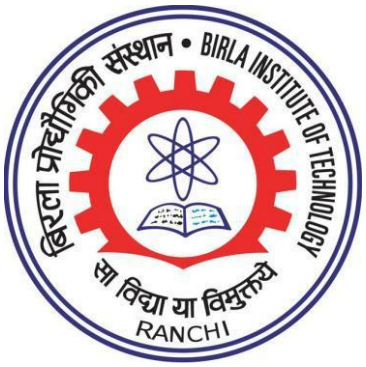
DEPT. OF COMPUTER SCIENCE AND ENGG.

BIRLA INSTITUTE OF TECHNOLOGY



HEART DISEASE PREDICTOR

A new hybrid knowledge-based system using PRISM classifier and parallel genetic algorithm for diagnosis of heart disease

A project report submitted on the partial fulfilment

of the requirement for the award of the degree

of Bachelors of Engineering in

Computer Science

By :

VIBHOR CHATURVEDI BE/10223/2014

RAJAT JAIN BE/10235/2014

S. SHREE VIGNESH BE/10242/2014

Project Mentor :

DR. B.K. SARKAR

Declaration Certificate

This is to certify that the work presented in the thesis entitled "
" in partial fulfilment of the requirement for the award of degree of "Bachelor of Engineering in Computer Science and Engineering of Birla Institute of Technology Mesra, Ranchi" is an authentic work carried out under my supervision and guidance.

To the best of my knowledge, the content of this thesis does not form a basis for award of any previous Degree to anyone else.

(Guide's Name and Signature)

Dept.Of Computer Science and Engineering,
Birla Institute of Technology,
Mesra, Ranchi.

Table of Contents

1. Objective of the project

In the present research, a new hybrid model (integrating PRISM learner and GA) is to be introduced for effective diagnosis of heart disease, addressing the limitations of the existing systems. In particular, the model suggests here a two-level optimization strategy, where *level*-1 first attempts to identify parallelly an *optimal* proportion (P_{opt}) for training and test sets for each Heart dataset using PRISM learner on Cluster-based HPC machine. Next, the best training set (T_{best}) for P_{opt} is to be searched again parallelly by the same learner. On the other hand, *level*-2 optimization aims to refine the rule-set(R) generated by the PRISM learner on T_{best} by employing parallel genetic algorithm(PGA).

This project consists of two parts, i.e, two levels of optimization. Our part was to carry the project forward with the second level of optimization.

2. Introduction

In this project we attempt to develop a model applying sequential genetic algorithms to enhance the performance for application to data sets in the medical domain.

Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved.

Algorithm is started with a **set of solutions** (represented by **chromosomes**) called **population**. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (**offspring**) are selected according to their fitness - the more suitable they are the more chances they have to reproduce.

This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

2.1 Why the medical domain?

Heart diseases are and have been very common in elderly people, complicating lives and posing danger. According to 2009 WHO survey, 1 out of every 4 deaths in the world on an average is caused from a heart disease. To get a premature knowledge and understanding of the ailment before it has taken a big form, is an important aspect in its diagnosis. This is why we chose this project so as to help make this world a better place to live in.

Natural Domain – Medical Data Sets derive their data from nature and natural occurrences which presents an interesting and pertinent area of work.

Presence of Conflicts – Medical Diagnoses are more challenging than regular inferences owing to the fact that a particular set of symptoms can be caused by more than one disease.

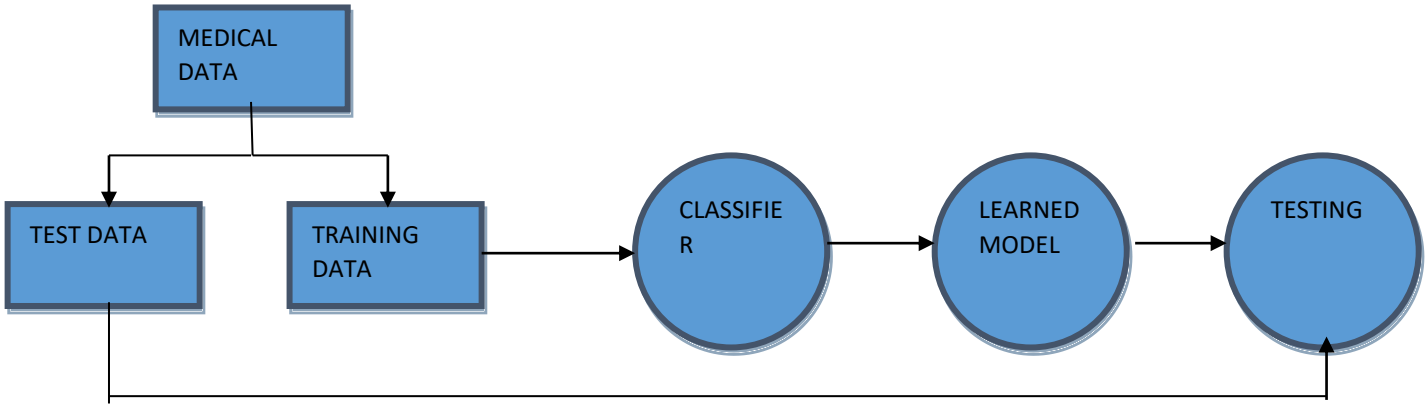
High Dimensionality – Medical Data Sets are comprised of several factors such as the patient's history, symptoms, modes of treatment and so on.

Missing Attributes – A patient's data can contain missing fields of information such as the illness history that present a challenge to arriving at a comprehensive analysis.

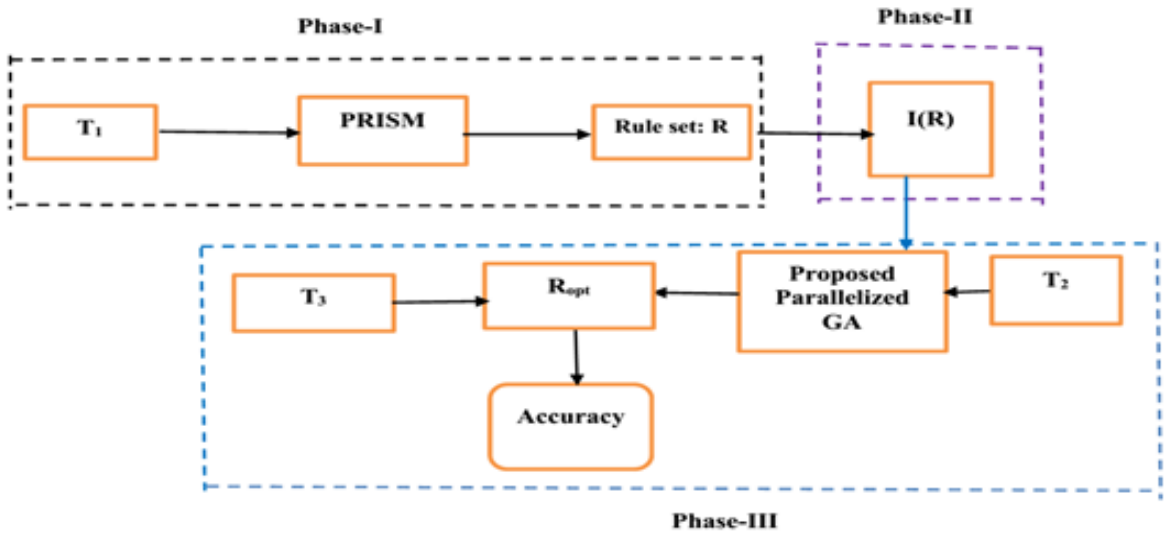
Humanitarian Aspect – Research into improving the efficiency of analyses in the medical domain leads to better healthcare.

