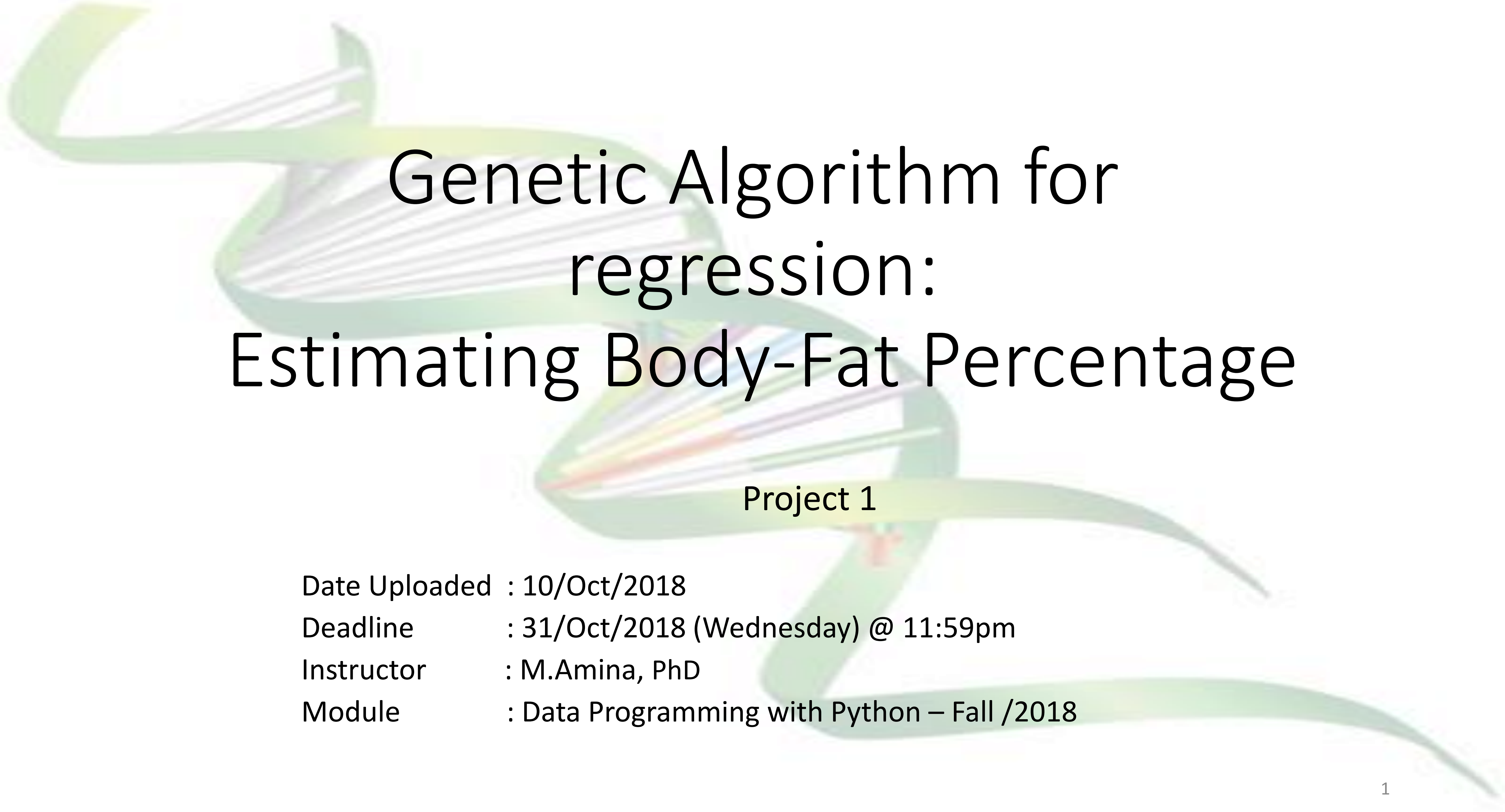
