ASSIGNMENT-4

Nandhini R Baladhandapani

Software Cost Estimation

Overview

A manufacturer can determine if the project or product is cost effective to manufacturer prior to the manufacturing process. There are various alternatives to manufacturing and cost estimating can determine the economical feasibility of manufacturing the project or product by different methods. A manufacturer can determine the selling price before manufacturing begins. This may or may not be a good idea because the product still needs to be priced at a value the consumer will purchase the item. This cost estimation on software is called as Software Cost Estimation.

Key Factor for the Software Cost Estimation is the requirements needed for manufacturing the item. The estimation are based on the size, domain, programming languages, expertise of development team.

Genetic Programming

GP is a set of instructions and a fitness function to measure how well a computer has performed a task. Genetic programming (GP) is a domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done. The user should know or specify the form or structure of solutions in advance

There are different approaches in making predictions:

Algorithmic-COCOMO

The COCOMO model uses a basic regression formula with parameters that are derived from historical project data and current as well as future project characteristics.

Historical data

The statistics, Expert judgement (Expert judgment techniques involve consulting with software cost estimation expert or a group of the experts to use their experience and understanding of the proposed project to arrive at an estimate of its cost), Machine Learning.

TinyGP:

TinyGP is a highly optimised GP system of the Genetic and Evolutionary Computation Conference (GECCO).

TinyGP is a symbolic regression system with the following characteristics:

Terminal set: It consists External inputs (x, y etc.) Numerical constants (0, 42, 3.14159 etc.) 0-arity functions (rand() etc.). Function Set: The set consists of some operations are addition, subtraction, multiplication, division, terminal. Fitness Measure: The fitness measure is calculated with the distance between the predicted and known effort value.Termination Criterion: The criteria is depended on the number of generations.

TinyGP problem:

I have tried tinyGP code from the link http://cswww.essex.ac.uk/staff/rpoli/TinyGP/. This is my output by using TinyGP problem.

```
-- TINY GP (Java version) --
SEED=-1
MAX_LEN=10000
POPSIZE=100000
DEPTH=5
CROSSOVER_PROB=0.9
PMUT_PER_NODE=0.05
MIN_RANDOM=-5.0
MAX_RANDOM=5.0
GENERATIONS=100
TSIZE=2
--------------------------------
Generation=0 Avg Fitness=4.6309192992837414E7 Best Fitness=1398.0897106207212 Avg Size=10.95096
```

ASSIGNMENT-4

Nandhini R Baladhandapani

Best Individual: ((4.880752728975795 - (((2.775519019992373 / -2.3070420617973877) * 0.9311673230283475) - (X6 + (-1.9729714703979848 * -4.752377591498008)))) / (-0.03806733626108283 + ((2.700299642985229 + (3.031232941871602 * X1)) - ((-1.0988683049199044 * -1.676667206085619) + -2.1809255776871304))))
Generation=1 Avg Fitness=117082.2808730116 Best Fitness=1270.0405021344593 Avg Size=11.5752
Best Individual: (((-2.8843060961117537 - (X3 * (2.1127687108125306 + 4.859139436241174))) - (((4.62743280685949 * X4) - -0.4606102922069857) - 1.0193647336774267)) / ((-2.8035170616588667 * 2.513939001422817) + (((-4.9955398995226306 / -2.3738976457687375) - 3.1251912473835777) / ((-4.625461399489113 / -3.542818051868477) * (X2 - 3.2198523079170887)))))
Generation=2 Avg Fitness=5544.685875305455 Best Fitness=1270.0405021344593 Avg Size=14.05418
Best Individual: (((-2.8843060961117537 - (X3 * (2.1127687108125306 + 4.859139436241174))) - (((4.62743280685949 * X4) - -0.4606102922069857) - 1.0193647336774267)) / ((-2.8035170616588667 * 2.513939001422817) + (((-4.9955398995226306 / -2.3738976457687375) - 3.1251912473835777) / ((-4.625461399489113 / -3.542818051868477) * (X2 - 3.2198523079170887)))))
Generation=3 Avg Fitness=9269.135295541975 Best Fitness=1170.9429871626644 Avg Size=16.38358
Best Individual: ((X4 + ((4.859139436241174 + ((-3.4871872205476007 + ((X4 + 2.1127687108125306) - -4.752377591498008)) - -4.9955398995226306)) * (-0.5513247404319461 / ((2.5147232336702166 + 4.93132245993748) + (4.372509575541088 * (1.059141648080768 - X2)))))) + 2.270346741087458)
Generation=4 Avg Fitness=6566.471287545069 Best Fitness=1170.9429871626644 Avg Size=17.58322
Best Individual: ((X4 + ((4.859139436241174 + ((-3.4871872205476007 + ((X4 + 2.1127687108125306) - -4.752377591498008)) - -4.9955398995226306)) * (-0.5513247404319461 / ((2.5147232336702166 + 4.93132245993748) + (4.372509575541088 * (1.059141648080768 - X2)))))) + 2.270346741087458)
Generation=5 Avg Fitness=7789.772717987868 Best Fitness=1170.9429871626644 Avg Size=19.57312
Best Individual: ((X4 + ((4.859139436241174 + ((-3.4871872205476007 + ((X4 + 2.1127687108125306) - -4.752377591498008)) - -4.9955398995226306)) * (-0.5513247404319461 / ((2.5147232336702166 + 4.93132245993748) + (4.372509575541088 * (1.059141648080768 - X2)))))) + 2.270346741087458)
Generation=6 Avg Fitness=36786.32201088987 Best Fitness=891.2927622874103 Avg Size=22.84676
Best Individual: (2.700299642985229 + (X4 + ((0.5318161458915753 - ((((((-0.03806733626108283 - 4.657918213747331) * -2.3738976457687375) + (4.64409378342658 / (((X5 - -0.8608170401354309) / 4.880752728975795) - (((-1.738539733206018 / (X5 / 4.859139436241174)) * (1.8234771075647203 + -3.5880048244561094)) * (2.25820335792943 + 4.93132245993748))))) - (4.93132245993748 - (X3 - -3.542818051868477))) + (0.47397372500964874 * ((1.5356965237338036 - -1.0988683049199044) + (3.031232941871602 / -4.830414941916759)))) / ((-3.899522699638908 + (4.520536171398568 - X2)) + 2.25820335792943))) * (2.5147232336702166 * 1.252649260297626))))
Generation=7 Avg Fitness=22262.232299496212 Best Fitness=891.1911051999328 Avg Size=27.12286
Best Individual: (2.700299642985229 + (X4 + ((0.5318161458915753 - ((((((-0.03806733626108283 - 4.657918213747331) * -2.3738976457687375) + (4.64409378342658 / (((X5 - -0.8608170401354309) / 4.880752728975795) - -1.8390759322160308))) - (4.93132245993748 - (X3 - -3.542818051868477))) + (0.47397372500964874 * ((1.5356965237338036 - -1.0988683049199044) + (3.031232941871602 / -4.830414941916759)))) / ((-3.899522699638908 + (4.520536171398568 - X2)) + 2.25820335792943))) * (2.5147232336702166 * 1.252649260297626))))
Generation=8 Avg Fitness=23933.463788589645 Best Fitness=763.9856815004844 Avg Size=32.84732
Best Individual: (2.775519019992373 + ((((((-2.887041260799721 - -4.830414941916759) / (-4.757521998161124 * (((2.139397381863853 + (2.513939001422817 * 0.4600702413755986)) - 0.26069919665784536) * -4.757521998161124))) + X5) / (2.043432823472732 + ((4.64409378342658 / (-3.3014239250298836 - ((2.25820335792943 - ((0.7722718669498976 - ((X2 + -3.0865007507811395) * ((((-2.1809255776871304 * (4.62743280685949 / -1.0988683049199044)) + (4.520536171398568 * 2.3328349014115517)) + 1.3063193735124585) + (-3.899522699638908 / -4.00635167170377)))) - -3.49328681532226)) / 2.1127687108125306))) + 3.2198523079170887))) * 1.0011374244164681) - -2.3738976457687375))
Generation=9 Avg Fitness=30931.591671464605 Best Fitness=742.6685384559103 Avg Size=40.09512
Best Individual: (2.775519019992373 + ((((-0.8608170401354309 + X5) / (2.043432823472732 + ((4.64409378342658 / (-3.3014239250298836 - ((2.25820335792943 - ((0.7722718669498976 - ((X2 + (2.152156456657134 - 0.8404443085864921)) * (((((X4 - ((((-3.0865007507811395 - X6) - (-1.738539733206018 + (((1.0011374244164681 + X4) / -4.1181045386908375) - (-4.752377591498008 + 3.2198523079170887)))) + ((4.657918213747331 / 0.21761876746571573) / 1.0193647336774267)) / (-3.5880048244561094 - ((-0.5513247404319461 + (4.859139436241174 / -3.446081842392016)) + 3.6704297031790176)))) / -3.899522699638908) + (4.520536171398568 * 2.3328349014115517)) + 1.3063193735124585) + (-3.899522699638908 / -4.00635167170377)))) - -3.49328681532226)) / -3.446081842392016))) + 3.072151322551674))) * 1.0011374244164681) - -2.3738976457687375))
Generation=10 Avg Fitness=65960.58452967372 Best Fitness=730.1362813160029 Avg Size=48.29514
Best Individual: ((((2.1127687108125306 + ((4.880752728975795 / (((-4.2590245336803525 / (1.9920389621954806 / 1.8519364385466375)) + (-1.4105985775497287 / ((3.1127341188803808 - (1.1072376188002364 + 2.139397381863853)) - 4.64409378342658))) * 0.8404443085864921)) + (X1 - -4.757521998161124))) + X5) / (2.043432823472732 + ((4.64409378342658 / (((X2 * (4.93132245993748 - (-1.9729714703979848 * -4.275197324544335))) + (-1.0988683049199044 - 4.208335448061069)) - (3.6704297031790176 * -3.8961592832558622))) + 3.072151322551674))) * 1.0011374244164681)

