INDEX SHEET
PARTICULARS OF EXPERIMENTS PERFORMED

Serial No.	Name of the Experiment	Page No.	Date of Expt.	Date of Submission	Remarks
1a	program for transforming	1	6-4-21	13-4-21	
	numericdata using z-score				
	normalization				
1b	program for transforming	3	6-4-21	13-4-21	
	numericdata using min-				
	max normalization				
2a	program for smoothing	5	20-4-21	27-4-21	
	numeric data with bin				
	2,3 & 5 size				
2b	program for smoothing	7	20-4-21	27-4-21	
	numeric data with binning				
	2,3,5 with bin means,				
	median, boundaries				
3	program for handling	10	4-5-21	8-5-21	
	missing data				
4	program for missing data	13	11-5-21	18-5-21	
5	TO create arff file with	15	18-5-21	21-5-21	
	all type attribute &				
	explore steps using				
	weka				

Serial No.	Name of the Experiment	Page No.	Date of Expt.	Date of Submission	Remarks
6	Attribute inf for numeric	17	1-6-21	4-6-21	
	&non numeric attribute and				
	loaded into weka				
7	create a transactional	19	8-6-21	14-6-21	
	data set suitable for				
	using weka and explore				
	the steps to data load				
8	create csv file with all	21	18-6-21	21-6-21	
	type of attributes				
9	prog to matrix(distance)	22	22-6-21	29-6-21	
10	Using weka list step	23	6-7-21	12-7-21	
	involved in decision tree				
11	Generating of decision tree	27	13-7-21	19-7-21	
12	program for clusters	30	20-7-21	26-7-21	
	using k-means				
13	Using weka list the steps	36	27-7-21	3-8-21	
	involved in sending				
	clusters using k-means				

Experiment - 1(a)

Aim:-

write a program for transformation numeric data using z-score normalisation.

Description:-

Normalization where the datavalues are scaled within the specific range. The techniques are:

1) Min-Max normalization
2) Z-score normalization
3) Decimal scaling

Min-Max Normalization:-

It performs a Linear transformation on the original data it maps a value 'v' to 'v' as the value.

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

Z-score Normalization:-

The values of attributes of 'A' are normalized based on the mean and standard deviation of attribute 'A'.

$$v' = \frac{v - \bar{A}}{\sigma A}$$

Decimal scaling:-

It normalizes the attributes by moving the decimal point of values.

$$v' = \frac{v}{10^j}$$

Program:-

```
import statistic as st
import math
array = []
temp = []
new_max = 1.0
new_min = 0.0
array = list(map(int, input("enter value into array").split())
Val = int(input("\n enter the value"))
print ("\n z-score normalization")
new_val = (val. st.mean(array))/st.stdev(array)
print(new_val)
```

Output:-

```
enter value into array  -123  256  -4548  -369  745
enter the value 47
z-score normalization
-0.122
```

Experiment-1(b)

Aim:-

write a program for transformation numeric data using z-score normalization.

Description:-

Normalization where the data values are scaled within the specific range of the techniques are:

1) Min-Max Normalization
2) z-score Normalization
3) Decimal scaling

Min-Max Normalization:-

It performs a linear transformation on the original data it maps a value 'v' to 'v'' as the value.

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A.$$

z-score Normalization:-

The values of attributes of 'A' are normalized based on the mean and standard deviation of attribute 'A'.

$$v' = \frac{v - \bar{A}}{\sigma A}$$

Decimal scaling:-

It normalizes the attributes by moving the decimal point of values.

$$v' = \frac{v}{10^j}$$

Program:-

```
import statistic as st
import math
array = []
temp = []
new_max = 1.0
new_min = 0.0
array = list(map(int, input("enter value into array").split()))
val = int(input("\n enter the value"))
print("\n Min Max Normalization")
new_val = ((val_min(array))/(max(array) - min(array))(new_max - new_
          min) + new_min
print(new_val)
for i in array:
    temp.append(len(str(i)))
    j = max(temp) - 1
    print("\n Decimal scaling")
    new_val = val((math.pow((10, i))
print(new_val)
```

Output:-

```
enter value into array -123 256 -4548 -369 745
enter the value 47
Min Max Normalization
0.373
```

Regd. No. 18BQ1A05H6

Experiment-2(a)

Aim:- write a program for smoothing numeric data with binsizes 2, 3 & 5

Description:-

Smoothing refers for removing noise data from dataset. This can be done in following methods.

1) Binning
   - Binning by mean
   - Binning by median
   - Binning by most probable value
   - Equal frequency binning
2) Clustering
3) combined human and computer inspection
4) Regression

Binning by mean:-

Step-1: Take the values to bin

Step-2: Take the sizes of each bin

Step-3: Distribute the values into each bin according to its capacity.

Step-4: Replace each of the value in the bin with the 'mean' of the values of that bin.

Program:-

```
import statistic as st
print ("Enter the values to bin")
a=list(map(int, input().split()))
bins = []
print("\n Enter number of bins")
```

```
n=int(input())
for i in range(n):
    print("\n Enter bin"+i+"size")
    size=int(input())
    bins.append([K:K+size])
    K+=size
for i in bins:
    bins.mean=st.mean(i)
    for j in i:
        i[j]=bins.mean
    print("\n Bin"+i+"mean is")
    print(bins.mean)
    print("\n After bining elements of bin"+i+"are:")
    for j in i:
        print(i[j], end=" ")
```

Output:

Enter the values to bin

3 2 2 4 3 3 10 0

Enter number of bins 3

Enter bin 0 size 2

Enter bin 1 size 3

Enter bin2 size 5

Bin : 1

3 2 0

Bin : 2

2 4 3

Experiment-2(b)

Aim:- write a program for smoothing numeric data using binning with bin mean, bin median, bin boundaries.

Description:-

Smoothing refers for removing noise data from dataset.

i) Binning method:

a) First sort data and partition into bins.

b) They can smooth by bin names, means smooth by bin smooth by bin boundaries etc.

ii) Clustering

Select and remove outliers.

iii) Combined computer & Human inspection

Detect suspicions & human inspection

iv) Regression:

smooth by fitting the data into regression functions.

Program:-

```
#include<stdio.h>
#include<stdlib.h>
int main() {
    int n, a[50], i, j, temp, b, m=1, s=0;
    float sum=0, mean;
    printf("enter no of elements in array");
    scanf("%d", &n);
    printf("enter elements into array");
    for(i=0; i<n; i++)
```

```
    scanf("%d", &a[i]);
for(i=0; i<n; i++){
        for(j=i+1; j<n; j++){
            if(a[i]>a[j])
                temp=a[i];
                a[i]=a[j];
    }}          a[j]=temp;
printf("enter bin capacity");
scanf("%d", &b);
for(i=0; i<n; i++)
        if(s==0)
            printf("\n bin --- %d", m);
        printf("\n %d \n", a[i]);
        sum=sum+a[i];
        s++;
        if(s==b)
            mean=sum/b;
            printf("\n mean is %f", mean);
            sum=0;
            s=0;
            m++;
        else if(i==(n-1) && (s<b))
            for(j=0; j<b-s; j++)
                printf("%d\n", j);
                mean=sum/b;
    printf("mean is %f", mean);
    printf("\n After replacing with mean");
    printf("\n Bin %d \n", i);
```

```
    for(int i=0; i<n; i++)
            printf("%.f", mean);
    }
  }}
```

Output:-

```
enter size: 9
enter values: 3 2 2 4 3 3 10 0
enter bin size: 3
Bin: 1
3 2 2
mean is 2.333
After replacing with mean
Bin: 1
2.333 2.333 2.333
Bin: 2
4 3 3
mean is 3.333
After replacing with mean
3.333 3.333 3.333
Bin: 3
1 0 0
mean is 0.3333
After replacing with mean
Bin: 3
0.3333 0.3333 0.3333
```

Experiment - 3

Aim:-

To implement the concept of data discretization.

Description:-

Data discretization converts a large number of data value into similar once. so that data evaluation & data management becomes very closely.

It can be used to divide the range of continous attribute into intervals. Numerous continous attribute values are replaced by small interval labels. This leads to a concise, easy to use, knowledge level representation of mining results.

If the process starts by first finding once or a few points to split the entire attribute range and then repeats this recursively on resulting interval then it is called top-down discretiza -tion or splitting.

If the process starts by considered all of value continous values as potential split-points. removes some by merging neighbourhood values to form intervals then it is called bottom-up discretization or merging.

program:-

```
import java.util.*;
public class Discretization {
    public static void main(String[] args) {
        int n, count=0, count1=0, count2=0, count3=0;
        int count4=0;
```

```
float a[] = new float[100];
String b[] = new String[100];
Scanner s = new Scanner(System.in);
System.out.println("enter n:");
n = s.nextInt();
System.out.println("enter elements:");
for(int i=0; i<n; i++){
        a[i] = s.nextfloat();
}
for(int i=0; i<n; i++){
    if(a[i]>=70){
        b[i]="D";
        count++;
    }
    else if(a[i]>=60){
        b[i]="fc";
        count1++;
    }
    else if(a[i]>=50){
        b[i]="sc";
        count2++;
    }
    else if(a[i]>=40){
        b[i]="tc";
        count3++;
    }
    else{
        b[i]="f";
        count4++;
    }
}
```

```
System.out.println();
System.out.println("Marks and their status:");
for(int i=0; i<n; i++){
    System.out.println("Marks and their status:");
for(int i=0; i<n; i++){
    System.out.println(a[i]+" "+b[i]);
}
System.out.println("NO.of student under distinction class are:"+
 count);
System.out.println("NO.of student under first class are:"+count1);
System.out.println("NO. of student under second class are:"+count2);
System.out.println("No. of student under third class are:"+count3);
System.out.println("NO.of students under fail are:"+count4);
}
}
```

Output:

```
Enter n=10
Enter elements: 90.0 85.1 76.4 27.8 38.5 67.8 88.7 49.8
71.9 30.8
Marks and their status
49.8    tc
90.0    D
71.9    D
85.1    D
30.8    F
76.1    F
29.8    F
67.8    FC
88.7    D
NO.of students under distinction class are: 5
NO.of students under first class are: 1
NO. of students under second class are:0
NO.of students under fail are:3
```

Experiment-4

Aim:-

write a program for handling missing value.

Description:-

Handling missing data

1) Ignore the tuple
2) Fill in missing values manually
3) Use a global constant to fill in missing value
4) Use a measure of central tendency for attribute
5) Use the attribute mean or median for all samples belonging to same class as given tuple
6) use the most propable value to fill missing value.

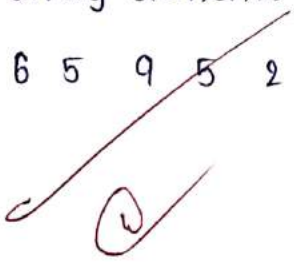
Program:-

```
#include <stdio.h>
void main(){
    int a[10], n, i, count=0, avg, sum=0;
    printf("enter no. of elements: ");
    scanf("%d", &n);
    for(i=0 ; i<n; i++)
        if(a[i]!=-1)
            sum+=a[i];
            count++;
    for(i=0; i<n; i++)
        scanf("%d", &a[i]);
    avg=sum/count;
    printf("avg of array element is %d \n", avg);
    for(i=0; i<n; i++)
```

```
        if (a[i] = -1)
                a[i] = avg;
        printf ("%d \t", a[i]);
    }
}
```

Output:-

```
enter no. of elements 10
5
-1
6
9 -1
9
-1
2
-1
3
-1
Avg of array elements is : 5
5 5 6 5 9 5 2 5 3 5
```



Regd. No. 18BQ1AD5H6

Experiment - 5

Aim:-

To create an arff file with all types of attributes and explore the steps using WEKA for data load.

