# **Implementation**

## **Representation of the Items**

I used an array of ArrayList to represent the features, each index of the array contains a linkedlist that contains the features values. The representation looks lock the following:

Index | values

1. | 3 0 2 1 4 3 2 1 1 1 0 0 2 0 0 2
2. | 4 2 1 4 4 0 0 1 2 2 2 3 0 0 0 1
3. | 2 3 4 1 4 2 0 2 0 1 2 3 0 0 1 2

## **Encoding of the chromosomes**

The chromosome is represented in an object called (Chromosome) that contains an array represent the genes. The array length is equal to the number of features (in our case it’s 8) and it contains zeros and ones, if the index I contains zero then the features is not included in the KNN classifier and if the index I contains one then the item is included to the KNN classifier. The gene looks as follow:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

Indexes 1,2,3,5 and 7 are represent the features that included in the KNN classifier, 0,4 and 6 are not included.

This chromosome represents a KNN classifier with 5 features out of 8. To represent the whole population, I used an Array List each node contains one chromosome. It’s possible to use an array of chromosomes but since the population is expanded dynamically it’s hard to use the array.

## **Termination conditions**

The population keep expanding over than it’s limit until some conditions occurs. The main condition is when the population size reach (4 x limit) size at least once and a better solution than the previous generation found then terminate the generation. If (10 x limit) chromosome are generated and didn’t found a better solution then terminate the generation and consider the previous generation as the new generation.

## **Fitness function**

I calculated the fitness by applying the KNN classifier to the selected features and calculate the accuracy of that classifier using the selected features, then I used the calculated accuracy to calculate the fitness by the equation given to us using alpha factor = 0.9 and beta factor = 0.1. after that, the calculated fitness will be compared with the fitness of KNN classifier when all features are selected (let’s call at FitnessOfAll), if the new fitness is greater or equal to the (FitnessOfAll) then we take it as a new chromosome, if not then we reject it and there is some function will try to fix it.

Example:

Let the FitnessOfAll equal to 0.8.

Chromosome a is:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

Fitness equal to 0.85 then we accept it.

Chromosome b is:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

Fitness equal to 0.78 then we reject it.

## **Selection functions**

In my project, I tried one method for selection: Roulette Wheel. When 2 chromosomes are selected they will be copied without change to the new generation so if the best solution found among them it will not be lost.

### **Roulette Wheel**

Roulette wheel selection or fitness proportionate is a genetic operator that used in genetic algorithms to select chromosomes to combine them to generate new solution. The roulette wheel algorithms give each chromosome a slice of circle based on its value. Then the wheel spin and the chromosome under the wheel’s marker will be chosen. The following example explain the algorithm:

Let the items as follows:

# **Results**

## **Data Set Accuracy**

Using 80% of data for training and 20% for test and k = 11.

Table Dataset Accuracy

|  |  |
| --- | --- |
| Dataset | Accuracy |
| zoo | 0.75 |
| Exactly | 0.69 |
| M-of-n | 0.88 |

## **Classification Accuracy for Datasets**

Using 80% of data for training and 20% for testing and k = 11.

Initial population = 20 and number of generation = 100.

Table Datasets Accuracy

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Zoo | Exactly | M-of-n |
| 1 | 0.85 | 0.82 | 1 |
| 2 | 0.95 | 0.75 | 0.99 |
| 3 | 0.9 | 0.72 | 1 |
| 4 | 0.95 | 0.71 | 0.92 |
| 5 | 0.85 | 0.71 | 1 |
| 6 | 0.9 | 0.72 | 0.93 |
| 7 | 0.95 | 1 | 1 |
| 8 | 0.95 | 0.74 | 1 |
| 9 | 0.9 | 1 | 0.97 |
| 10 | 0.95 | 0.74 | 0.98 |
| Avg. | 0.91 | 0.79 | 0.98 |

## **Performance of KNN when using original dataset and selected features**

Table Performance of KNN when using original dataset and selected features

|  |  |  |
| --- | --- | --- |
| Dataset | Accuracy (original dataset) | Accuracy (selected dataset) |
| Zoo | 0.75 | 0.91 |
| Exactly | 0.69 | 0.79 |
| M-of-n | 0.88 | 0.98 |