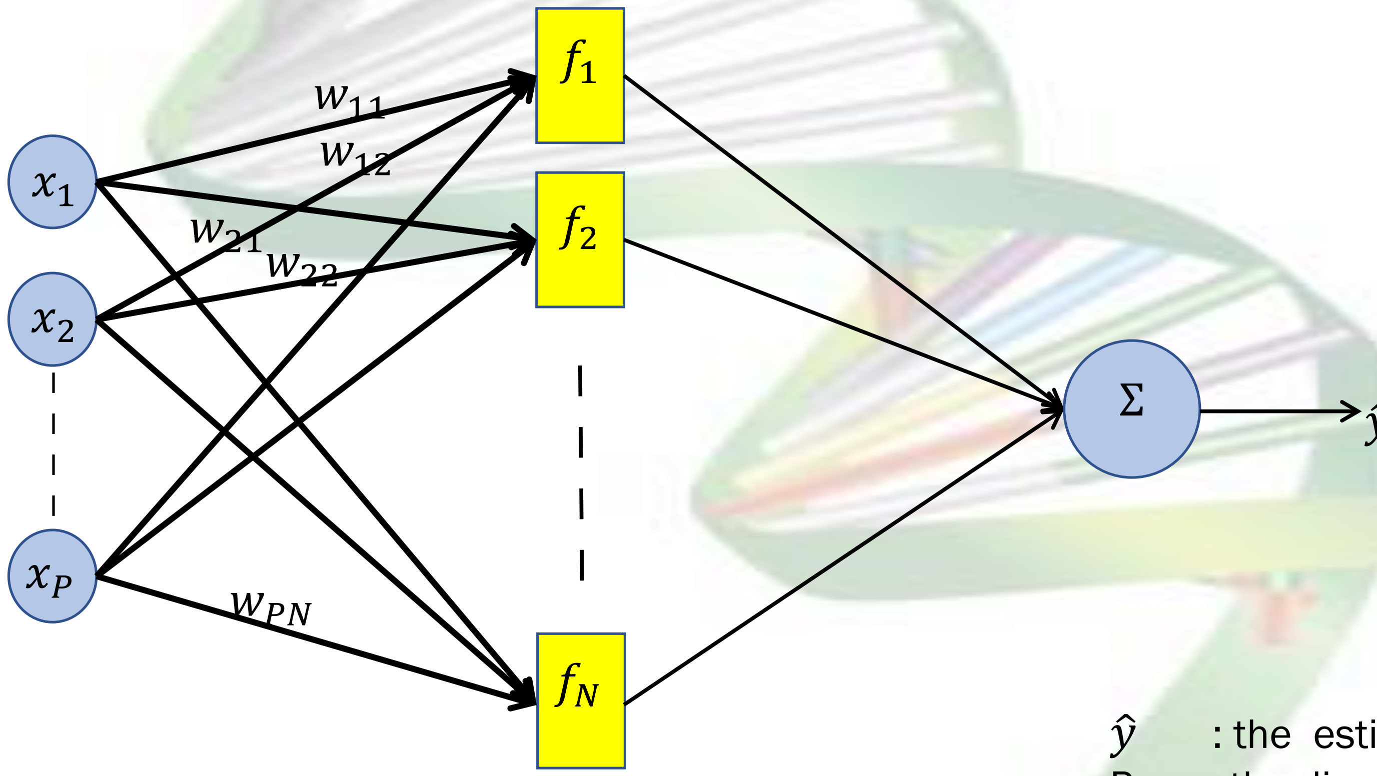