Procedure:-

How to create arff file

1. First we define the name of a relation.
2. List all attributes with their type specified.
3. Enter values for all attributes of dataset. If missing values are there, specify it as "?".
4. Save the file as filename.arff.

Student.arff

```
@ relation student
@ attribute regdno numeric
@ attribute name string
@ attribute branch {cse, it, ece}
@ data
1. Ravi, CSE
2. dev, It
3. Ram, CSE
4. sita, ECE
```

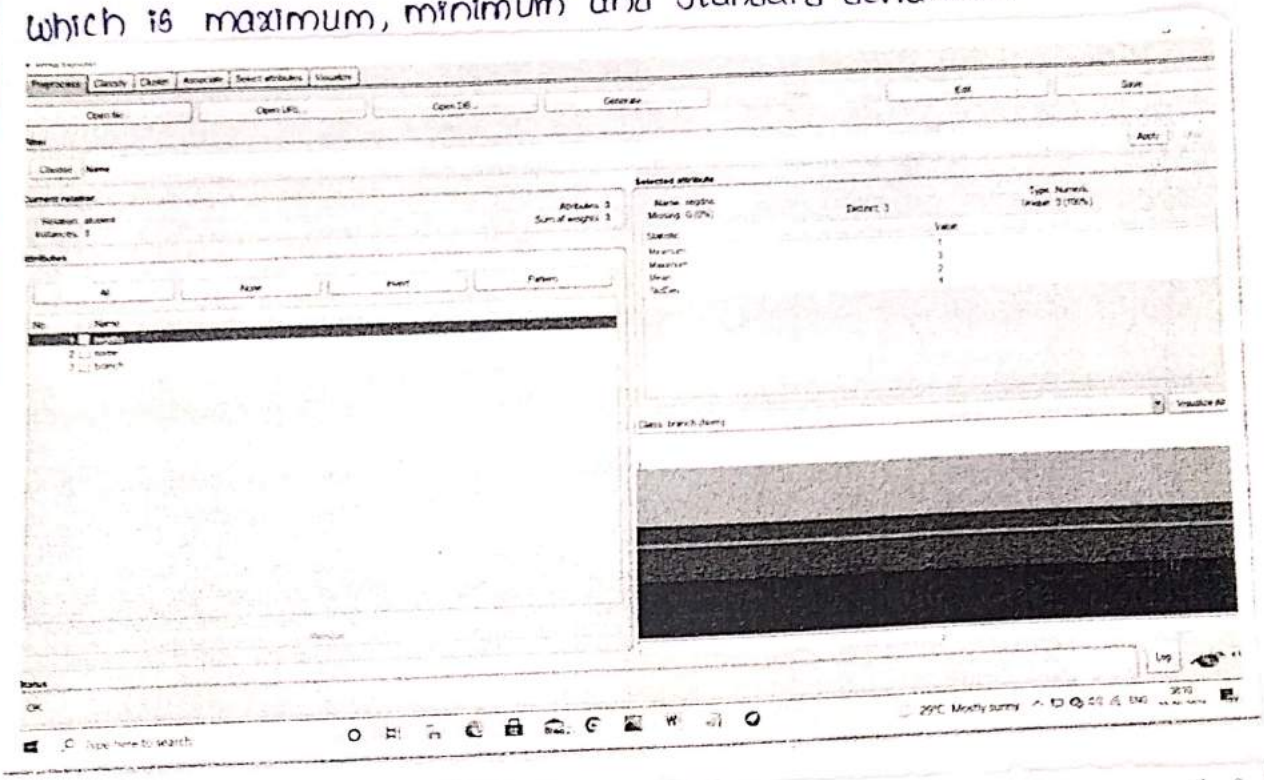
Steps to load arff file:

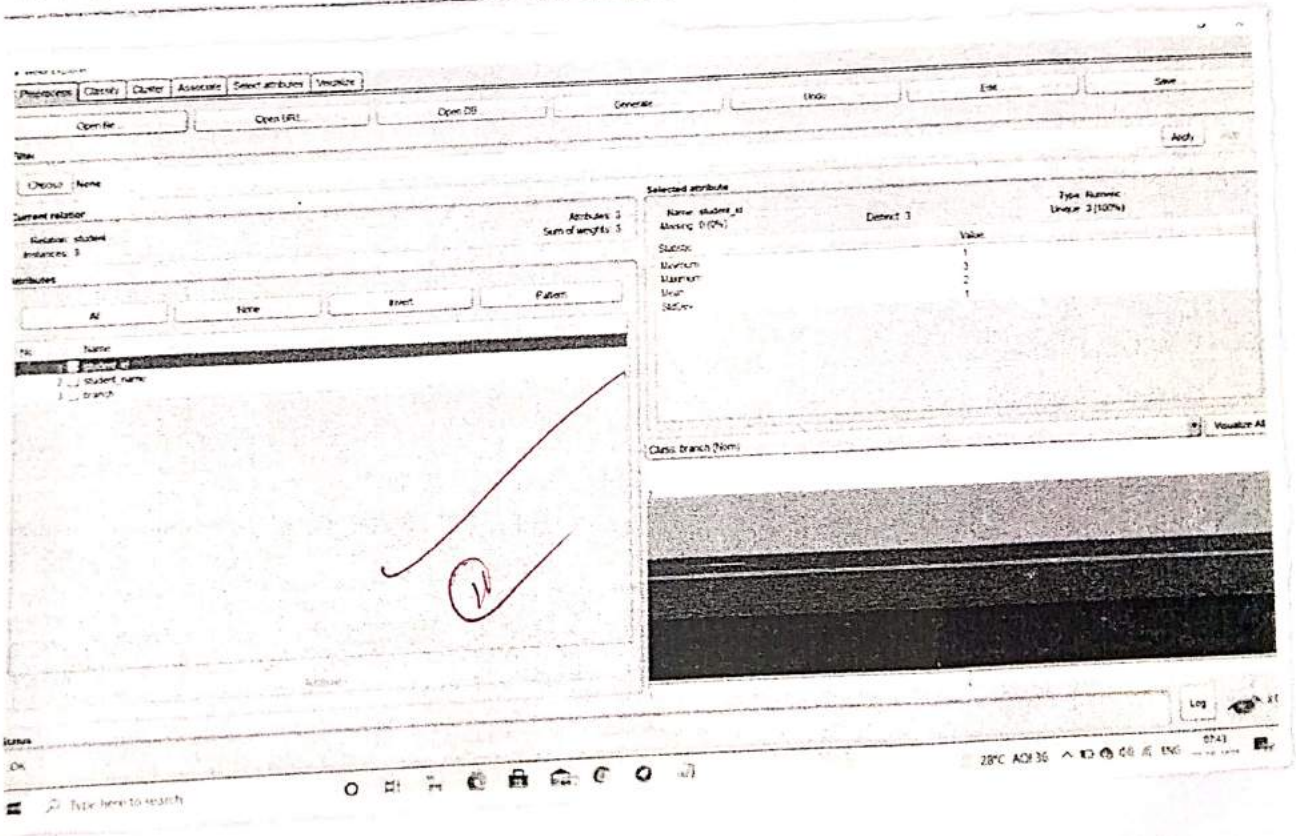
1. Open explores in weka interface
2. preprocessor is the default tab of explorer.
3. select open file & search for required arff file.

4. Load the dataset

5. Screenshot the graph

The attached screenshoot represents the relation of the studentid which is maximum, minimum and standard deviation.





Regd. No. 19BQ1A05H6

Experiment-6

Aim:- write attribute information for numeric attribute & non-numeric attribute when loaded into weka.

Description:-

The weather numeric is available in the arff format in the weka tool.

