CHAPTER III : DECISION TREES

In this chapter we present decision trees, a model widely used in Data Mining.

3.1 INTRODUCTION

For certain fields of application, it is essential to produce classification procedures that the user can understand. This is particularly the case for medical diagnosis, where the doctor needs to be able to interpret the reasons for the diagnosis. Decision trees meet this requirement because they graphically represent a set of rules and are easy to interpret.

3.2 CONCEPT OF DECISION TREES

3.2.1 Aim

A decision tree models a hierarchy of tests on the values of a set of variables called *attributes*. At the end of these tests, the model (the decision tree) produces a numerical value or selects an element from a discrete set of conclusions. The former is known as *regression* and the latter as *classification*. For example, the following decision tree (figure 3.1) models a problem where we wish to classify individuals into two classes {sick, healthy} according to the values taken by two descriptors: "temperature" and "sore throat".

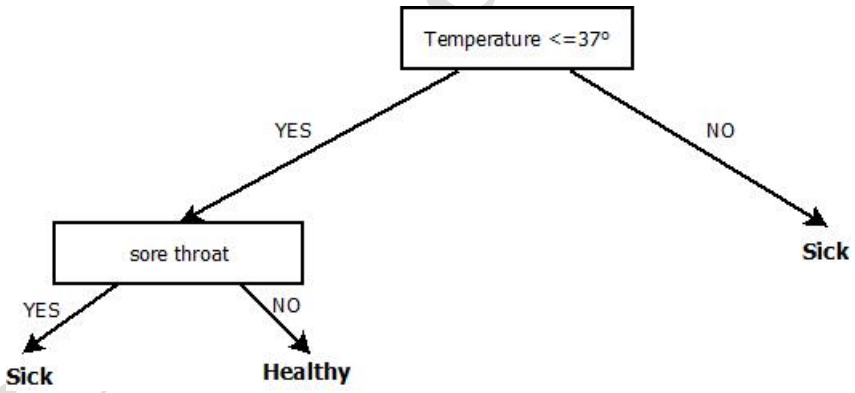

Fig 3.1. Example of a classification decision tree.

You can also find on the website (https://www.medg.fr/informations/arbres-decisionnels-medicaux/) several decision trees used in the medical field.

The objective of the following regression-type decision tree (figure 3.2) is to estimate the price of a vehicle as a function of the two attributes "Fuel type" and "Power".

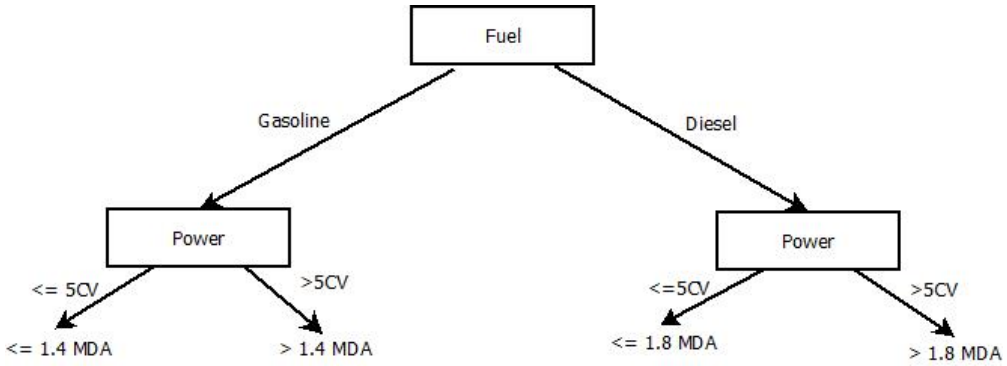

Fig 3.1. Example of a regression decision tree.

The internal *nodes* of a decision tree are called decision nodes. These nodes are labelled with a *test* that can be applied to any description of an individual in the population. In general, each test examines the value of a single attribute in the description space. The possible answers to the test correspond to the labels of the arcs originating from this node. In the case of binary decision nodes, the labels of the arcs are omitted and, by convention, the left arc corresponds to a positive response to the test. Leaves are labelled by a *class*.

