{middle_aged}) &= - (4/4) * \log_2(4/4) - (0/4) * \log_2(0/4) \\ &= \mathbf{0.0} \end{aligned}$$

The Entropy of *middle_aged* is **0** i.e., *middle_aged* has only “**yes**” component.

So, once we are ready with Entropy for each value in **age** column then we will find out the *Entropy* for complete **age** column with respect to **buys_computer** column.

Entropy (buys_computer, age) =

$$\begin{aligned} &P(\text{youth}) * E(\text{youth}) + P(\text{middle_aged}) * E(\text{middle_aged}) + P(\text{senior}) * E(\text{senior}) \\ &= (5/14) * 0.97 + (4/14) * 0 + (5/14) * 0.97 \\ &= \mathbf{0.693 \text{ bits}} \end{aligned}$$

Entropy (buys_computer, age) = 0.693 bits

Now, we need to calculate the *information gain* of column **age** is

Information_Gain (buys_computer, age) =

Entropy(buys_computer) - Entropy (buys_computer, age)

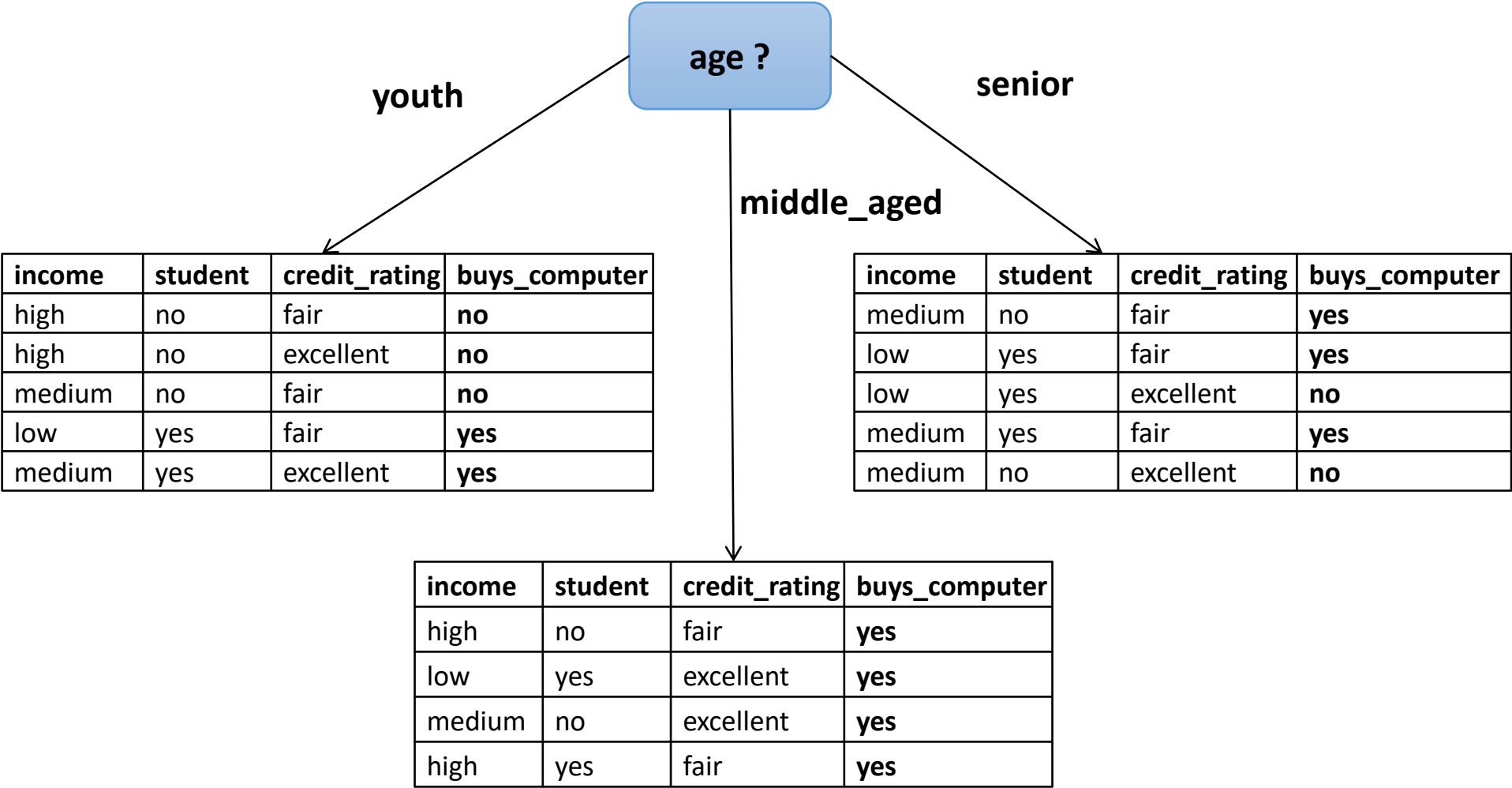
= 0.94 – 0.693

*= **0.247** bits*

Information Gain (buys_computer, age) = 0.247 bits

Similarly, we can compute *Information Gain (buys_computer, income)* is **0.029** bits, *Information Gain (buys_computer, student)* is **0.151** bits, and *Information Gain (buys_computer, credit_rating)* is **0.048** bits.

The attribute ***age*** has the ***highest information gain*** and therefore becomes the splitting attribute at the **root node** of the decision tree. Branches are grown for each outcome of age. The tuples are shown partitioned accordingly.



Notice that the tuples falling into the partition for **age = middle_aged** all belong to the **same class**. Because they all belong to class “yes,” a leaf should therefore be created at the end of this branch and labelled “yes.”