Genetic Algorithm for regression: Estimating Body-Fat Percentage

Project 1

Date Uploaded	: 10/Oct/2018
Deadline	: 31/Oct/2018 (Wednesday) @ 11:59pm
Instructor	: M.Amina, PhD
Module	: Data Programming with Python – Fall /2018

Our STRUCTURE



$$\hat{y} = \sum_{n=1}^{N} f\left(\sum_{p=1}^{P} W_{pn} * x_p\right)$$ **Eq(1)**

$\hat{y}$: the estimation of y

P : the dimension of input data (number of columns).

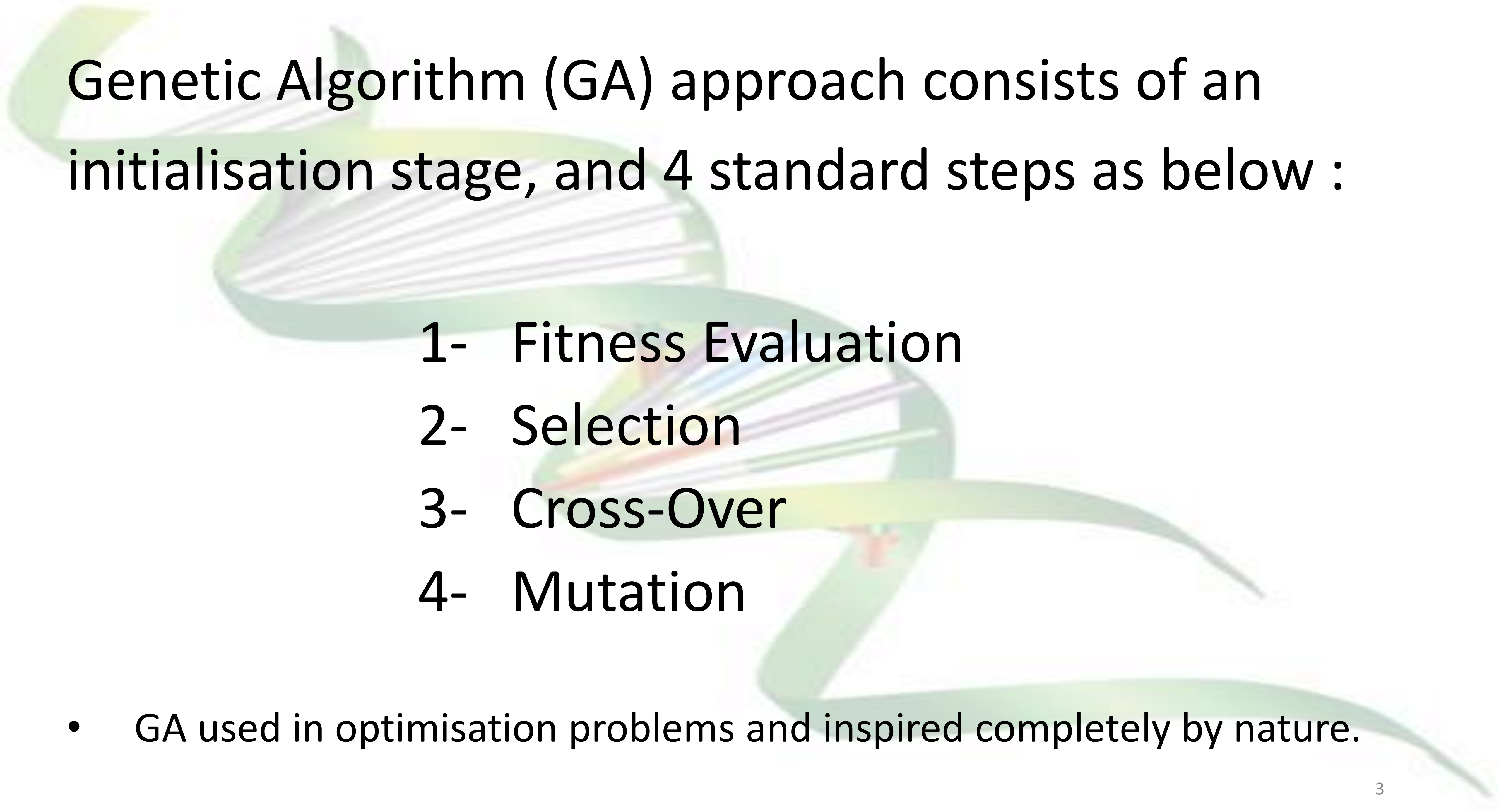
N : the number of functions (equations). Chosen arbitrarily.

Wpn : the weight which links the input feature p, to function n.

Our mission is to find W_{pn} .

f(x) : $f_1 = \ldots = f_N = \frac{1}{1+e^{-x}}$

Mission : Finding Wpn with the help of Genetic Algorithm, to have the best estimation for $\hat{y}$ in above structure.



Genetic Algorithm (GA) approach consists of an initialisation stage, and 4 standard steps as below :

1- Fitness Evaluation
2- Selection
3- Cross-Over
4- Mutation

- GA used in optimisation problems and inspired completely by nature.

