CLUSTERING IRIS DATASET

E / 16 / 103

Fernando P.D.R

K-means Clustering

For Linear , Euclidean Distance, number of clusters = 2

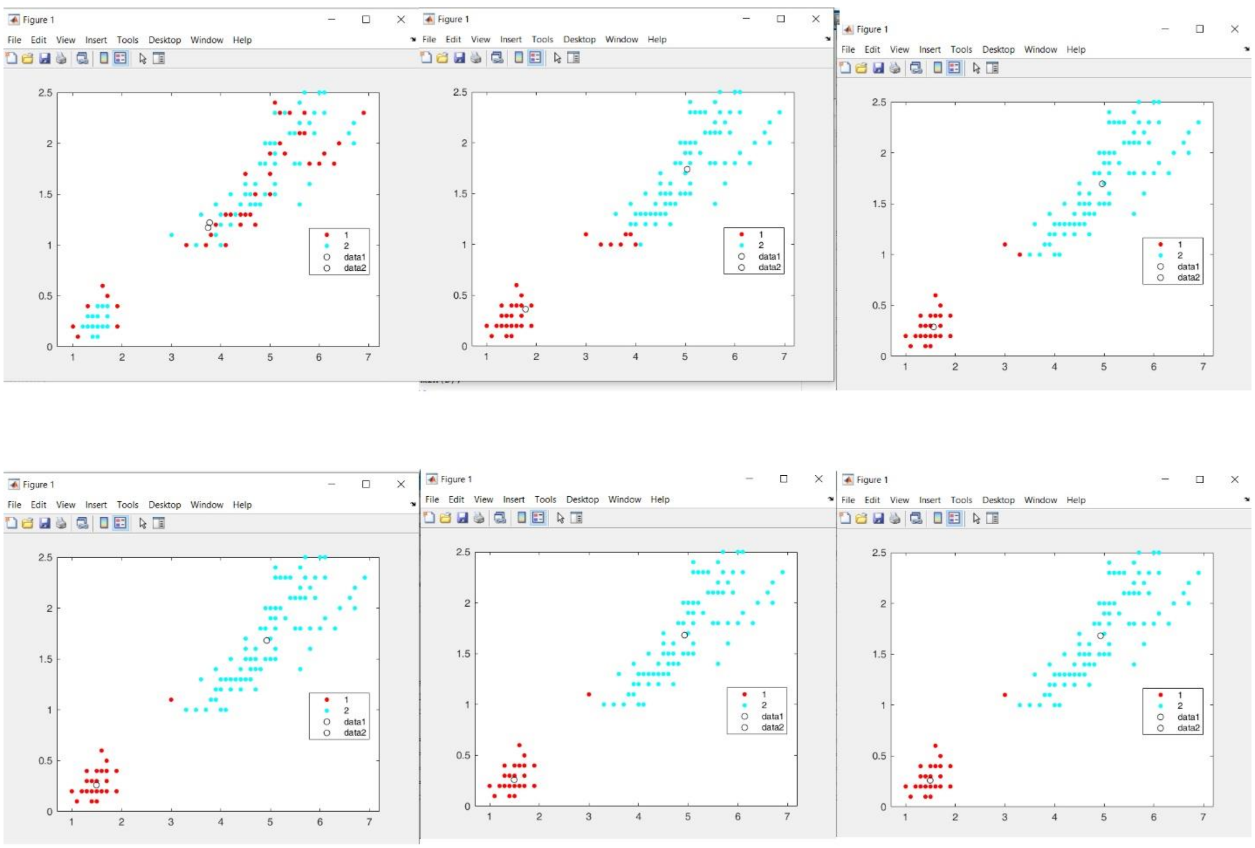
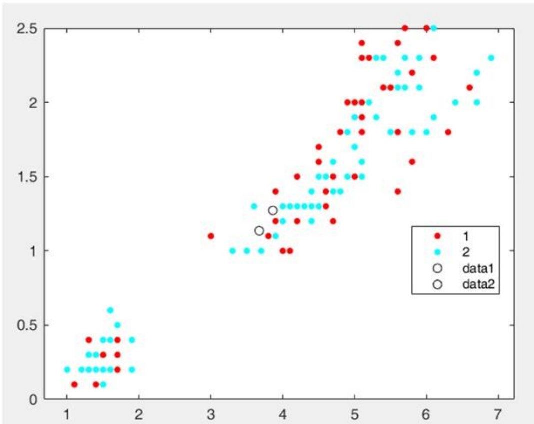