A decision tree is the graphical representation of a classification procedure. Each complete description is associated with a single leaf of the decision tree. This association is defined by starting at the root of the tree and moving down the tree according to the responses to the tests that label the internal nodes. The associated class is then the default class associated with the leaf that corresponds to the description. The classification procedure obtained is immediately translated into decision rules. The rule systems obtained are special in that the order in which the attributes are examined is fixed and the decision rules are mutually exclusive.

3.2.2 Translating a decision tree into rules

A decision tree can be interpreted as a series of rules. For example, a patient with a temperature of 39 and a non-irritated throat will be classified as "sick" by the tree in the previous example. The translation of this tree into decision rules is shown in figure 3.3.

```
If  (Temperature<=37)
        then    if (sore throat)
                        then Class= « sick »
                        else
                                Class= « healthy »

        else
                Class= « sick »
```

Fig 3.3. Translating a decision tree into rules

3.2.3 Notations used with decision trees

The following is an introduction to some of the notations used with decision trees.

Position of a node: The nodes of a tree are identified by positions, which are numbers where the level of the node and its position (from the left) are concatenated (figure 3.4). The root is noted Ø. For example, position 11 refers to the node at level 1, and is the first position from the left.

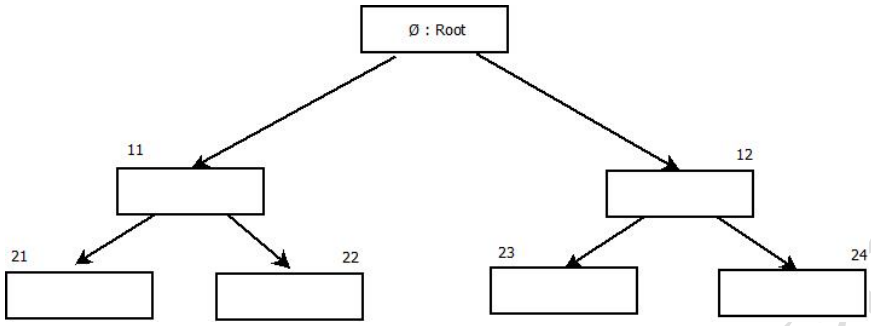

Fig 3.4. Positions in a decision tree

Given a sample S, a set of classes {1,...,*c*} and a decision tree *t*, at each position *p* of *t* corresponds a subset of the sample which is the set of examples which satisfy the tests from the root up to that position. Consequently, for any position *p* of *t*, we can define the following quantities:

N(p) is the cardinal of the set of examples associated with *p,*

N(k/p) is the cardinal of the set of examples associated with p that belong to class *k,*

P(k/p) = N(k/p)/N(p) the proportion of elements of class *k* at position *p.*

Example*: Consider the decision tree from the previous example. We also have a sample of 200 patients. We know that 100 are sick and 100 are healthy. The distribution between the two classes M (for sick) and S (for healthy) is given by :*

	Sore throat	*Non sore throat*
temperature <= 37	*(0 Healthy, 38 Sick)*	*(100 Healthy, 0 Sick)*

We return to the tree, adding the associated examples to each node (figure 3.4).

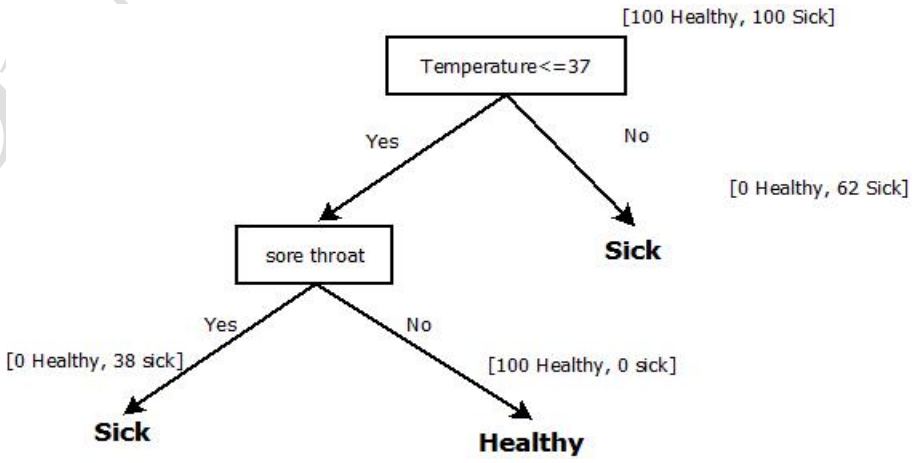

Fig 3.4. Decision tree with examples associated with each node

Here is the calculation of the different cardinal values at the root.
We then have : *N(Ø)=200; N(H/Ø)=100; N(S/Ø)=100; P(H/Ø)=100/200 and P(S/Ø)=100/200.*