3. Research Methodology

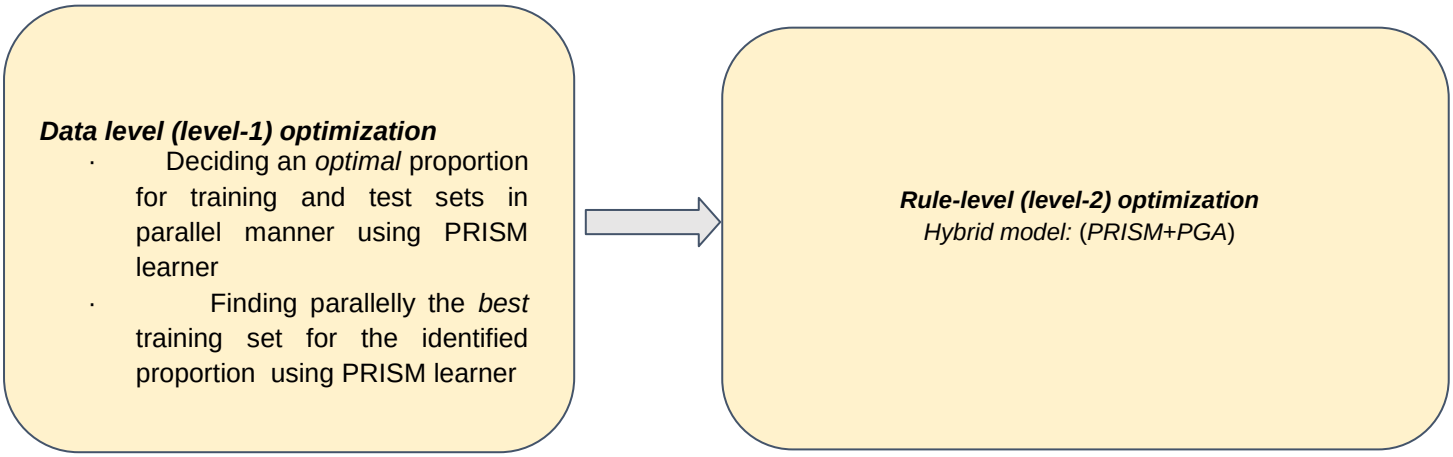
THE GENERALISED CLASSIFICATION MODEL



HYBRID MODEL



a. Workflow overview of the project



4. Rule based classification

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

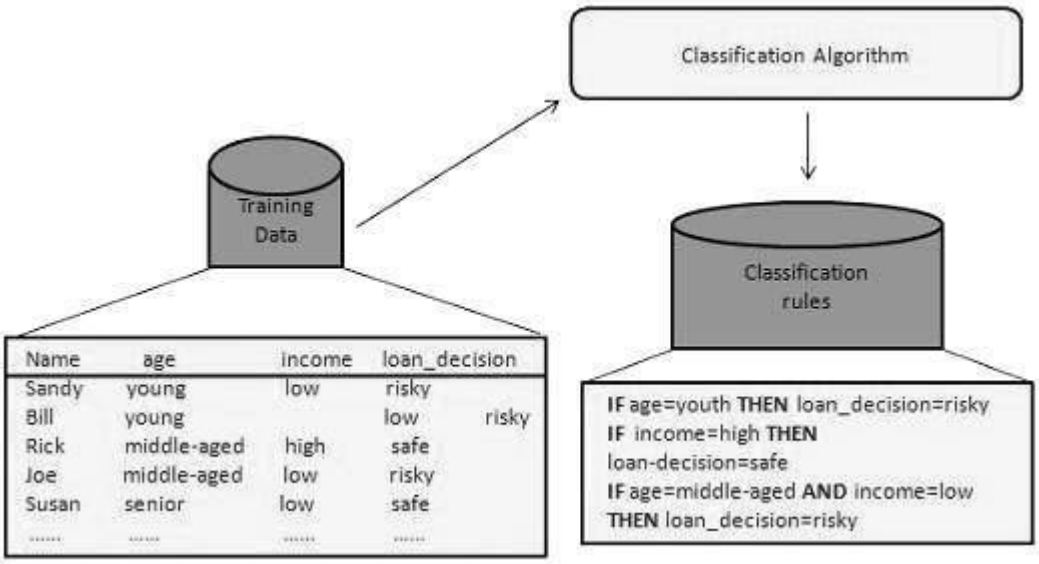
Rule Based Classification is the preferred approach when dealing in the medical domain because of the following reasons :-

- Modularity - Each rule can be seen like a "unit of knowledge".
- Uniformity - All the knowledge is expressed in the same format.
- Naturalness - The rules are a natural format to express knowledge in a domain.

As a result of these, it becomes easier for physicians and experts to express knowledge and diagnoses more naturally which is desirable and advantageous in the medical domain as it leads to simplicity and better readability of data.

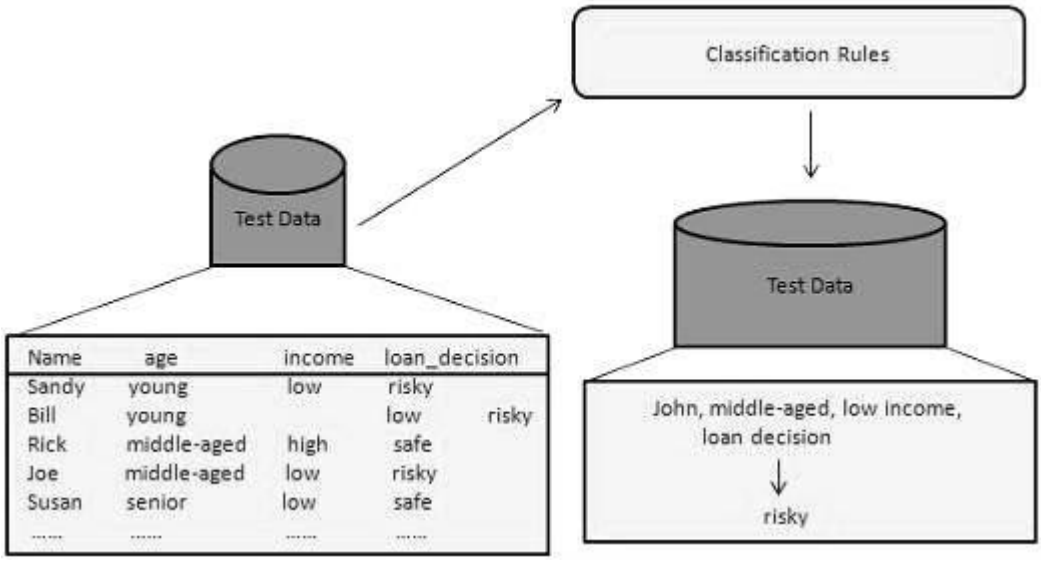
5. Building a rule based classifier

- This step is the learning step or the learning phase.
- In this step the classification algorithms build the classifier.
- The classifier is built from the training set made up of database tuples and their associated class labels.
- Each tuple that constitutes the training set is referred to as a category or class. These tuples can also be referred to as sample, object or data points.



5.1 Using classifier for classification

In this step, the classifier is used for classification. Here the test data is used to estimate the accuracy of classification rules. The classification rules can be applied to the new data tuples if the accuracy is considered acceptable.



5.3 Advantages and Disadvantages of PRISM

ADVANTAGES :-

- It uses rule based classification, and rules are easier to interpret.
- It is easier to implement and understand as compared to other classification algorithms.
- As medical data set is an imbalanced data set, so it is necessary to include all the classes including the minority classes. PRISM
  Does not ignore minority classes.