ASSIGNMENT-4

Nandhini R Baladhandapani

Generation=11 Avg Fitness=32222.11617157459 Best Fitness=728.3038046219456 Avg Size=57.91442
Best Individual: ((((2.1127687108125306 + ((4.880752728975795 / (((-4.2590245336803525 / (1.9920389621954806 / 1.8519364385466375)) + (-1.4105985775497287 / ((3.1127341188803808 - (1.1072376188002364 + 2.139397381863853)) - 4.64409378342658))) * -3.5880048244561094)) + 1.587055663767858)) + X5) / (2.043432823472732 + ((4.64409378342658 / (((X2 * (4.93132245993748 - (-1.9729714703979848 * -4.275197324544335))) + (-1.0988683049199044 - 4.208335448061069)) - (3.6704297031790176 * -3.8961592832558622))) + 3.072151322551674))) * 1.0011374244164681)
Generation=12 Avg Fitness=34977.12899489811 Best Fitness=724.4893776430986 Avg Size=70.03984
Best Individual: ((((2.1127687108125306 + -4.757521998161124) + X5) / (2.043432823472732 + ((4.64409378342658 / (((X2 * (4.93132245993748 - (-1.9729714703979848 * -4.275197324544335))) + (-1.0988683049199044 - 4.208335448061069)) - (3.6704297031790176 * -3.8961592832558622))) + 3.072151322551674))) * 1.0011374244164681)

From the tiny_gp code, I have understood that tiny_gp will take more iterations, and not completely functional algorithm. It is having more limitations in terms of functions. The above results shows the average fitness value for each individual.

Work of Tiny_gp:

The system is based on the linear representation for trees, which effectively corresponds to listing the primitives in prefix notation but without any brackets. A program is simply a vector of characters. The operators used are sub tree crossover and point mutation. The selection of the crossover points is performed at random with uniform probability above. The code uses recursion for the creation of the initial population (grow), for the identification of the sub tree rooted at a particular crossover point (traverse), for program execution (run), and for printing programs (print individuals). A small number of global variables have been used.

JGAP

JGAP supports to do quite a bit of work for the developer, but allows you to extend things if you so choose. The example from the genetic programming is mentioned below.

For example, there is an two inputs and output. The point of the problem is to discover what equation will return the result given the 2 input values. The answer is x^2+2y+3x+5 where x is input 1 and y is the input 2 and that is the output.

