C++ Code

```
#include<iostream>
#include<bits/stdc++.h>
#include<fstream>
#include <sstream>
#include<vector>
#include<cstring>
#include<map>
using namespace std;
#define mp make_pair
#define pb push_back

class Point
{       public:
                int id;
                vector<float> values;
                Point(){
                }
                Point(int i,vector<float> val)
                {       id=i;
                        values=val;     }
                void PrintP()
                {       int attr=values.size();
                        cout<<id<<" -> ";
                        for(int j=0;j<attr;j++)
                        {       cout<<values[j]<<" ";       }
                        cout<<endl;     }
};

Point* getPoint(vector<Point*> data, int id)
{       int n=data.size();
        for(int i=0;i<n;i++)
        {       if(data[i]->id==id)
                {return data[i];}       }       }

float getDistance(Point* x,Point* x1)
{       int l=x->values.size();
        float dist=0;
        for(int j=0;j<l;j++)
        {       dist+=(x->values[j]-x1->values[j])*(x->values[j]-x1->values[j]);        }
        dist=sqrt(dist);
        return dist;    }
```

```
float calcminClu(map<Point*,vector<Point*> > cluster, Point* &cid1 )
{       float minm=100000;
        map<Point*, vector<Point*> >:: iterator it;
        for(it=cluster.begin();it!=cluster.end();it++)
        {       int n=it->second.size();
                if(n<minm)
                {       minm=n;
                        cid1=it->first;       }       }
        return minm;
}

float calcminxC(map<Point* , vector<Point*> > cluster, Point* x ,Point* &cid)    //O(K.M)
{       float minm=100000.0;
        map<Point*, vector<Point*> >:: iterator it;
        for(it=cluster.begin();it!=cluster.end();it++)
        {       float dist=getDistance(it->first,x);
                if(minm>=dist) {
                minm=dist;cid=it->first;    }       }
        return minm;
}

void REASSIGN(vector<Point*> data, map<Point* ,vector<Point*> > &cluster)
{       int tuples=data.size();
        int attr=data[0]->values.size();
        map<Point*, vector<Point*> >:: iterator it;

        vector<Point*> ms;
        for(it=cluster.begin();it!=cluster.end();it++)
        {       it->second=ms;    }
        for(int i=0;i<tuples;i++)
        {       Point* cid;
                float minxC=calcminxC(cluster,data[i],cid);
                cluster[cid].push_back(data[i]);
        }

         map<Point* ,vector<Point*> >  cluster1=cluster;
        for(it=cluster1.begin();it!=cluster1.end();it++)
        {       Point* cid=it->first;
                vector<Point*> ms=cluster[cid];
                vector<float> newleader;
                int l=ms.size();
                for(int j=0;j<attr;j++)
                {       newleader.push_back(0);         }

                for(int k=0;k<l;k++)
        {   for(int j=0;j<attr;j++)
            {   newleader[j]+=ms[k]->values[j];     }
        }
        for(int j=0;j<attr;j++)
        {   newleader[j]/=l;        }

        cluster.erase(cid);
        cid=new Point(-1,newleader);
        cluster.insert(mp(cid,ms));
    }
}

void run_kmeans(vector<Point*> data, map<Point* ,vector<Point*> > &cluster, int k, int max_iter)
{
    int tuples=data.size();
    int attr=data[0]->values.size();

    for(int i=0;i<k;i++)
    {   int id=rand()%tuples;
        Point* p=getPoint(data,id);
        vector<Point*> c;
        cluster.insert(mp(p,c));    }

    int i=1;
    map<Point* ,vector<Point*> > cluster1;
    while(i<max_iter && !same(cluster,cluster1))
    {   cluster1 = cluster;
        REASSIGN(data,cluster);
        i++;    }   }

void run_mod_kmeans(vector<Point*> data, map< Point*, vector<Point*> > &cluster_kp, int k, int max_iter)
{   int tuples=data.size();
         int arr[]={3,9};
    int attr=data[0]->values.size();
    int i=1;
    map<Point* ,vector<Point*> > cluster1;
    while(i<max_iter && !same(cluster_kp,cluster1))
    {   cluster1 = cluster_kp;
        REASSIGN(data,cluster_kp);
        i++;
    }
}

void findoutliers(map<Point*,vector<Point*> > cluster, int attr , vector<pair<Point*,float> > &outliers)