3.2.4 Concept of Entropy

At any position *pos* in a decision tree, we can associate a quantity i(*pos*) which represents the degree of mixing of classes at position *pos*. The higher i(*pos*) is, the greater the mixing of classes will be. The function i should reach its maximum when the examples are equally distributed between the different classes and its minimum when one class contains all the examples (there is no mixing: the node is said to be pure).

Several functions have been proposed to measure class mixing: Shannon entropy, Ginni measure, etc. In the rest of this course, we will only use Shannon entropy, whose formula is :

$$Entropy(pos) = -\left[\sum_{k=1}^{C} P\left(\frac{k}{pos}\right).Log_2\left(\frac{k}{pos}\right)\right]$$

(equation 3.1)

This function takes its values in the interval [0, 1].

The next section explains how this notion of entropy is used to determine whether a node is terminal or not when constructing a decision tree.

Example: Calculation of the entropy at the root node (Ø) and node 11 of the medical decision tree (paragraph 3.1).

$$Entropy(\acute{Ø}) = -\left[\left(\frac{Healthy}{\acute{Ø}}\right).Log_2\left(\frac{Healthy}{\acute{Ø}}\right)+\left(\frac{Sick}{\acute{Ø}}\right).Log_2\left(\frac{Sick}{\acute{Ø}}\right)\right]$$

$$Entropy(\acute{Ø}) = -\left[\left(\frac{100}{200}\right).Log_2\left(\frac{100}{200}\right)+\left(\frac{100}{200}\right).Log_2\left(\frac{100}{200}\right)\right] = 1.00$$

$$Entropy(11) = -\left[\left(\frac{Healthy}{11}\right).Log_2\left(\frac{Healthy}{11}\right)+\left(\frac{Sick}{11}\right).Log_2\left(\frac{Sick}{11}\right)\right]$$

$$Entropy(\acute{Ø}) = -\left[\left(\frac{100}{138}\right).Log_2\left(\frac{100}{138}\right)+\left(\frac{38}{138}\right).Log_2\left(\frac{38}{138}\right)\right] = 0.849$$

We can show that entropy decreases as we go down the decision tree.

3.3 DECISION TREES CONSTRUCTION

The problem of constructing a decision tree is to propose learning algorithms, i.e. algorithms which, given a sample S as input, construct a decision tree.

3.3.1 General principle

The general principle of decision tree construction methods is to recursively divide the examples in the training set as efficiently as possible by tests defined using attributes, until we obtain subsets of examples that all belong to the same class.

Dans toutes les méthodes, on trouve les trois opérateurs suivants :

1. *Decide whether a node is terminal,* i.e. decide whether a node should be labelled as a leaf. For example: all examples are in the same class, there are fewer than a certain number of errors, ...

2. *Select a test to associate with a node.* For example: randomly, using statistical criteria, etc.

3. *Assigning a class to a sheet.* The majority class is assigned, except where cost or risk functions are used.

The methods will differ in the choices made for these different operators, i.e. the choice of test (for example, use of the gain and entropy function) and the stopping criterion (when to stop the growth of the tree, i.e. when to decide whether a node is terminal). The general outline of the algorithms is as follows:

Algorithme 3.2 : Generic algorithm for building a decision tree

```
Algorithme Decision Tree Construction
Input : Data set S
Output : Decision Tree
Begin
        Initialise the empty tree; the root is the current node
        Repeat
                Check whether the current node is terminal
                If the node is terminal
                        then assign it a class
                else
                        Select a test and create the sub-tree
                EndIf
                Go to the next unexplored node if there is one
        Until getting a decision tree
End.
```

With such an algorithm, it is possible to calculate a decision tree with little or no apparent error. A perfect decision tree is one in which all the examples in the training set are correctly classified. Such a tree does not always exist (if there are two examples such that two different classes correspond to two identical descriptions). The aim is to build a tree with the smallest possible classification error.