Input 1	Input 2	Output
26	35	829
8	24	141
20	1	467
33	11	1215
37	16	1517

The JGAP document lists 4 steps involved in creating a GP using JGAP:

1. Create a GP Configuration Object
2. Create an initial Genotype
3. Evolve the population
4. Optionally implement custom functions and terminals (a mutable static number)

CREATE A GP CONFIGURATION OBJECT

```
GPConfiguration config = getGPConfiguration();
//THE VARIABLES ARE USED ACCORDING TO THE ATTRIBUTES IN THE FILE
    _xVariable = Variable.create(config, "X", CommandGene.IntegerClass);
    _yVariable = Variable.create(config, "Y", CommandGene.IntegerClass);

    config.setGPFitnessEvaluator(new DeltaGPFitnessEvaluator());
    config.setMaxInitDepth(4);
    config.setPopulationSize(1000);
    config.setMaxCrossoverDepth(8);
```

```
    config.setFitnessFunction(new SimpleMathTestFitnessFunction(INPUT_1, INPUT_2, OUTPUT, _xVariable,
_yVariable));
    config.setStrictProgramCreation(true);
```

CREATE A INTIAL GENOTYPE

```
CommandGene[][] nodeSets = {
        {
//  CREATE A INITIAL GENOTYPE FOR EVERY FUNCTION
            _xVariable,
            _yVariable,
            new Add(config, CommandGene.IntegerClass),
            new Multiply(config, CommandGene.IntegerClass),
            new Terminal(config, CommandGene.IntegerClass, 0.0, 10.0, true)
        }
    };
```

EVOLVE THE FUNCTION

```
GPGenotype gp = problem.create();
    gp.setVerboseOutput(true);
    gp.evolve(30);


    System.out.println("Formulaiscover: x^2 + 2y + 3x + 5");
    gp.outputSolution(gp.getAllTimeBest());
```

By using all this codes, I have generated an GP program for the two files. I have tried to implement for Albercht and kemerer files. I have used kemerer file from the http://openscience.us/repo/effort/. and saved the file with the extension .dat. In the file, I have changed the attributes from 8 to 7. It uses the functions are addition, subtraction, multiplication and division and it calculates the fitness value from the attributes(inputs).

The kemerer file with all alphabetical data and the numbers(attributes). Looks like:

```
@relation kemerer

@attribute ID numeric
@attribute Language numeric
@attribute Hardware numeric
@attribute Duration numeric
@attribute KSLOC numeric
@attribute AdjFP numeric
@attribute RAWFP numeric
@attribute EffortMM numeric

@data
1,1,1,17,253.6,1217.1,1010,287
2,1,2,7,40.5,507.3,457,82.5
3,1,3,15,450,2306.8,2284,1107.31
4,1,1,18,214.4,788.5,881,86.9
5,1,2,13,449.9,1337.6,1583,336.3
6,1,4,5,50,421.3,411,84
7,2,4,5,43,99.9,97,23.2
```

ASSIGNMENT-4
Nandhini R Baladhandapani

8,1,2,11,200,993,998,130.3
9,1,1,14,289,1592.9,1554,116
10,1,1,5,39,240,250,72
11,1,1,13,254.2,1611,1603,258.7
12,1,5,31,128.6,789,724,230.7
13,1,6,20,161.4,690.9,705,157
14,1,1,26,164.8,1347.5,1375,246.9
15,3,1,14,60.2,1044.3,976,69.9

I have ignored all the alphabetical data, first column(which is a line number), commas and I have used the file for implementing JGAP. I have declared the attributes as the inputs from the file and created the configuration object. By using the object, I have implemented the mathematical function.

Albercht file:

@relation albrecht

@attribute Input numeric
@attribute Output numeric
@attribute Inquiry numeric
@attribute File numeric
@attribute FPAdj numeric
@attribute RawFPcounts numeric
@attribute AdjFP numeric
@attribute Effort numeric

@data
25,150,75,60,1,1750,1750,102.4
193,98,70,36,1,1902,1902,105.2
70,27,0,12,0.8,535,428,11.1
40,60,20,12,1.15,660,759,21.1
10,69,1,9,0.9,478.89,431,28.8
13,19,0,23,0.75,377.33,283,10
34,14,0,5,0.8,256.25,205,8
17,17,15,5,1.1,262.73,289,4.9
45,64,14,16,0.95,715.79,680,12.9
40,60,20,15,1.15,690.43,794,19
41,27,29,5,1.1,465.45,512,10.8
33,17,8,5,0.75,298.67,224,2.9
28,41,16,11,0.85,490.59,417,7.5
43,40,20,35,0.85,802.35,682,12
7,12,13,8,0.95,220,209,4.1
28,38,24,9,1.05,487.62,512,15.8
42,57,12,5,1.1,550.91,606,18.3
27,20,24,6,1.1,363.64,400,8.9
48,66,13,50,1.15,1073.91,1235,38.1
69,112,21,39,1.2,1310,1572,61.2
25,28,4,22,1.05,476.19,500,3.6
61,68,0,11,1,694,694,11.8
15,15,6,3,1.05,189.52,199,0.5
12,15,0,15,0.95,273.68,260,6.1

I have ignored the alphabetical data and commas and I have implemented the file in the JGAP. In JGAP, I have stored all the attributes as inputs in an variable names. I have created the configuration object and using the object, it sets the values for config.setGPFitnessEvaluator

```
config.setMaxInitDepth(4);
config.setPopulationSize(1000);
config.setMaxCrossoverDepth(8);
config.setFitnessFunction(new jgap_fitness_function(InputofA, InputofB,InputofC,InputofD,InputofE,InputofF, OUTPUT, a, b,c,d,e,f))
config.setStrictProgramCreation(true);
```

I have set all these values by initialising with some random values. I have parsed by deleting the header in the file and applying the code for JGAP. I have stored all the values in the list and send it to the function. It calculates the fitness function by the formula longResult += Math.*abs*(datavalues - output.get());

ASSIGNMENT-4
Nandhini R Baladhandapani

Output for the JGAP

(Kemerer file): I have printed only the 3 iterations.

File found!
DATA1: [1.0, 1.0, 1.0, 1.0, 1.0, 2.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 3.0]
DATA2: [2.0, 3.0, 1.0, 2.0, 4.0, 4.0, 2.0, 1.0, 1.0, 1.0, 5.0, 6.0, 1.0, 1.0]
DATA3: [7.0, 15.0, 18.0, 13.0, 5.0, 5.0, 11.0, 14.0, 5.0, 13.0, 31.0, 20.0, 26.0, 14.0]
DATA4: [40.5, 450.0, 214.4, 449.9, 50.0, 43.0, 200.0, 289.0, 39.0, 254.2, 128.6, 161.4, 164.8, 60.2]
DATA5: [507.3, 2306.8, 788.5, 1337.6, 421.3, 99.9, 993.0, 1592.9, 240.0, 1611.0, 789.0, 690.9, 1347.5, 1044.3]
DATA6: [457.0, 2284.0, 881.0, 1583.0, 411.0, 97.0, 998.0, 1554.0, 250.0, 1603.0, 724.0, 705.0, 1375.0, 976.0]
OUTPUT: [82.5, 1107.31, 86.9, 336.3, 84.0, 23.2, 130.3, 116.0, 72.0, 258.7, 230.7, 157.0, 246.9, 69.9]

