Εργασία Υπολογιστική Νοημοσύνη

Για την δημιουργία των δύο πινάκων ΣΔΟ και ΣΔΤ δημιουργήσαμε δυο κλάσεις την Create_SDO και την Create_SDT :

```
1 usage
public double[][] create_SDT_array() {
    Random rand = new Random();
    int row = 8000;
    int col = 2;
    double[][] point_data_arr = new double[row][col];
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            double num = rand.nextDouble() * 2 - 1;
            point_data_arr[i][j] = num;
        }
    }
    return point_data_arr;
}
```

```java
2 usages
public class Create_SDO {
    1 usage
    public double[][] Create_SDO_array() {
        Random rand = new Random();
        int row = 1200;
        int col = 2;
        double[][] point_data_arr = new double[row][col];
        for (int i = 0; i < row; i++) {
            //150 points in the rectangle [0.8,1.2]x[0.8,1.2]
            if(i<150){
                point_data_arr[i][0] = rand.nextDouble()  * 0.4 + 0.8;
                point_data_arr[i][1] = rand.nextDouble()  * 0.4 + 0.8;
            }
            //150 points in the rectangle [0,0.5]x[0,0.5]
            else if(i>=150 && i < 300){
                point_data_arr[i][0] = rand.nextDouble()  *0.5;
                point_data_arr[i][1] = rand.nextDouble()  *0.5;
            }
            //150 points in the rectangle [1.5,2]x[0,0.5]
            else if(i>=300 && i < 450){
                point_data_arr[i][0] = rand.nextDouble() * 0.5 + 1.5;
                point_data_arr[i][1] = rand.nextDouble() * 0.5;
            }
```

```java
            //150 points in the rectangle [0,0.5]x[1.5,2],
            else if(i>=450 && i < 600){
                point_data_arr[i][0] = rand.nextDouble() * 0.5 ;
                point_data_arr[i][1] = rand.nextDouble() * 0.5 + 1.5;
            }
            //150 points in the rectangle [1.5,2] x [1.5,2]
            else if(i>=600 && i < 750){
                point_data_arr[i][0] = rand.nextDouble() * 0.5 + 1.5;
                point_data_arr[i][1] = rand.nextDouble() * 0.5 + 1.5;
            }
            //75 points in the rectangle [0,0.4]x[0.8, 1.2]
            else if(i>=750 && i < 825) {
                point_data_arr[i][0] = rand.nextDouble() * 0.4;
                point_data_arr[i][1] = rand.nextDouble() * 0.4 + 0.8;
            }
            //75 points in the rectangle [1.6,2]x[0.8, 1.2]
            else if(i>=825 && i < 900) {
                point_data_arr[i][0] = rand.nextDouble() * 0.4 + 1.6;
                point_data_arr[i][1] = rand.nextDouble() * 0.4 + 0.8;
            }
            //75 points in the rectangle [0.8,1.2]x[0.3,0.7]
            else if(i>=900 && i < 975) {
                point_data_arr[i][0] = rand.nextDouble() * 0.4 + 0.8;
                point_data_arr[i][1] = rand.nextDouble() * 0.4 + 0.3;
            }
```

```java
            //75 points in the rectangle [0.8,1.2]x[1.3,1.7]
            else if(i>=975 && i < 1050) {
                point_data_arr[i][0] = rand.nextDouble() *  0.5 + 0.8;
                point_data_arr[i][1] = rand.nextDouble() * 0.4 + 1.3;
            }
            //150 points in the rectangle [0,2]x[0,2]
            else if(i>=1050 && i < 1200){
                point_data_arr[i][0] = rand.nextDouble() * 2;
                point_data_arr[i][1] = rand.nextDouble() * 2;
            }
        }
        return point_data_arr;
    }
```

