DBSCAN(Density-Based Spatial Clustering of Applications with Noise)

DBSCAN is most widely used density based algorithm. It uses the concept of  **density reachability** and **density connectivity**.   
**Density Reachability**- A point "p" is said to be density reachable from a point "q" if point "p" is within ε distance from point "q" and "q" has sufficient number of points in its neighbors which are within distance ε.  
**Density Connectivity** - A point "p" and "q" are said to be density connected if there exist a point "r" which has sufficient number of points in its neighbors and both the points "p" and "q" are within the ε distance. This is chaining process. So, if "q" is neighbor of "r", "r" is neighbor of "s", "s" is neighbor of "t" which in turn is neighbor of "p" implies that "q" is neighbor of "p".

**Algorithmic steps for DBSCAN clustering**

Let  X = {x1, x2, x3, ..., xn} be the set of data points. DBSCAN requires two parameters: ε (eps) and the minimum number of points required to form a cluster (minPts).

1) Start with an arbitrary starting point that has not been visited.

2) Extract the neighborhood of this point using ε (All points which are within the ε distance are neighborhood).

3) If there are sufficient neighborhood around this point then clustering process starts and point is marked as visited else this point is labeled as noise (Later this point can become the part of the cluster).

4) If a point is found to be a part of the cluster then its ε neighborhood is also the part of the cluster and the above procedure from step 2 is repeated for all ε neighborhood points. This is repeated until all points in the cluster is determined.

5) A new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

6) This process continues until all points are marked as visited.

**Algorithm**

dbscanAlgo<-function(x,eps,minPts = 4){

dist\_mat<-dist(x,method="euclidean",diag = TRUE, upper = TRUE)

dist\_mat<-as.matrix(dist\_mat)

noiseptscount=nrow(dist\_mat)

corepts<-c()

for(i in 1:nrow(dist\_mat)){

count=0;

#print(i)

for(j in 1:ncol(dist\_mat)){

#(new)here consider i=j also

if(dist\_mat[i,j]<=eps)

count=count+1

#sprintf(count)

}

if(count>=minPts){

corepts<-union(corepts,c(i))

}

}

#print(dist\_mat)

#print(corepts)

unvisited<-corepts

stack<-c()

clustercount=0

#print(length(unvisited))

clusterno<-matrix(0, 1,nrow(dist\_mat)) #(new)

while(length(unvisited)!=0){

stack<-union(stack,unvisited[1])

unvisited<-unvisited[-1]

cluster<-c();

clustercount<-clustercount+1

while(length(stack)!=0){

point<-stack[1]

clusterno[1,point]=clustercount #(new)added clusterno to corepoint

stack<-stack[-1]

cluster<-union(cluster,point)

#add neighbours of point

for(j in 1:ncol(dist\_mat)){

if(dist\_mat[point,j]<=eps && point!=j){

#(new)don't add if already in cluster

if(is.element(point,cluster))

cluster<-union(cluster,j)

#(new)giving class to j

clusterno[1,j]=clustercount

#neighbours of point added

if(is.element(j,unvisited)){

stack<-union(stack,j)

#(new)removing j from unvisited as now it will be visited example is given below

remove <- c(j)

unvisited<-unvisited [! unvisited %in% remove]

}

}

}

}

#(new)removing duplicates

cluster<-unique(cluster)

noiseptscount=noiseptscount-length(cluster)

print(paste("Cluster:", clustercount,"No of point:",length(cluster)))