[16:41:55] INFO GPGenotype - Creating initial population
[16:41:55] INFO GPGenotype - Mem free: 25.2 MB
[16:41:55] INFO GPPopulation - Prototype program set
[16:41:55] INFO GPGenotype - Mem free after creating population: 25.2 MB
[16:41:56] INFO GPGenotype - Your configuration does not contain unused commands, this is good
Best Random solution :
[16:41:56] INFO GPGenotype - Best solution fitness: 1309.0
[16:41:56] INFO GPGenotype - Best solution: (E / 6.0) / A
[16:41:56] INFO GPGenotype - Depth of chrom: 2
[16:41:56] INFO GPGenotype - Best solution fitness: 1309.0
[16:41:56] INFO GPGenotype - Best solution: (E / 6.0) / A
[16:41:56] INFO GPGenotype - Depth of chrom: 2
Random COMPLETE
Randomly picked solution :
[16:41:56] INFO GPGenotype - Best solution fitness: 1309.0
[16:41:56] INFO GPGenotype - Best solution: (E / 6.0) / A
[16:41:56] INFO GPGenotype - Depth of chrom: 2
Random COMPLETE
ITERATIONS for genetic:0
Genetic solution :
[16:41:56] INFO GPGenotype - Evolving generation 0, memory free: 22.6 MB
[16:41:56] INFO GPGenotype - Best solution fitness: 1306.0
[16:41:56] INFO GPGenotype - Best solution: ((D * E) / (D + E)) + ((8.0 / A) + C)
[16:41:56] INFO GPGenotype - Depth of chrom: 3
[16:41:56] INFO GPGenotype - Best solution fitness: 1269.0
[16:41:56] INFO GPGenotype - Best solution: ((E / 6.0) / A) + (B + B)
[16:41:56] INFO GPGenotype - Depth of chrom: 3
[16:41:57] INFO GPGenotype - Best solution fitness: 1249.0
[16:41:57] INFO GPGenotype - Best solution: (B + (((E / 6.0) / A) + (B + B))) + (B / D)
[16:41:57] INFO GPGenotype - Depth of chrom: 5
[16:41:57] INFO GPGenotype - Best solution fitness: 1245.0
[16:41:57] INFO GPGenotype - Best solution: (B + (((E / 6.0) / A) + (B + B))) + (D / D)
[16:41:57] INFO GPGenotype - Depth of chrom: 5
[16:41:57] INFO GPGenotype - Best solution fitness: 1234.0
[16:41:57] INFO GPGenotype - Best solution: B + ((B + (((E / 6.0) / A) + (B + B))) + (D / D))
[16:41:57] INFO GPGenotype - Depth of chrom: 6
[16:41:58] INFO GPGenotype - Evolving generation 25, memory free: 12.2 MB
[16:41:58] INFO GPGenotype - Best solution fitness: 1223.0
[16:41:58] INFO GPGenotype - Best solution: (D + F) / (6.0 * A)
[16:41:58] INFO GPGenotype - Depth of chrom: 2
[16:41:59] INFO GPGenotype - Best solution fitness: 1219.0
[16:41:59] INFO GPGenotype - Best solution: (A + (B + (B + ((E / 6.0) / A)))) + C
[16:41:59] INFO GPGenotype - Depth of chrom: 6
[16:42:00] INFO GPGenotype - Evolving generation 50, memory free: 5.5 MB
[16:42:00] INFO GPGenotype - Best solution fitness: 1218.0
[16:42:00] INFO GPGenotype - Best solution: B + (B + ((A + (((E / 6.0) / A) - A)) + C))
[16:42:00] INFO GPGenotype - Depth of chrom: 7
[16:42:01] INFO GPGenotype - Best solution fitness: 1207.0
[16:42:01] INFO GPGenotype - Best solution: A + (B + ((A + (((C + (E / 7.0)) / A) - A)) + C))
[16:42:01] INFO GPGenotype - Depth of chrom: 8
[16:42:02] INFO GPGenotype - Best solution fitness: 1190.0
[16:42:02] INFO GPGenotype - Best solution: B + ((A + (B + (A + (((C + (E / 7.0)) / A) - A)))) + C)
[16:42:02] INFO GPGenotype - Depth of chrom: 9