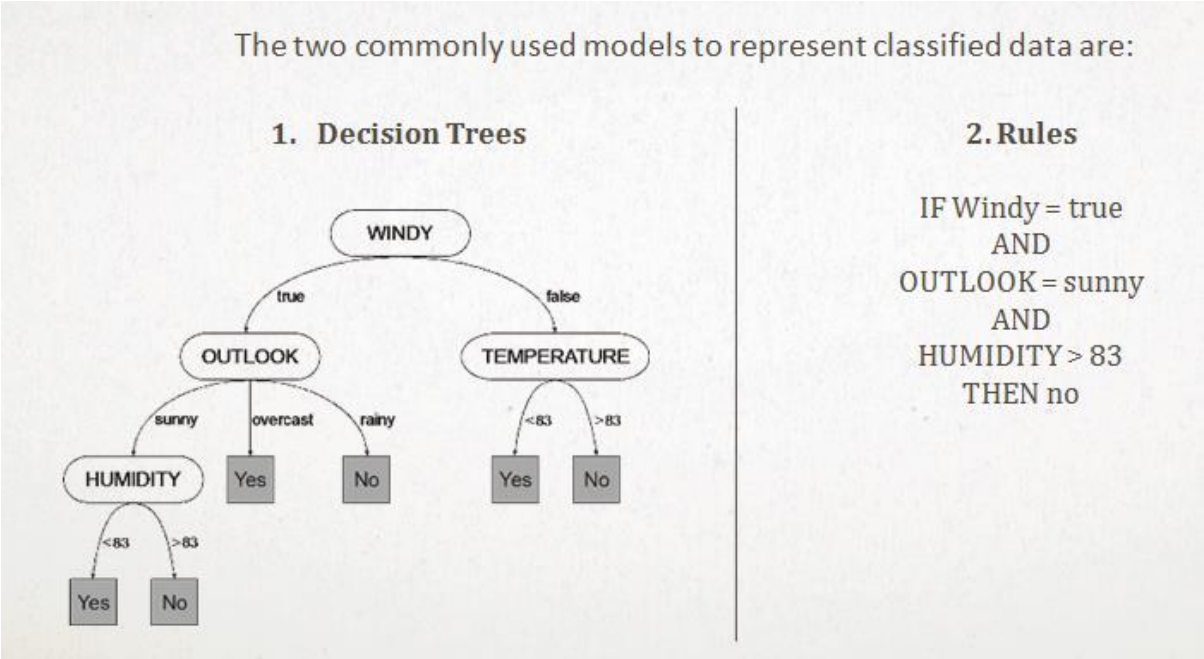
DISADVANTAGES :-

- It does not have rule pruning.
- Better algorithms have been developed but they have implicit knowledge.
- Statistical machine learning methods give better accuracy as compared to PRISM.

5.4 Why PRISM over Decision-Tree Classifiers ?

The decision tree output of some classifiers is one of their major weaknesses. Not only can it be incomprehensible and difficult to manipulate, but its use in expert

systems frequently demands irrelevant information to be supplied. This report argues that the problem lies in the induction algorithm itself and can only be remedied by radically altering the underlying strategy. It describes a new algorithm, PRISM which uses a different induction strategy to induce rules which are modular, thus avoiding many of the problems associated with decision trees.



5.5 PRISM Algorithm

Input: Initialize examples (E) to the training set (T).

For each class *c,*

While T contains examples of class c

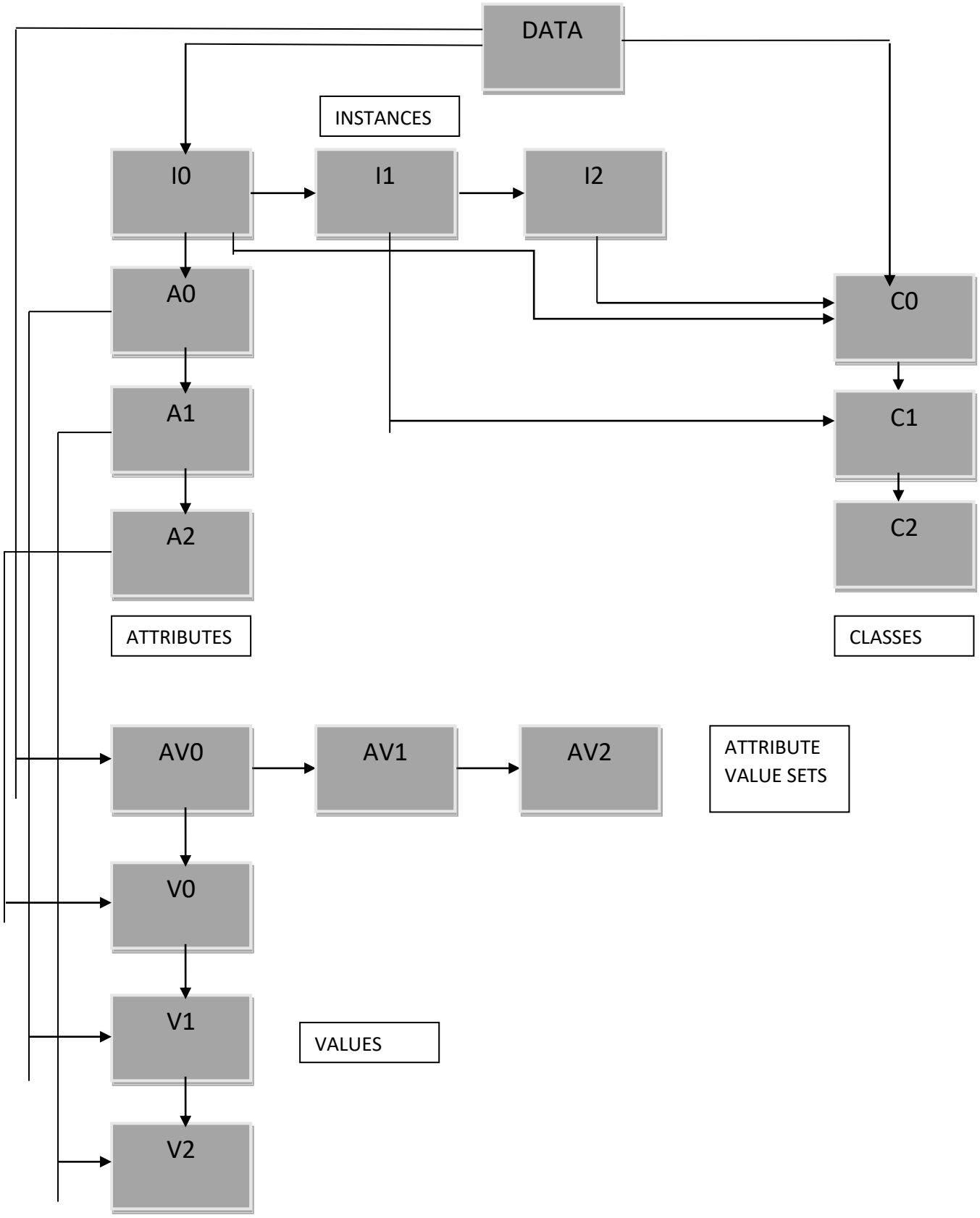
- Create a rule *r* with an empty left-hand side (LHS) that predicts class c