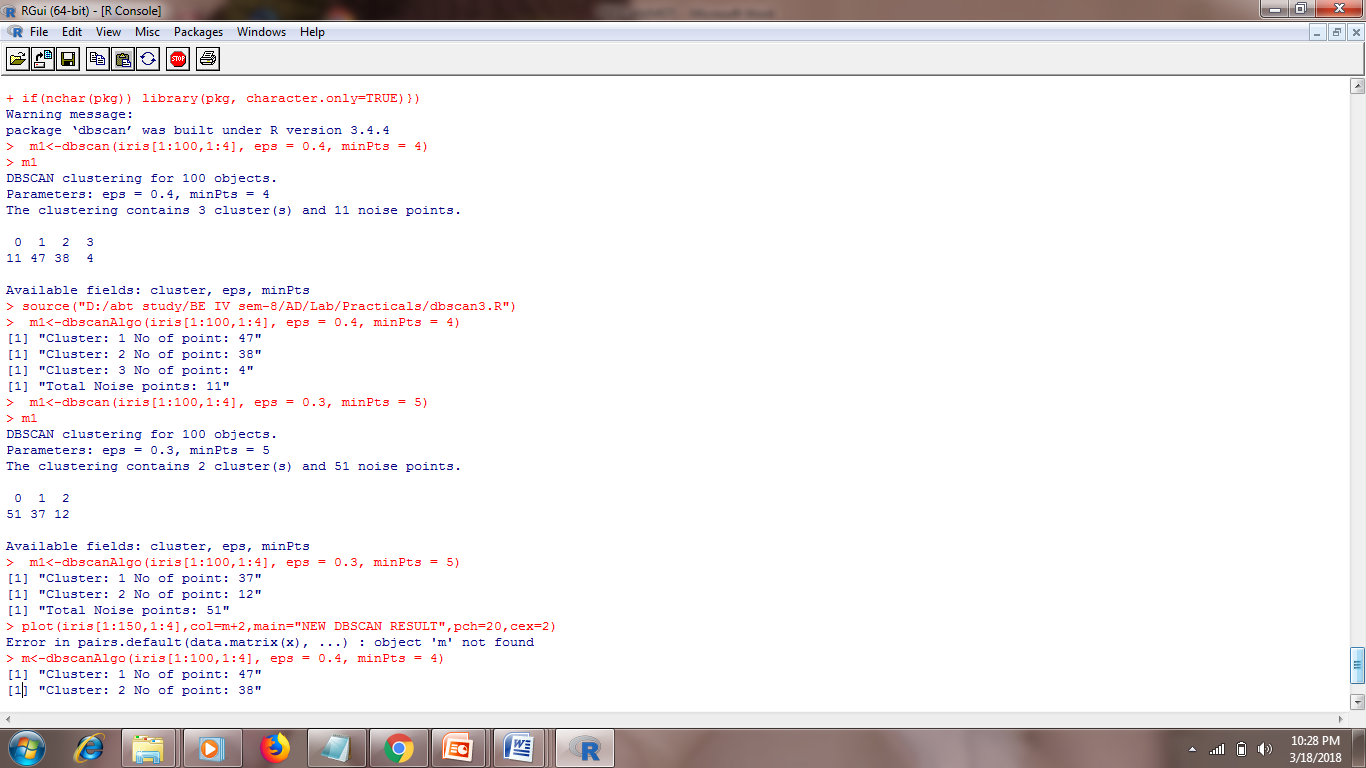
}

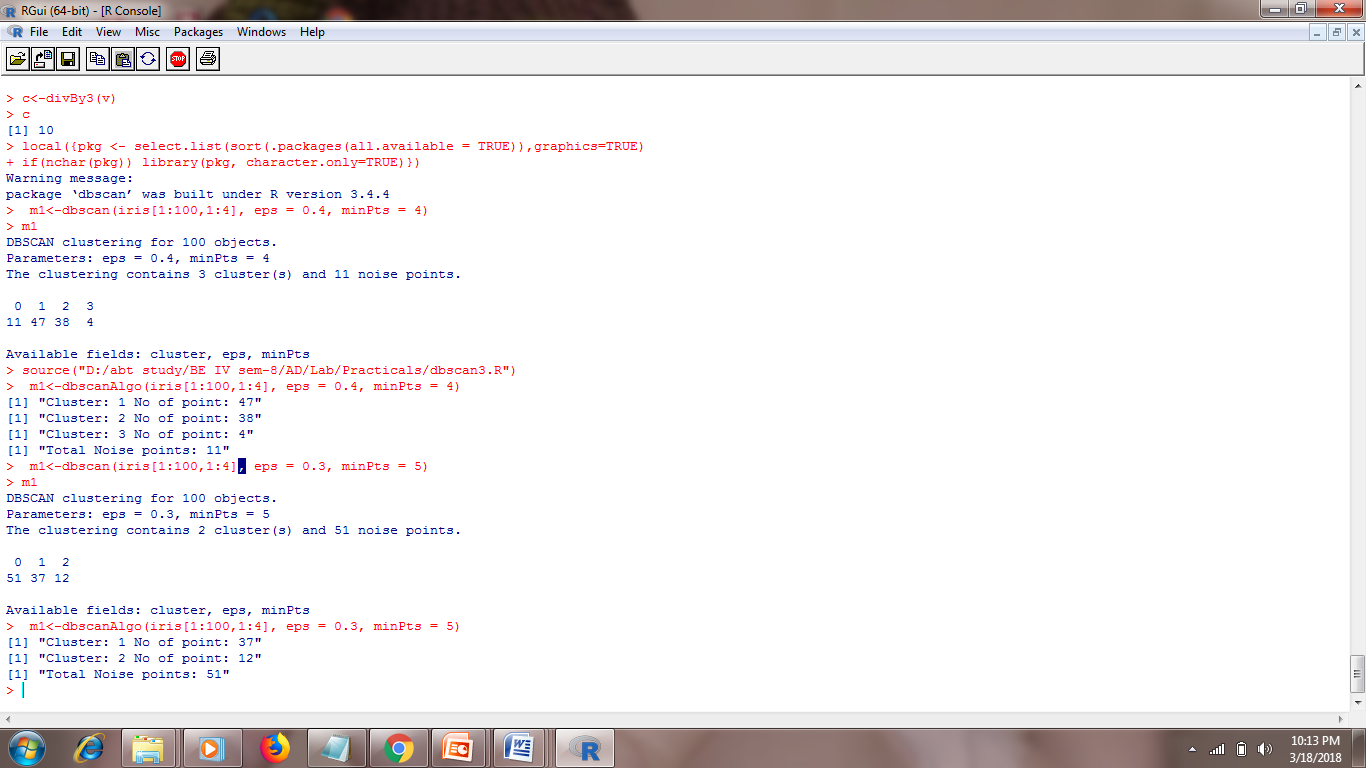
print(paste("Total Noise points:",noiseptscount))

return (clusterno)