Figure 1 : Converging of means for Linear , Euclidean Distance, number of clusters = 2 ,Iterations

For Linear , Manhatten Distance, number of clusters = 2



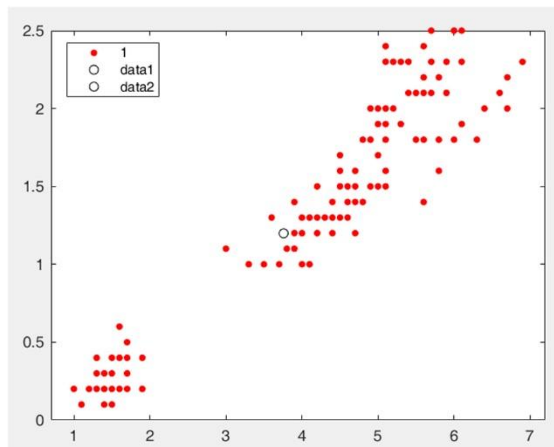
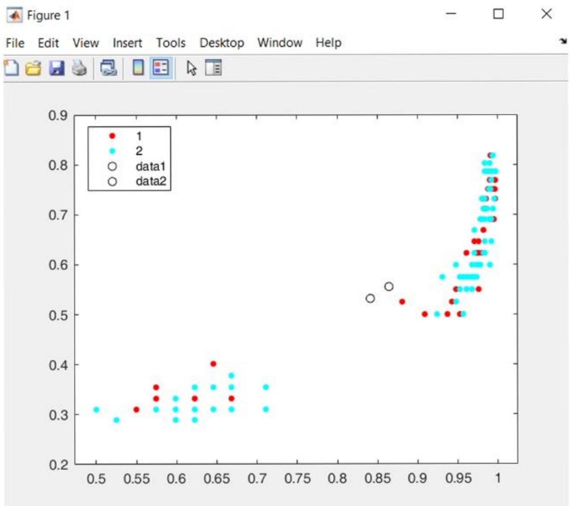


Figure 2 : Converging of means for Linear , Manhatten Distance, number of clusters = 2, Iterations

For Non - Linear , Euclidean Distance, number of clusters = 2



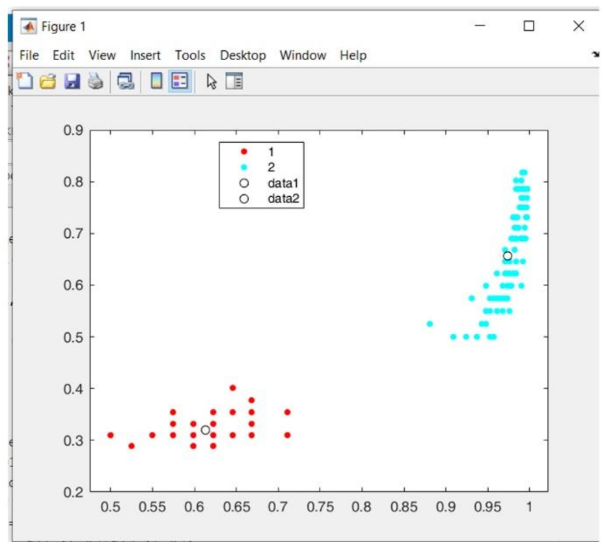


Figure 3 : Converging of means for Non - Linear , Euclidean Distance, number of clusters = 2 ,Iterations