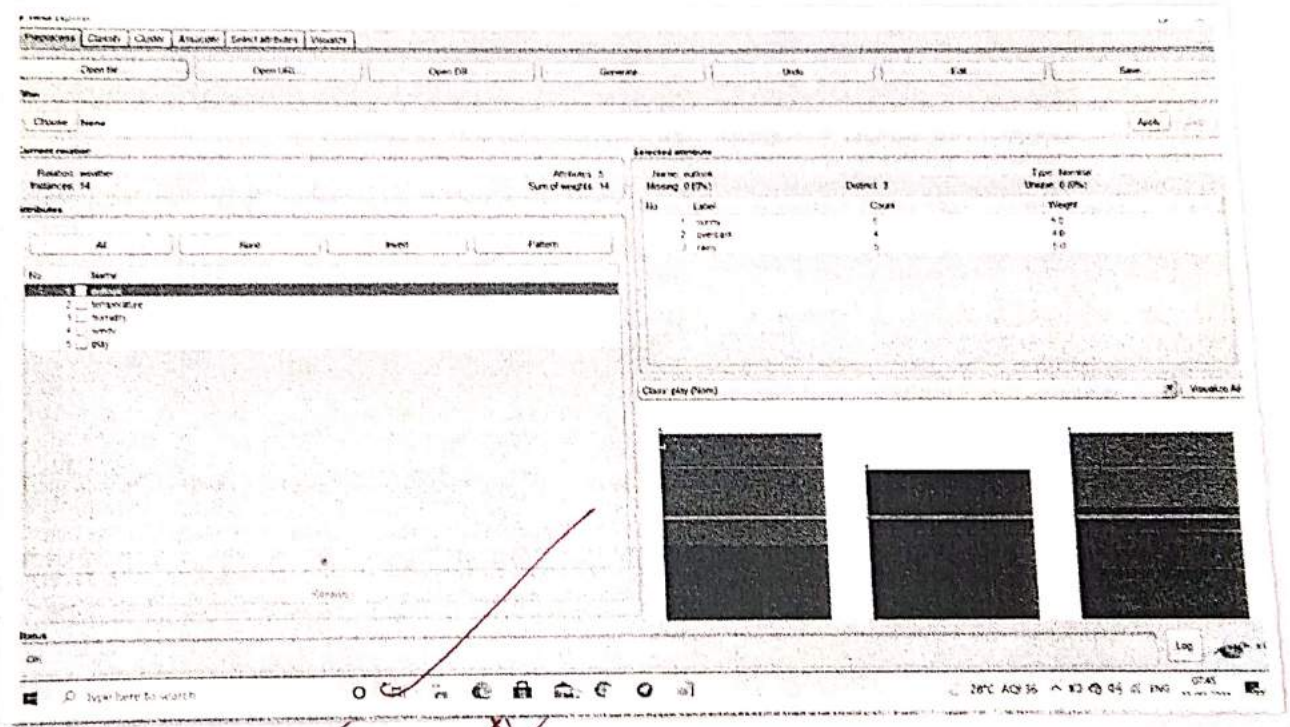
i) Open explorer in weka interface

ii) preprocessor is default tab of explorer

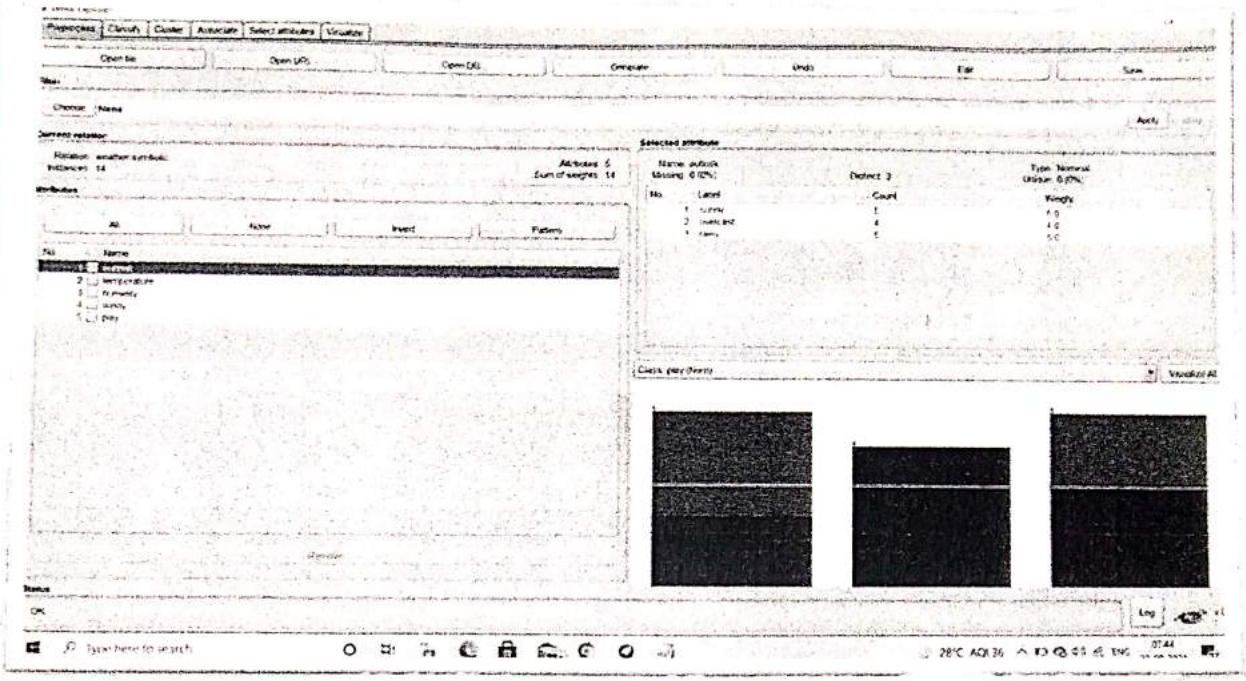
iii) Select open file & search for required file

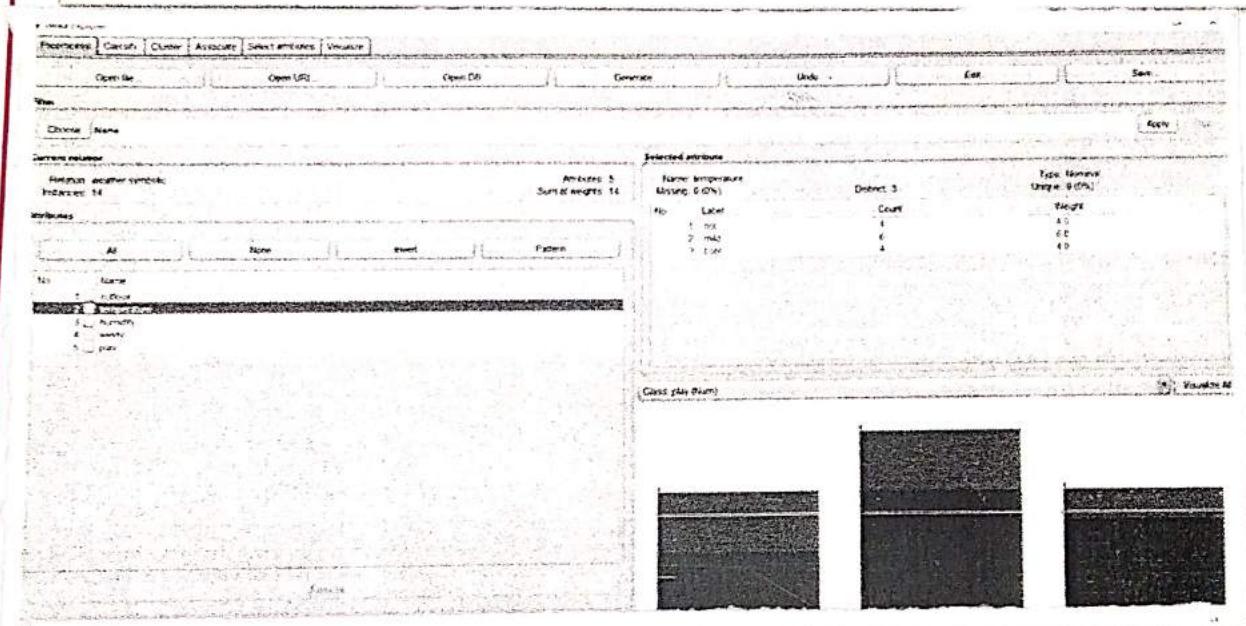
iv) Load the data set

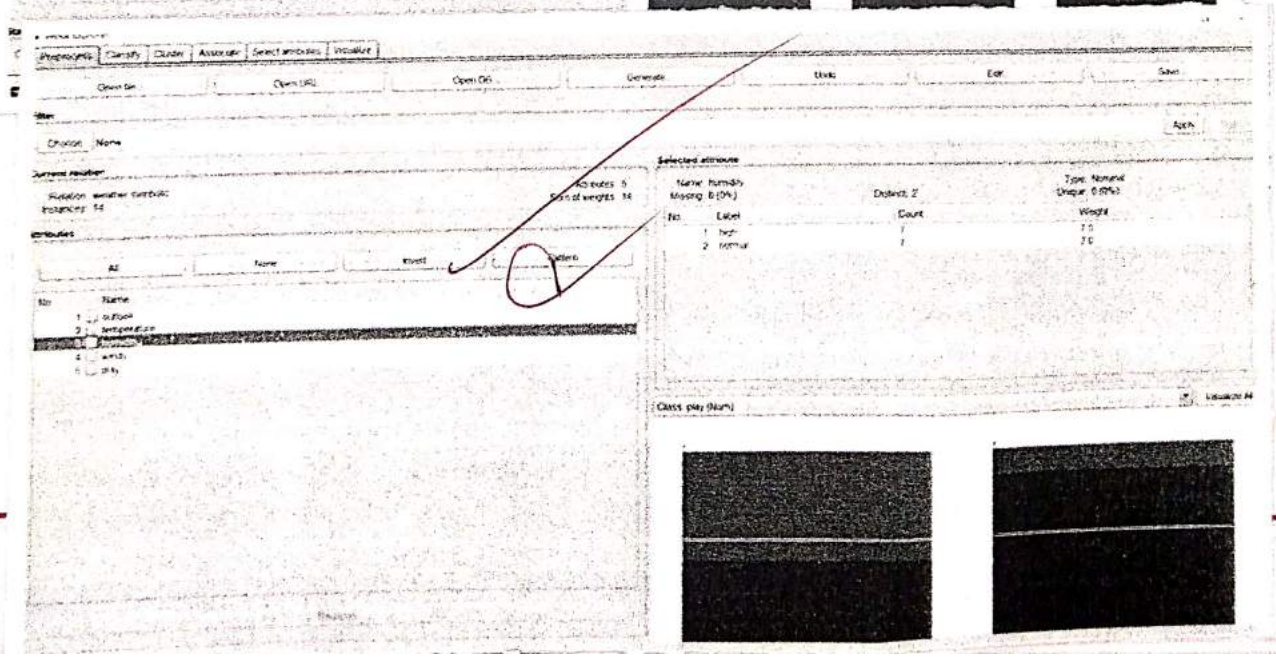
v) Screen the graph



Regd. No. 5H6







Experiment - 7

Aim:-

Create a transactional Dataset suitable for weka & explain the steps to data load.

Description:-

create the following transaction dataset in texteditor

@ relation transaction dataset

@ attribute l1 {1}

@ attribute l2 {1}

@ attribute l3 {1}

@ attribute l4 {1}

@ attribute l5 {1}

@ data

1, 1, ?, ?, 1

2, 1, ?, 1, ?

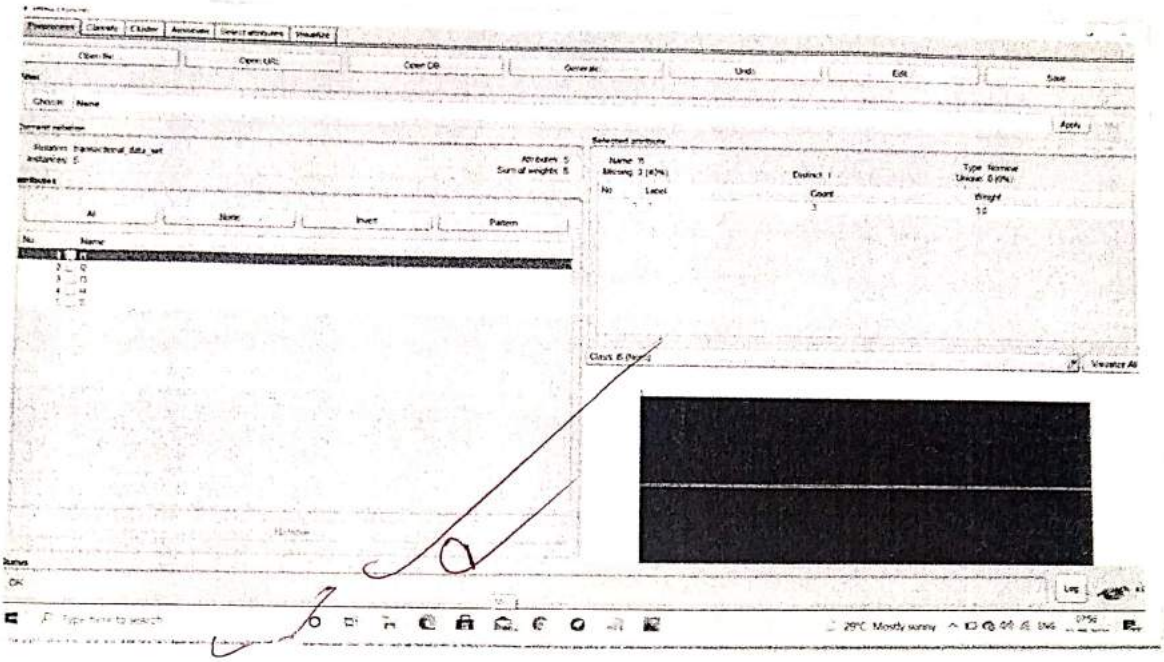
?, 1, 1, ?, ?

1, 1, ?, 1, ?

1, ?, 1, ?, ?

Steps to load:-

1. open explorer in weka interface
2. preprocessor is the default tab of explorer.
3. select openfile & search for required arff file.
4. Load the dataset
5. screenshot the graph.