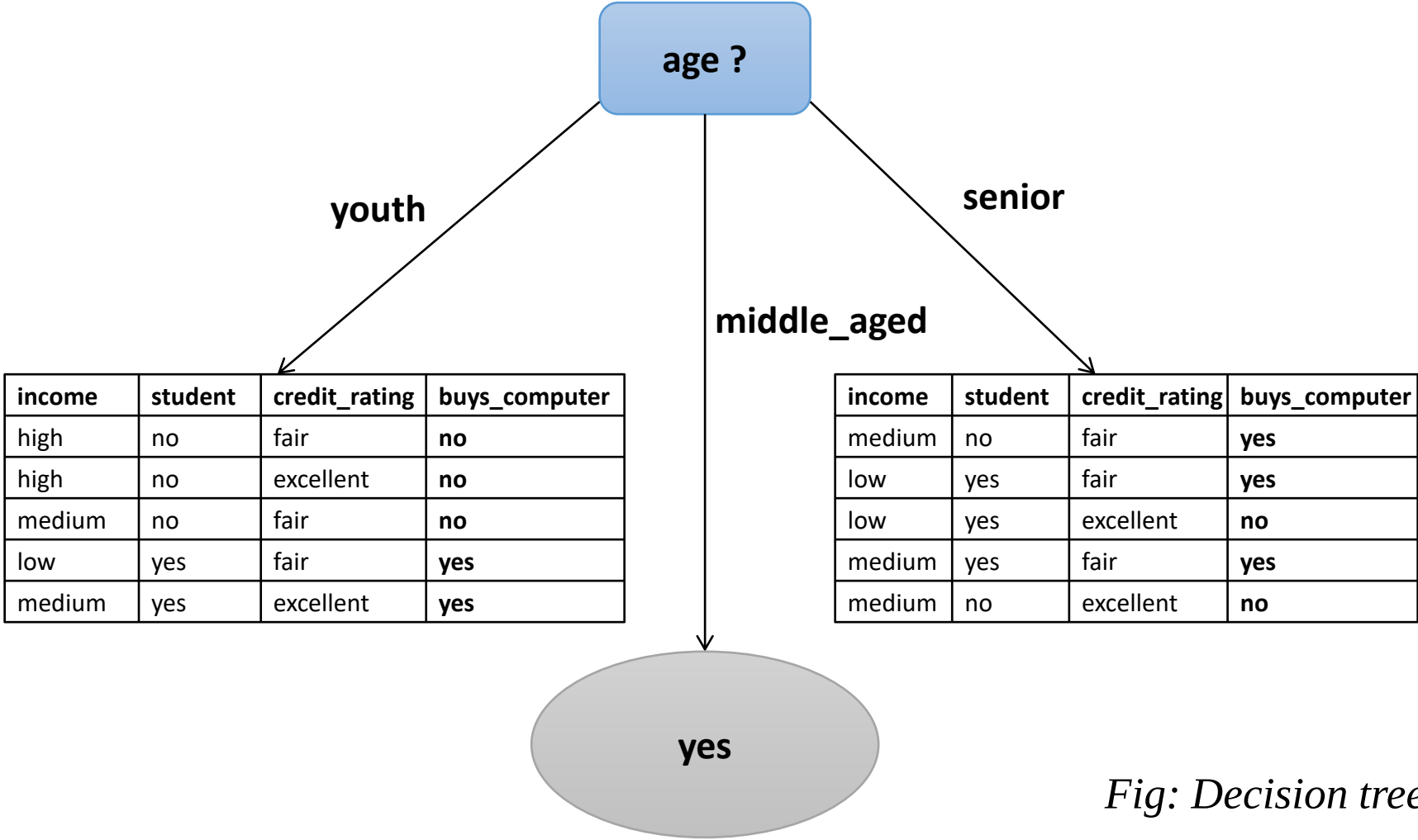


Fig: Decision tree for **age** attribute

Let's continue with the *next level*. If the age is “youth” then the dataset is

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

- In the above, *income*, *student* and *credit_rating* are categorized as input attributes and *buys_computer* is categorized as output attribute. There are 5 instances among those **2 yes** and **3 no** for *buys_computer* attribute.
- Now, we will calculate the *Information Gain* for all the attributes with respect to target/class attribute called *buys_computer*. To calculate *Information Gain* first, we need to calculate the *Entropy* for all the attributes.

Let's calculate the **Entropy** for class/target attribute **buys_computer** is

$$\text{Entropy}(\text{buys_computer}) = -P(\text{yes}) * \log_2(P(\text{yes})) - P(\text{no}) * \log_2(P(\text{no}))$$

$$= - (2/5) * \log_2(2/5) - (3/5) * \log_2(3/5)$$

$$= \mathbf{0.97} \text{ bits}$$

Once we are ready with the *Entropy* of target variable, we will find out the Entropy for each **column/variable/attribute** with respect to target variable to find most homogeneous sample.

