DMWA Lab eval

Teghdeep Kapoor

18104050

B12

Q1

#include<iostream.h>

#include<conio.h>

void main() {

  int i, j, t1, k, l, m, f, f1, f2, f3;

  //Initial item-purchase

  int a[5][5];

  for (i = 0; i < 5; i++) {

    cout << "\n Enter items from purchase " << i + 1 << ":";

    for (j = 0; j < 5; j++) {

      cin >> a[i][j];

    }

  }

  //Defining minimum level for acceptence

  int min;

  cout << "\n Enter minimum acceptance level";

  cin >> min;

  //Printing initial input

  cout << "\nInitial Input:\n";

  cout << "\nTrasaction\tItems\n";

  for (i = 0; i < 5; i++) {

    cout << i + 1 << ":\t";

    for (j = 0; j < 5; j++) {

      cout << a[i][j] << "\t";

    }

    cout << "\n";

  }

  cout << "\nAssume minimum support: " << min;

  //First pass

  int l1[5];

  for (i = 0; i < 5; i++) {

    t1 = 0;

    for (j = 0; j < 5; j++) {

      for (k = 0; k < 5; k++) {

        if (a[j][k] == i + 1) {

          t1++;

        }

      }

    }

    l1[i] = t1;

  }

  //Printing first pass

  cout << "\n\nGenerating C1 from data\n";

  for (i = 0; i < 5; i++) {

    cout << i + 1 << ": " << l1[i] << "\n";

  }

  //Second pass

  //Counting number of possibilities for pass2

  int p2pcount = 0;

  int p2items[5];

  int p2pos = 0;

  for (i = 0; i < 5; i++) {

    if (l1[i] >= min) {

      p2pcount++;

      p2items[p2pos] = i;

      p2pos++;

    }

  }

  //Printing selected items for second pass

  cout << "\nGenerating L1 From C1\n";

  for (i = 0; i < p2pos; i++) {

    cout << p2items[i] + 1 << "\t" << l1[p2items[i]] << "\n";

  }

  //Joining items

  int l2[5][3];

  int l2t1; //will hold first item for join

  int l2t2; //will hold second item for join

  int l2pos1 = 0; //position pointer in l2 array

  int l2ocount = 0; //product join occruance counter

  int l2jcount = 0; //join counter

  for (i = 0; i < p2pcount; i++) {

    for (j = i + 1; j < p2pcount; j++) {

      l2t1 = p2items[i] + 1;

      l2t2 = p2items[j] + 1;

      if (l2t1 == l2t2) {

        //it is self join

        continue;

      }

      //join the elements

      l2[l2pos1][0] = l2t1;

      l2[l2pos1][1] = l2t2;

      l2jcount++;

      //count occurances

      l2ocount = 0; //reset counter

      for (k = 0; k < 5; k++) {

        f1 = f2 = 0; //resetting flag

        //scan a purcahse

        for (l = 0; l < 5; l++) {

          if (l2t1 == a[k][l]) {

            //one of the element found

            f1 = 1;

          }

          if (l2t2 == a[k][l]) {

            //second elements also found

            f2 = 1;

          }

        }

        //one purchase scanned

        if (f1 == 1 && f2 == 1) //both items are present in purchase

        {

          l2ocount++;

        }

      }

      //assign count

      l2[l2pos1][2] = l2ocount;

      l2pos1++;

    }

  }

  //Printing second pass

  cout << "\n\nGenerating L2\n";

  for (i = 0; i < l2jcount; i++) {

    for (j = 0; j < 3; j++) {

      cout << l2[i][j] << "\t";

    }

    cout << "\n";

  }

  //Third pass

  int p3pcount = 0;

  int p3items[5] = {

    -1,

    -1,

    -1,

    -1,

    -1

  };

  int p3pos = 0;

  for (i = 0; i < 5; i++) {

    if (l2[i][2] >= min) {

      f = 0;

      for (j = 0; j < 5; j++) {

        if (p3items[j] == l2[i][0]) {

          f = 1;

        }

      }

      if (f != 1) {

        p3items[p3pos] = l2[i][0];

        p3pos++;

        p3pcount++;

      }

      f = 0;

      for (j = 0; j < 5; j++) {

        if (p3items[j] == l2[i][1]) {

          f = 1;

        }

      }

      if (f != 1) {

        p3items[p3pos] = l2[i][1];

        p3pos++;

        p3pcount++;

