Data Mining Program
K-means Algorithms:

(Partitioning based clustering)

```
import math
import random
import numpy as np
import statistics
k=int(input("enter the value of k"))



n=int(input("Enter the number of object "))

p=[[1,2,3],[3,2,4],[4,3,5],[6,5,9],[2,8,5],[5,6,9],[6,4,2],[3,5,7],
[4,5,8],[6,3,4],[5,8,9],[1,7,8],[10,13,24],[20,15,40],[13,9,8]]



print(p)


Centroidd=[]
```

```
for i in range(k):
    Centroidd.append(random.choice(p))


print(Centroidd)


def dist(point,centroid):
    E=0
    for i in range(len(point)):
        E+=(point[i] - centroid[i])**2
    E=math.sqrt(E)
    return E
print(dist(p[0],Centroidd[0]))


Cluster=[[1,2,3],[2,3,4],[3,4,5],[4,5,6],[5,6,7]]
Cluster[0].append(1)
print(Cluster)


print(list(np.array(p).mean(axis=0)))
```

```
def K_mean_algorithm():

    for iter in range(100):
        print("Eteration ",iter,"\n")
        C1=[]
        C2=[]
        C3=[]
        C4=[]
        C5=[]
        C=[C1,C2,C3,C4,C5]
        for c in range(len(C)):
            print(C[c])
        count=0

        for i in p:
            min=1000
            m=[]
            for ci in Centroidd:
                if(dist(i,ci)<min):
                    min=dist(i,ci)
                    m=ci
            for j in range(len(Centroidd)):
                if(m==Centroidd[j]):
                    C[j].append(i)
                    print(i," ",Centroidd[j])
                    break
```

```
        for index in range(len(C)):
            print(C[index])

if(Centroidd[index]==list(np.array(C[index]).mean(axis=0)))
:
                count=count+1
Centroidd[index]=list(np.array(C[index]).mean(axis=0))
        if(count==len(C)):
            for q in C:
                print("cluster     ",q,"\n")
            break
print(K_mean_algorithm())
```

Aglomarative Clustering:

(Hirarichal clustering)

```
import math
C=[[[1,2,3]],[[3,5,2]],[[2,4,8]],[[3,7,0]],[[4,9,9]],[[3,4,5]],[[4,6,
8]],[[7,5,3]],[[9,7,5]],[[8,4,6]],[[11,14,3]]]

for i in C:
    print(i)
print(C[0])
record=[]
info=[]
```

```
def Agglomative_Clustering():
    print("Merging the cluster\n",C,"\n")
    length=len(C)
    for iteration in range(length-1):
        min=1000
        info=[]
        record=[]
        for i in range(len(C)):
            for j in range(i+1,len(C)):

record.append([i,j,min_distance_cluster(C[i],C[j])])
                if(min>min_distance_cluster(C[i],C[j])):
                    min=min_distance_cluster(C[i],C[j])
                    info=[i,j,min]
        print("record\n\n",record,"\n\n\n")
        if(len(info)>0):
            print("mergin cluster
are\n",C[info[0]],"\t\t",C[info[1]],"\n")
            appending(C[info[0]],C[info[1]])
            C.pop(info[1])
            print(C,"\n")


Agglomative_Clustering()
```

```
def appending(p,q):
        for j in q:
            p.append(j)
```