ASSIGNMENT-4
Nandhini R Baladhandapani

[16:42:02] INFO GPGenotype - Evolving generation 75, memory free: 12.9 MB
[16:42:02] INFO GPGenotype - Best solution fitness: 1189.0
[16:42:02] INFO GPGenotype - Best solution: C + (A + (A + (B + (B + (B + ((C + (E / 7.0)) / A))))))
[16:42:02] INFO GPGenotype - Depth of chrom: 9
[16:42:03] INFO GPGenotype - Best solution fitness: 1189.0
[16:42:03] INFO GPGenotype - Best solution: C + (A + (A + (B + (B + (B + ((C + (E / 7.0)) / A))))))
[16:42:03] INFO GPGenotype - Depth of chrom: 9
Genetic COMPLETE
ITERATIONS for genetic:1
Genetic solution :
[16:42:03] INFO GPGenotype - Evolving generation 90, memory free: 2.2 MB
[16:42:04] INFO GPGenotype - Best solution fitness: 1182.0
[16:42:04] INFO GPGenotype - Best solution: C + (A + (B + (B + (B + ((C + (E / 7.0)) / A)))))
[16:42:04] INFO GPGenotype - Depth of chrom: 8
[16:42:04] INFO GPGenotype - Evolving generation 115, memory free: 11.5 MB
[16:42:04] INFO GPGenotype - Best solution fitness: 1180.0
[16:42:04] INFO GPGenotype - Best solution: A + (B + (B + (B + ((C + (C + (E / 7.0))) / A))))
[16:42:04] INFO GPGenotype - Depth of chrom: 8
[16:42:05] INFO GPGenotype - Best solution fitness: 1177.0
[16:42:05] INFO GPGenotype - Best solution: (C + (B + (B + (B + ((C + (E / 7.0)) / A))))) - A
[16:42:05] INFO GPGenotype - Depth of chrom: 8
[16:42:05] INFO GPGenotype - Best solution fitness: 1077.0
[16:42:05] INFO GPGenotype - Best solution: (((C + (E / 7.0)) - A) / (8.0 / E)) / ((B + C) * (C / B))
[16:42:05] INFO GPGenotype - Depth of chrom: 5
[16:42:06] INFO GPGenotype - Best solution fitness: 1069.0
[16:42:06] INFO GPGenotype - Best solution: ((((E / 4.0) / 8.0) / (8.0 / E)) / (3.0 * (C / B))) / A
[16:42:06] INFO GPGenotype - Depth of chrom: 5
[16:42:06] INFO GPGenotype - Best solution fitness: 815.0
[16:42:06] INFO GPGenotype - Best solution: B + ((((E / 4.0) / 8.0) / (8.0 / E)) / (4.0 * (C / B)))
[16:42:06] INFO GPGenotype - Depth of chrom: 5
[16:42:06] INFO GPGenotype - Evolving generation 140, memory free: 9.6 MB
[16:42:06] INFO GPGenotype - Best solution fitness: 742.0
[16:42:06] INFO GPGenotype - Best solution: C + (((((E / 4.0) / A) / 8.0) / (8.0 / E)) / (4.0 * (C / B)))
[16:42:06] INFO GPGenotype - Depth of chrom: 6
[16:42:08] INFO GPGenotype - Evolving generation 165, memory free: 11.2 MB
[16:42:09] INFO GPGenotype - Best solution fitness: 742.0
[16:42:09] INFO GPGenotype - Best solution: C + (((((E / 4.0) / A) / 8.0) / (8.0 / E)) / (4.0 * (C / B)))
[16:42:09] INFO GPGenotype - Depth of chrom: 6
Genetic COMPLETE
ITERATIONS for genetic:2
Genetic solution :
[16:42:09] INFO GPGenotype - Evolving generation 180, memory free: 8.8 MB
[16:42:09] INFO GPGenotype - Best solution fitness: 656.0
[16:42:09] INFO GPGenotype - Best solution: C + ((((F / 5.0) / 8.0) / (6.0 / E)) / (4.0 * (C / B)))
[16:42:09] INFO GPGenotype - Depth of chrom: 5
[16:42:10] INFO GPGenotype - Evolving generation 205, memory free: 14.0 MB
[16:42:14] INFO GPGenotype - Evolving generation 230, memory free: 8.3 MB
[16:42:14] INFO GPGenotype - Best solution fitness: 643.0
[16:42:14] INFO GPGenotype - Best solution: C + (C + ((((F / 5.0) / 8.0) / (6.0 / E)) / (4.0 * (C / B))))
[16:42:14] INFO GPGenotype - Depth of chrom: 6
[16:42:15] INFO GPGenotype - Evolving generation 255, memory free: 16.3 MB
[16:42:16] INFO GPGenotype - Best solution fitness: 643.0
[16:42:16] INFO GPGenotype - Best solution: C + (C + ((((F / 5.0) / 8.0) / (6.0 / E)) / (4.0 * (C / B))))
[16:42:16] INFO GPGenotype - Depth of chrom: 6
Genetic COMPLETE

(Albercht file): I have printed only the 3 iterations

File found!
DATA1: [150.0, 98.0, 27.0, 60.0, 69.0, 19.0, 14.0, 17.0, 64.0, 60.0, 27.0, 17.0, 41.0, 40.0, 12.0, 38.0, 57.0, 20.0, 66.0, 112.0, 28.0, 68.0, 15.0, 15.0]
DATA2: [75.0, 70.0, 0.0, 20.0, 1.0, 0.0, 0.0, 15.0, 14.0, 20.0, 29.0, 8.0, 16.0, 20.0, 13.0, 24.0, 12.0, 24.0, 13.0, 21.0, 4.0, 0.0, 6.0, 0.0]
DATA3: [60.0, 36.0, 12.0, 12.0, 9.0, 23.0, 5.0, 5.0, 16.0, 15.0, 5.0, 5.0, 11.0, 35.0, 8.0, 9.0, 5.0, 6.0, 50.0, 39.0, 22.0, 11.0, 3.0, 15.0]
DATA4: [1.0, 1.0, 0.8, 1.15, 0.9, 0.75, 0.8, 1.1, 0.95, 1.15, 1.1, 0.75, 0.85, 0.85, 0.95, 1.05, 1.1, 1.1, 1.15, 1.2, 1.05, 1.0, 1.05, 0.95]

ASSIGNMENT-4

Nandhini R Baladhandapani

DATA5: [1750.0, 1902.0, 535.0, 660.0, 478.89, 377.33, 256.25, 262.73, 715.79, 690.43, 465.45, 298.67, 490.59, 802.35, 220.0, 487.62, 550.91, 363.64, 1073.91, 1310.0, 476.19, 694.0, 189.52, 273.68]
DATA6: [1750.0, 1902.0, 428.0, 759.0, 431.0, 283.0, 205.0, 289.0, 680.0, 794.0, 512.0, 224.0, 417.0, 682.0, 209.0, 512.0, 606.0, 400.0, 1235.0, 1572.0, 500.0, 694.0, 199.0, 260.0]
OUTPUT: [102.4, 105.2, 11.1, 21.1, 28.8, 10.0, 8.0, 4.9, 12.9, 19.0, 10.8, 2.9, 7.5, 12.0, 4.1, 15.8, 18.3, 8.9, 38.1, 61.2, 3.6, 11.8, 0.5, 6.1]