Experiment-8

Aim:-

Create csv file with all types of attributes & explore the steps using weka for data load.

Description:-

How to create csv file

1. Open text editor
2. List all attributes we want to specify seperated by ','.
3. Enter data to attributes.
4. Save file with filename.csv

Student.csv

student, id, student_name, branch

1. Karishma, cse
2. Afreen, cse
3. Heena, eee

Steps to load csv file:

1. Open explorer in weka interface
2. preprocess is default tab in explorer.
3. select open file & search for required csv file
4. Load the dataset
5. Screenshot the graph.

Experiment-9

Aim:-

write a program to calculate dist matrix for given set of 2 dimensional points.

Description:-

The dist b/w 2 points (x_1, y_1) and (x_2, y_2) is

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

program:-

```
import math
x = list(map(int,input().split()))
y = list(map(int,input().split()))
for i in range(len(x)):
        for j in range(len(y)):
            d = ((x[i]-x[j])**2+(y[i]-y[j])**2)**0.5)
            print("%.2f", %d , end=' ')
        print()
```

Output:

```
1 2 3
4 5 6
0.00 1.41 2.83
1.41 0.00 1.41
2.83 1.41 0.00
```

Experiment -10

Aim:-

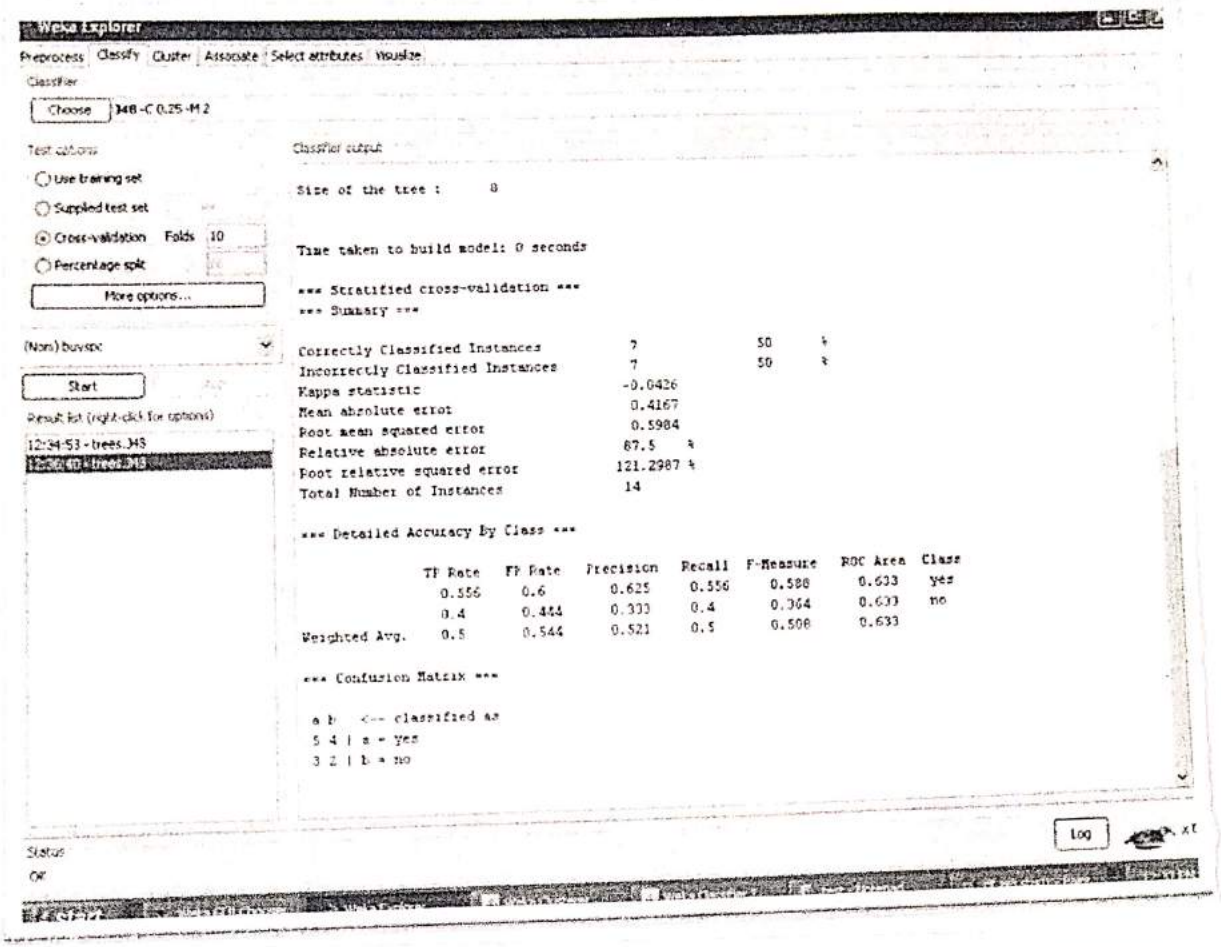
Using weka list the steps involved for generating decision tree (J48, Naive Bayesian algorithm)

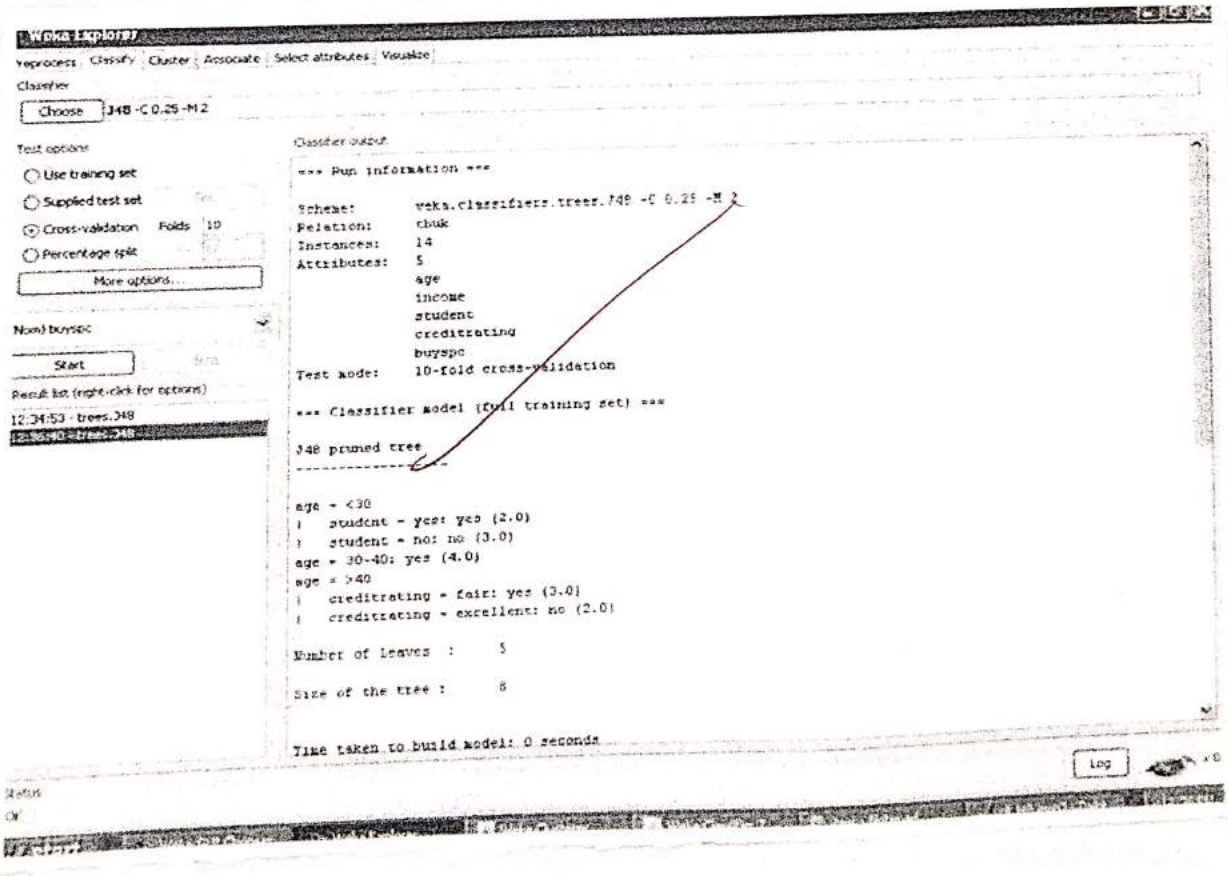
Description:-

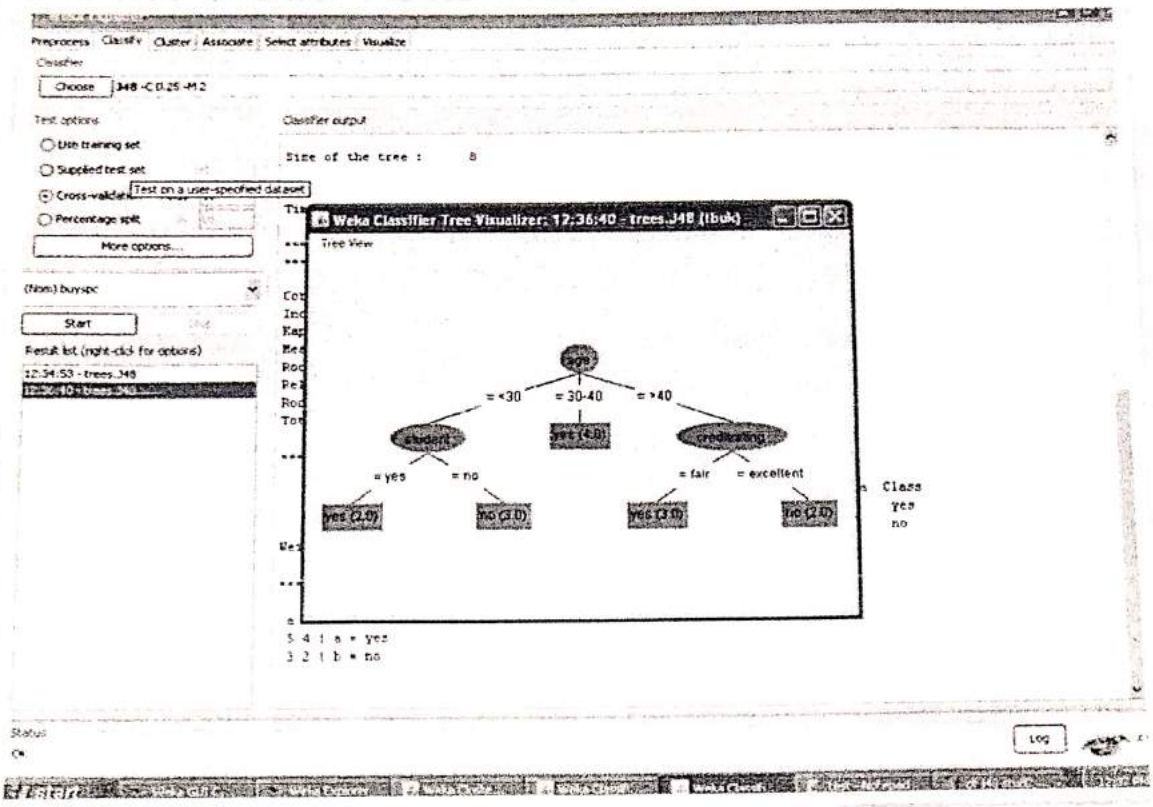
This exp illustrate the use of J48 classifier in weka. the sample dataset used in this exp is "employee" data available at arff format. This doc assumes that app data preprocessing has been performed.