Let's extract the **income** attribute from given dataset.

So, here we can see that

- low has **1 yes, 0 no**
- medium has **1 yes** and **1 no**
- high has **0 yes** and **2 no**.

		buys_computer		
		yes	no	
income	low	1	0	1
	medium	1	1	2
	high	0	2	2
				5

Now we can calculate the Entropy for each value of income column.

$$\text{Entropy (low)} = - (1/1) * \log_2(1/1) - (0/1) * \log_2(0/1) = 0$$

$$\text{Entropy (medium)} = - (1/2) * \log_2(1/2) - (1/2) * \log_2(1/2) = 1$$

$$\text{Entropy (high)} = - (0/2) * \log_2(0/2) - (2/2) * \log_2(2/2) = 0$$

So, once we are ready with **Entropy** for each value in **income** column then we will find out the **Entropy** for complete income column with respect to **buys_computer** column.

Entropy (buys_computer, income) =

$$P(\text{low}) * E(\text{low}) + P(\text{medium}) * E(\text{medium}) + P(\text{high}) * E(\text{high})$$

$$= (1/5) * 0 + (2/5) * 1 + (1/5) * 0$$

$$= \mathbf{0.4 \text{ bits.}}$$

$$\text{Entropy (buys_computer, income)} = \mathbf{0.4 \text{ bits}}$$

Now, we need to calculate the **information gain** of column **income** is

Information Gain (buys_computer, income) =

Entropy (buys_computer) - Entropy (buys_computer, income)