For Linear , Euclidean Distance, number of clusters = 3

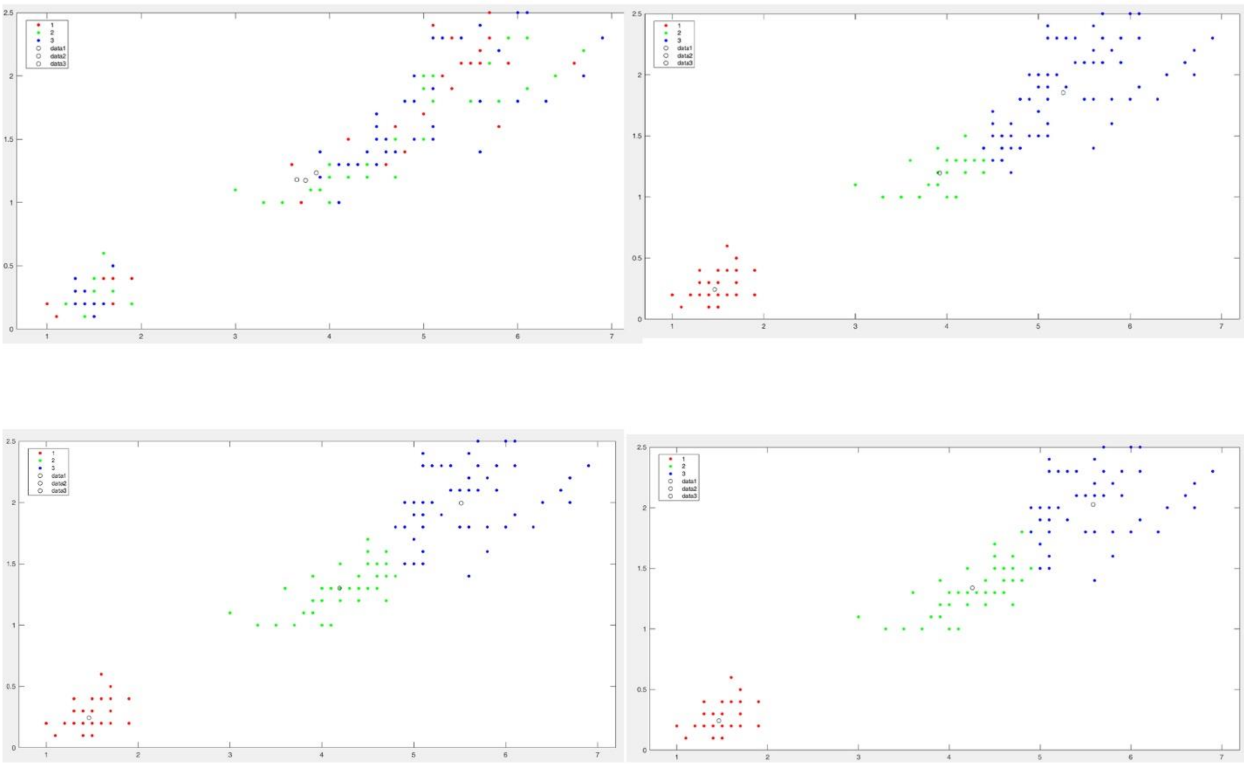


Figure 4 : Converging of means for Linear , Euclidean Distance, number of clusters = 3 ,Iterations