Until *r* is perfect (or there are no more attributes to use) *do*

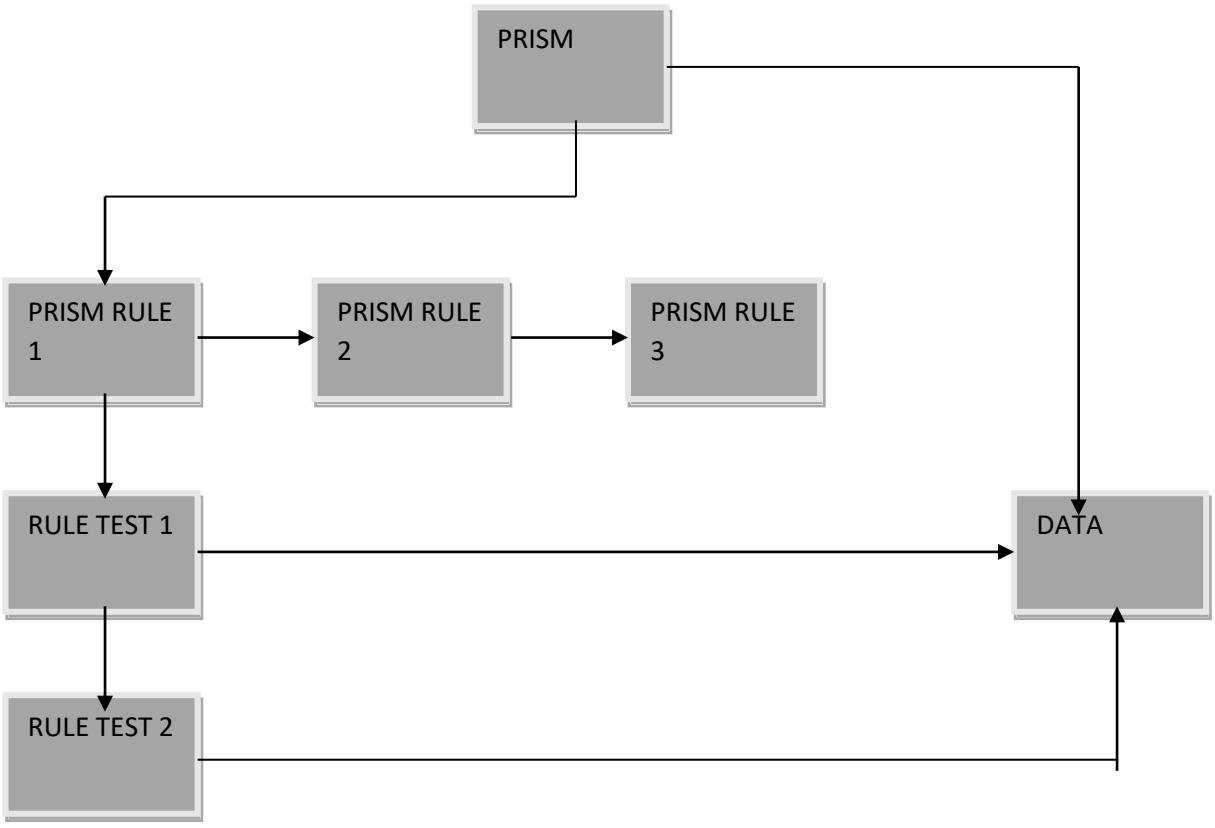
For each attribute: *A* not mentioned in *r*, and each value *v,*

- Consider adding the condition A = v to the LHS of *r*
  - Select *A* and v to maximize the *accuracy* p/t , where p= number of instances covered by rule that belong to the positive class and *t*= number of instances covered by rule (r)
    (*break ties by choosing the condition with the largest p*)
  - Add A = v to r
  - Remove the instances covered by *r* from T

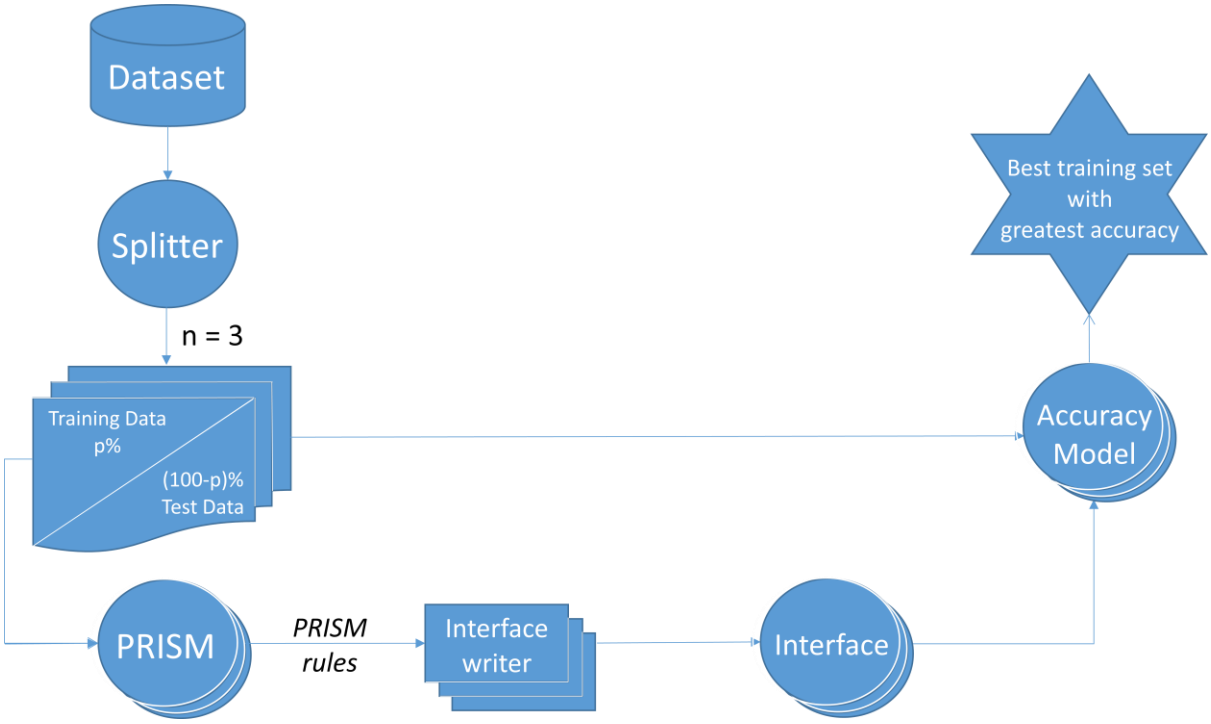
5.5.1 Input Design



5.5.2 PRISM Design



5.6 Level – I (Detailed Workflow)



- Once the number of processors to be used has been determined, the splitter splits the dataset into n sets of <Training data, Test data>.
- The Training data for each is then fed to the PRISM algorithm as input, in parallel.
- Following this, each processor writes the rules using the interface writer to the interface.
- The accuracy model then uses these rules in combination with the test data to determine the accuracy for each.
- The best result i.e., the one with the greatest accuracy is chosen.

6. Understanding Genetic Algorithms

Genetic algorithms are inspired by Darwin's theory about evolution. Solution to a problem solved by genetic algorithms is evolved.

Algorithm is started with a **set of solutions** (represented by **chromosomes**) called **population**. Solutions from one population are taken and used to form a new population. This is motivated by a hope, that the new population will be better than the old one. Solutions which are selected to form new solutions (**offspring**) are selected according to their fitness - the more suitable they are the more chances they have to reproduce.

This is repeated until some condition (for example number of populations or improvement of the best solution) is satisfied.