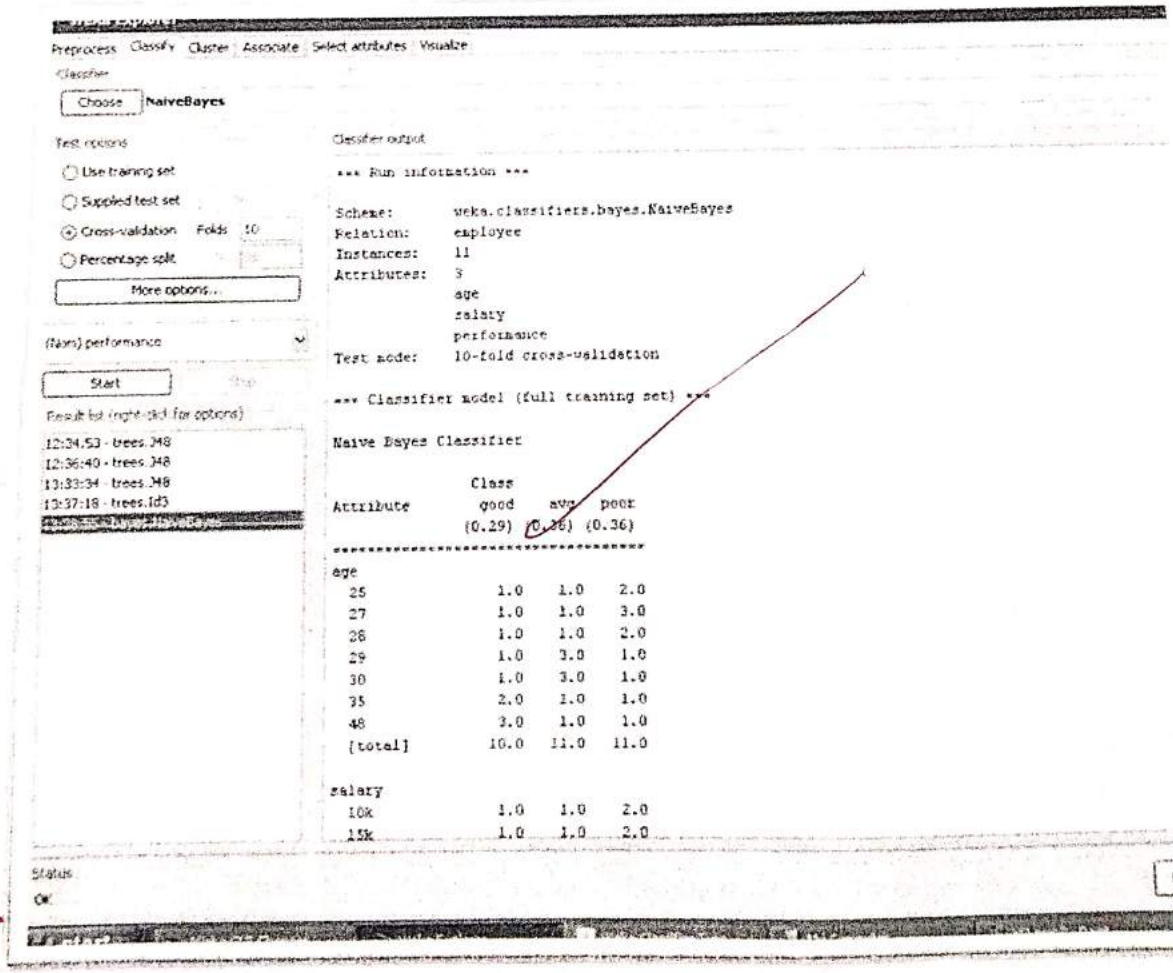
Steps:

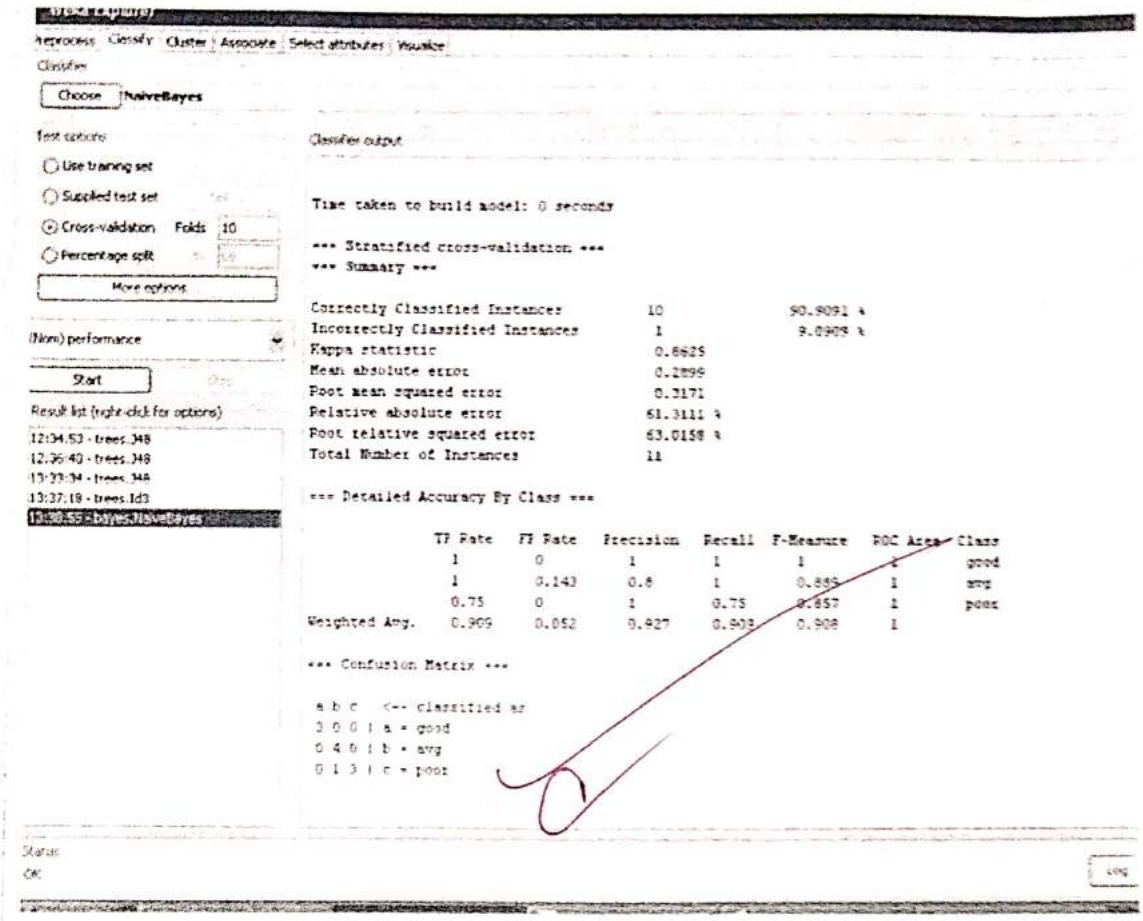
1. Loading the data into weka
2. select 'classify' tab & click 'choose' button to select 'J46' classifier.
3. specify various parameters, They can be specified by clicking text box to right of choose button.
4. Under 'text' options in main panel. we select the 10- fold cross validation as our evaluation app. se we don't have seperable data, this is necessary to get idea of accuracy of model.
5. click 'start' to generate model
6. Note the classification accuracy of model is about.


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Cross-validation Folds 10
Percentage split
More options...
(Nom) buyspc
Start
Result list (right-click for options)
12:34:53 - trees.J48
Classifier output
Size of the tree : 8
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 7 50 %
Incorrectly Classified Instances 7 50 %
Kappa statistic -0.0426
Mean absolute error 0.4167
Root mean squared error 0.5984
Relative absolute error 87.5 %
Root relative squared error 121.2987 %
Total Number of Instances 14
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
0.556 0.6 0.625 0.556 0.588 0.633 yes
0.4 0.444 0.333 0.4 0.364 0.633 no
Weighted Avg. 0.5 0.544 0.521 0.5 0.508 0.633
=== Confusion Matrix ===
a b <-- classified as
5 4 | a = yes
3 2 | b = no
Log
Status
OK


Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Cross-validation Folds 10
Percentage split
More options...
(Nom) buyspc
Start
Result list (right-click for options)
12:34:53 - trees.J48
Classifier output
=== Run information ===
Scheme: weka.classifiers.trees.J48 -C 0.25 -M 2
Relation: cbuk
Instances: 14
Attributes: 5
age
income
student
creditrating
buyspc
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
----
age = <30
| student = yes: yes (2.0)
| student = no: no (3.0)
age = 30-40: yes (4.0)
age = >40
| creditrating = fair: yes (3.0)
| creditrating = excellent: no (2.0)
Number of Leaves : 5
Size of the tree : 8
Time taken to build model: 0 seconds
Log
Status
OK


Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Cross-validation
Percentage split
More options...
(Nom) buyspc
Start
Result list (right-click for options)
12:34:53 - trees.J48
Classifier output
Size of the tree : 8
Weka Classifier Tree Visualizer: 12:36:40 - trees.J48 (ibuk)
Tree View
age
= <30
= 30-40
= >40
student
yes (4.0)
credit rating
= yes
= no
= fair
= excellent
yes (2.0)
no (3.0)
yes (3.0)
no (2.0)
Class
yes
no
5 4 | a = yes
3 2 | b = no
Status
OK
Log


Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose NaiveBayes
Test options
Use training set
Supplied test set
Cross-validation Folds 10
Percentage split
More options...
(Nom) performance
Start
Result list (right-click for options)
12:34:53 - trees.J48
12:36:40 - trees.J48
13:33:34 - trees.J48
13:37:18 - trees.Id3
Classifier output
=== Run information ===
Scheme: weka.classifiers.bayes.NaiveBayes
Relation: employee
Instances: 11
Attributes: 3
age
salary
performance
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
Naive Bayes Classifier
Class
Attribute good avg poor
(0.29) (0.36) (0.36)
age
25 1.0 1.0 2.0
27 1.0 1.0 3.0
28 1.0 1.0 2.0
29 1.0 3.0 1.0
30 1.0 3.0 1.0
35 2.0 1.0 1.0
48 3.0 1.0 1.0
[total] 10.0 11.0 11.0
salary
10k 1.0 1.0 2.0
15k 1.0 1.0 2.0
Status
OK
Log


Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose NaiveBayes
Test options
Use training set
Supplied test set
Cross-validation Folds 10
Percentage split
More options
(Nom) performance
Start
Result list (right-click for options)
12:34:53 - trees.J48
12:36:40 - trees.J48
13:33:34 - trees.J48
13:37:18 - trees.Id3
13:38:55 - bayes.NaiveBayes
Classifier output
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 10 90.9091 %
Incorrectly Classified Instances 1 9.0909 %
Kappa statistic 0.8625
Mean absolute error 0.2899
Root mean squared error 0.3171
Relative absolute error 61.3111 %
Root relative squared error 63.0158 %
Total Number of Instances 11
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure ROC Area Class
1 0 1 1 1 1 good
1 0.143 0.8 1 0.889 1 avg
0.75 0 1 0.75 0.857 1 poor
Weighted Avg. 0.909 0.052 0.927 0.909 0.908 1
=== Confusion Matrix ===
a b c <-- classified as
3 0 0 | a = good
0 4 0 | b = avg
0 1 3 | c = poor
Status
OK
Log

Experiment-11

Aim:- using weka list the steps involved in generating frequent pattern

Description:-

Steps:

1. Open datafile in weka explorer. It is presume that required datafields have been discretized. In this it is age attribute.
2. Clicking an attribute will bring up interface for association rule algorithm
3. we will use apriopri algorithm
4. Inorder to change the parameter for run, click on testbox.