For Non - Linear , Euclidean Distance, number of clusters = 3

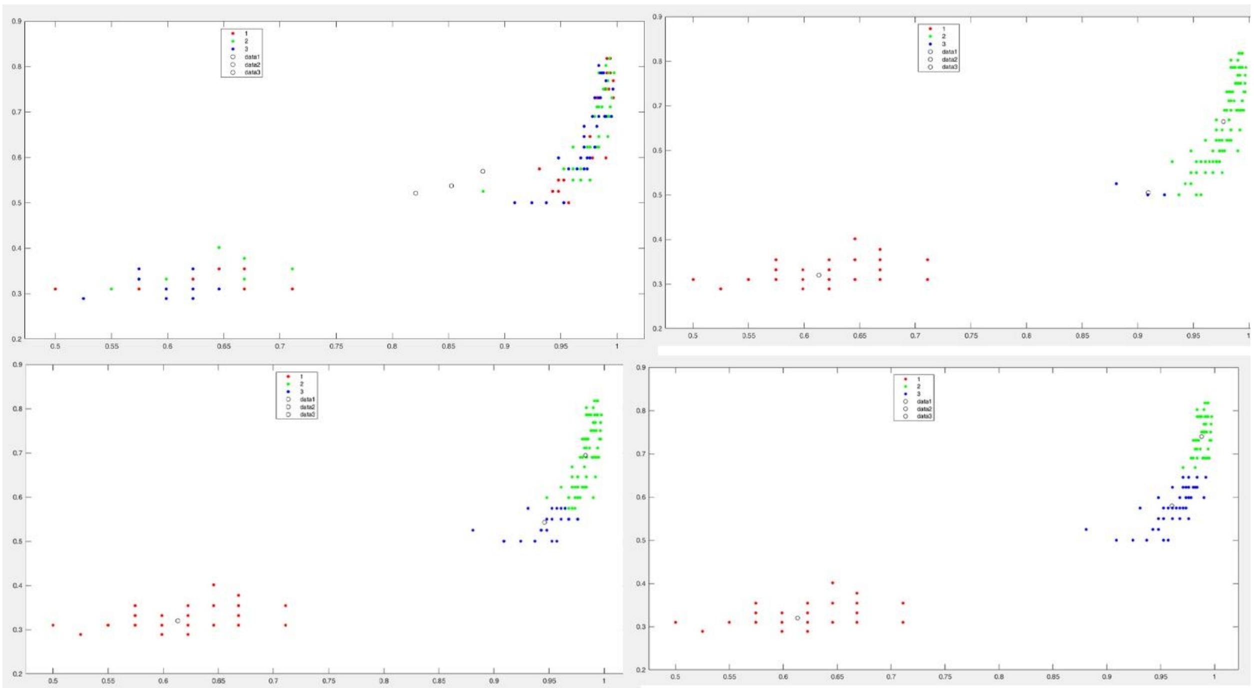


Figure 5 : Converging of means for Non - Linear , Euclidean Distance, number of clusters = 3 ,Iterations