6.1 Outline of basic genetic algorithm

1. **[Start]** Generate random population of *n* chromosomes (suitable solutions for the problem)
2. **[Fitness]** Evaluate the fitness *f(x)* of each chromosome *x* in the population
3. **[New population]** Create a new population by repeating following steps until the new population is complete
    1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)
    2. **[Crossover]** With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
    3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
    4. **[Accepting]** Place new offspring in a new population
4. **[Replace]** Use new generated population for a further run of algorithm
5. **[Test]** If the end condition is satisfied, stop, and return the best solution in current population
6. **[Loop]** Go to step 2

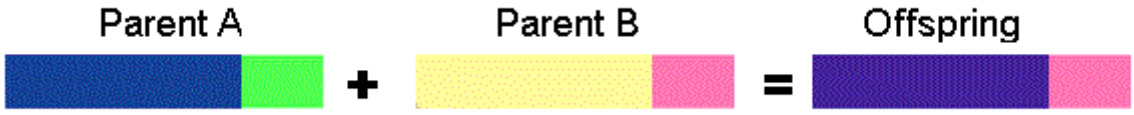
6.2 Genetic Operators

Selection, crossover and mutation

As you can see from the genetic algorithm outline, the crossover and mutation are the most important part of the genetic algorithm. The performance is influenced mainly by these two operators

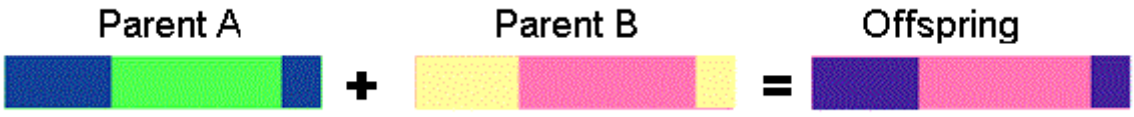
- **Selection of rules :** As we already know from the GA outline, chromosomes are selected from the population to be parents to crossover. The problem is how to select these chromosomes. According to Darwin's evolution theory the best ones should survive and create new offspring. There are many methods how to select the best chromosomes, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others.
- **Crossover (Reproduction operator) :**
  - **Single point crossover** - one crossover point is selected, binary string from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent

    11001011+11011111 = 11001111



  - **Two point crossover** - two crossover point are selected, binary string from beginning of chromosome to the first crossover point is copied from one parent, the part from the first to the second crossover point is copied from the second parent and the rest is copied from the first parent

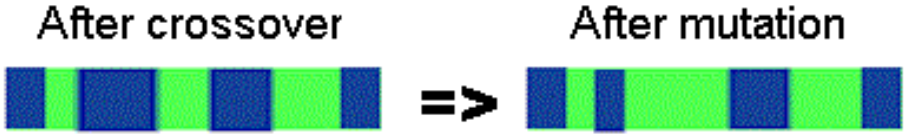
    11001011 + 11011111 = 11011111



- **Mutation :**

  Bit inversion - selected bits are inverted

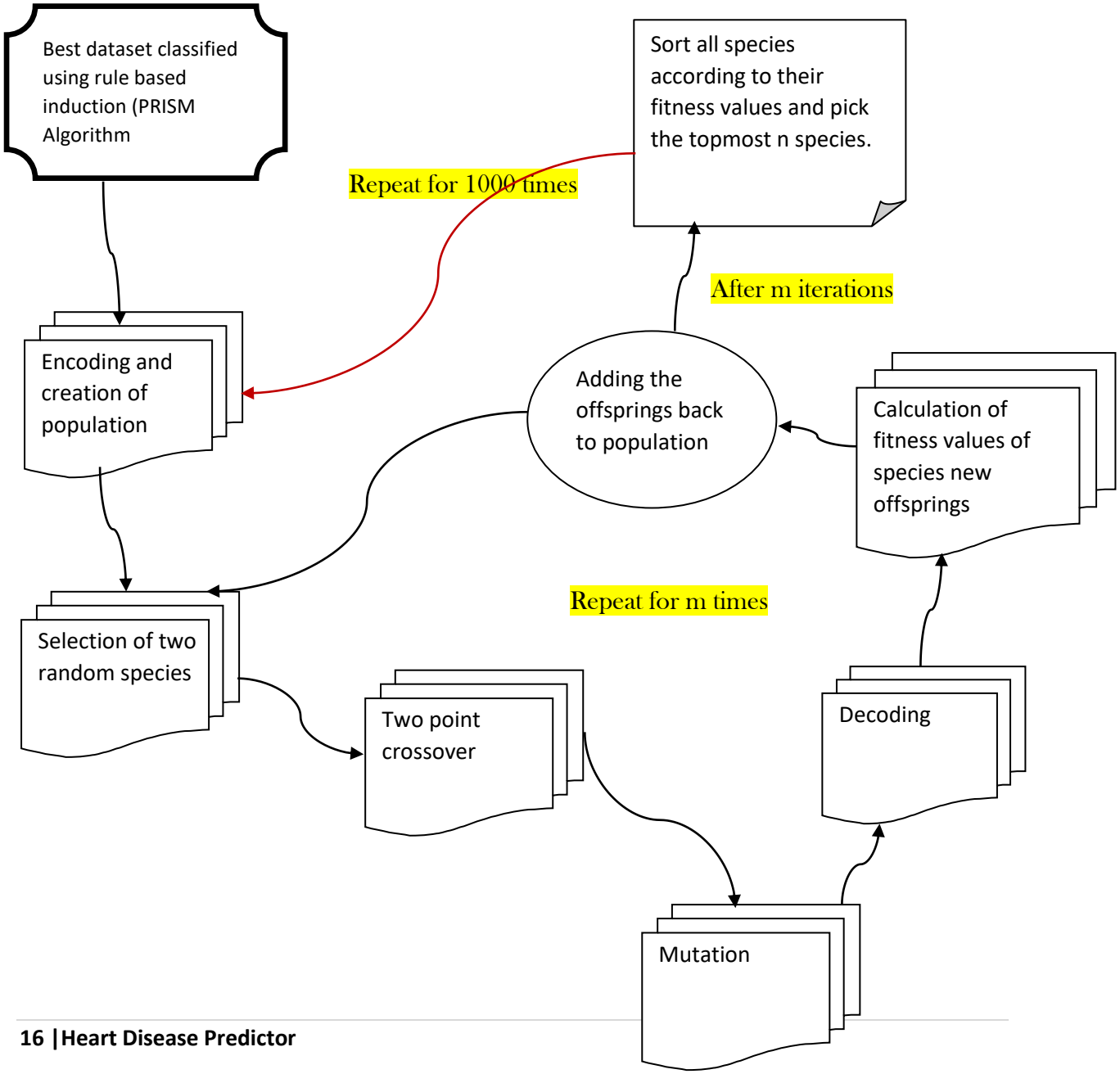
  11001001 => 10001001



6.2.1 Fitness Function

A fitness function is essentially an objective function for any problem, and it depends upon the nature of the problem. The function gives a means of evaluating a solution for its inclusion in the resulting set. From the perspective of decision making, factors such as *prediction-accuracy*, *error* (i.e., misclassification) *rate*, *true positive rate*, etc. are essentially considered for constructing fitness function.

7.Proposed Methodology



7.1 Encoding Scheme :

From the chosen discretized heart datasets, it has been observed that the largest value of any attributes among all the datasets does not exceed 7. Hence, 3 bits are sufficient to encode each attribute. Accordingly, 3x13=39 bits in total are necessary to represent every rule of the datasets, viz., Heart(Cleveland), Heart(Hungarian) and Heart(Swiss), since each of these databases has total 13 attributes (including) the class attribute. However, this size becomes 3x14=42 bits for Heart(Statlog) because this set contains total 14 attributes (including the class attribute).