[16:51:20] INFO GPGenotype - Creating initial population
[16:51:20] INFO GPGenotype - Mem free: 25.2 MB
[16:51:20] INFO GPPopulation - Prototype program set
[16:51:20] INFO GPGenotype - Mem free after creating population: 25.2 MB
[16:51:20] INFO GPGenotype - Your configuration does not contain unused commands, this is good
Best Random solution :
[16:51:20] INFO GPGenotype - Best solution fitness: 223.0
[16:51:20] INFO GPGenotype - Best solution: (C + A) / 4.0
[16:51:20] INFO GPGenotype - Depth of chrom: 2
[16:51:20] INFO GPGenotype - Best solution fitness: 223.0
[16:51:20] INFO GPGenotype - Best solution: (C + A) / 4.0
[16:51:20] INFO GPGenotype - Depth of chrom: 2
Random COMPLETE
Randomly picked solution :
[16:51:20] INFO GPGenotype - Best solution fitness: 223.0
[16:51:20] INFO GPGenotype - Best solution: (C + A) / 4.0
[16:51:20] INFO GPGenotype - Depth of chrom: 2
Random COMPLETE
ITERATIONS for genetic:0
Genetic solution :
[16:51:20] INFO GPGenotype - Evolving generation 0, memory free: 22.6 MB
[16:51:21] INFO GPGenotype - Best solution fitness: 221.0
[16:51:21] INFO GPGenotype - Best solution: A / 2.0
[16:51:21] INFO GPGenotype - Depth of chrom: 1
[16:51:22] INFO GPGenotype - Best solution fitness: 207.0
[16:51:22] INFO GPGenotype - Best solution: (C + (C + A)) / 4.0
[16:51:22] INFO GPGenotype - Depth of chrom: 3
[16:51:22] INFO GPGenotype - Best solution fitness: 205.0
[16:51:22] INFO GPGenotype - Best solution: (D * 2.0) + (D * ((C + A) / 4.0))
[16:51:22] INFO GPGenotype - Depth of chrom: 4
[16:51:23] INFO GPGenotype - Best solution fitness: 202.0
[16:51:23] INFO GPGenotype - Best solution: ((C + A) + A) / 6.0
[16:51:23] INFO GPGenotype - Depth of chrom: 3
[16:51:23] INFO GPGenotype - Best solution fitness: 198.0
[16:51:23] INFO GPGenotype - Best solution: ((C + (C + A)) + A) / 6.0
[16:51:23] INFO GPGenotype - Depth of chrom: 4
[16:51:25] INFO GPGenotype - Best solution fitness: 196.0
[16:51:25] INFO GPGenotype - Best solution: D * (((C + A) + (C * D)) / 4.0)
[16:51:25] INFO GPGenotype - Depth of chrom: 4
[16:51:25] INFO GPGenotype - Best solution fitness: 194.0
[16:51:25] INFO GPGenotype - Best solution: D * (((D * (D * 3.0)) / 6.0) + (D * (((C + A) + D) / 4.0)))
[16:51:25] INFO GPGenotype - Depth of chrom: 6
[16:51:26] INFO GPGenotype - Best solution fitness: 193.0
[16:51:26] INFO GPGenotype - Best solution: D * (((C + ((A / 4.0) + C)) + A) / 5.0)
[16:51:26] INFO GPGenotype - Depth of chrom: 6
[16:51:26] INFO GPGenotype - Evolving generation 25, memory free: 15.0 MB
[16:51:26] INFO GPGenotype - Best solution fitness: 189.0
[16:51:26] INFO GPGenotype - Best solution: (A / 4.0) + ((((C + A) / 6.0) + (D * (D * C))) / 3.0)
[16:51:26] INFO GPGenotype - Depth of chrom: 5
[16:51:26] INFO GPGenotype - Best solution fitness: 188.0
[16:51:26] INFO GPGenotype - Best solution: (((A / 5.0) + (D * (D * C))) / 3.0) + (A / 4.0)
[16:51:26] INFO GPGenotype - Depth of chrom: 5
[16:51:27] INFO GPGenotype - Best solution fitness: 185.0
[16:51:27] INFO GPGenotype - Best solution: (((B - 8.0) / 5.0) + (A / 3.0)) + ((D * (D * (D * C))) / 5.0)
[16:51:27] INFO GPGenotype - Depth of chrom: 5
[16:51:28] INFO GPGenotype - Best solution fitness: 180.0
[16:51:28] INFO GPGenotype - Best solution: (((B - 8.0) / 4.0) + (A / 5.0)) + (((C * D) + (D * C)) / 4.0)
[16:51:28] INFO GPGenotype - Depth of chrom: 4
[16:51:29] INFO GPGenotype - Evolving generation 50, memory free: 5.5 MB

ASSIGNMENT-4
Nandhini R Baladhandapani

[16:51:29] INFO GPGenotype - Best solution fitness: 178.0
[16:51:29] INFO GPGenotype - Best solution: (((C * D) / 2.0) + ((C + (B - 8.0)) + A)) / 4.0
[16:51:29] INFO GPGenotype - Depth of chrom: 5
[16:51:30] INFO GPGenotype - Best solution fitness: 148.0
[16:51:30] INFO GPGenotype - Best solution: ((C + (((C * (A / 6.0)) / 2.0) / 4.0)) + A) / 4.0
[16:51:30] INFO GPGenotype - Depth of chrom: 7
[16:51:31] INFO GPGenotype - Best solution fitness: 143.0
[16:51:31] INFO GPGenotype - Best solution: (((C * (A / 6.0)) / 5.0) + (C + A)) / 5.0
[16:51:31] INFO GPGenotype - Depth of chrom: 5
[16:51:32] INFO GPGenotype - Evolving generation 75, memory free: 13.4 MB
[16:51:33] INFO GPGenotype - Best solution fitness: 140.0
[16:51:33] INFO GPGenotype - Best solution: (((C * (A / 6.0)) / 5.0) + ((D * (A / 4.0)) + A)) / 5.0
[16:51:33] INFO GPGenotype - Depth of chrom: 5
[16:51:33] INFO GPGenotype - Best solution fitness: 132.0
[16:51:33] INFO GPGenotype - Best solution: ((((C + 5.0) * (A / 6.0)) / 5.0) + ((D * (A / 6.0)) + A)) / 5.0
[16:51:33] INFO GPGenotype - Depth of chrom: 5
[16:51:33] INFO GPGenotype - Best solution fitness: 132.0
[16:51:33] INFO GPGenotype - Best solution: ((((C + 5.0) * (A / 6.0)) / 5.0) + ((D * (A / 6.0)) + A)) / 5.0
[16:51:33] INFO GPGenotype - Depth of chrom: 5
Genetic COMPLETE
ITERATIONS for genetic:1
Genetic solution :
[16:51:33] INFO GPGenotype - Evolving generation 90, memory free: 10.6 MB
[16:51:35] INFO GPGenotype - Best solution fitness: 127.0
[16:51:35] INFO GPGenotype - Best solution: ((((D * C) * (A / 6.0)) / 5.0) + ((D + B) + A)) / 5.0
[16:51:35] INFO GPGenotype - Depth of chrom: 5
[16:51:36] INFO GPGenotype - Evolving generation 115, memory free: 11.9 MB
[16:51:40] INFO GPGenotype - Evolving generation 140, memory free: 5.2 MB
[16:51:44] INFO GPGenotype - Evolving generation 165, memory free: 16.5 MB
[16:51:45] INFO GPGenotype - Best solution fitness: 127.0
[16:51:45] INFO GPGenotype - Best solution: ((((D * C) * (A / 6.0)) / 5.0) + ((D + B) + A)) / 5.0
[16:51:45] INFO GPGenotype - Depth of chrom: 5
Genetic COMPLETE
ITERATIONS for genetic:2
Genetic solution :
[16:51:45] INFO GPGenotype - Evolving generation 180, memory free: 11.9 MB
[16:51:47] INFO GPGenotype - Evolving generation 205, memory free: 17.6 MB
[16:51:48] INFO GPGenotype - Evolving generation 230, memory free: 5.7 MB
[16:51:49] INFO GPGenotype - Best solution fitness: 125.0
[16:51:49] INFO GPGenotype - Best solution: (A + ((B + (((D * C) * (A / 6.0)) / 5.0)) - 5.0)) / 5.0
[16:51:49] INFO GPGenotype - Depth of chrom: 7
[16:51:49] INFO GPGenotype - Best solution fitness: 124.0
[16:51:49] INFO GPGenotype - Best solution: (A + ((B + (((D * C) * (A / 6.0)) / 5.0)) - 7.0)) / 5.0
[16:51:49] INFO GPGenotype - Depth of chrom: 7
[16:51:50] INFO GPGenotype - Evolving generation 255, memory free: 17.8 MB
[16:51:51] INFO GPGenotype - Best solution fitness: 124.0
[16:51:51] INFO GPGenotype - Best solution: (A + ((B + (((D * C) * (A / 6.0)) / 5.0)) - 7.0)) / 5.0
[16:51:51] INFO GPGenotype - Depth of chrom: 7
Genetic COMPLETE