3.3.2 ID3 Algorithm

ID3 (Iterative Dichotomiser) is one of a number of algorithms that have been proposed for generating a decision tree from a training dataset. This algorithm was developed in 1986 by Ross Quinlan. An improvement on ID3 was published by Quinlan in 1990 under the name C4.5.

The following table represents the PlayTennis dataset presented by Quilan himself to introduce the ID3 algorithm. Note that all the variables (corresponding to the columns) have been made discrete.

The ID3 algorithm starts with a table whose data has already been classified (labelled). From this table, the algorithm constructs a decision tree which can predict the class of each of the data items in the table, and even the class of new data (which does not appear in the dataset).

Table 3.1 : Discretised Quinlan PlayTennis dataset (Quinlan, 1986)

N°	Sky	Temperature	Humidity	Wind	Class
1	Sunny	Warm	High	Weak	No
2	Sunny	Warm	High	Strong	No
3	Overcast	Warm	High	Weak	Yes
4	Rainy	Medium	High	Weak	Yes
5	Rainy	Cool	Normal	Weak	Yes
6	Rainy	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Medium	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rainy	Medium	Normal	Weak	Yes
11	Sunny	Medium	Normal	Strong	Yes
12	Overcast	Medium	High	Strong	Yes
13	Overcast	Warm	Normal	Weak	Yes
14	Rainy	Medium	High	Strong	No

Sky, *Temperature*, *Humidity* and *Wind* are the four attributes that describe the data. We can see that the dataset contains just *14* rows corresponding to situations in which tennis players accept or refuse to play depending on the values taken by the attributes describing the weather conditions. But in reality, there are *36* different examples if we vary the attributes with all the possible values they can take on:

|{Sunny,Overcast,Rainy}| × |{Warm,Medium,Cool}| x |{High,Normal}| × |{Weak, Strong}| =3×3×2×2=9×4=**36**

The ID3 algorithm is based on the concept of attributes and classes from machine learning (discrete classification). This algorithm looks for the most relevant attribute to test so that the tree is as short and optimised as possible.

To find the attribute to test, we use the entropy defined in the previous section.

Initially, the algorithm takes the whole dataset S = {J_1, J_2, J_3, ..., J_{14}}. And as 9 out of 14 examples give the decision (or class) *Yes* and 5 out of 14 give the decision No, we can calculate the following proportions:

$$P_{Yes} = 9/14 \qquad P_{No} = 5/14$$

The entropy of S can be calculated as follows:

$$Entropy(S) = -\left[\left(\frac{9}{14}\right).Log_2\left(\frac{9}{14}\right) + \left(\frac{5}{14}\right).Log_2\left(\frac{5}{14}\right)\right] = 0.94$$ (equation 3.2)

Now that we know that the initial entropy of the dataset is *0.94*, we need to know which attribute to test first, then second, and so on.

To find out which attribute to test, we need to use the notion of *entropy gain*. The gain is defined by a set of examples and by an attribute. This formula is used to calculate the contribution of this attribute to the disorder of the set. The more an attribute contributes to disorder, the more important it is to test it in order to separate the set into smaller sets with lower entropy.

Here is the formula that calculates the entropy gain for a set S and an attribute A.

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} x\, Entropy(S_v)$$ (equation 3.3)

The attribute that will be tested at this node of the tree is the node that will reduce entropy the most.

Taking the example again, and considering S as the initial set, to determine which attribute to test, we need to calculate the gain of all the attributes.

Calculating the entropy gain of the "Sky" attribute :

The "Sky" attribute has three possible values: {Sunny, Overcast, Rainy}. The proportion for each of these values in the initial set is 5/14, 4/14 and 5/14 respectively. This gives the following calculation of the entropy gain for this attribute.

$$Gain(S, Sky) = Entropy(S) - \frac{5}{14} Entropy(S_{Sunny}) - \frac{4}{14} Entropy(S_{overcast}) - \frac{5}{14} Entropy(S_{rainy})$$

$$Gain(S, Sky) = 0.94 - \frac{5}{14}\left(-\frac{3}{5}.Log_2\left(\frac{3}{5}\right) - \frac{2}{5}.Log_2\left(\frac{2}{5}\right)\right) - \frac{4}{14}\left(-\frac{0}{4} - \frac{4}{4}.Log_2\left(\frac{4}{4}\right)\right) - \frac{5}{14}\left(-\frac{3}{5}.Log_2\left(\frac{3}{5}\right) - \frac{2}{5}.Log_2\left(\frac{2}{5}\right)\right)$$

$$Gain(S, Sky) = 0.247$$

Calculating the entropy gain of the "Temperature" attribute :

