

## Experiment no. 10

**Name:** Sonali Dattatray Kaingade

**PRN:** 21620002

**Title:** Distance and cluster

- Assume some points (multidimensional)

Compute centre of cluster assuming all points belonging to one cluster.

Find distance of all points with obtained cluster centre using suitable distance function

Display results in the form of upper or lower triangular matrix

**code:**

```
#include <bits/stdc++.h>

using namespace std;

double finddist(pair<double,double>p1, pair<double,double>p2){

    double x1 = p1.first;

    double x2 = p2.first;

    double y1 = p1.second;

    double y2 = p2.second;

    return sqrt((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2));

}

int main(){

    string line;

    ifstream input("cluster_input.csv");

    vector<pair<double, double>>points;

    int j = 0;
```

```
while(getline(input,line)){  
    stringstream str(line);  
  
    if(j==0){  
        j++;  
        continue;  
    }  
  
    string name,x,y;  
    getline(str,name,',');  
    getline(str,x,',');  
    getline(str,y);  
  
    points.push_back({stoi(x), stoi(y)});  
}  
input.close();  
  
double x_sum, y_sum, x_mean, y_mean;  
int n = points.size();  
for(auto it : points){  
    x_sum += it.first;  
    y_sum += it.second;  
}  
x_mean = x_sum/n;  
y_mean = y_sum/n;
```

```

//calculate distance of each point from every point
ofstream output("output.csv");

output<<" "<<"p1," <<"p2,"<<"p3,"<<"p4,"<<"p5"<<endl;

for(int i = 0; i<points.size();i++){

    output<<"p"<<i+1<<" ";

    for(int j = 0; j<=i;j++){

        if(i==j){

            output<<"0,";

        }

        else{

            double ans = finddist(points[i],points[j]);

            output<<ans<<" ";

        }

    }

    output<<endl;

}

output<<"centroid,";

for(int i = 0; i<points.size(); i++){

    double ans = finddist(points[i],{x_mean,y_mean});

    output<<ans<<" ";

}

cout << endl;

output.close();

```

```
}
```

**Output:**

**Input.csv:**

cluster_input.csv	
cluster_input.csv	
1	Points,x,y
2	p1,10,40
3	p2,20,10
4	p3,15,20
5	p4,25,30
6	p5,15,5
7	

**Output.csv:**

output.csv	
output.csv	
1	,p1,p2,p3,p4,p5
2	p1,0,
3	p2,31.6228,0,
4	p3,20.6155,11.1803,0,
5	p4,18.0278,20.6155,14.1421,0,
6	p5,35.3553,7.07107,15,26.9258,0,
7	centroid,20.2485,11.4018,2.23607,12.0416,16.1245,

## Knime:

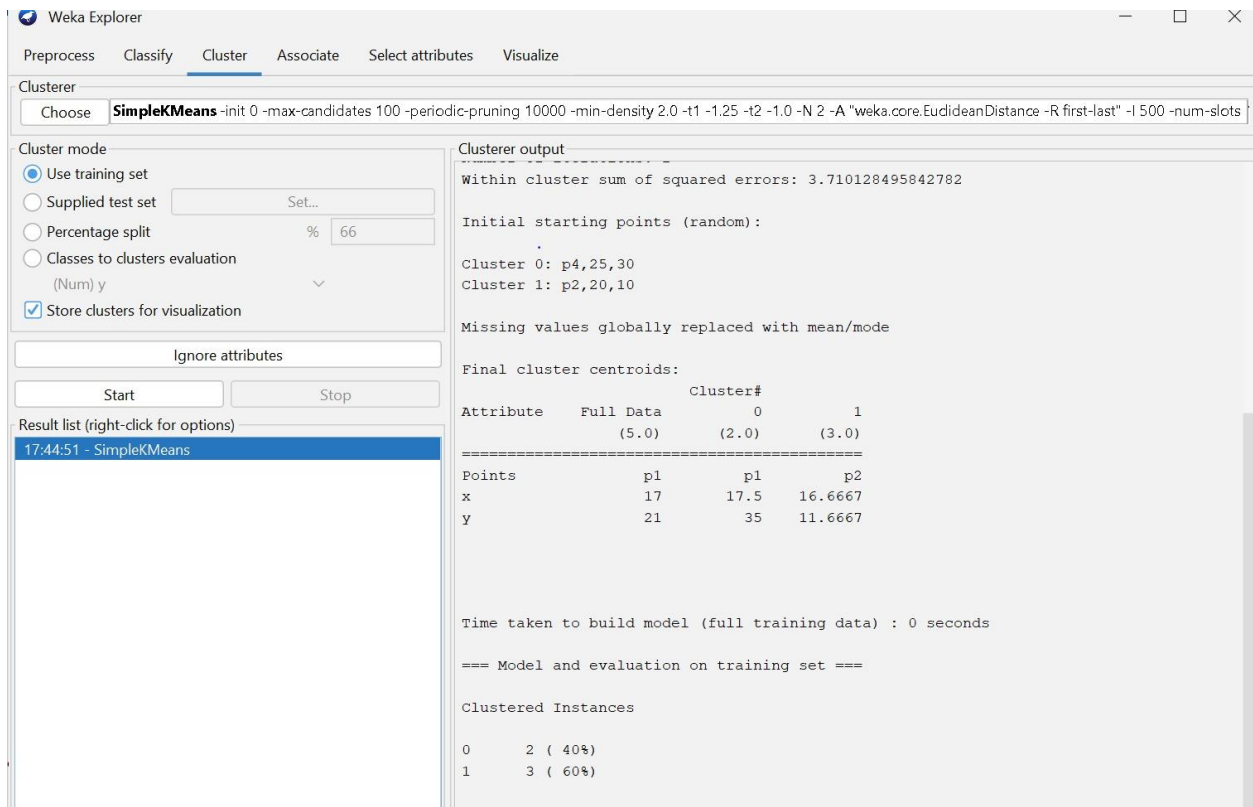


The Knime workflow consists of two nodes: a **CSV Reader** and a **k-Means** node. The CSV Reader is connected to the k-Means node. Below the k-Means node, there is a text box that says "Add comment".

The k-Means node output is displayed in a table view with 5 rows and 4 columns. The columns are: #, Row..., points, x, y, and Cluster. The data is as follows:

#	Row...	points	x	y	Cluster
1	Row0	p1	10	40	cluster_0
2	Row1	p2	20	10	cluster_1
3	Row2	p3	15	20	cluster_2
4	Row3	p4	25	30	cluster_2
5	Row4	p5	15	5	cluster_1

## Weka:



The Weka Explorer interface shows the **SimpleKMeans** clustering algorithm settings. The **Clusterer** dropdown is set to **SimpleKMeans**. The **Cluster mode** is set to **Use training set**. The **Ignore attributes** field is empty. The **Start** button is visible.

The **Clusterer output** pane displays the following information:

Within cluster sum of squared errors: 3.710128495842782

Initial starting points (random):

Cluster 0: p4,25,30  
Cluster 1: p2,20,10

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute	Full Data	Cluster#	0	1
	(5.0)	(2.0)	(3.0)	

Points

	p1	p1	p2
x	17	17.5	16.6667
y	21	35	11.6667

Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0	2 ( 40%)
1	3 ( 60%)