Initialisation

- Generate random population of possible weights with values between -1 and +1.
- Normalize the input and output columns to numbers between 0 – 1
- Normalise the weights population to numbers between 0 – 1.
- Multiply the weights to 1000, and then round them so that you make integer numbers. By this stage your generated weights should be integer figures less than 1000.
- Binarize the generated weights by getting the base 2 of them.

<u>**Important :**</u> you need to have fixed size of bits for binarization.

I would suggest 10 bits.

- Concatenate all these 10-bit and make a long '*chromosome*' .

Step 1 : Fitness Evaluation

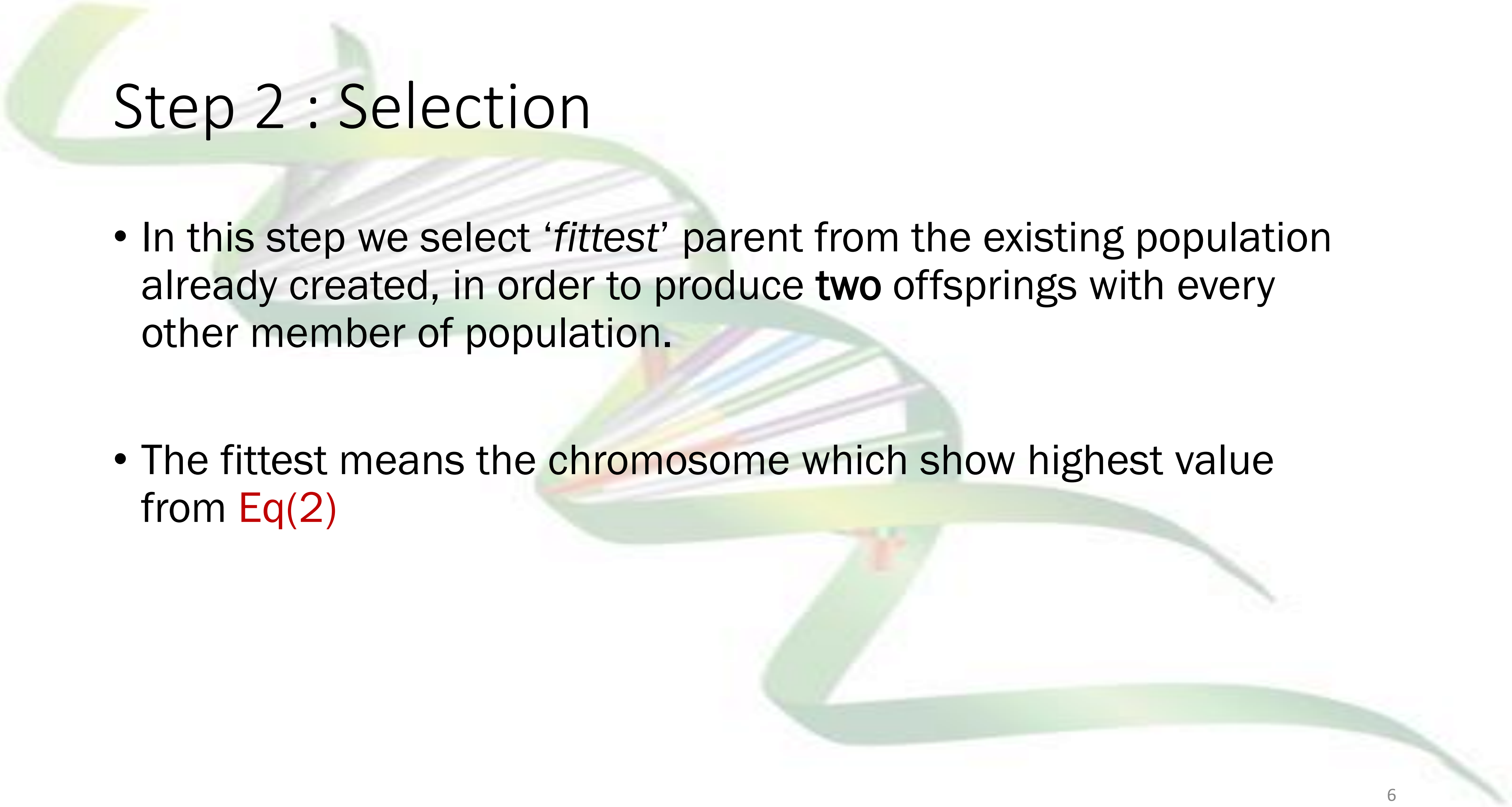
- Fitness function shows how good is a chromosome as solution to your problem. Its mainly user-defined function. In this problem we define it as below :

