NORMALIZATION

DATE :
CYCLE :
Exp. No. :

Aim:- Program to implement Normalization.

```
#include<iostream.h>
#include < math.h>
using namespace std;
int main()
{
   int n;
   double sum=0;
   cout<< "Enter the number of values";
    cin>>n;
   int a[n], r1, r2;
    for (int i=0;i<n;i++)
    {
        cin>>a[i];
     }
   cout<< "Enter starting & ending range for normalization:";
   cin>>r1>>r2;
   double max = a[0];
   for(int i=0;i<n;i++)
   {
       if (a[i] >max)
           max =a[i];
        else if
                min = a[i];
    }
        float r[n];
      double temp;
      for(int i=0; i<n; i++)
      {
          temp = double(double(double(a[i]-min)*(r2-r1))/
                                      (max-min)) +r1;
          v[i] = temp;
       }
     cout<<"Normalization by min-max"<<endl;
      cout<<"Values are:"<<endl;
      for(int i=0; i<n; i++)
            cout<<v[i]<<endl.
}
```

Output:-

```
Enter the number of values : 5
 85  80  83  70  68
Enter the starting & ending range for normalization: 0 1
Normalization by min-max
 Values are: 1
 0.705882
 0.882353
 0.117647
 0.
```

EUCLIDEAN, MANHATTAN, MINKOWSKI DISTANCE	DATE : CYCLE : Exp. No. :

Aim:- Program to calculate Euclidean, ~~d~~ manhattan & minkowski distances.

```
from decimal import Decimal
import math
from math import *
x =(5,6,7)
 y =(8,9,9)
 distance=math.sqrt(sum([a-b]**2 for a,b in zip(x,y)))}
 print("Euclidean distance", distance);
  n=len(x).
 for i in range (n):
        for j in range (i+1,n):
             sum+ =(abs(x[i]-x[j])+abs(y[i]-y[j]))
  print("Manhattan Distance:",sum);

vector 1 = [0,2,3,4]
vector 2 = [2,4,3,7]
 p=3
 def p-root(value, root):
    root_value = 1/ float (root).
  return round (Decimal(value)** Decimal (root_value), 3)
  def minkowski_distance (x, y, p_value):
      return (p_root (sum (pow (abs (a-b), p-value) for a,b in
                        zip (x,y)), p-value))
  print ("minkowski Distance:", minkowski_distance(vector1,
                                                    vector2, p))
```

Output:

```
Euclidean distance: 4.69041575982343
Manhattan distance: 22
Minkowski distance: 0.503.
```

CHI-SQUARE

DATE :
CYCLE :
Exp. No. :

Aim: Program to calculate Chi-square value

```
from scipy.stats import chi2-contigency
 #defining the table
  data = [[207,282,241],[234,242,232]]
  stat, p, dof, expected = chi2_contigency(data)
  # interpret p-value
    alpha = 0.05
   print ("p value is"+str(p))
   if p<=alpha:
        print('Dependent (reject Ho)').
   else
        print('Independent (Ho holds true)')
```

Output:

```
P value is 0.1031971404730939
Independent (Ho holds true)
```

CENTRAL TENDENCY

DATE :
CYCLE :
Exp. No. :

Aim: Program to implement measures of central tendancy.

```
#include <bits/stdc++.h>
 using namespace std;
float mean(float arr[], int n)
{    float sum=0;
      for(int i=0;i<n;i++)
           sum+=arr[i];
      return sum/n;
 }
float median(float arr[], int n)
{
      sort(arr, arr+n);
      if(n%2==0)
           return (arr[n/2-1] + arr[n/2])/2;

      return arr[n/2];
 }
float mode(float arr[], int n)
{
    sort(arr, arr+n);
    int max_count=1, res=arr[0], count=1;
```

```
for(int i=1; i<n; i++)
{   if (arr[i] == arr[i-1])
        count ++;
    else {
        if(count > max-count)
        {   max-count = count;
            res = arr[i-1];
        }
        count = 1;
    }
}
if (count > max_count)
{
    max-count = count;
    res~~t~~ = arr[n-1];
}
return res;
}
int main()
{
    int n;
    float arr[50];
    cout<<"Enter the size of array:";
    cin>>n;
    cout<<" Enter the elements of array:";
```

```
  for(int i=0;i<n;i++)
        cin>>arr[i];
 cout<<"Mean="<<mean(arr,n);
 cout<<"\n Median="<<median(arr,n);
 cout<<"\n Mode="<<mode(arr,n);
 return 0;
}
```

Output:-

```
Enter the size of array:10
Enter the elements of array: 4 8 7 6 4 2 4 4 1 2
 Mean = 4.2
 Median = 4
 Mode = 4
```