Άσκηση 1η

Ερώτημα 1ο

Για την αρχικοποίηση των νευρώνων, του αριθμού εισόδων, αριθμού κατηγοριών και της συνάρτησης ενεργοποίησης Κάναμε στατική την αρχικοποίηση των νευρώνων και δυναμική της συνάρτησης ενεργοποίησης και του αριθμού εισόδων.

```
private static int first_layer_neurons =4;
4 usages
private static int second_layer_neurons = 4;
3 usages
private static int third_layer_neurons = 4;
```

```
this.number_of_inputs = data_array.length;
this.activation_function = activation_function;
```

```
1 usage
public double activation_ReLU(double x){
    if (x > 0) {
        return x;
    } else {
        return 0;
    }
}
1 usage
public double activation_TanHU(double x) { return (float) Math.tanh(x); }

1 usage
public double sigmoid(double x) { return (float) 1.0 / (1.0 + Math.exp(-x)); }
```

Ερώτημα 2ο

Για το φόρτωμα των συνόλων εκπαίδευσης και ελέγχου και την κατηγοριοποίηση των κατηγοριών δημιουργήσαμε δυο συναρτήσεις την load_data για την φόρτωση και διαχωρισμό των δεδομένων και την classify_data για την κατηγοριοποίηση τους

```
1 usage
public void load_data(double[][] data_array){
    this.learning_set = new double[number_of_inputs/2][2];
    this.testing_set = new double[number_of_inputs/2][2];
    for(int i=0; i<this.number_of_inputs; i++){
        if(i < this.number_of_inputs/2) {
            this.learning_set[i][0] = data_array[i][0];
            this.learning_set[i][1] = data_array[i][1];
        }
        else{
            this.testing_set[i-learning_set.length][0] = data_array[i][0];
            this.testing_set[i-learning_set.length][1] = data_array[i][1];
        }
    }
}
```

```
public String classify_data(double x1, double x2){
    String category = "";
    if((Math.pow((x1-0.5),2) + Math.pow((x2-0.5),2)) <0.2){
        if(x1>0.5){
            category = "C1";
        }
        else if(x1<0.5){
            category = "C2";
        }
    }
    else if(Math.pow((x1+0.5),2) + Math.pow((x2+0.5),2)>0.2) {
        if(x1>-0.5){
            category = "C1";
        }
        else if(x1<-0.5){
            category = "C2";
        }
    }
    else if(Math.pow((x1-0.5),2) + Math.pow((x2+0.5),2)<0.2){
        if(x1>0.5){
            category = "C1";
        }
        else if(x1<0.5){
            category = "C2";
        }
```

```
    }
    else if(Math.pow((x1+0.5),2) + Math.pow((x2-0.5),2)<0.2){
        if(x1>-0.5){
            category = "C1";
        }
        else if(x1<-0.5){
            category = "C2";
        }
    }
    if(category==""){
        if(x1>0){
            category = "C3";
        }
        else{
            category = "C4";
        }
    }
    return category;
}
```

Ερώτημα 3ο

Για τον καθορισμό της αρχιτεκτονικής του δικτύου MLP ορίσαμε τους απαραίτητους πίνακες για την αρχικοποίηση των

```
this.weights1 = instantiate_weights(learning_set[1].length,first_layer_neurons);
this.weights2 = instantiate_weights(first_layer_neurons,second_layer_neurons);
this.weights3 = instantiate_weights(second_layer_neurons,third_layer_neurons);
this.biases1 = instantiate_biases(first_layer_neurons);
this.biases2 = instantiate_biases(second_layer_neurons);
this.biases3 = instantiate_biases(third_layer_neurons);
```