Dataset:

```
@ relation contactlenses
@ attribute age {young, pre-prubyopic, presbyopic}
@ attribute spectaculus - prescrip {myope, hyper metrope}
@ attribute astigmatism {yes, no}
@ attribute tear-prod-rate {normal, reduced}
@ attribute contact-lenses {none, suff, hard}
@ data
    young, myope, no, reduced, none
        == Run  Inf ==

Scheme: weka. associations apriopri-N 10 -To -C 0.9
    -D  0.05  -O  1.0  -M0.1  -S 1.0  -C  -1

Relation: contact_lenses
    Instances: 24
    Attributes: 5
```

Attributes : 5

age

spectacle_prescrip

tear_prod_rate

contact_lenses

== Associator model ===

Apriopri

===

Min support = 0.2

Min metric <confidence> : 0.9

NO. Of cycle : 16

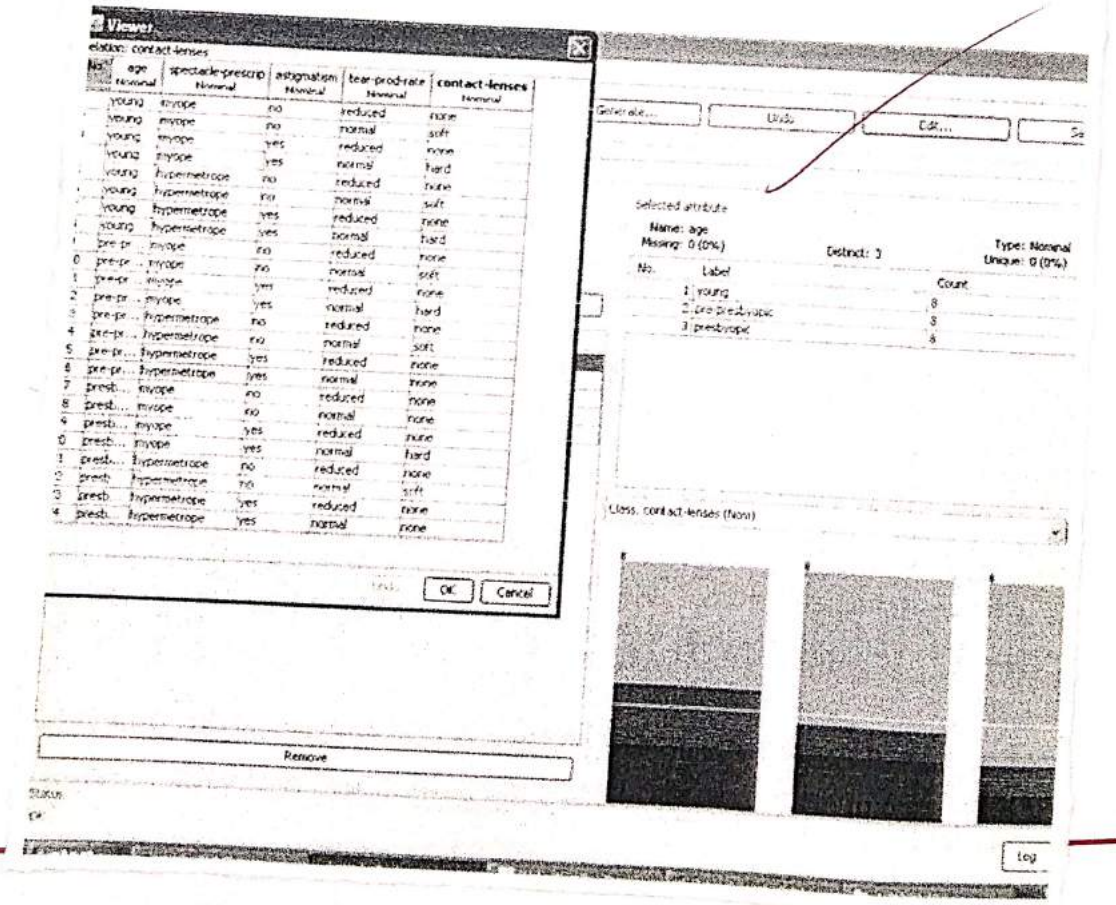
Generated sets of large

Size of set of Large itemset L(1) = 11

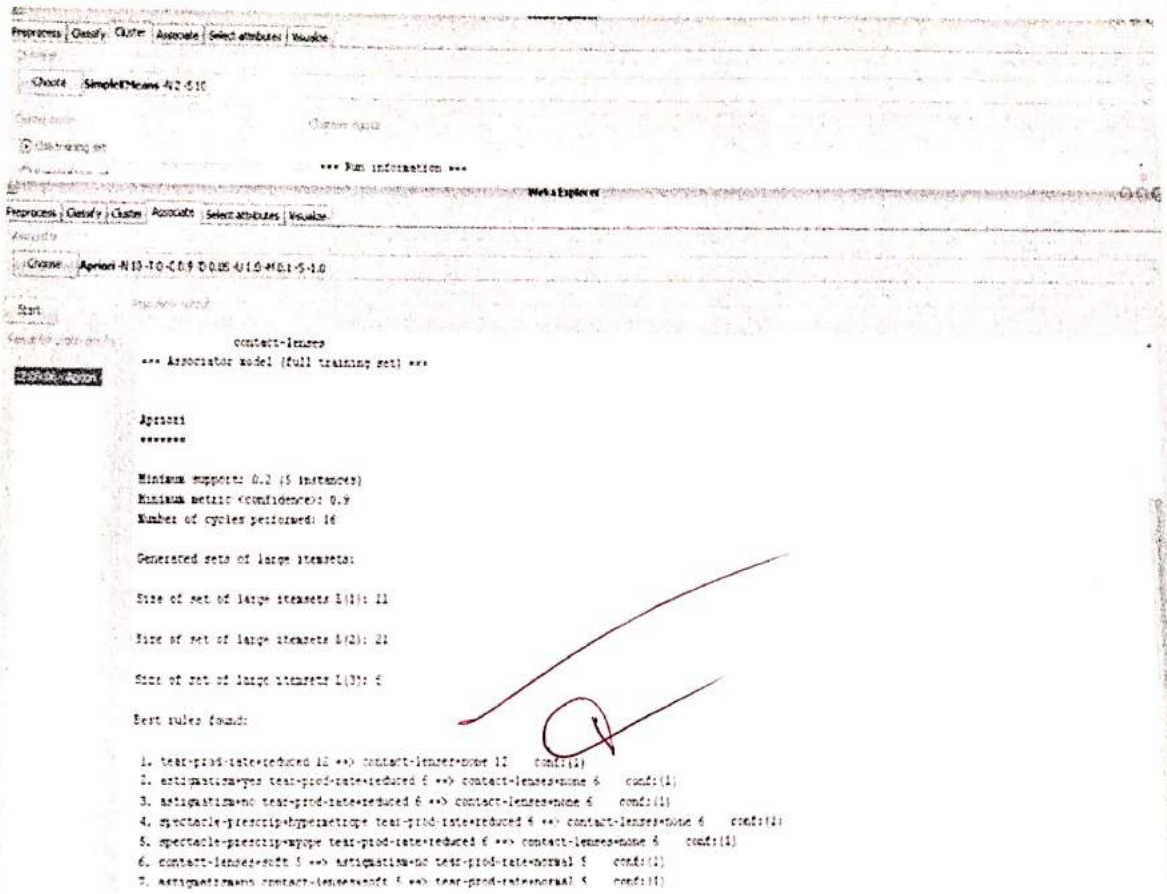
L(2) = 21

L(3) = 6

contactlenses.arff



Regd. No. 5H6



```
Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Choose  SimpleKMeans -N 2 -S 10

                                  === Run information ===

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Choose  Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0

Start

                    contact-lenses
=== Associator model (full training set) ===


Apriori
=======

Minimum support: 0.2 (5 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 16

Generated sets of large itemsets:

Size of set of large itemsets L(1): 11

Size of set of large itemsets L(2): 21

Size of set of large itemsets L(3): 6

Best rules found:

 1. tear-prod-rate=reduced 12 ==> contact-lenses=none 12    conf:(1)
 2. astigmatism=yes tear-prod-rate=reduced 6 ==> contact-lenses=none 6    conf:(1)
 3. astigmatism=no tear-prod-rate=reduced 6 ==> contact-lenses=none 6    conf:(1)
 4. spectacle-prescrip=hypermetrope tear-prod-rate=reduced 6 ==> contact-lenses=none 6    conf:(1)
 5. spectacle-prescrip=myope tear-prod-rate=reduced 6 ==> contact-lenses=none 6    conf:(1)
 6. contact-lenses=soft 5 ==> astigmatism=no tear-prod-rate=normal 5    conf:(1)
 7. astigmatism=no contact-lenses=soft 5 ==> tear-prod-rate=normal 5    conf:(1)
```

Experiment-12

Aim:-

write a program clustering using K-means.

Description:-

K-means clustering is a type of unsupervised learning which is used when you have unlabeled data.