The results generated in randomly by giving the gp.evolve(0) in the code and results evolved by genetic program by giving the gp.evolve(90). I found that the fitness value gets minimised for every iterations. I have run this code for many times to get the minimise value for fitness function. Because, it is randomly generated. The same method, I have followed for two files. The fitness values evolved should predict the output(last column) in the file. In the result, the depth of chrome represents level of the tree in the file.

I have used kemerer file at first, it shows the fitness value with the lot of difference when compared with the output value in the value. So, I have chosen an another file “albercht” that gives fitness which is nearer to the output in the file.

WEKA(Waikato Environment for Knowledge Analysis)

For example, The data mining is used to predict the cost of one requirements in company means it can predicts the other cost for other companies as well. But, WEKA is not expensive like other data mining tools and it provides all the resources needed.

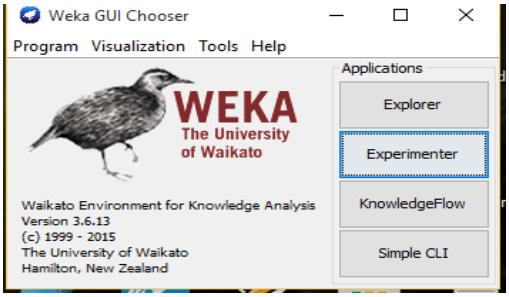
ASSIGNMENT-4
Nandhini R Baladhandapani

WEKA is the data mining software in java. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. In most cases, weka is used for big dataset.

Weka is the tool mining kit that is used by giving the file directly in to the too. (WEKA), free and open source software you can use to mine your own data and turn what you know about your users, your clients, and your business into useful information for increasing your revenue. The software is written in the Java language and contains a GUI for interacting with data files and producing visual results.

Implementation on WEKA

I have stored the file with the extension .arff format and parsed it into the WEKA tool. It directly takes the input from the file and it produces the results.



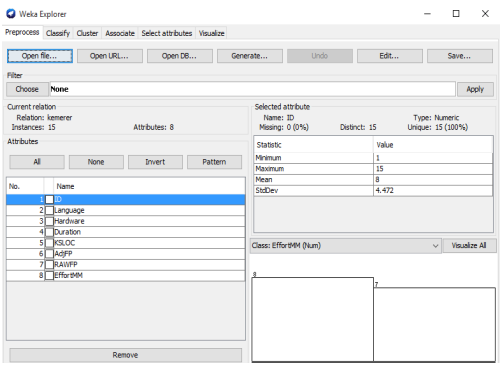
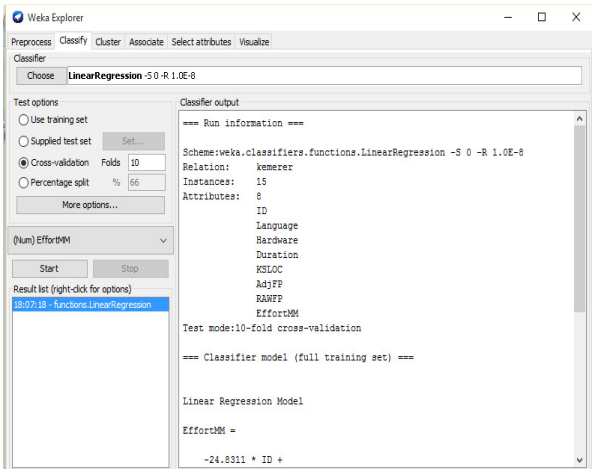


Figure1:starting into weka GUI software tool

figure2: WEKA explorer- choosing the file into the WEKA

These figures says that gets started with WEKA



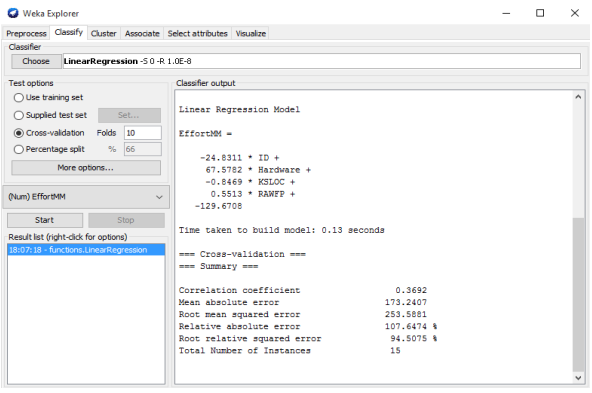


Figure3: choosing linear regression

figure4: output

Figure 3 and 4: It says that the file is chosen as the relation. It's having instances as 15. It reads the attributes from the file containing topmost data. It automatically calculating the effort value from the dataset in the file for each attributes .It calculates the standard measures how good the prediction is. MAE - Mean Absolute Error :- abs(actual effort - estimated effort) , correlation coefficient, root mean squared error, relative absolute error and root relative absolute error are calculated and the results are printed in above screenshots.