- $FitnessValue = \left(1 - \frac{\sum_{m=1}^{M}(\hat{y}_m - y_m)^2}{M}\right) * 100$ **Eq(2)**

M : total number of training samples

$\hat{y}$: obtained from **Eq(1)**

Step 2 : Selection

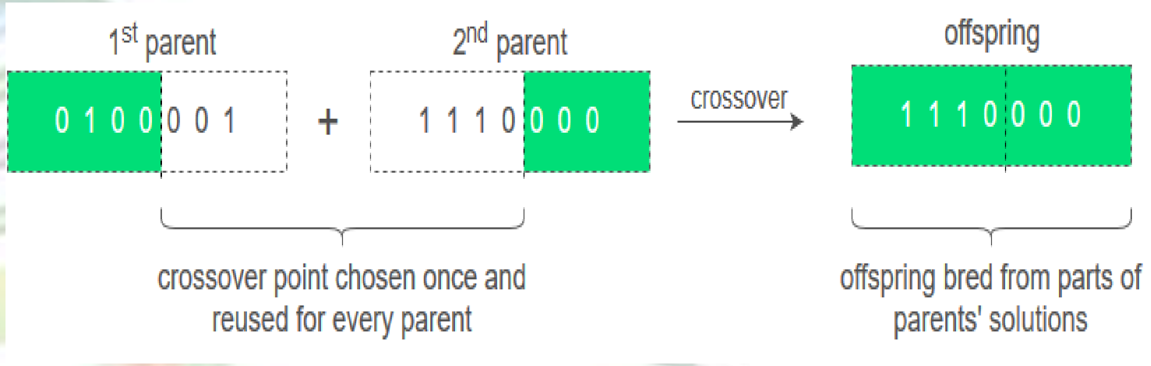


- In this step we select ‘*fittest*’ parent from the existing population already created, in order to produce **two** offsprings with every other member of population.

- The fittest means the chromosome which show highest value from Eq(2)

Step 3: Cross Over (mating)

- The most common type of Cross-Over is single point crossover.
- In single point crossover, you choose a point at which you swap the remaining bits(genes) from one parent to the other.
- The illustration in this slide help you understand it visually.
- As you can see, the offspring takes one section of the chromosome from each parent.
- The point at which the chromosome is broken depends on the randomly selected crossover point.


1st parent
0 1 0 0 0 0 1
+
2nd parent
1 1 1 0 0 0 0
crossover
offspring
1 1 1 0 0 0 0
crossover point chosen once and reused for every parent
offspring bred from parts of parents' solutions

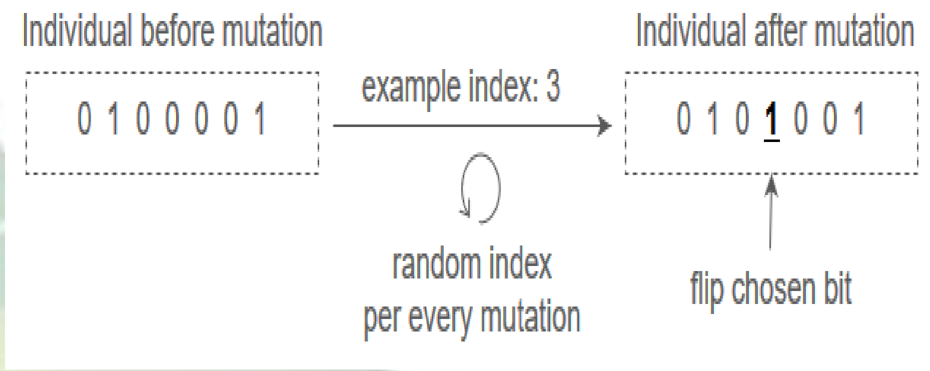
- This particular method is called single point crossover because only one crossover point exists. Sometimes only child 1 or child 2 is created, but oftentimes both offspring are created and put into the new population.