Classification of 3 clusters

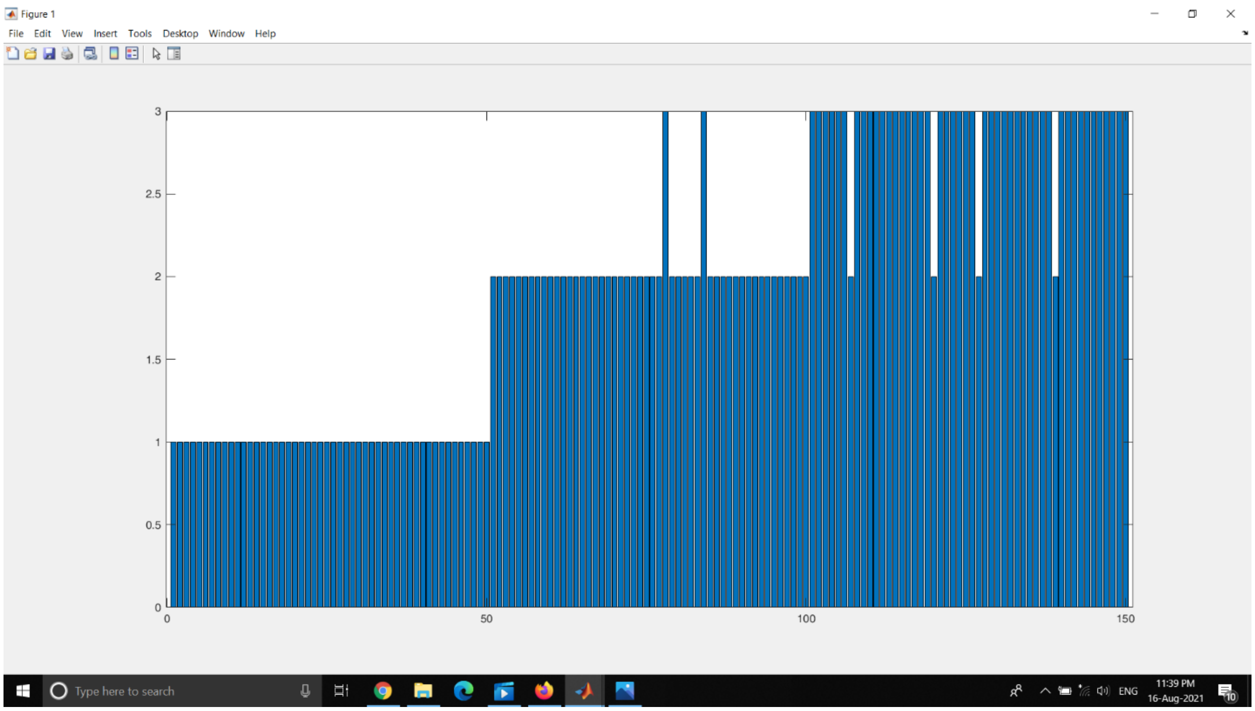


Figure 6 : Classification of cluster obtained by k means algorithm

DBSCAN Clustering

Steps Used in the DBSCAN code below

1. Finding Core Points By defining a minimum number of neighbors that should exist around a so called “Core Points”.
2. Assigning core points to clusters by assuming that core points which belongs to a certain cluster has at least one neighboring core point within its radius (ε).
3. Finding the boundary points and assigning them to clusters by considering the nearest neighboring core point’s cluster.

Core Points ,Boundary Points and Outliers

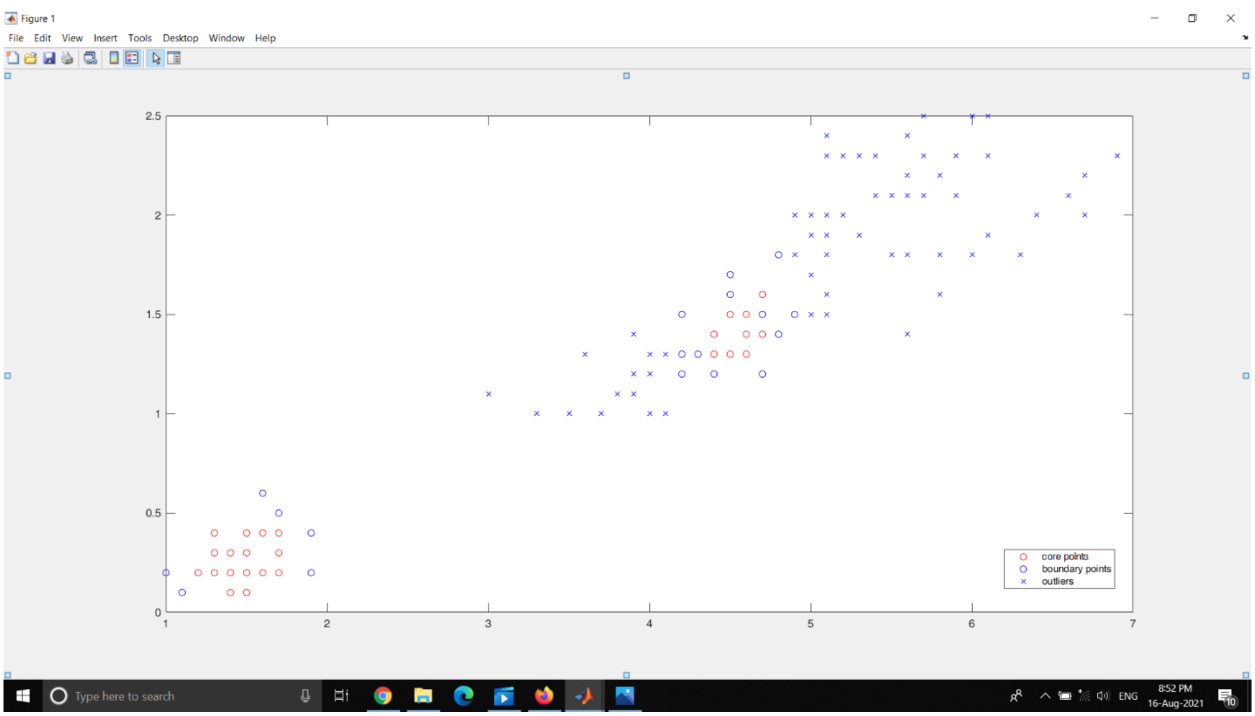


Figure 7 : Core Points, Boundary points & Outliers obtained through DBSCAN algorithm