Screenshots for the file albercht

ASSIGNMENT-4
Nandhini R Baladhandapani

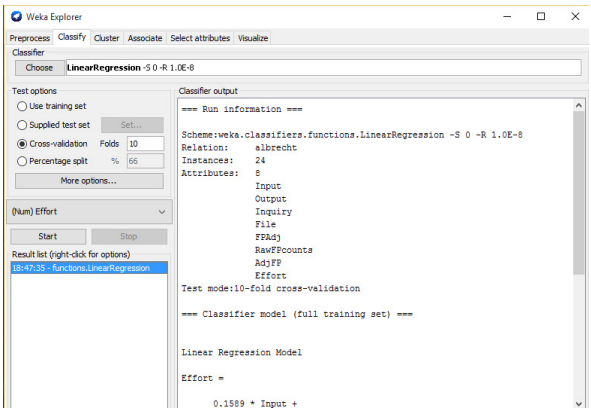


Figure5:output

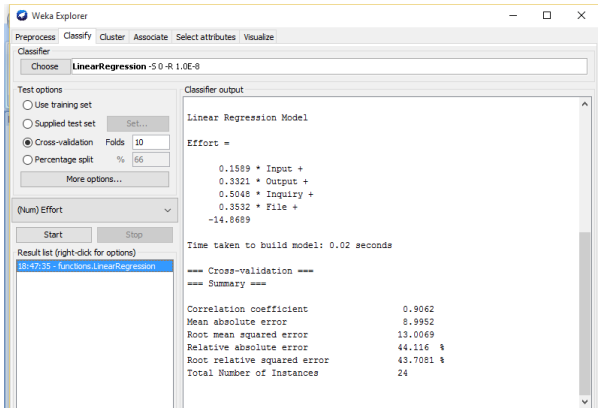


Figure6:output

Figure 5 and 6: The same results as like as the kemerer file but with the different values.

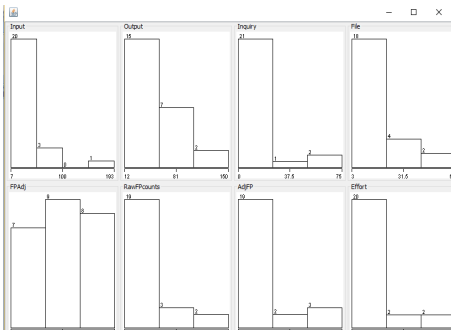


Figure7: It creates from the clusters as a graph for all the attributes in the file.

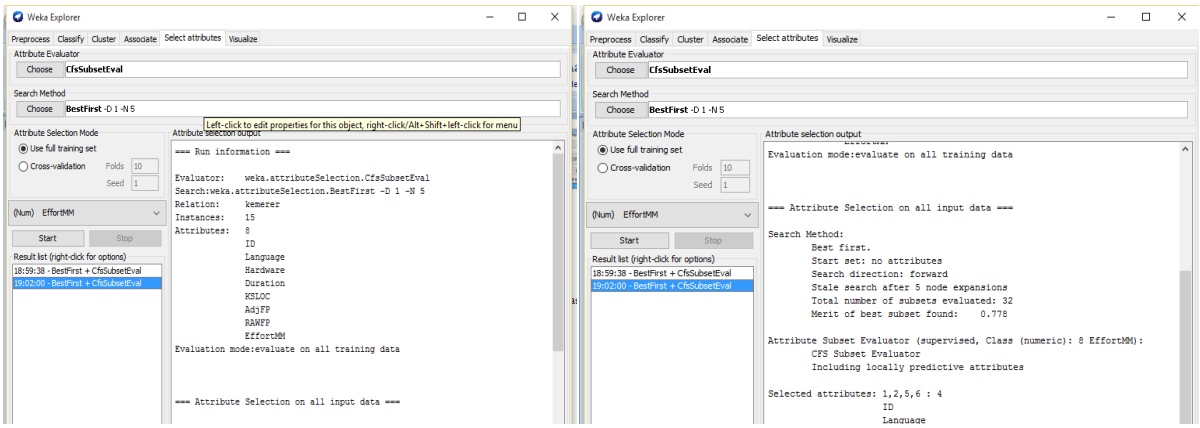
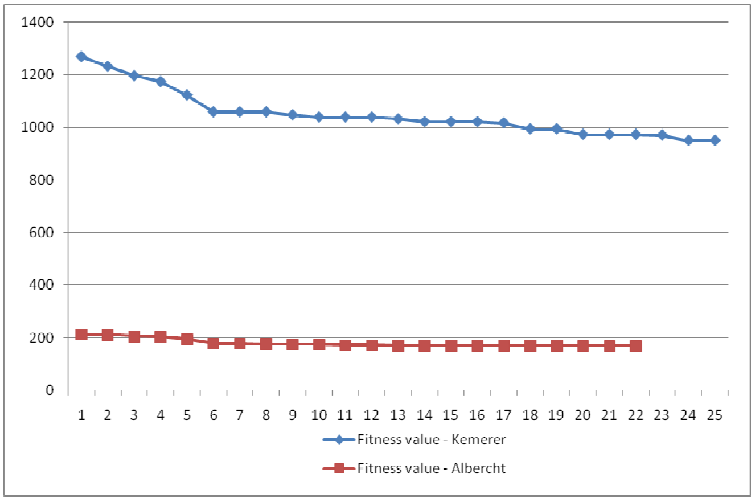


Figure8: WEKA in selected attributes. It gives the information about the search method- how the search method goes in the direction, attribute selected evaluator, selected attribute. It gives some more information in screenshots.

Comparing fitness value between the two files:

ASSIGNMENT-4
Nandhini R Baladhandapani



In comparing with the two files, it shows the fitness value almost nearer to the predicted value in the albercht file than kemerer file.

I have found interesting is that getting same fitness value for each iterations even it is randomly generated.

Conclusion:

The accurate prediction of software development costs is a critical issue to make the good management decisions and accurately determining how much effort and time a project required for both project managers as well as system analysts and developers.

References

[1] Mark Harman, Phil McMinn, Jefferson Teixeira de Souza, and Shin Yoo, Search Based Software Engineering: Techniques, Taxonomy, Tutorial

[2] A FIELD GUIDE TO GENETIC PROGRAMMING. Available from: http://www.gp-field-guide.org.uk/. [March 2008]

[3] A Tiny Genetic Programming Implementation. Available from: http://cswww.essex.ac.uk/staff/rpoli/TinyGP/. [28 February 2008]

[4] JGAP: A First/Simple Tutorial. Available from: https://cvalcarcel.wordpress.com/2009/08/04/jgap-a-firstsimple-tutorial/. [4 August 2009]

[5] Data mining with WEKA, Part 1: Introduction and regression. Available from: http://www.ibm.com/developerworks/opensource/library/os-weka1/index.html. [27 April 2010]