Experiment - 1a

Aim: Write a program for transformation numeric data using z-score normalization.

Description:

Normalization: where the data values 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 lineal transfomation, 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 / z-score Normalization")
new_val = (val - st.mean(array))/ st.stdev(array)
print(new_val)
```

O/P: enter value into array -123 256 -4548 -36a 745

enter the value 47

z-score normalization

-0.122.

Experiment - 1b

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

Description:

Normalization: where the data values 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 normalized the attributes by moving the decimal point of values

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

program:

```
import statistics 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,j))
print(new_val)
```

o/p:

```
enter value into array -193 256 -4548 -369 745
enter the value 47.
Min max Normalization
0.373.
```

Experiment - 2a.

Aim: write a program for smoothing numeric data with data-bin sizes 2, 3, 5.

Description: Smoothing refers for removing noise data from data set. 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:
    bin_mean = st.mean(i)
    for j in i:
        i[j] = bin_mean
    print("\n print Bin "+i+"mean is")
    print (bin_mean)
    print ("\n after binning elements of bin"+i+"are")
    for j in i:
        print(i[j],end=" ")
```

O/P:-

```
Enter the values to bin
3 2 2 4 3 3 1 0 0
Enter number of bins 3
Enter bin 0 size 2
Enter bin 1 size 3
Enter bin 2 size 5
Bin :1
3 2
Bin : 2
2 4 3
```

Experiment - 3

Aim: To implement the concept of Data discretization

Description:- Data discretization converts a large number of data value into simlas once. So that data evaluation and 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 minining results.

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

If the process starts by considered all of the valve 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,
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]="C";
    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 States :");
for(int i=0; i<n; i++)
{
    System.out.println("Marks and their States:");
    for(int i=0; i<n; i++)
    {
        System.out.println(a[i]+" "+b[i]);
    }
System.out.println("No. of student under distenction
        class are: " +count);
System.out.println("No. of student under firstclass
        are:" +count1);
System.out.println("No. of students under secondclass
        are:" +count2);
System.out.println("No. of students under thirdclass
        are:" + count3);
System.out.println("No. of students under fail
        are:" +count4);
    }
}
```

O/P: 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 States

90.0 D

85.1 D

76.1 D

29.8	f
38.5	f
67.8	fc
88.7	D
49.8	fc
71.9	D
30.8	f

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 Student under third class are : 1

No. of Student under fail are : 3.

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 the 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, Anjana, cse
2, Puja, It
3, Devi, cse
4, Geetha, ece
```

Steps to load arff file:-

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

4. Load the dataset

5. Screenshot the graph.

The attached screenshot represents the relation of the StudentId which is maximum, minimum and standard deviation.