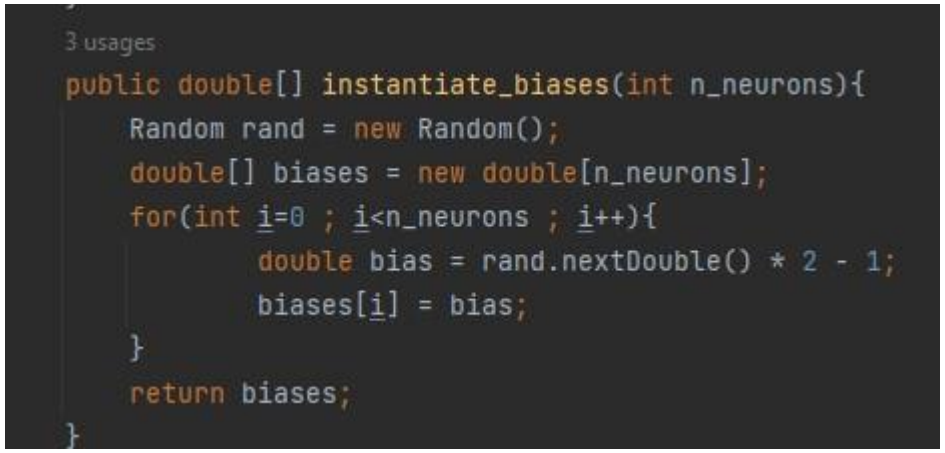
```
3 usages
public double[][] instantiate_weights(int n_inputs, int n_neurons){
    Random rand = new Random();
    double[][] weights = new double[n_inputs][n_neurons];
    for(int i=0 ; i<n_inputs ; i++){
        for(int j=0 ;j< n_neurons; j++){
            double weight = rand.nextDouble() * 2 - 1;
            weights[i][j] =weight;
        }
    }
    return weights;
}
```

```
no usages
private double learning_rate =0.01;
```

3 usages

```
2 usages
private static double[][] weights1;

2 usages
private static double[][] weights2;

2 usages
private static double[][] weights3;

2 usages
private static double[] biases1;
2 usages
private static double[] biases2;
2 usages
private static double[] biases3;
```

βαρών και των πολώσεων καθώς και του ρυθμού μάθησης



```
3 usages
public double[] instantiate_biases(int n_neurons){
    Random rand = new Random();
    double[] biases = new double[n_neurons];
    for(int i=0 ; i<n_neurons ; i++){
            double bias = rand.nextDouble() * 2 - 1;
            biases[i] = bias;
    }
    return biases;
}
```

Ερώτημα 4ο

Όσο αναφορά την συνάρτηση forward pass την ορίσαμε να παίρνει ως ορίσματα τα βάρη ένα πίνακα από δεδομένα του προηγούμενου επιπέδου και τις πολώσεις και στην συνέχεια να υπολογίζει την επιθυμητή έξοδο.

```
3 usages
public double[][] forward_pass(double[][] x,int d, int K,double[][] weights, double[] biases){
    double[][] output = new double[d][K];
    for (int i = 0; i <d; i++) {
        for (int j = 0; j < K; j++) {
            for (int k = 0; k < x[i].length; k++) {
                output[i][j] += x[i][k] * weights[k][j];
            }
            output[i][j] += biases[j];
            if(this.activation_function == "relU"){
                output[i][j] = activation_ReLU(output[i][j]);
            }
            else if(this.activation_function == "tahU"){
                output[i][j] = activation_TanHU(output[i][j]);
            }
            else{
                output[i][j] = sigmoid(output[i][j]);
            }
        }
    }
    return output;
```

Άσκηση 2η

Για την δεύτερη άσκηση για να προσομοιώσουμε την λειτουργία του αλγορίθμου K-Means δημιουργήσαμε δυο κλάσεις μια την Entry για την διαχείριση των δεδομένων που παίρνουμε από τον πίνακα ΣΔΟ και την Cluster για την διαχείριση των συμπλεγμάτων που δημιουργούνται κατά του τρεξίματος του αλγορίθμου.

```java
public class Entry {


    2 usages
    private double x;
    2 usages
    private double y;

    1 usage
    public Entry(double x,double y){
        this.x = x;
        this.y = y;
    }

    3 usages
    public double getX() { return x; }

    3 usages
    public double getY() { return y; }

}
```

```java
17 usages
public class Cluster {
    4 usages
    private double x_Center;
    4 usages
    private double y_Center;
    2 usages
    private int clusterNum;
    3 usages
    private double error;
    1 usage
    public Cluster(double x, double y, int clusterNum){
        this.x_Center = x;
        this.y_Center = y;
        this.clusterNum= clusterNum ;
        this.error = 0;
    }
    1 usage
```