Program:-

```
import math
def euclidean_distance(a,b):
    V1 = (b[0]-a[0])**2
    V2 = (b[1]-a[1])**2
    return math.sqrt(V1+V2)
def check(clusters):
    belongs to = []
    for i in range(0,n):
        values = []
        for j in range(0,k):
            V = euclidean_distance(points[i], clusters[j])
            values.append(v)
        m = values[0]
        index = 0
        for j in range(0,len(values)):
            if values[j]<m:
                m = values[j].
                index = j
        belongs_to.append(index)
```

```
print ("- - - -")
for j in range(0, K):
        print ("cluster", j+1, "contains the pts"),
        for i in range(0, len(belongs_to)):
                if belongs_to [i] == j:
                        print (points [i])
  return belong_to
  x = int(input("enter no.of clusters: "))
  n = int(input("enter no.of points: "))
  if K>n:
        print ("each pt can be taken as cluster as number of
        pts is equal to the no. of clusters").
        exit(0)
points = []
for i in range(0, n):
        print ("enter the point", i+1)
        points.append ((float(input()), float(input())))
char clusters = []
for i in range(0,K):
      print ("enter cluster,", i+1, "point")
      clusters.append((float(input()), float(input())))
iteration = 1
belongs_to = []
for i in range(0, n):
            values = []
```

```
for j in range (0,K):
        v= euclidean_distance (points[i], clusters[j])
        values.append(v)
m=values[0]
index=0
for j in range(0, len(values)):
        if value[j]<m:
                m=values[j]
                index=j
belongs_to.append(index)
print ("------")
print ("iteration" iteration)
print ("------")
for j in range(0,K):
        print (clusters[j])
for j in range(0,K):
        print ("cluster", j+1, "contains pts")
        for i in range(0,len(belongs_to)):
                if belong_to[i]==j:
                        print (points[i])
c=999
while c>0:
        iteration = iteration-1
        print ("iteration", iteration)
        c=0
```

```
Clusters = []
for j in range(0, n):
        l = []
        x = 0
        y = 0
        count = 0
        for j in range(0,n):
                if belongs_to[i] == j:
                        count += 1
                x = x + points[i][0]
                y = y + points[i][1]
        Clusters.append((x(count * 1.0), y(count * 1.0))
print ("Centroid values are:")
for j in range(0, K):
        print (clusters[j])
belongs_to_new = check(clusters)
count = 0
for j in range(0,n):
        if belongs_to[j] == belongs_to_new[j]:
                continue
        else:
                c = c+1
for j in range(0, n):
        belongs_to[j] = belongs_to_new[j].
```

Output:-

```
enter no. of clusters: 3
enter no. of pts: 8
enter the pt 1
2
10
enter the pt 2
2
5
enter the pt 3
8
4
enter the pt 4
5
8
enter the pt 5
7
5
enter the pt 6
6
4
enter the pt 7
1
2
enter the pt 8
4
9
enter the cluster 1 point
2
10
enter the cluster 2 point
5
8
enter the cluster 3 point
1
2
```

Iteration 1

Centrord values are

[2.0, 10.0]

[5.0, 8.0]

[1.0, 2.0]

cluster 1 contains the points

[2.0, 10.0]

cluster 2 contains the points

[8.0, 4.0]

[5.0, 8.0]

[7.0, 5.0]

[6.0, 4.0]

[4.0, 9.0]

cluster 3 contains the points

[2.0, 5.0]

[1.0, 2.0]

Iteration 2

centrod values are:

[2.0, 10.0]

[6.0, 6.0]

[1.5, 3.5]

cluster1 contain the points

[2.0, 1.0]

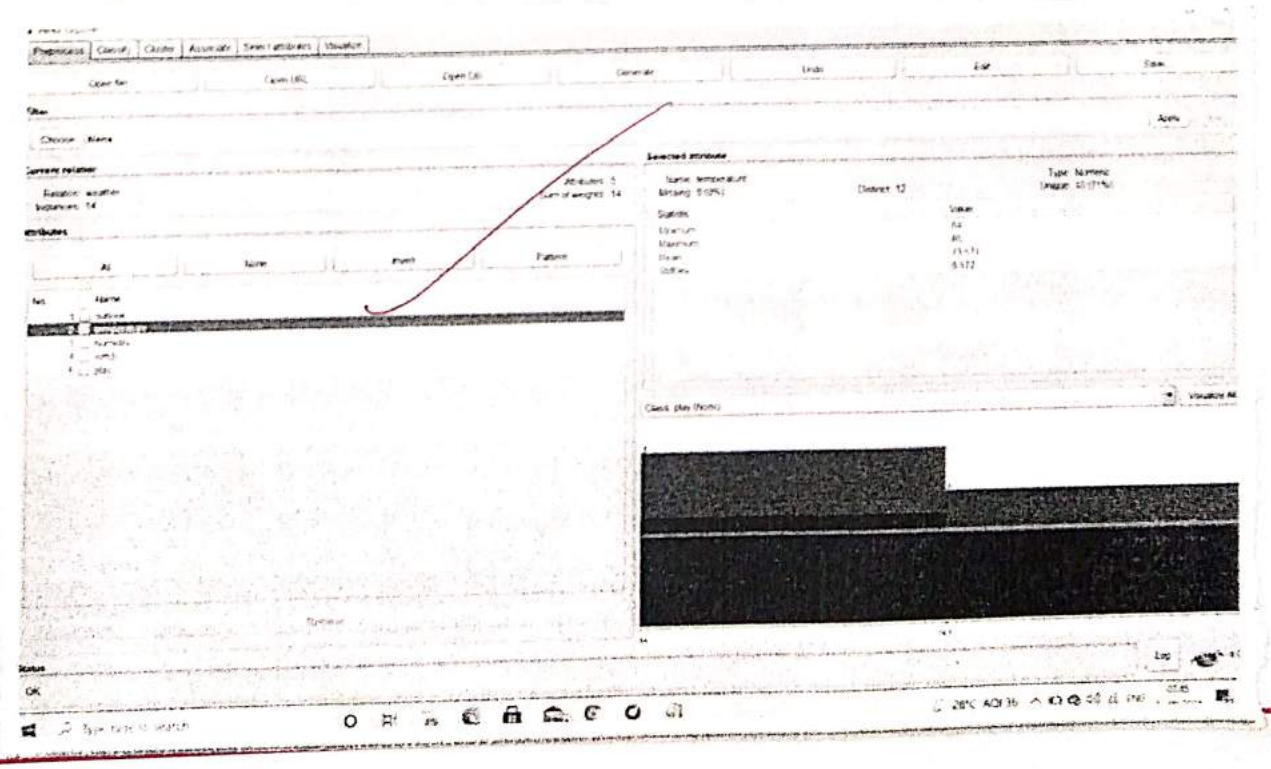
Experiment-13

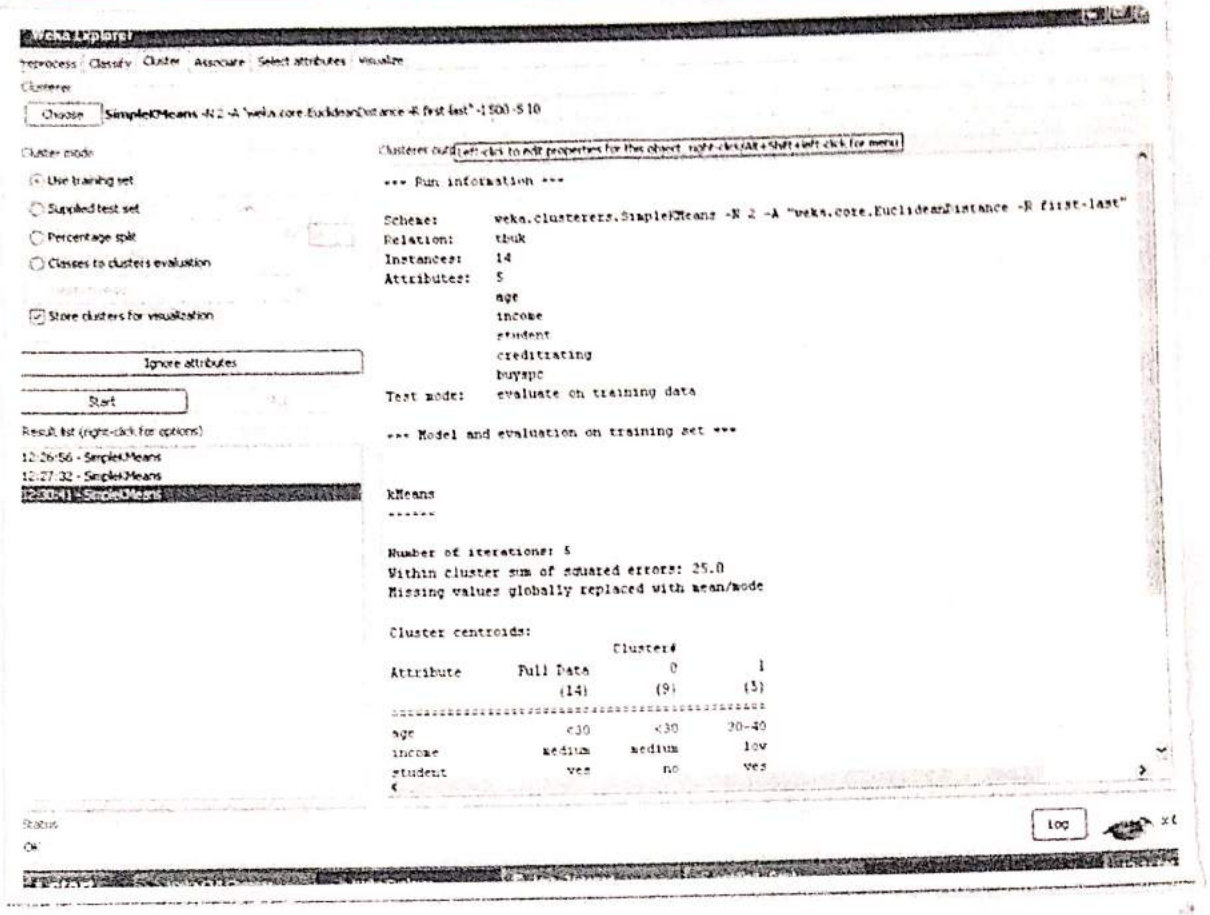
Aim:-

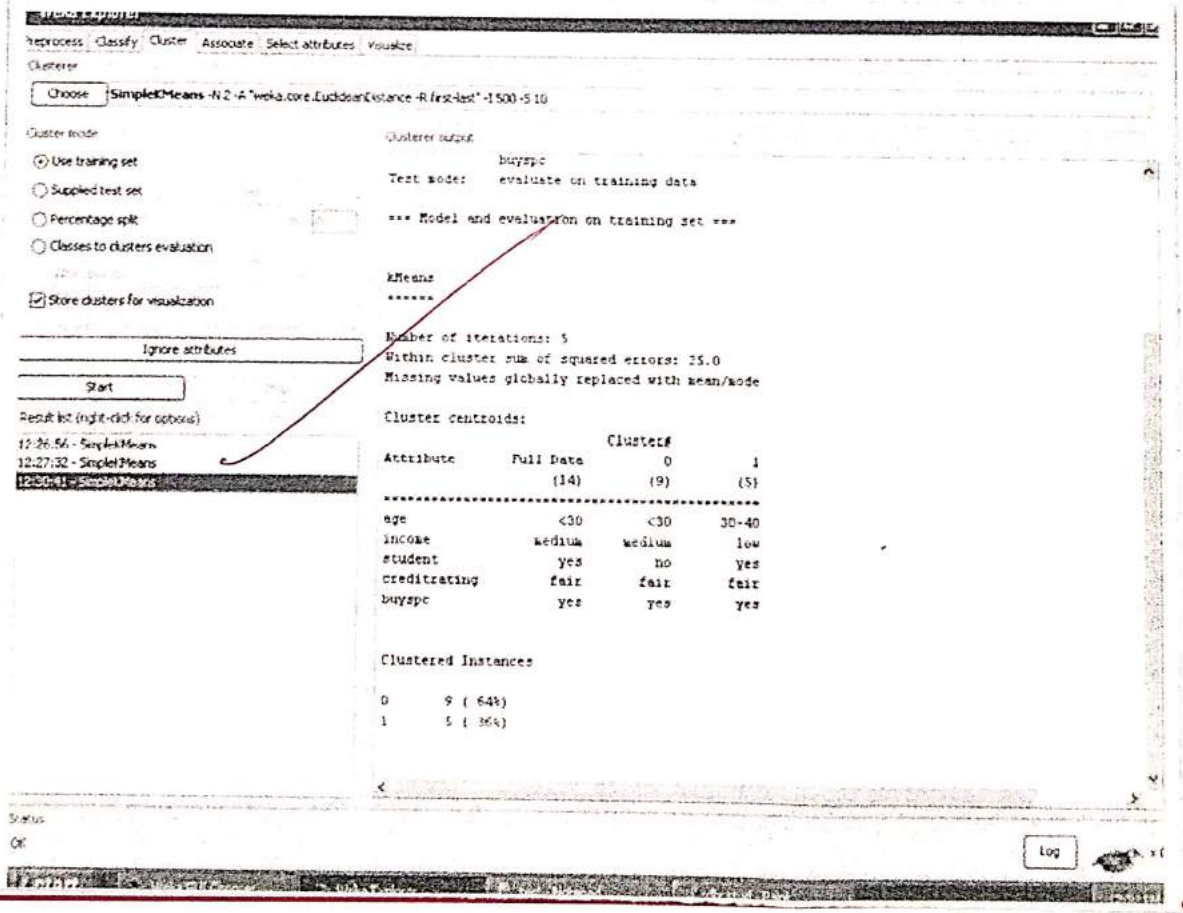
Using weka list the steps involved for generating cluster using K-means