      }

    }

  }

  int l3[5][4];

  int l3ocount = 0;

  int l3jcount = 0;

  for (i = 0; i < p3pcount; i++) {

    for (j = i + 1; j < p3pcount; j++) {

      for (k = j + 1; k < p3pcount; k++) {

        l3[i][0] = p3items[i];

        l3[i][1] = p3items[j];

        l3[i][2] = p3items[k];

        l3jcount++;

        l3ocount = 0;

        for (k = 0; k < 5; k++) {

          f1 = f2 = f3 = 0;

          for (l = 0; l < 5; l++) {

            if (l3[i][0] == a[k][l]) {

              f1 = 1;

            }

            if (l3[i][1] == a[k][l]) {

              f2 = 1;

            }

            if (l3[i][2] == a[k][l]) {

              f3 = 1;

            }

          }

          if (f1 == 1 && f2 == 1 && f3 == 1) //all items are present in purchase

          {

            l3ocount++;

          }

        }

        l3[i][3] = l3ocount;

      }

    }

  }

  cout << "\n\nGenerating L3\n";

  for (i = 0; i < l3jcount; i++) {

    for (j = 0; j < 4; j++) {

      cout << l3[i][j] << "\t";

    }

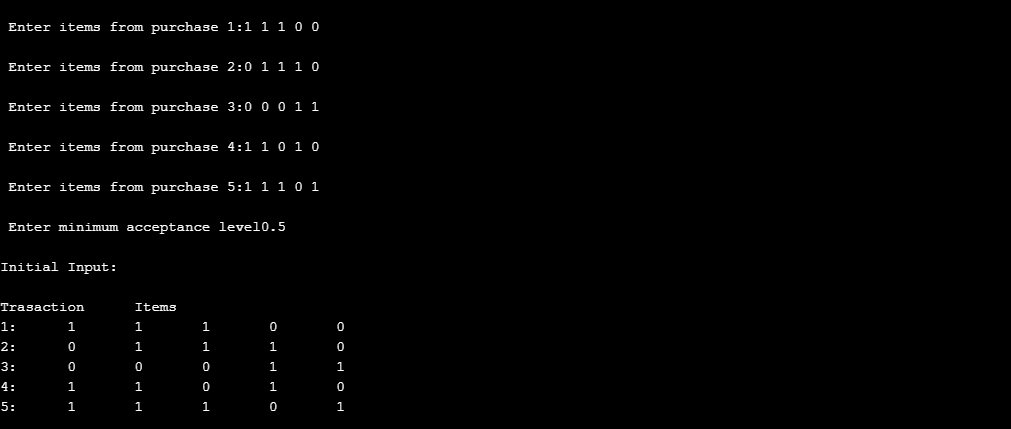
    cout << "\n";