Για τον διαμοιρασμό των εργασιών η κλάση Cluster είναι υπεύθυνη και για τον εκσυγχρονισμό των κέντρωνκαθώς και για τον υπολογισμό της ευκλείδειας απόστασης

```
1 usage
public double calc_euclidean_distance(Entry entry){
    return Math.sqrt((Math.pow(entry.getX()-this.x_Center,2))+(Math.pow(entry.getY()-this.y_Center,2)));
}

1 usage
public void updateCenter(List<Entry> entries){
    double sum_x=0;
    double sum_y=0;
    for(int j=0;j< entries.size() ;j++){
        Entry curr_entry =entries.get(j);
        sum_x += curr_entry.getX();
        sum_y += curr_entry.getY();
    }
    this.x_Center= sum_x/entries.size();
    this.y_Center= sum_y/entries.size();
}
```

```
public double getX_Center() {
    return x_Center;
}

1 usage
public double getY_Center() {
    return y_Center;
}

1 usage
public int getClusterNum() {
    return clusterNum;
}
2 usages
public double getError() {
    return error;
}

2 usages
public void setError(double error) {
    this.error = error;
}
```

Για την υλοποίηση του αλγορίθμου K-Means

1)Φτιάξαμε 2 λίστες, μια για την διαχείριση των οντοτήτων Entries, μια για την διαχείριση των συμπλεγμάτων Clusters Και ένα χάρτη ο οποίος μας βοηθά στην χαρτογράφηση του αλγορίθμου το οποίο σημαίνει ότι κρατά πληροφορία για τα Entries και για τα συμπλέγματα στα οποία βρίσκονται

```
private List<Entry> data = new ArrayList<~>();
6 usages
private List<Cluster> clusters = new ArrayList<~>();
8 usages
private Map<Cluster,List<Entry>> clusterElements = new HashMap<~>();
```

```
public void initialize_entries(){
    for(int i=0;i<sdo.length; i++){
        Entry entry = new Entry(sdo[i][0],sdo[i][1]);
        data.add(entry);
    }
}
1 usage
public void initialize_centers(){
    Random rand = new Random();
    for(int i=0;i<M;i++){
        double x = rand.nextDouble()*2;
        double y = rand.nextDouble()*2;
        Cluster cluster = new Cluster(x,y,i);
        clusters.add(cluster);
    }
}
```

Για την αρχικοποίηση των συμπλεγμάτων φτιάξαμε την μέθοδο create_clusters και για την ενημέρωση τους την update_clusters. Η πρώτη είναι υπεύθυνη να δημιουργεί τα συμπλέγματα το οποίο επαναλαμβάνεται μετά από κάθε ενημέρωση και η δευτερη είναι υπεύθυνη για την ενημέρωση των κέντρων

```java
public void create_clusters(){
    for (int i = 0; i < data.size(); i++) {
        Entry entry = data.get(i);
        double distance = Double.POSITIVE_INFINITY;
        Cluster cluster = clusters.get(0);
        for(int j=0;j< clusters.size() ;j++){
            Cluster current_cluster = clusters.get(j);
            double curr_distance= current_cluster.calc_euclidean_distance(entry);
            if(curr_distance<distance) {
                distance = curr_distance;
                cluster = current_cluster;
            }
        }
        double error = cluster.getError();
        error += distance;
        cluster.setError(error);
        if(clusterElements.containsKey(cluster)) {
            clusterElements.get(cluster).add(entry);
        }
        else {
            List<Entry> entries= new ArrayList<~>();
            entries.add(entry);
            clusterElements.put(cluster,entries);
        }
    }
}
```

```java
public void updateClusters(){
    int centers_changed = 0;
    for(Cluster groupName : clusterElements.keySet()){
        groupName.updateCenter(clusterElements.get(groupName));
    }
}
```