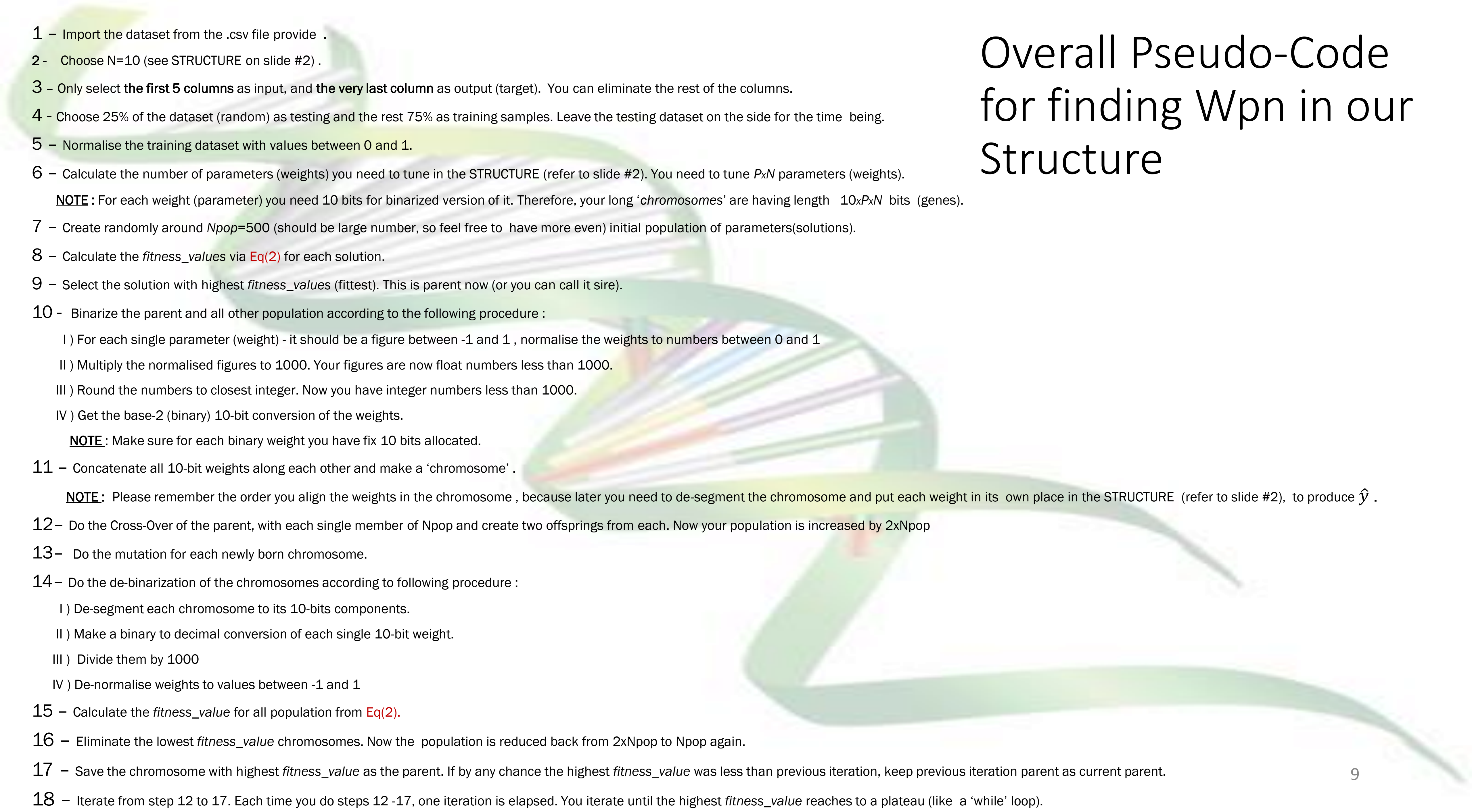
- PLEASE NOTE : when two offsprings created size of initial population becomes doubled.

Step 4: Mutation

- In mutation you randomly select - let’s say – 5 % of the bits in the chromosome, and flip them to 0 if they are 1, and flip them to 1 if they are zero.