{   map<Point*,vector<Point*> >:: iterator it;
    for(it=cluster.begin();it!=cluster.end();it++)
    {   Point* leader=it->first;
        vector<Point*> corr_clust=it->second;
        int l=corr_clust.size();

        float radius=0;
        for(int i=0;i<l;i++)
        {   radius+=getDistance(leader,corr_clust[i]);  }
        radius/=l;

        for(int i=0;i<l;i++)
        {   float dist=getDistance(leader,corr_clust[i]);
            if(dist>radius)
            {   outliers.push_back(mp(corr_clust[i],dist));
                corr_clust[i]->PrintP();    }   }   }   }

bool mycmp(pair<Point*,float> a,pair<Point*,float> b)
{   if(a.second>=b.second) return true;
    else
    return false;   }

int main()
{   fstream file;
    file.open("iri.txt",fstream::in |fstream::out| fstream::binary);

    if(file.fail())
    {   cout<<"Error opening the file"<<endl;
        exit(0);  }

    string str;
    getline(file,str);
    int attr=0,tuples=1,classes=0;
    stringstream ss(str);

    while( ss.good() )
    {   string substr;
       getline( ss, substr, ',' );
       attr++;
    }
    attr-=1;
    while(getline(file,str))
    {   tuples++;   }

    file.clear();
    file.seekg(0, ios::beg);

    vector<Point*> data;
```

```
for(int i=0;i<tuples;i++)
{           string str1;
            getline(file,str1);
            stringstream ss1(str1);
            vector<float> result;

            while( ss1.good() )
            {   string substr1;
               getline( ss1, substr1, ',' );
               result.push_back( atof(substr1.c_str()) );               }
            result.pop_back();
            data.push_back(new Point(i+1,result));      }

map<Point* , vector<Point*> > cluster;
int k=3,max_iter=10;
run_kmeans(data,cluster,k,max_iter);
vector<pair<Point*,float> > outliers1,outliers2,outliers3,outliers4;
findoutliers(cluster,attr,outliers1);

//k+1 starts.....................
            //Finding clusters for K+1...............

                        vector<Point*> ms;
                        map<Point* , vector<Point*> > cluster_kp;
                        map<Point* , vector<Point*> >::iterator it;
                        for(it=cluster.begin();it!=cluster.end();it++)
                        {           cluster_kp.insert(mp(it->first,ms));            }

                        vector<float> ncleader;
                        for(int j=0;j<attr;j++)
                        {           ncleader.push_back(0);    }

                        for(it=cluster.begin();it!=cluster.end();it++)
                        {           for(int j=0;j<attr;j++)
                                    {           ncleader[j]+=it->first->values[j];     }           }

                        int l=cluster.size();
                        for(int j=0;j<attr;j++)
                        {           ncleader[j]/=l;     }
                        cluster_kp.insert(mp(new Point(-1,ncleader),ms));
                        run_mod_kmeans(data,cluster_kp,k+1,max_iter);
                        findoutliers(cluster_kp,attr,outliers2);

// k+1 TILL HERE.....................

//k-1 starts.....................
```

```
            Point* cid1;
            map<Point* , vector<Point*> > cluster_km;
            for(it=cluster.begin();it!=cluster.end();it++)
            {
                cluster_km.insert(mp(it->first,ms));
            }

            calcminClu(cluster,cid1);
            cluster_km.erase(cid1);

            run_mod_kmeans(data,cluster_km,k-1,max_iter);
            findoutliers(cluster_km,attr,outliers3);

//FINDING COMMON OUTLIERS
        int mss=outliers1.size();
        int ns=outliers2.size();
        int oss=outliers3.size();

int cnt=0;
for(int i=0;i<mss;i++)
{   for(int j=0;j<ns;j++)
    {   for(int l=0;l<oss;l++)
        {if((outliers1[i].first->id==outliers2[j].first->id)&&(outliers1[i].first->id==outliers3[l].first->id))
        {   cnt++;
            outliers4.push_back(outliers1[i]);      }
        }   }   }

sort(outliers4.begin(),outliers4.end(),mycmp);
cout<<"Common Outlers"<<endl;
for(int i=0;i<cnt;i++)
{   cout<<outliers4[i].second<<" ";outliers4[i].first->PrintP();   }

return 0;
}
```