= 0.97 – 0.4

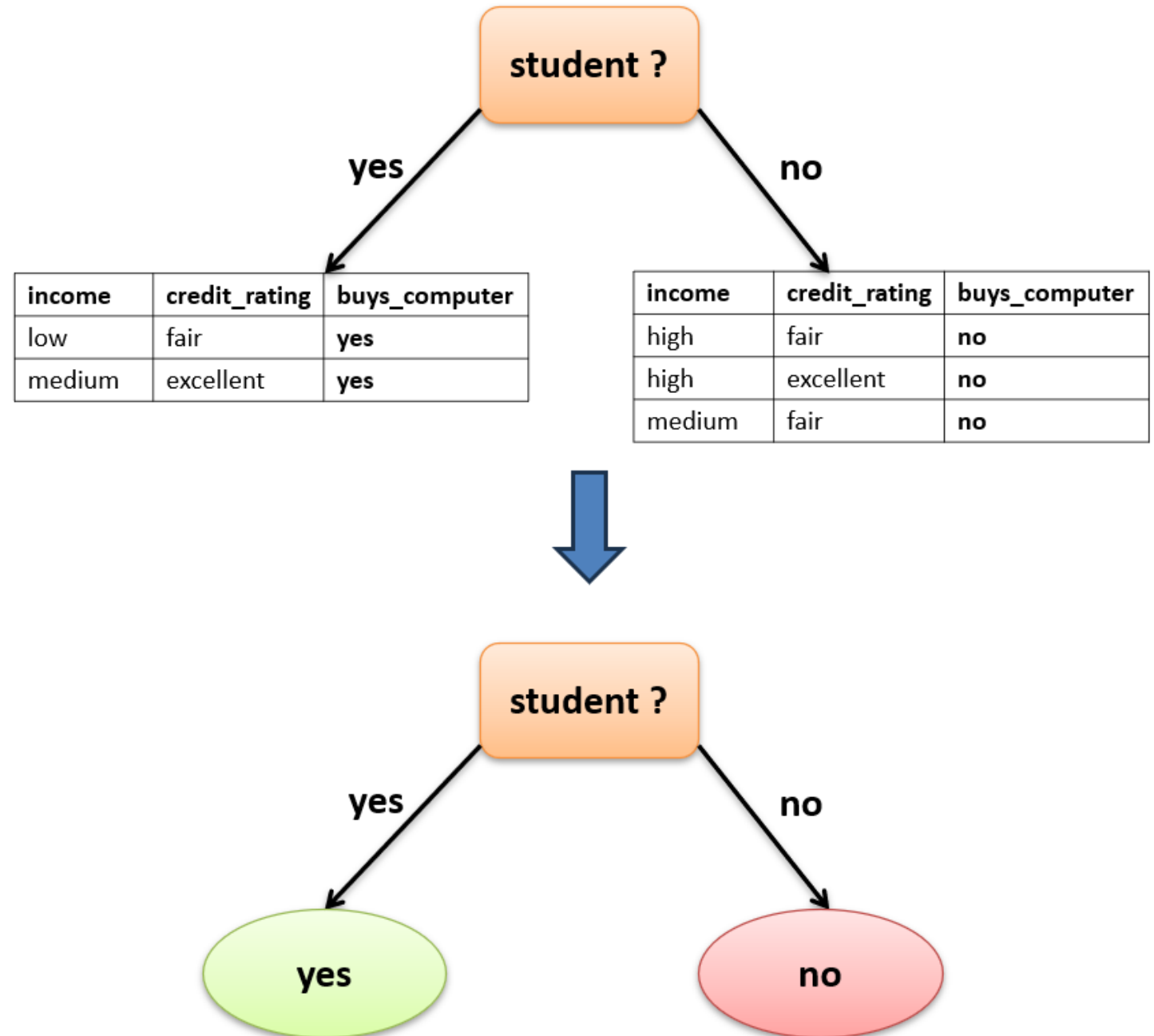
= **0.57** bits

Information Gain (buys_computer, income) = 0.57 bits

Similarly, we can compute ***Information Gain*** (buys_computer, student) = **0.97** bits

and ***Information Gain*** (buys_computer, credit_rating) = **0.024** bits.

The attribute ***student*** has the ***highest information*** gain and therefore becomes the splitting attribute at the *next level node of the decision tree*.