- These bits need not to be beside each other.





Overall Pseudo-Code for finding Wpn in our Structure

```
1 – Import the dataset from the .csv file provide .
2 - Choose N=10 (see STRUCTURE on slide #2) .
3 – Only select **the first 5 columns** as input, and **the very last column** as output (target). You can eliminate the rest of the columns.
4 - Choose 25% of the dataset (random) as testing and the rest 75% as training samples. Leave the testing dataset on the side for the time being.
5 – Normalise the training dataset with values between 0 and 1.
6 – Calculate the number of parameters (weights) you need to tune in the STRUCTURE (refer to slide #2). You need to tune *PxN* parameters (weights).
    NOTE : For each weight (parameter) you need 10 bits for binarized version of it. Therefore, your long ‘*chromosomes*’ are having length 10x*PxN* bits (genes).
7 – Create randomly around *Npop*=500 (should be large number, so feel free to have more even) initial population of parameters(solutions).
8 – Calculate the *fitness_values* via Eq(2) for each solution.
9 – Select the solution with highest *fitness_values* (fittest). This is parent now (or you can call it sire).
10 - Binarize the parent and all other population according to the following procedure :
    I ) For each single parameter (weight) - it should be a figure between -1 and 1 , normalise the weights to numbers between 0 and 1
    II ) Multiply the normalised figures to 1000. Your figures are now float numbers less than 1000.
    III ) Round the numbers to closest integer. Now you have integer numbers less than 1000.
    IV ) Get the base-2 (binary) 10-bit conversion of the weights.
        NOTE : Make sure for each binary weight you have fix 10 bits allocated.
11 – Concatenate all 10-bit weights along each other and make a ‘chromosome’ .
    NOTE : Please remember the order you align the weights in the chromosome , because later you need to de-segment the chromosome and put each weight in its own place in the STRUCTURE (refer to slide #2), to produce $\hat{y}$ .
12– Do the Cross-Over of the parent, with each single member of Npop and create two offsprings from each. Now your population is increased by 2xNpop
13– Do the mutation for each newly born chromosome.
14– Do the de-binarization of the chromosomes according to following procedure :
    I ) De-segment each chromosome to its 10-bits components.
    II ) Make a binary to decimal conversion of each single 10-bit weight.
    III ) Divide them by 1000
    IV ) De-normalise weights to values between -1 and 1
15 – Calculate the *fitness_value* for all population from Eq(2).
16 – Eliminate the lowest *fitness_value* chromosomes. Now the population is reduced back from 2xNpop to Npop again.
17 – Save the chromosome with highest *fitness_value* as the parent. If by any chance the highest *fitness_value* was less than previous iteration, keep previous iteration parent as current parent.
18 – Iterate from step 12 to 17. Each time you do steps 12 -17, one iteration is elapsed. You iterate until the highest *fitness_value* reaches to a plateau (like a ‘while’ loop).
```

What you need to do and percentage of the mark.

- Make a python programme from the pseudo-code and do the training with training dataset (55%).
- Scatter Plot the highest fitness_value for each iteration (15%).
- Scatter Plot in 3D, the first and second input and the estimated output ($\hat{y}$), together with real output (y), for testing dataset(20%).

  (Plot y and $\hat{y}$ with different colours)
- Find out the overall error for testing dataset from below (10%) :

$$\frac{\sum_{m\_test=1}^{M\_test}\left(\hat{y}_{m\_test} - y_{m\_test}\right)^2}{M\_test}$$

Some hints (help)

- For binarization in python with 10 bits you may try the following :

```
binary_x  =  bin(decimal_x)[2:].zfill(10)
```

- For converting from binary to decimal, try the following :

```
decimal_x = int('binary_x', 2)
```