The "Temperature" attribute has three possible values: {Warm, Medium, Cool}. Its entropy gain is :

$$Gain(S, Temperature) = Entropy(S) - \frac{4}{14} Entropy(S_{warm}) - \frac{6}{14} Entropy(S_{medium}) - \frac{4}{14} Entropy(S_{cool})$$

$$Gain(S, Temperature) = 0.028$$

Calculating the entropy gain of the "Humidity" attribute :

The "Humidity" attribute has two possible values: {High, Normal}. Its entropy gain is :

$$Gain(S, Humidity) = Entropy(S) - \frac{7}{14} Entropy(S_{high}) - \frac{7}{14} Entropy(S_{normal})$$

$$Gain(S, Humidity) = 0.153$$

Calculating the entropy gain of the "Wind" attribute :

The "Wind" attribute has two possible values: {Weak, Strong}. Its entropy gain is :

$$Gain(S, Wind) = Entropy(S) - \frac{8}{14} Entropy(S_{weak}) - \frac{6}{14} Entropy(S_{strong})$$

$$Gain(S, Wind) = 0.048$$

Here is a summary of the calculations made :

Gain(S, Sky) = **0.247**
Gain(S, Temperature) = 0.028
Gain(S, Humidity) = 0.153
Gain(S, Wind) = 0.048

The calculations show that: Gain(S, Temperature) < Gain(S, Wind) < Gain(S, Humidity) < Gain(S, Sky). The greatest gain is for *Sky*. *Sky* is therefore the first attribute tested in the tree. If we look at each child node, we see that for the overcast node, all the results are positive. So there's no attribute to test here, we can label Yes directly. The following figure shows the decision tree to be constructed after this first iteration.

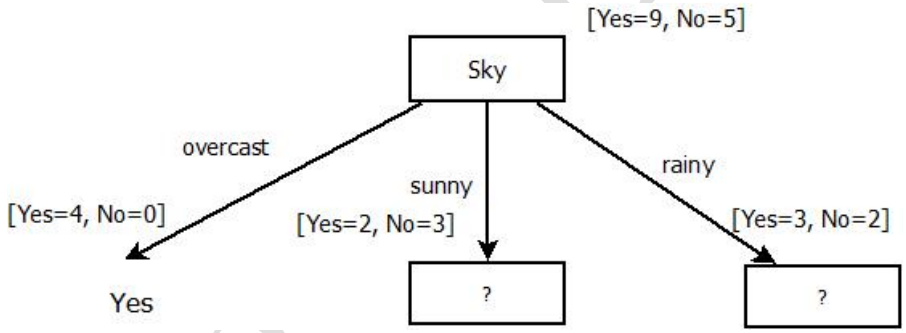

Fig. Tree after the first iteration of its creation with ID3

We now need to continue adding test nodes after *Sunny* and *Rainy* because there is a mixture of classes between the examples. So let's determine for *Sunny* which is the best attribute to test using entropy gain again. However, it is no longer useful to test the gain of *Sky*, as it has just been used. The results of the calculation are given directly:

Gain(S_{sunny}, Temperature) = 0.571
Gain(S_{sunny}, Humidity) = **0.971**
Gain(S_{sunny}, Wind) = 0.019

We can see that : Gain(S_{sunny}, Wind) < Gain(S_{sunny}, Temperature) < Gain(S_{sunny}, Humidity)

The biggest gain is for *Humidity*. You can see that the gain is equal to S_{sunny} entropy. This means that all the children of Humidity will give a class (label). Here's our tree after a second iteration of ID3.