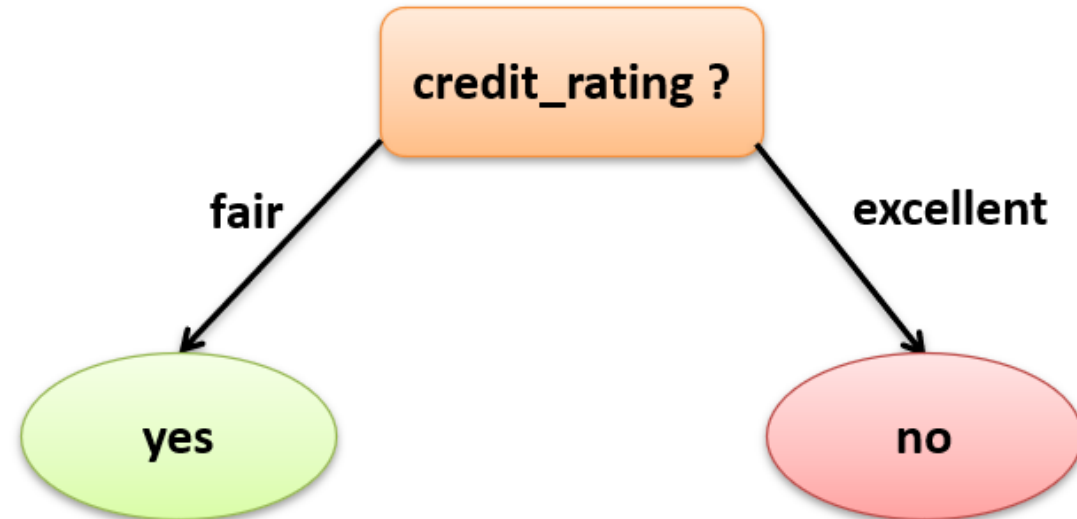
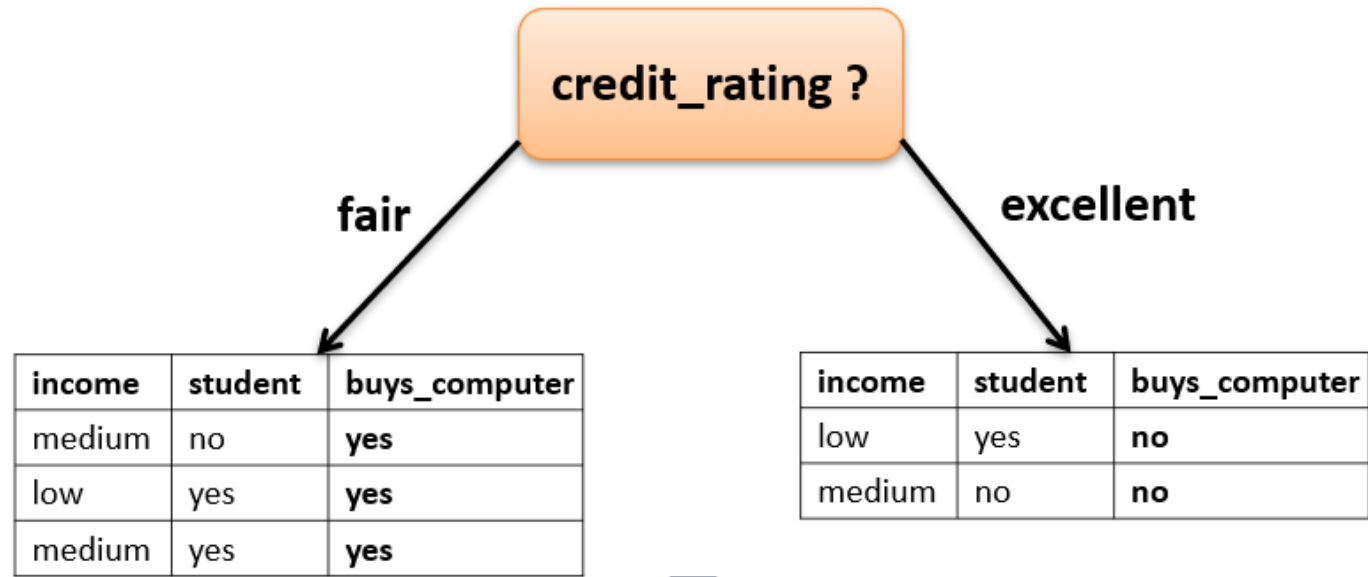


Branches are grown for each outcome of ***student***. The tuples are shown partitioned accordingly.

Let's continue the same process for the age is “**senior**”.

If you continue the above process then we will get the attribute ***credit_rating*** has the ***highest information gain*** and therefore becomes the splitting attribute at the next level node of the decision tree.

Branches are grown for each outcome of ***credit_rating***. The tuples are shown partitioned accordingly.



The final ***decision tree*** for all the attributes in the above dataset using ***ID3 algorithm***.