}

**Output**

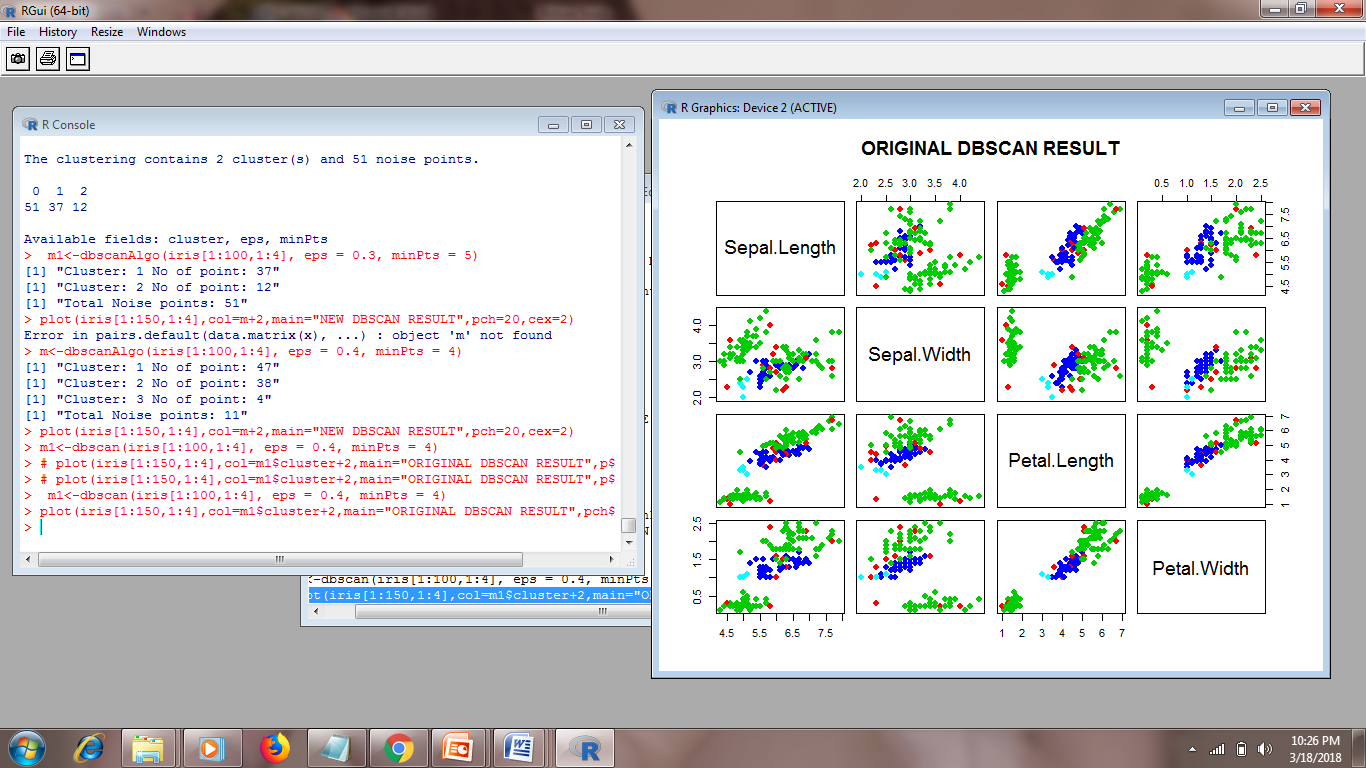
****

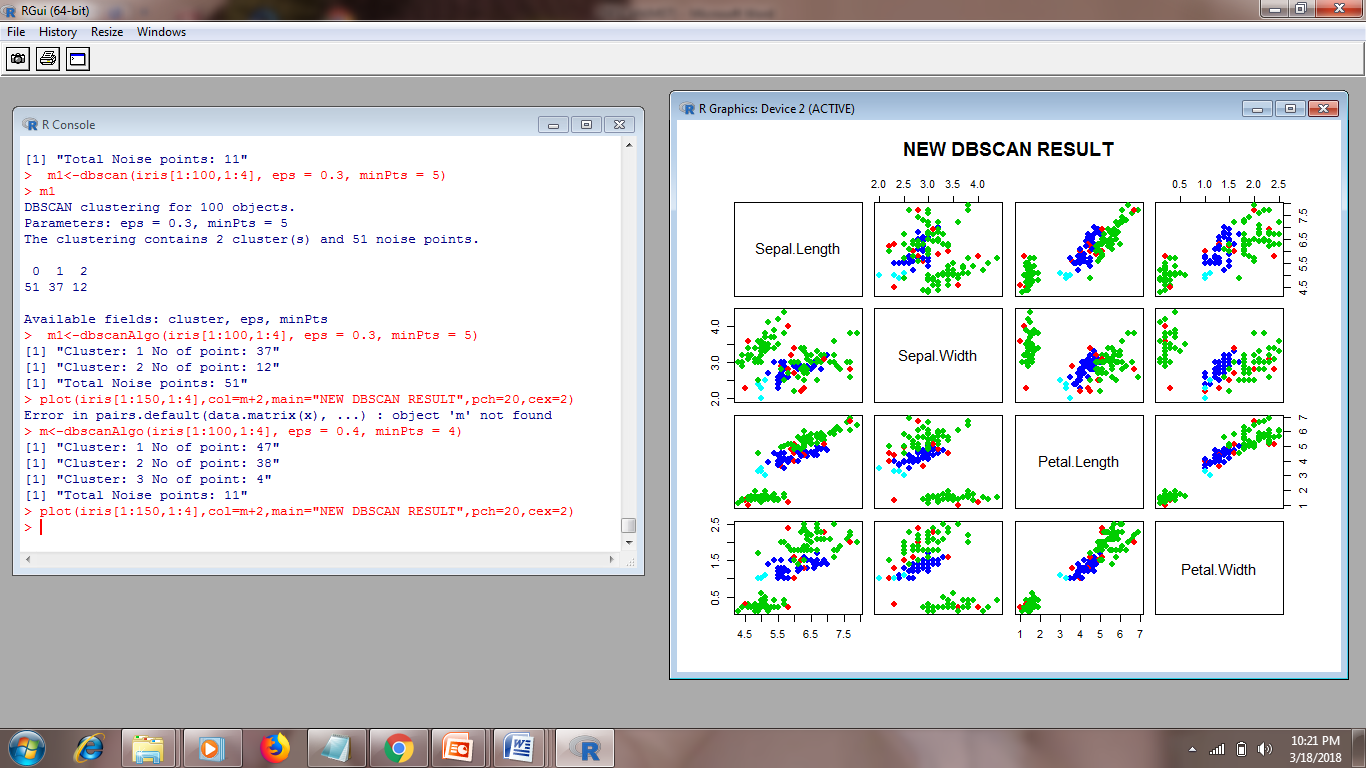
****

**Space Complexity :** o(n2)

**Time Complexity :** o(n2)

**Graphs:**

****

****

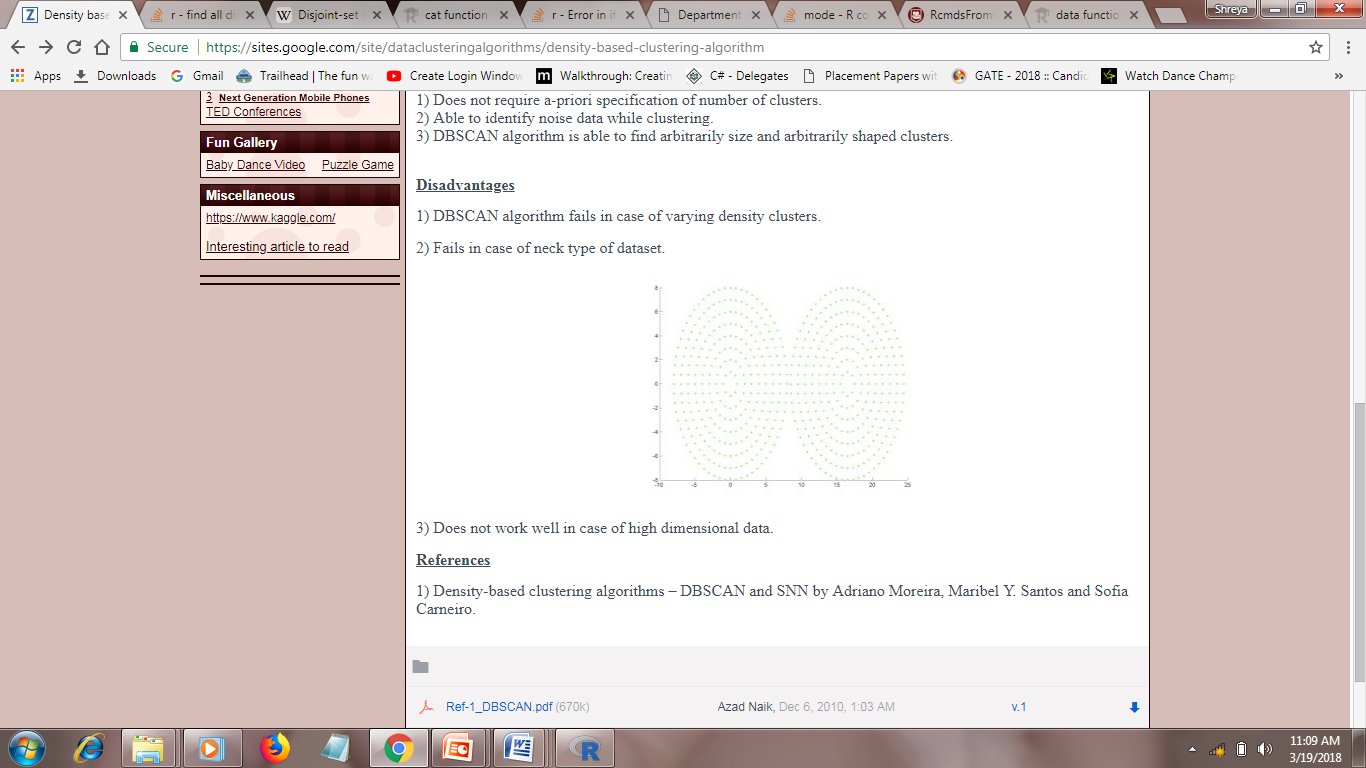
**Advantages**

1) Does not require a-priori specification of number of clusters.  
2) Able to identify noise data while clustering.  
3) DBSCAN algorithm is able to find arbitrarily size and arbitrarily shaped clusters.

**Disadvantages**

1) DBSCAN algorithm fails in case of varying density clusters.

2) Fails in case of neck type of dataset.



3)Does not work well in case of high dimensional data.