As per this representation, if '*' appears in a rule for an attribute, then this symbol is simply encoded as: 000. On the other hand, an equivalent binary coding is adopted for values other than '*'. This technique is undoubtedly one of the easiest implementation techniques for encoding the rules.

```
char* encoding(char* d)
{
    char s[60];
    char* b=s;
    while(*d!='\0')
    {
        int dec=(int)(*d-'0');
        d++;
        if(dec==0)
        {
            *b='0';
            b++;
            *b='0';
            b++;
            *b='0';
            b++;
        }
        else if(dec==1)
        {
            *b='1';
            b++;
            *b='0';
            b++;
            *b='0';
            b++;
        }
        else if(dec==2)
        {
            *b='0';
            b++;
            *b='1';
```

7.2 Selection :

In this research, the selection of parents for performing *crossover* is made at random. Here, the rules in the rule set (*i.e.*, population) are numbered as: 1, 2, . . ., *n*. Now, two parents say: p_1 and p_2 are picked *randomly* from the

current rule set with *n* rules, and placed them into a mating pool (say M). Thus, the probability of a selecting rule in a rule set (with *n* rules) is:

```
pair selection(chrom data[] , int n){

    int sel1 = _rand()%n , sel2 = _rand()%n ;

    while(sel2 == sel1)
        sel2 = _rand()%n;

    pair p1;
    p1.first = data[sel1];
    p1.second = data[sel2];

    return p1;
}

pair selectionRW(chrom data[] , int n){

    int i;
    float sum = 0 , temp_sum = 0 ;

    for(i = 0; i < n ; i++)
        sum += data[i].fit;

    float sel1 = _rand()%sum;
    temp_sum = sel1;

    for(i=0;i<n;i++){
        if(sel1 <= 0){
            p1.first = data[i];
            break;
        }else
            sel1 -= data[i].fit;
    }

    float sel2 = _rand()%sum;

    while(sel2 == temp_sum)
        sel2 = rand()%sum;

    for(i=0;i<n;i++){
        if(sel2 <= 0){
            p1.second = data[i];
            break;
        }else
            sel2 -= data[i].fit;
    }

    return p1;
}
```

7.3 Crossover :

The present study performs two-point cross-over technique, where two distinct points: x_i (i = 1, 2) are again chosen *randomly* within $1 < x_i < L$. Finally,

the heads and tails (the parts respectively before and after the cutting points) of the parents are swapped. Obviously, probability (p_c) of selecting any crossover point is : 1/L.

```
pair two_pt(chrom ch1,chrom ch2)
{
    int len=strlen(ch1.s);
    int random1=_rand()%len+1;
    int random2=_rand()%len+1;
    while(random1>random2)
    {
        random1=_rand()%len+1;
        random2=_rand()%len+1;
    }
    printf("%d %d\n",random1,random2);
    int i=0;
    chrom ch1_d,ch2_d; // duplicate chromes of ch1 & ch2
    for(i=0;i<random1;i++)
    {
        ch1_d.s[i]=ch1.s[i];
        ch2_d.s[i]=ch2.s[i];
    }
    for(i=random1;i<random2;i++)
    {
        ch2_d.s[i]=ch1.s[i];
        ch1_d.s[i]=ch2.s[i];
    }
    for(i=random2;i<len;i++)
    {
        ch1_d.s[i]=ch1.s[i];
        ch2_d.s[i]=ch2.s[i];
    }
    ch1_d.s[len]='\0';
    ch2_d.s[len]='\0';
    pair p;

p.first=ch1_d;
p.second=ch2_d;
return p;
```

7.4 Mutation :

As per the basic concept of mutation operation in GA, a zero ('0') is changed to one ('1') and vice versa for a binary coded solution. Here, single point mutation (selected at random within the length L) is performed after crossover. Thus, p_m(probability of selecting any mutation point) is 1/L . Now, the above mentioned three steps combinedly result two new individuals: say O_1 and O_2.

```
void mutate(chrom &ch){

    int len = 0;

    for(int i=0;ch.s[i] != '\0' ;i++)
        len++;

    int flip_pos = _rand()%len;

    if(ch.s[flip_pos] == '0')
        ch.s[flip_pos] = '1';
    else
        ch.s[flip_pos] = '0';

}
```

7.5 Decoding Scheme :

We know that decoding is the reverse process of encoding. The decoding strategy adopted in the present study is briefly stated below. In particular, the following steps are applied during decoding.

Step-1: Access the binary stream block by block, each of size 3 bits.

Step-2: For each block, the value *v* is computed as the decimal equivalent of that stream.

Step-3: *If* v=0, *then* place '*' at the position of the attribute corresponding to its block,
(except for the target attribute)

else if ($v > a_{max}$), *then* place the scaled value v_s = ceil(v_s/a_{max}) at that position, where a_{max} denotes the *maximum* value yield by that attribute,
else place simply the computed value at that position.

```
char* decoding(char* b)
{
    char s[60];
    char* d=s;
    while(*b!='\0')
    {
        int c=0;
        int dec=0;
        int j=0;
        for(j=0;j<3;j++)
        {
            dec=dec+(int)(*b-'0')*pow(2,c);
            b++;
            c++;
        }
        if(dec==0)
        {
            *d='0';
            d++;
        }
        else if(dec==1)
        {
            *d='1';
            d++;
        }
        else if(dec==2)
        {
            *d='2';
            d++;
        }
```

7.6 Fitness Function :

Keeping the point in mind that the heart databases are imbalanced in nature, *fitness function* for measuring fitness value of each rule (r) in rule set (I(R))

$$f(r) = ((m - n) / n_{ci}) + (1/k)$$

,where m and n represent respectively the numbers of training examples *correctly* and *incorrectly* classified by r over T_2. Further, k represents the number of *pre-conditions* present in the rule r and n_{ci} is the number of examples of class c_i present in T_2 such that the class-label of rule r is also c_i.

```
float fittness(char *s,chrom train[],int size_train)     //train_de
{
    char st[60];
    char *b=st;
    int i=0,pos=0,neg=0,clas=0,k=0,j=0;
    float f1,f2,f3;
    strcpy(st,s);
    for(i=0;i<size_train;i++)
    {
        for(j=0;train[i].s[j]!='\0';j++)
        {

            if(st[j+1]=='\0' && train[i].s[j]==st[j])
            {
                pos++;
                continue;
            }

            else if(st[j+1]=='\0' && train[i].s[j]!=st[j])
            {
                neg++;
                continue;
            }

            else if(st[j]=='0')
                continue;

            else if(train[i].s[j]=='0')
                continue;

            else if(st[j]!='0' && train[i].s[j]!='0' && train[i].s[j]==st[j])
```

```
                continue;

            else if(st[j]!='0' && train[i].s[j]!='0' && train[i].s[j]!=st[j])
                break;
        }

        for(j=0;train[i].s[j]!='\0';j++)
        {
            if(st[j+1]=='\0' && train[i].s[j]==st[j])
            {
                clas++;
                continue;
            }
        }

    }

    k=precon(st);
    f1=(float)pos-(float)neg;
    f2=f1/(float)clas;
    f3=f2+1/(float)k;

    return f3;
}
```

8.Algorithm used for the SGA model :

Variable: R: rule set where each rules in 'IF- THEN' format

I(R) : Rule set (R) in tabular form

O_1, O_2, r_1, r_2 : rule of I(R) format

ctr = 0, Max_itr // Max_itr says the *maximum number of iterations*

Step-1 Randomly selects two rules (solutions), say r_1 and r_2 from I(R)

Step-2: Apply the suggested *two-point* crossover and then *mutation* points on r_1 and r_2 to result two new *distinct* offspring: O_1 and O_2 (in decoded form).

Step-3: Check the *rule-redundancy* and *rule-conflict* cases for each of O_i (i =1, 2) with the rules present in R following the strategy presented in the next section, and take action(s) appropriately.