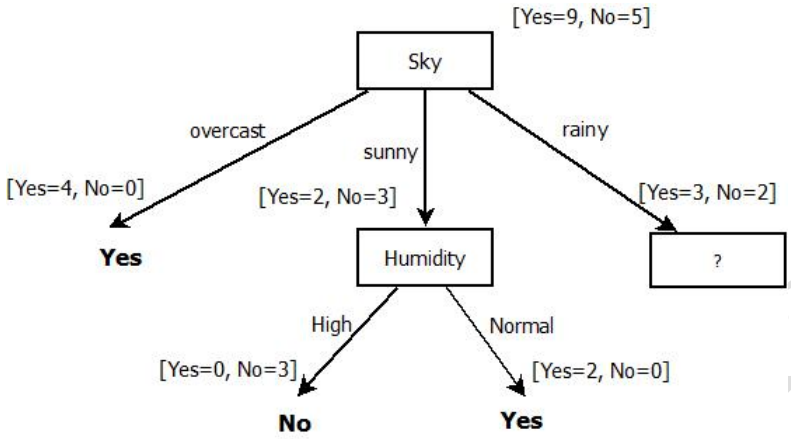

Fig. Tree after the second iteration of its creation with ID3

We still have to continue the tree on the *Rainy* edge. Here are the gains for the different attributes:

$Gain(S_{rainy}, Temperature) = 0.019$

$Gain(S_{rainy}, Humidity) = 0.019$

$Gain(S_{rainy}, Wind) = \mathbf{0.971}$

We can see that : $Gain(S_{rainy}, Temperature) \leq Gain(S_{rainy}, Humidity) < Gain(S_{rainy}, Wind)$
The largest gain is 0.971 and is for Wind. We therefore need to test *Wind*, and since the gain is equal to the entropy of S_{rainy}, each of *Wind*'s child nodes will be a label (pure node).

So here's our final tree.

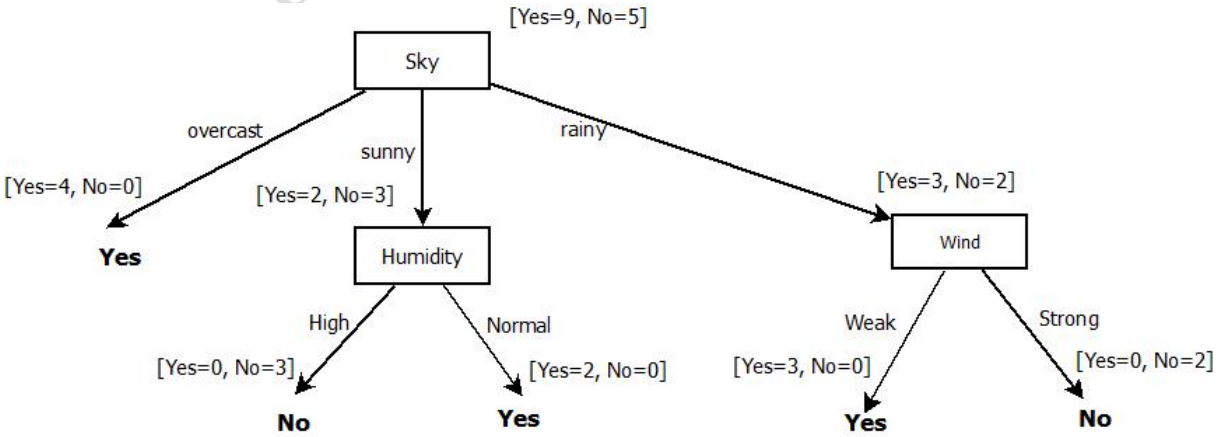

Fig. Final tree after being created with ID3

We can check that this tree gives the correct prediction for each of the 14 cases in the dataset used to construct it. For example, for case number 1 (Sky="Sunny", Temperature="Warm", Humidity="High", Wind="Weak"), the tree gives the class "No", which is consistent with what exists in the training dataset.

But the tree also allows predictions to be made about new cases that do not exist in the dataset. For example, for a new case (Sky='Sunny', Temperature='Cool', Humidity='High', Wind='Weak'), the tree gives the class 'No'.

3.3.3 Algorithm C.45 vs Algorithm ID3

The ID3 algorithm was improved by Ross Quinlan in 1990, under the name C4.5. This latest algorithm introduces the following new features:

- The ability to manipulate continuous values.

- The ability to generate a tree even if data is missing for certain attributes.

CONCLUSION OF THE CHAPTER

…….

EXERCISES

…….