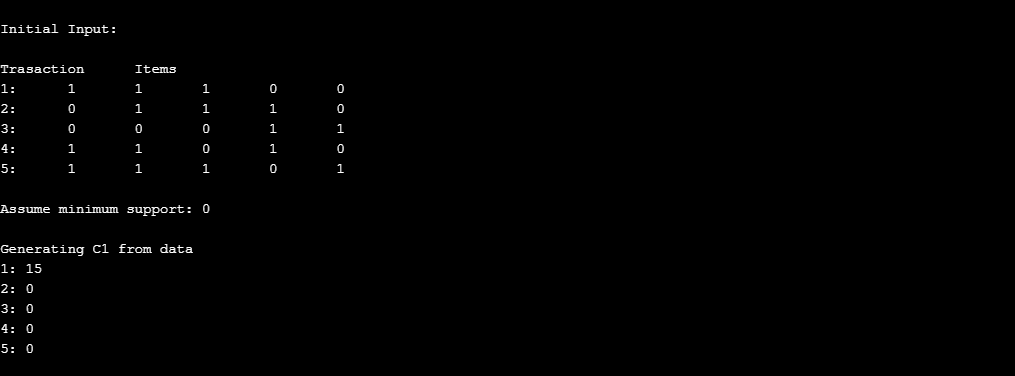
  }

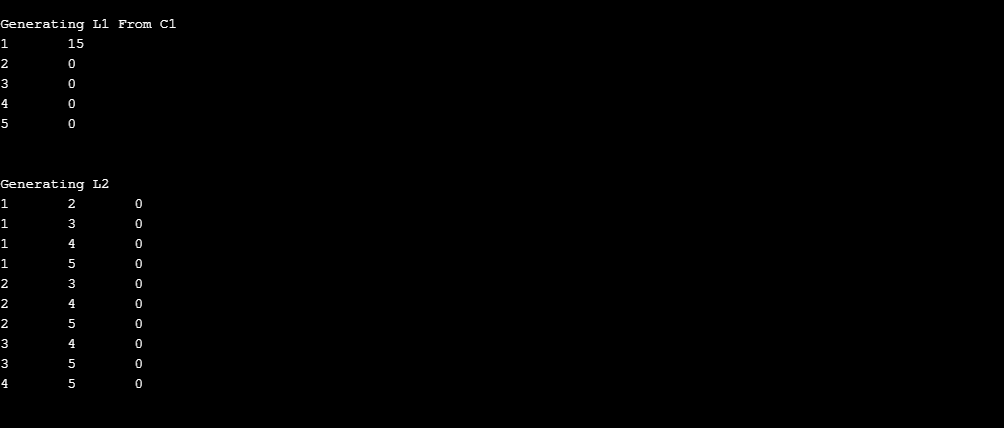
  getch();

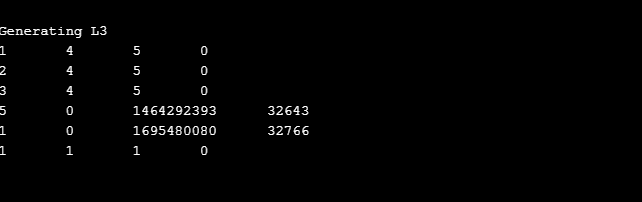
}

Output









Q2

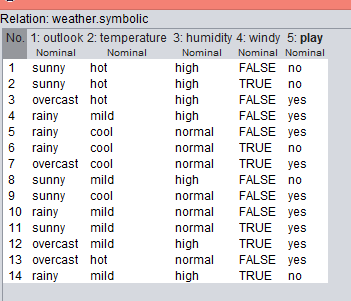
A.

Step-1. Make a File Weather-nominal.arff and save it.

Step-2. Open the arff file on weka explorer.

Step-3. Click on Edit and View the File.

Weather-nominal.arff



Step-4. Click on Choose Filters Button

Step-5.Under Unsupervised>Instances>RemoveWithValues

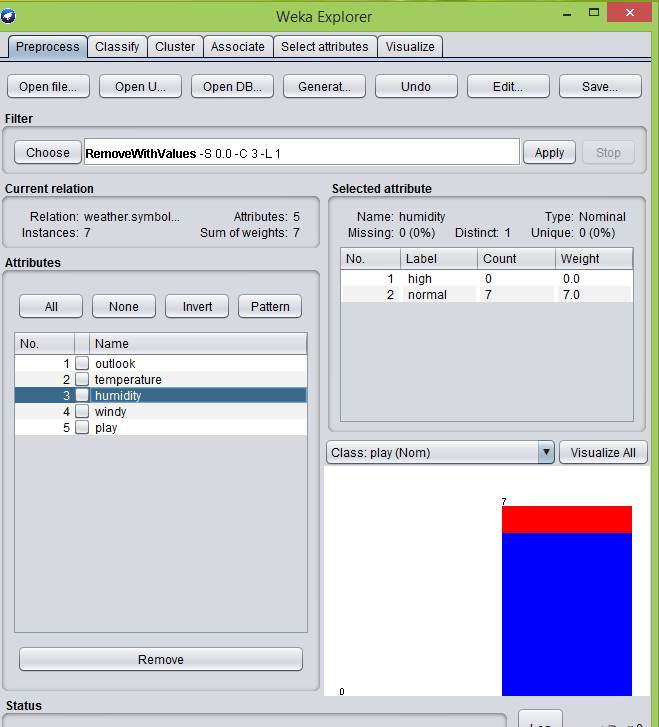
Step-6. After selecting Click on it and a dialog box appears

Step-7. Set Attributeindices as 3(humidity index)

Step-8. Set The nominal index as 1.

Step-9. Click on Ok. And then Apply.

Step-10. Click on Visualize all see the result.