- For Cross-Over, you can get some hints from the follwing:

  a) for choosing a random cross-over point

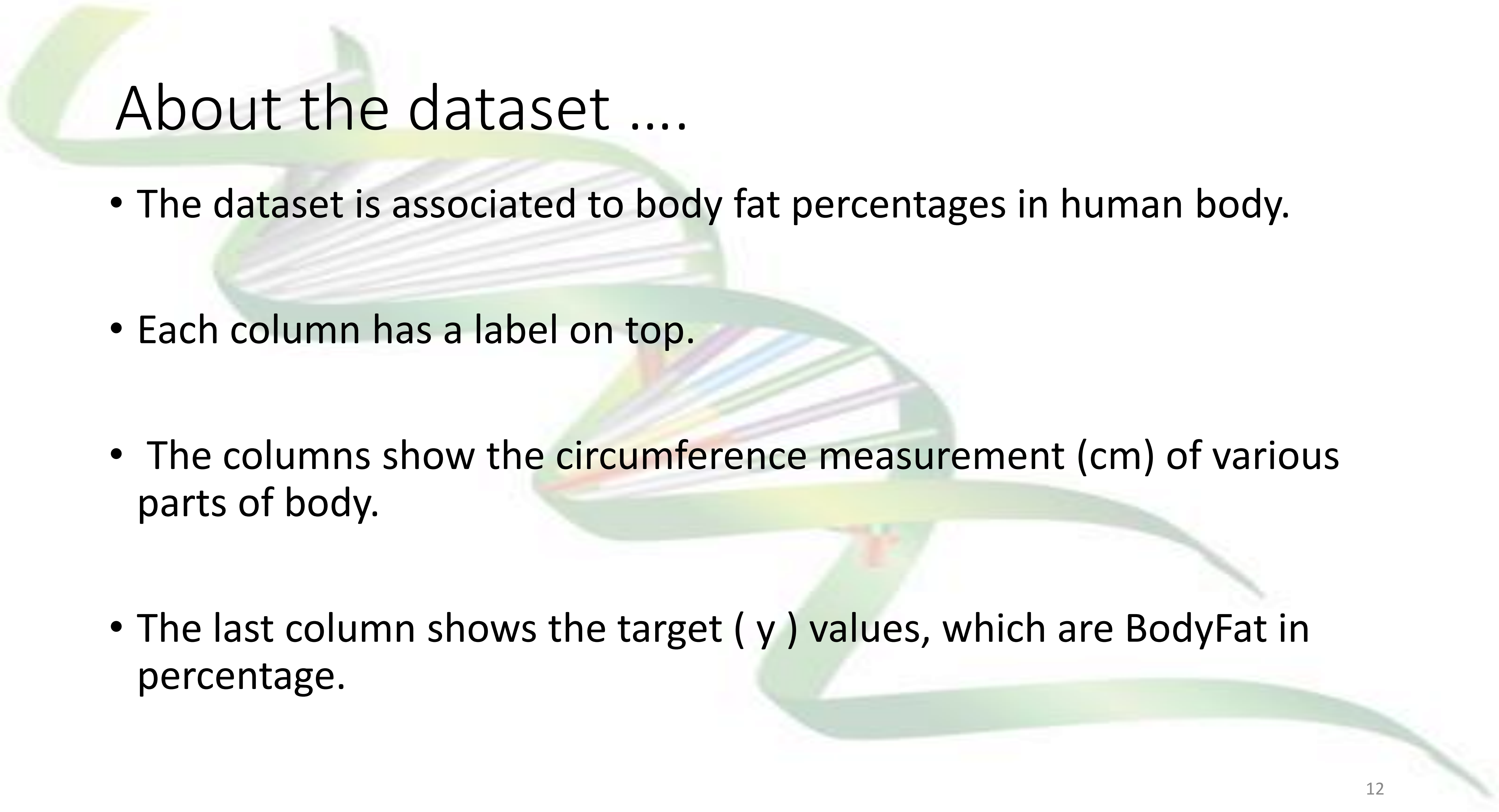
```
import numpy as np
C_Point = np.random.random_integers(2,length_of_chromosome-1)
```

  b) for creating an offspring from two parents (parents[0] & parent[1])

```
import numpy as np
np.hstack((parents[0,:C_Point],parents[1,C_Point:]))
```

- For 3D scatter plot you can use and import the following library :

```
from mpl_toolkits.mplot3d import Axes3D
```



About the dataset ….

- The dataset is associated to body fat percentages in human body.
- Each column has a label on top.
- The columns show the circumference measurement (cm) of various parts of body.
- The last column shows the target (y) values, which are BodyFat in percentage.