Step-4: If O_i (i=1, 2) is correct (*i.e.*, neither redundant nor conflict), then evaluate fitness score of each O_i (i=1, 2).

Step-5: Choose the best rule (r_{best}) based on fitness score over T_2.

Step-6: Replace the worst rule: r_{worst} with the r_{best} if possible.

Step-7: ctr =ctr +1. If (ctr < Max_itr), then goto Step-1.

8.1 Managing redundant and conflicting rules:

8.1.1 Managing Redundant Rules :

Two rules: r_1 and r_2 are identical if *min* ($|pre(r_1)|$, $|pre(r_2)|$) = *match*(pre(r_1), pre(r_2)), where $|pre(r_i)|$ results the number of *pre-conditions* (each with a numerical value) present in rule: r_i and the function: *min*(m_1, m_2) returns the *minimum* between two numbers: m_1, m_2.

Let r_1 and r_2 be two decision rules for P as follows:

- r_1: *If* (A_1=4) *and* (A_2= 2) *and* (A_4=1), *then* C=1
- r_2: *If* (A_1=4) *and* (A_4=1), *then* C=1

The number of pre-conditions present in r_1 is 3, whereas it is 2 in r_2. In fact, the rules *match* at *two* places except for (A_2=2) in r_1. Clearly, the number of *matched* pre-conditions (m) is here 2 (*i.e.*,m =2). Again, *min* ($|pre(r_1)|$, $|pre(r_2)|$)= *min*(3,2) returns 2, and it equals to m.

Hence, both the rules are *identical* but one supersedes the other. In other words, out of these two rules, one is the *super* rule of the other. It is observed that $|pre(r_2)| = 2$, and it is *less* than $|pre(r_1)| = 3$. So, rule r_2 is here treated as the *super* rule of r_1, and r_2 (instead of r_1) is well expected to be present in rule set with the aim to classify more *test* examples. Definitely, r_1 is *redundant*, and it is to be removed from rule set, R.

8.1.2 Managing Conflicting Rules :

Two rules are termed as *conflict* rules if their antecedent parts are *identical* but consequent parts (*i.e.,* class values) are *different*. For better realization, let r_1 and r_2 be two rules of P as:

- r_1: *If* $(A_1=4)$ *and* $(A_2=2)$ *and* $(A_4=1)$, *then* C=1
- r_2: *If* $(A_1=4)$ *and* $(A_2=2)$ *and* $(A_4=1)$, *then* C=2

Clearly, these two rules present an example of conflict rules, since their *antecedent* parts are same but class values are different (*i.e.*, these are C=1 and C=2 respectively). Now, if r_1 and r_2 are *conflict*, then keep the rule with maximum prediction rate on T_2 and discard the other.

```
bool is_redundant(char a[],int sz,int ruleset[][14])
{
    int flag = 0;
    for(int i=0;i<sz;i++)
    {
        char b[14];
        for(int j=0;j<13;i++)
        {
            b[j] = ruleset[i][j];
        }
        if(min(precon(a),precon(b))==match(a,b))
        {
            flag = 1;
            break;
        }
    }
    if(flag==1)return true;
    return false;
}
bool is_conflicting(char a[],char b[])
{
    for(int i=0;i<13;i++)
    {
            if(a[i]!=b[i]&&b[i]!='*')return false;
    }
    return true;
}
```

9.Code Snippets

9.1 Reading dataset from file

```
void dataReadAndEncode(chrom data[] , chrom inp[]){

    char ch ;
    FILE *myfile = fopen("Rule Set.txt" , "r");
    int cur = 0, i = 0;
    do{
        ch = fgetc(myfile);
        if(ch == '\n'){
            data[i].s[cur] = '\0';
            inp[i].s[cur] = '\0';
            data[i].fit = 0;
            inp[i].fit = 0;
            strcpy(data[i].s,encoding(data[i].s));
            printf("%s\n",data[i].s);
            i++;
            cur = 0;
        }else{
            if(ch == ',')
                continue;
            if(ch =='*')
                ch = '0';
            data[i].s[cur] = ch;
            inp[i].s[cur] = ch;
            cur++;
        }
    }while(ch != EOF);

    fclose(myfile);
}
```

9.2 Random selection operator in C

```
long long _rand(){

    static const char filename[] = "Random Number.txt";
    FILE *file = fopen(filename, "r");
    int count = 0;
    static int lineNumber = 0;
    long long rett = -1;

    if ( file != NULL ){
        char line[256]; /* or other suitable maximum line size */
        while (fgets(line, sizeof(line), file) != NULL){
            if(count == lineNumber){
                long long ret = 0 , i;
                for(i=0;line[i] != '\0';i++){
                    ret = ret*10 + (line[i]-'0');
                }
                lineNumber = (lineNumber + 1)%10000;
                rett = ret;
            }else{
                count++;
            }
        }
        fclose(file);
    }
    else{
        printf("Unable to access file\n");
        fclose(file);
    }
    return rett;
}
```