Clustered Data Using DBSCAN

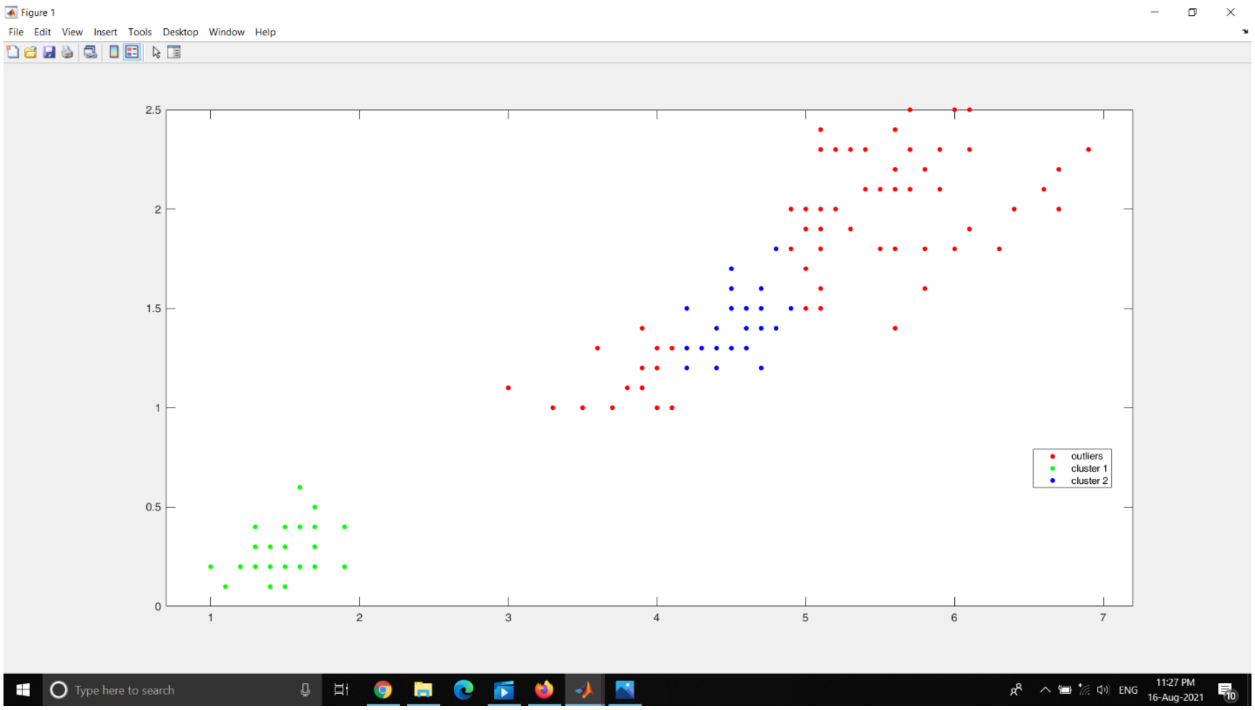


Figure 8 : Scatter plot of clustered data

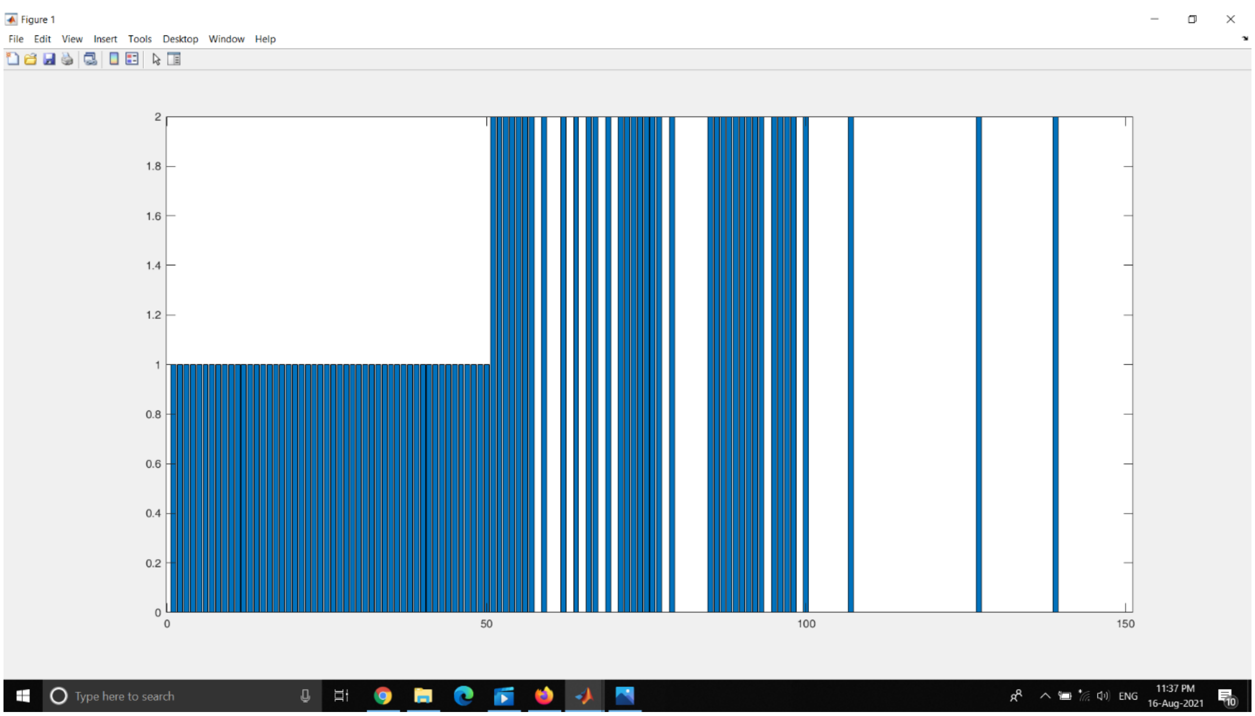


Figure 9 : Bar plot of clustered data

APPENDIX

MatLab code for k-means algorithm

```
x=iris_dataset;
x=x(3:4,:);

clsses=[ones(1,50) 2*ones(1,50) 3*ones(1,50)];
gscatter(x(1,:),x(2,:),clsses); hold on;

N = size(x,2);
k=3;
ix = randi(k,1,N);

for i=1:10
    C=[];
    D=[];
    gscatter(x(1,:),x(2,:),ix); hold on;
    for p=1:k
        ixd = find(ix==p);
        if (~isempty(ixd))
        mC = mean(x(:,ixd)');
        d = x-mC';
        d = d(1,:).^2+d(2,:).^2;
        D = [D;d];
        C = [C;mC];
        plot(mC(1),mC(2),'ko');
        end
    end

    [m,ix]=min(D);
    hold off;
    drawnow;
    pause;
end
```

MatLab code for DBSCAN algorithm

```
x=iris_dataset;
x = x(3:4,:);

min_neighbor_distance = .08;
min_neighbors_core = 14;

N = size(x,2);
core_pts = [];


for i=1:N
    d = x - x(1:2,i);
    d =  (d(1,:).^2 + d(2,:).^2)';
    neighbors = setdiff(find(d<min_neighbor_distance),i);