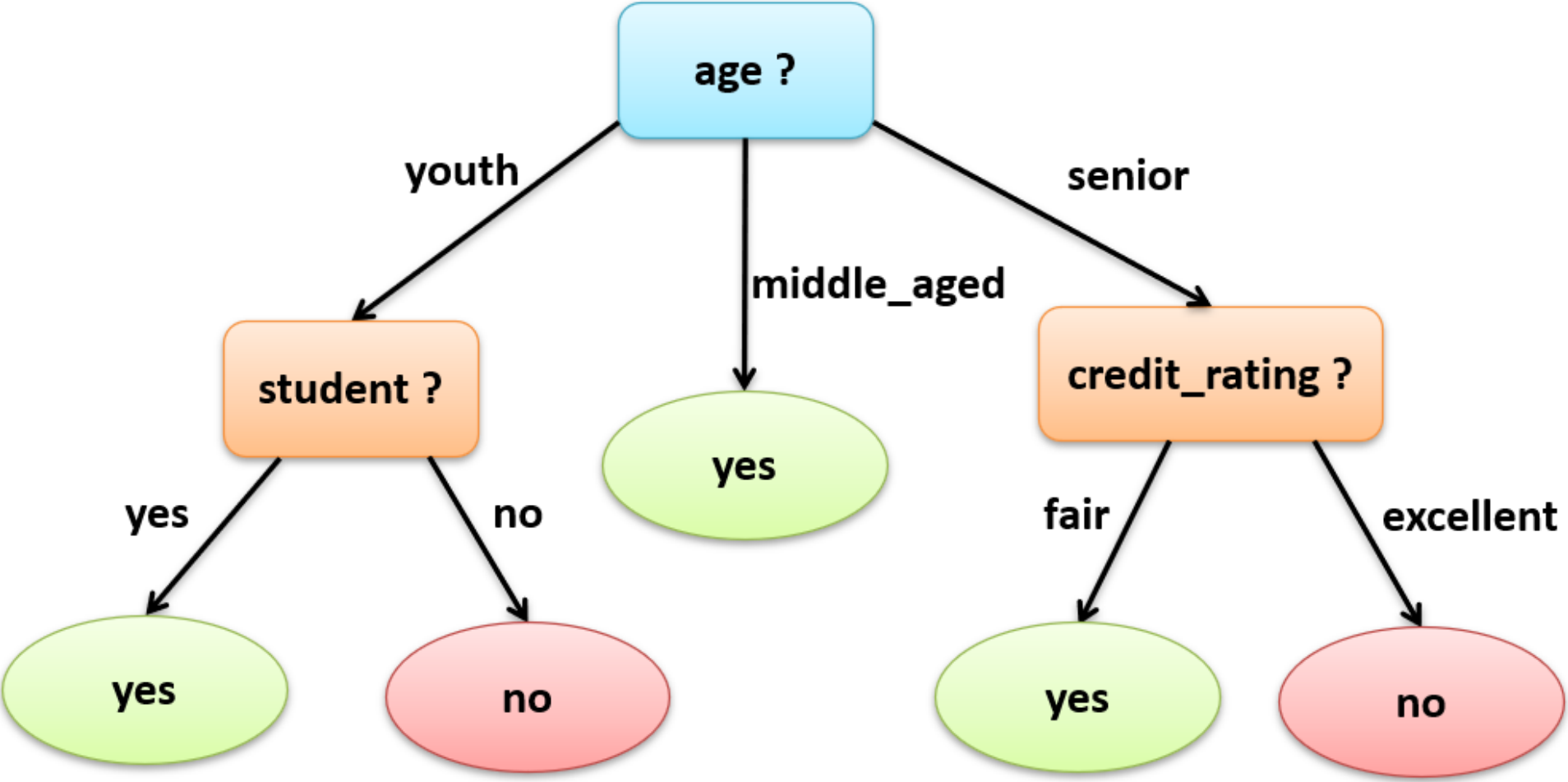


Fig: Decision tree

- Let us illustrate the **ID3 algorithm** with a simple example of classifying ***golf play*** based on weather conditions. (or) Construct a ***decision tree*** to classify “***golf play***.” Consider the following dataset:

Outlook	Temperature	Humidity	Wind	Golf_Play
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No