10. Samples

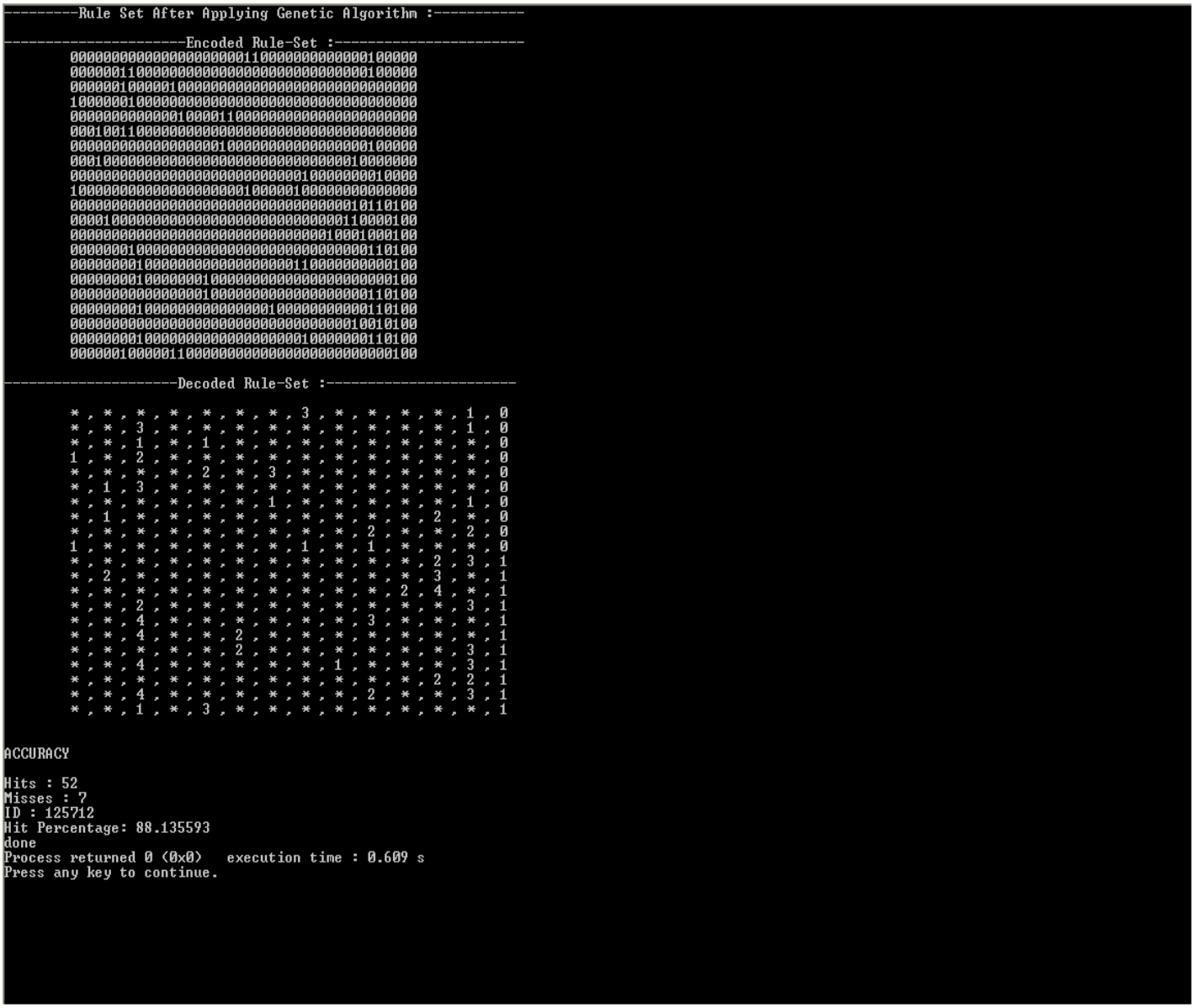
heart_data

1,2,2,2,1,2,1,3,1,1,1,1,1,0
2,1,3,2,3,2,3,2,1,1,1,2,1,0
2,2,3,2,1,2,1,3,1,1,1,2,3,0
2,2,1,2,2,2,3,2,1,3,3,1,2,0
2,2,2,2,2,1,3,1,1,1,1,1,1,0
2,1,4,1,1,1,1,1,1,1,1,3,1,0
2,2,2,2,2,1,1,3,1,1,1,1,1,0
2,2,2,2,1,2,3,1,1,1,1,1,1,0
2,2,4,2,3,1,3,2,1,1,2,1,1,0
1,1,3,2,1,1,1,3,1,1,2,1,1,0
1,1,3,2,3,2,1,3,1,1,1,1,1,0
1,1,3,2,1,1,1,2,1,1,1,2,1,0
2,1,3,2,3,1,1,3,1,1,1,2,1,0

heart_ruleset

,,*,*,*,*,*,3,*,*,*,*,1,0
,,3,*,*,*,*,*,*,*,*,*,1,0
,,1,*,1,*,*,*,*,*,*,*,*,0
1,*,2,*,*,*,*,*,*,*,*,*,*,0
,,*,*,2,*,3,*,*,*,*,*,*,0
,1,3,,*,*,*,*,*,*,*,*,*,0
,,*,*,*,*,1,*,*,*,*,*,1,0
,1,,*,*,*,*,*,*,*,*,2,*,0
,,*,*,*,*,*,*,*,2,*,*,2,0
1,*,*,*,*,*,*,1,*,1,*,*,*,0
,,*,*,*,*,*,*,*,*,*,2,3,1
,2,,*,*,*,*,*,*,*,*,3,*,1
,,*,*,*,*,*,*,*,*,2,4,*,1
,,2,*,*,*,*,*,*,*,*,*,3,1
,,4,*,*,*,*,*,*,3,*,*,*,1
,,4,*,*,2,*,*,*,*,*,*,*,1
,,*,*,*,2,*,*,*,*,*,*,3,1
,,4,*,*,*,*,*,1,*,*,*,3,1
,,*,*,*,*,*,*,*,*,*,2,2,1
,,4,*,*,*,*,*,*,2,*,*,3,1
,,1,*,3,*,*,*,*,*,*,*,*,1
2,*,4,*,3,*,*,*,*,*,*,1,*,1

11. Result :


---------Rule Set After Applying Genetic Algorithm :----------
----------Encoded Rule-Set :----------
00000000000000000000000110000000000000100000
00000011000000000000000000000000000000100000
00000010000010000000000000000000000000000000
10000001000000000000000000000000000000000000
00000000000001000011000000000000000000000000
00010011000000000000000000000000000000000000
00000000000000000010000000000000000100000
00010000000000000000000000000000010000000
00000000000000000000000000000010000000010000
10000000000000000000001000001000000000000000
00000000000000000000000000000000010110100
00001000000000000000000000000000011000100
00000000000000000000000000000010001000100
00000001000000000000000000000000010100
00000000100000000000000000011000000000100
00000001000000010000000000000000000100
00000000000000001000000000000000000110100
00000000100000000000001000000000110100
00000000000000000000000000000010010100
00000000100000000000000000010000000110100
00000010000011000000000000000000000000100
Decoded Rule-Set :----------
* , * , * , * , * , * , 3 , * , * , * , * , 1 , 0
* , * , 3 , * , * , * , * , * , * , * , * , 1 , 0
* , * , 1 , * , 1 , * , * , * , * , * , * , * , 0
1 , * , 2 , * , * , * , * , * , * , * , * , * , 0
* , * , * , * , 2 , * , 3 , * , * , * , * , * , 0
* , 1 , 3 , * , * , * , * , * , * , * , * , * , 0
* , * , * , * , * , * , 1 , * , * , * , * , 1 , 0
* , 1 , * , * , * , * , * , * , * , * , 2 , * , 0
* , * , * , * , * , * , * , * , * , 2 , * , 2 , 0
1 , * , * , * , * , * , 1 , * , 1 , * , * , * , 0
* , * , * , * , * , * , * , * , * , * , 2 , 3 , 1
* , 2 , * , * , * , * , * , * , * , * , 3 , * , 1
* , * , * , * , * , * , * , * , * , 2 , 4 , * , 1
* , * , 2 , * , * , * , * , * , * , * , * , 3 , 1
* , * , 4 , * , * , * , * , * , 3 , * , * , * , 1
* , * , 4 , * , * , 2 , * , * , * , * , * , * , 1
* , * , * , * , * , 2 , * , * , * , * , * , 3 , 1
* , * , 4 , * , * , * , * , * , 1 , * , * , 3 , 1
* , * , * , * , * , * , * , * , * , * , 2 , 2 , 1
* , * , 4 , * , * , * , * , * , * , 2 , * , 3 , 1
* , * , 1 , * , 3 , * , * , * , * , * , * , * , 1
ACCURACY
Hits : 52
Misses : 7
ID : 125712
Hit Percentage: 88.135593
done
Process returned 0 (0x0)   execution time : 0.609 s
Press any key to continue.

12. Conclusion :

By applying genetic algorithm on the classified dataset, an accuracy of **88%** was achieved, compared to **less than 70%** previously.
Future prospective : Moving forward with the same project, we will try achieving parallelism and thus enhancing the efficiency multifolds.

13. Bibliography :

[1] **Jadzia Cendrowska**, C/O The Faculty of Mathematics, The Open University, Walton Hall, Milton Keynes, MK7 6AA, U.K. (29/05/1987)

[2] **Issa Qabajeh**, **Francisco Chiclana**, Centre for Computational Intelligence, De Montfort University, UK; **Fadi Thabtah**, EBusiness Dept, CUD, Dubai (UAE) : A Classification Rules Mining Method based on Dynamic Rules' Frequency (2015)

[3] Article on Malnutrition Detection and Management System

[4] **J. Han, M. Kamber**, Data Mining: Concepts and Techniques, Elsevier Inc..

[5] **I. H. Witten and E. Frank**, Data Mining: Practical Machine Learning Tools and Techniques, 2nd Rule-Based Classification Edition, Elsevier Inc., 2005

[6] **Marek Obitko**: Genetic Algorithm Library, https://www.codeproject.com

[7] **Johannes Furnkranz**, Separate-and-Conquer Rule Learning, Austrian Research Institute of Artificial Intelligence, 3-54, 1999