    %C = [C neighbors];
    %core points
    if(length(neighbors) > min_neighbors_core)
       core_pts=[core_pts i];
    end
end


C=[];
k=0;
ix =[];
for i=core_pts
    d = x(:,core_pts) - x(1:2,i);
    d =  (d(1,:).^2 + d(2,:).^2)';
    neighbors = find(d<min_neighbor_distance);

    if(  sum(ismember(C,neighbors)) >  1    )
        ix = [ix k];
    else
        k=k+1;
        ix = [ix k];

    end

    C = neighbors;

end
```

```
cluster = zeros(1,150);
L=1;
for i=core_pts
    cluster(1,i)=ix(1,L);
    L=L+1;
end
%boundary points
boundary_pts = [];

for i= setdiff(1:150,core_pts)

    d = x - x(1:2,i);
    d =  (d(1,:).^2 + d(2,:).^2)';
    neighbors = setdiff(find(d<min_neighbor_distance),i);
         if ( ~isempty( intersect(neighbors,core_pts) ) )

         boundary_pts = [boundary_pts i];

         cluster(1,i) = round( sum(cluster(
1,intersect(neighbors,core_pts) ))/ length(
intersect(neighbors,core_pts) ) ) ;
        end

end

gscatter(x(1,:),x(2,:),cluster);hold on;
legend('outliers','cluster 1','cluster 2');
```