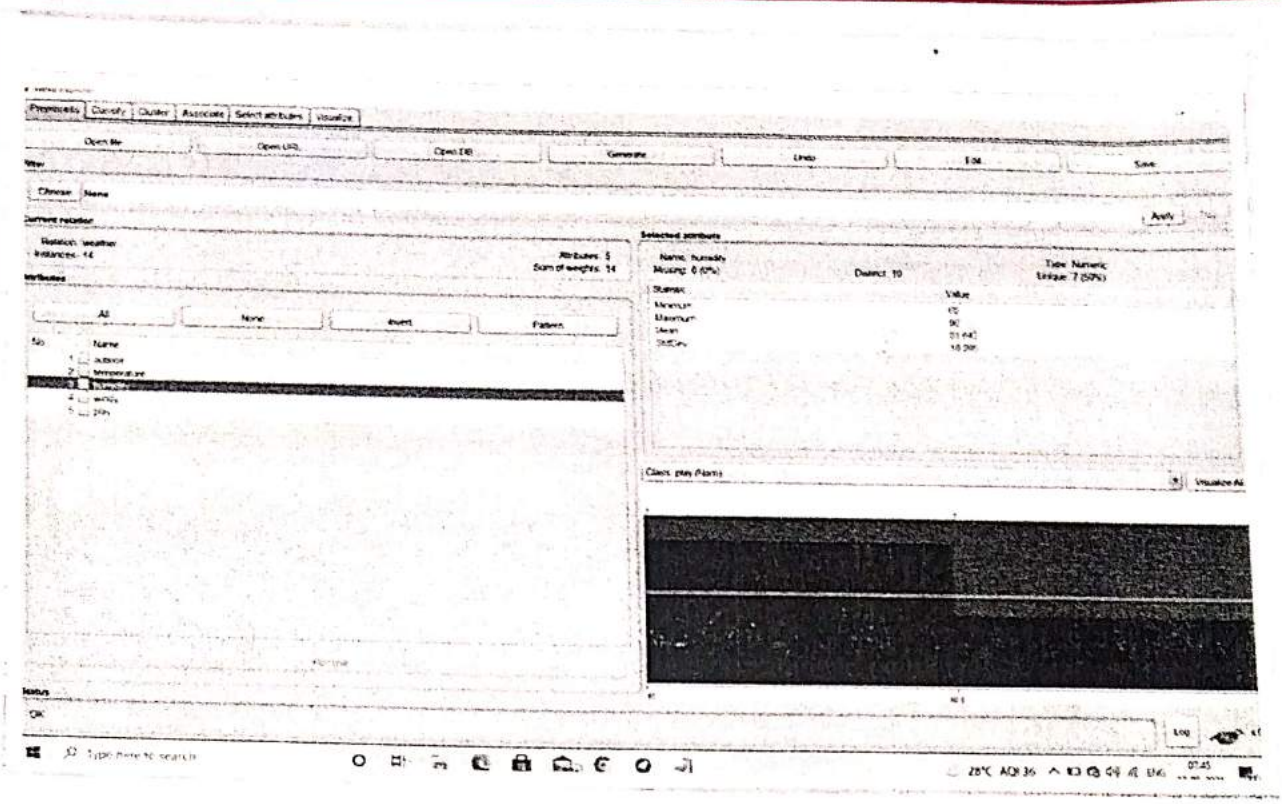
Steps:

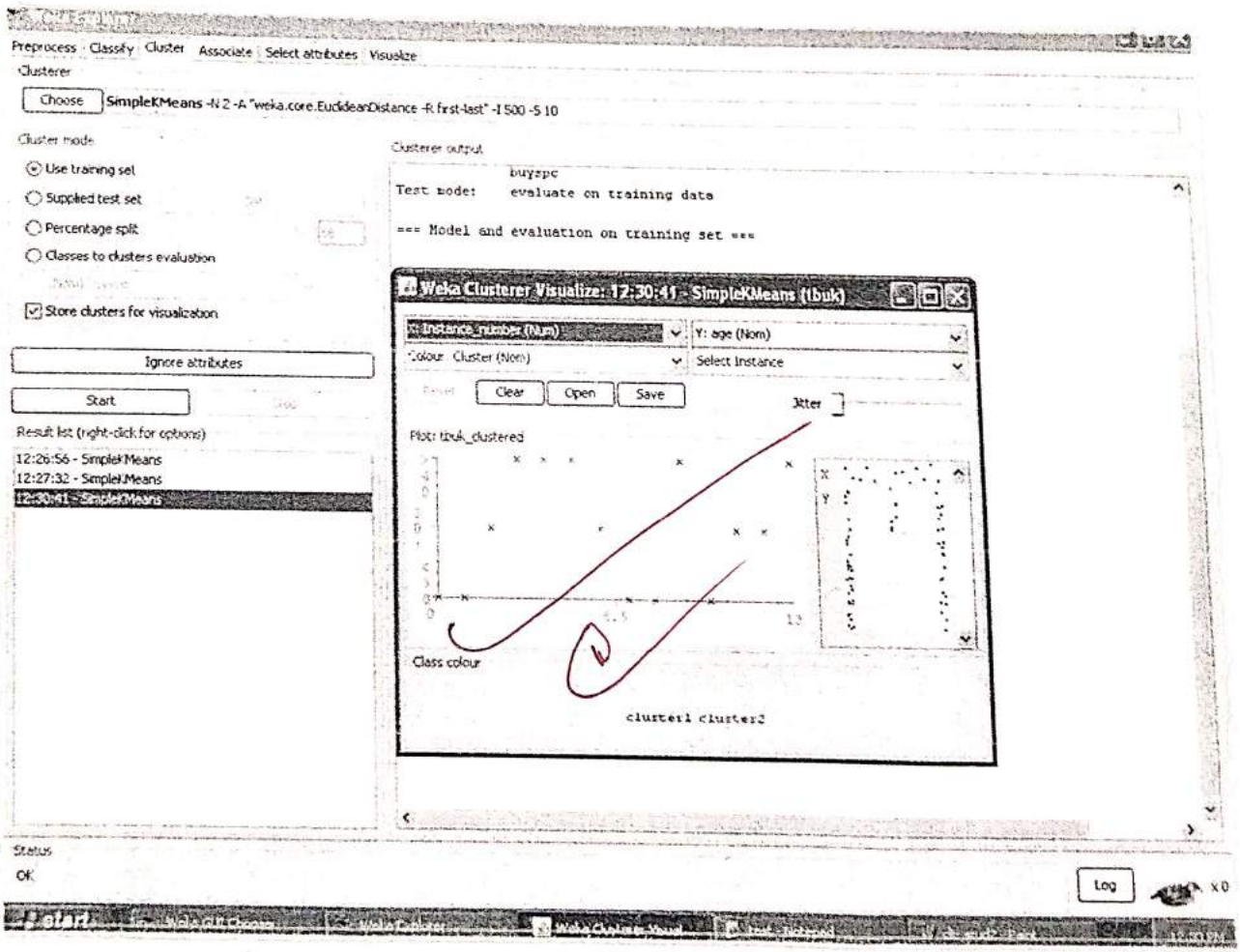
1. Run the weka explorer & load the file iris arff in preprocessing interface
2. Inorder to perform clustering select the cluster tab in the explorer & click on 'choose'.
3. In case, we select simple K-means
4. Click text button to get popup window shown in ss
5. One of options have been specified. we ran the clustering alg these we must make that they are in cluster mode panel
6. The result window of centroid of each cluster is shown
7. Another way of understanding characteristics of each cluster through visualization we can do this




Preprocess Classify Cluster Associate Select attributes Visualize
Clusterer
Choose SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Cluster mode
Use training set
Supplied test set
Percentage split
Classes to clusters evaluation
Store clusters for visualization
Ignore attributes
Start
Result list (right-click for options)
12:26:56 - SimpleKMeans
12:27:32 - SimpleKMeans
12:30:41 - SimpleKMeans
Clusterer output
=== Run information ===
Scheme: weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last"
Relation: tbuk
Instances: 14
Attributes: 5
age
income
student
creditrating
buyspc
Test mode: evaluate on training data
=== Model and evaluation on training set ===
kMeans
Number of iterations: 5
Within cluster sum of squared errors: 25.0
Missing values globally replaced with mean/mode
Cluster centroids:
Cluster#
Attribute Full Data 0 1
(14) (9) (5)
age <30 <30 30-40
income medium medium low
student yes no yes
Status
OK
Log


Preprocess Classify Cluster Associate Select attributes Visualize
Clusterer
Choose SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Cluster mode
Use training set
Supplied test set
Percentage split
Classes to clusters evaluation
Store clusters for visualization
Ignore attributes
Start
Result list (right-click for options)
12:26:56 - SimpleKMeans
12:27:32 - SimpleKMeans
12:30:41 - SimpleKMeans
Clusterer output
buyspc
Test mode: evaluate on training data
=== Model and evaluation on training set ===
kMeans
Number of iterations: 5
Within cluster sum of squared errors: 25.0
Missing values globally replaced with mean/mode
Cluster centroids:
Cluster#
Attribute Full Data 0 1
(14) (9) (5)
age <30 <30 30-40
income medium medium low
student yes no yes
creditrating fair fair fair
buyspc yes yes yes
Clustered Instances
0 9 (64%)
1 5 (36%)
Status
OK
Log




Preprocess Classify Cluster Associate Select attributes Visualize
Clusterer
Choose SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Cluster mode
Use training set
Supplied test set
Percentage split
Classes to clusters evaluation
Store clusters for visualization
Ignore attributes
Start
Result list (right-click for options)
12:26:56 - SimpleKMeans
12:27:32 - SimpleKMeans
12:30:41 - SimpleKMeans
Clusterer output
buyspc
Test mode: evaluate on training data
=== Model and evaluation on training set ===
Weka Clusterer Visualize: 12:30:41 - SimpleKMeans (tbuk)
X: Instance_number (Num)
Y: age (Nom)
Colour: Cluster (Nom)
Select Instance
Clear
Open
Save
Jitter
Plot: tbuk_clustered
Class colour
cluster1 cluster2
Status
OK
Log