Τέλος για να επιτύχουμε την ομαλή λειτουργία του αλγορίθμου στον constructor της κλάσης K_Means κάνουμε 1000 επαναλήψεις στη δημιουργία και επαναϋπολογισμό των κέντρων για να λειτουργεί ομαλά ο αλγόριθμος και επίσης του δωσαμε την ευθυνη να υπολογίζει το σφάλμα ομαδοποίησης .

```
public K_Means(double[][] sdo, int M){
    this.M = M;
    this.sdo=sdo;
    initialize_entries();
    initialize_centers();
    create_clusters();
    for(int i=0;i<1000;i++){
        updateClusters();
        for(Cluster cluster : clusters ){
            cluster.setError(0);
        }
        clusterElements.clear();
        create_clusters();
    }
    setError();
}
```

```
1 usage
public void setError(){
    for(Cluster groupName : clusterElements.keySet()){
        this.full_error += groupName.getError();
    }
```

Τέλος για την δημιουργία των Scatterplots χρησιμοποιήσαμε την βιβλιοθήκη Jfree και φτιάξαμε μια συνάρτηση CreateScatterPlot η οποία παίρνει ως όρισμα τον χάρτη που επιστρέφει ο αλγόριθμος K_Means και φτιάχνει ένα plot για την απεικόνιση του

```java
public CreateScatterPlot(String title,K_Means my_data) {
    super(title);
    this.clusters =my_data.getClusters();
    this.data = my_data.getData();
    this.clusterElements = my_data.getClusterElements();
    // Create dataset
    XYDataset dataset = createDataset();

    // Create chart
    JFreeChart chart = ChartFactory.createScatterPlot(
            title: "K-Means Clustering",
            xAxisLabel: "X-Axis", yAxisLabel: "Y-Axis", dataset);


    //Changes background color
    XYPlot plot = (XYPlot) chart.getPlot();
    plot.setBackgroundPaint(new Color( r: 255, g: 228, b: 196));


    // Create Panel
    ChartPanel panel = new ChartPanel(chart);
    setContentPane(panel);
}
```

```java
private XYDataset createDataset() {
    XYSeriesCollection dataset = new XYSeriesCollection();
    XYSeries series1 = new XYSeries( key: "Centers");
    for(int j=0;j< clusters.size() ;j++){
        Cluster current_cluster = clusters.get(j);
        series1.add(current_cluster.getX_Center(),current_cluster.getY_Center());
    }
    dataset.addSeries(series1);
    for (Cluster groupName : clusterElements.keySet()) {
        XYSeries series = new XYSeries(groupName.getClusterNum());
        List<Entry> data = clusterElements.get(groupName);
        for (int i = 0; i < data.size(); i++) {
            Entry entry = data.get(i);
            series.add(entry.getX(), entry.getY());
        }
        dataset.addSeries(series);
    }
    return dataset;
}
```

Για τις 15 επαναλήψεις είναι υπεύθυνη η main καθώς και για την εύρεση της βέλτιστης λύσης και την δημιουργία των plots

```
4 usages
public static void createPlots(K_Means curr){
    SwingUtilities.invokeLater(() -> {
        CreateScatterPlot example = new CreateScatterPlot( title: "Scatter Chart Example",curr);
        example.setSize( width: 800,  height: 800);
        example.setLocationRelativeTo(null);
        example.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
        example.setVisible(true);
    });
}
```