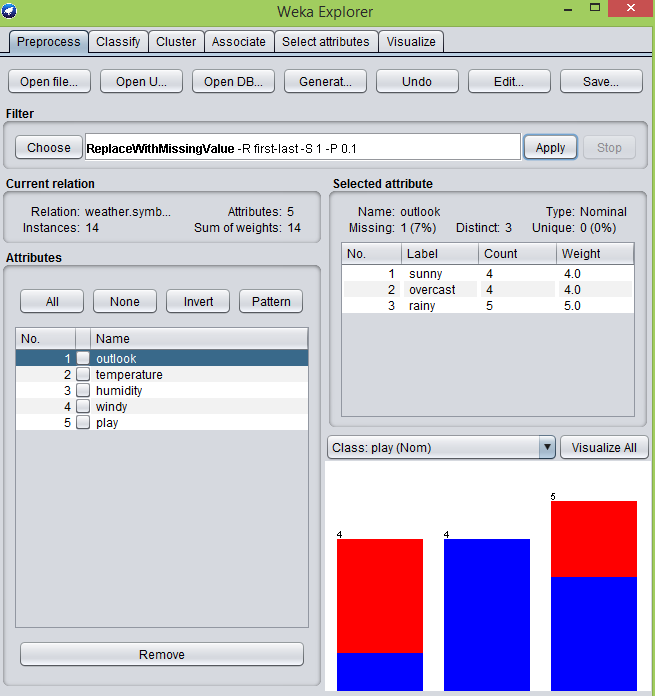


B.

Step-1. Open File WeatherNominal.arff on Weka

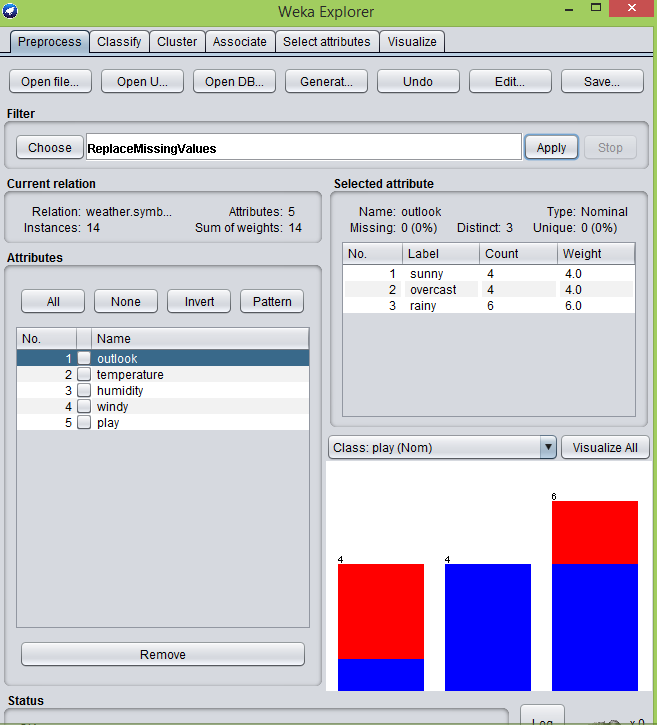
Step-2. choose > unsupervised > attributes > ReplaceWithMissingValues

Step-3. Under default setting click Apply.



Step-4 choose > unsupervised > attributes > ReplaceMissingValues

Step-5 Under default settings click Apply.



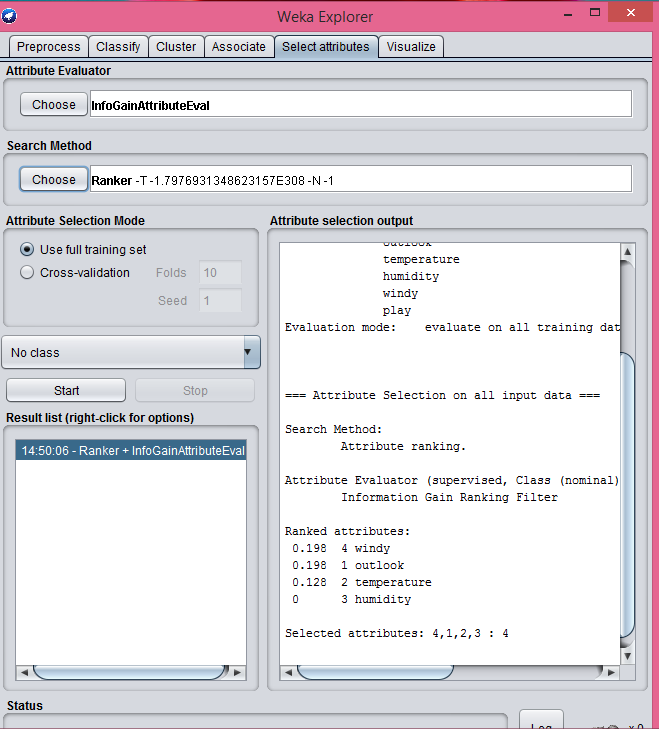
Step-6. Click on Visualize all see the result.

C.

Step-1. Go under Select Attribute Tab on Weka

Step2. Click on choose and select InfoGainAttributeEval

Step-3. Click on Yes on dialog box (Ranker Dailog box\_)



Step-4 Click on Start