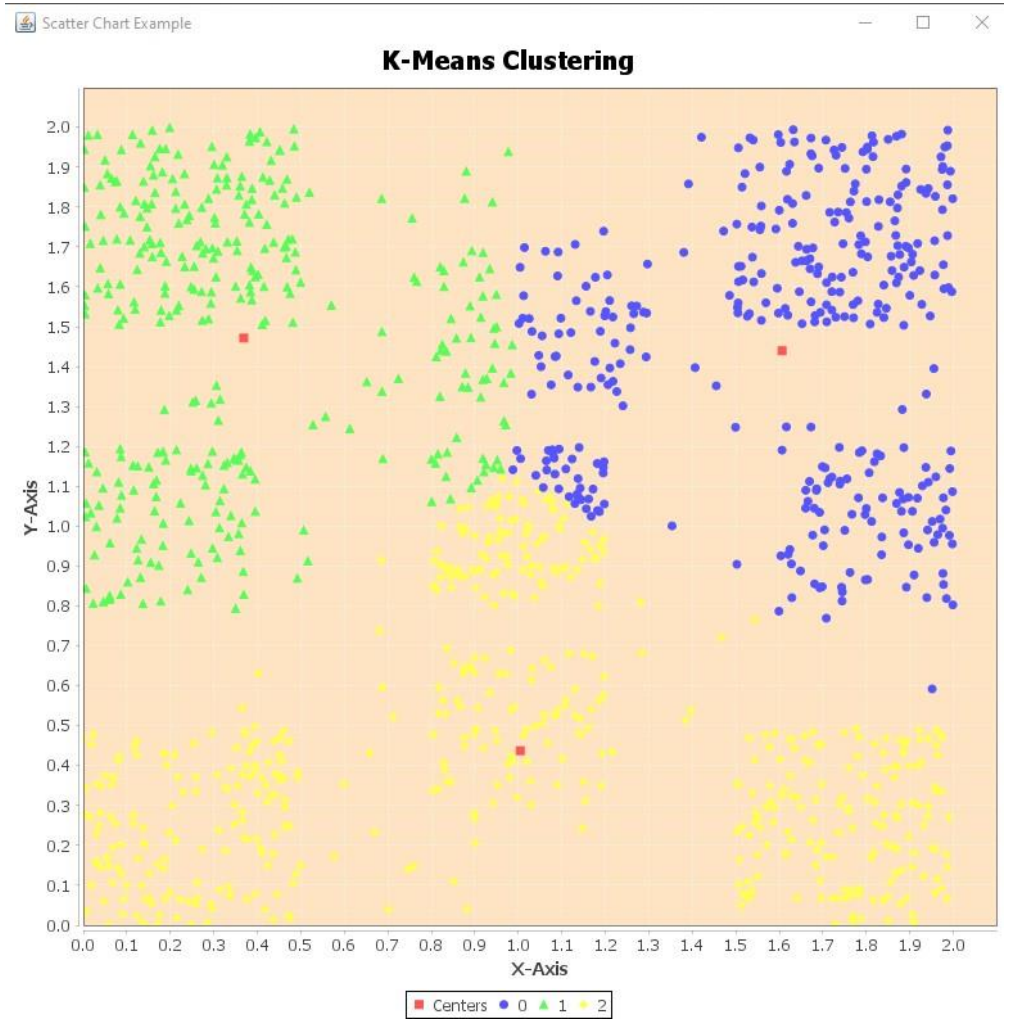
```java
public class Main {

    public static void main(String[] args) {
        Create_SDO array = new Create_SDO();
        double[][] SDO_array = array.Create_SDO_array();
        K_Means three= new K_Means(SDO_array, M: 3);
        K_Means six= new K_Means(SDO_array, M: 6);
        K_Means nine= new K_Means(SDO_array, M: 9);
        K_Means twelve= new K_Means(SDO_array, M: 12);
        three = update(SDO_array,three, M: 3);
        six = update(SDO_array,six, M: 6);
        nine = update(SDO_array,nine, M: 9);
        twelve = update(SDO_array,twelve, M: 12);
        createPlots( three);
        createPlots( six);
        createPlots( nine);
        createPlots( twelve);

    }
```

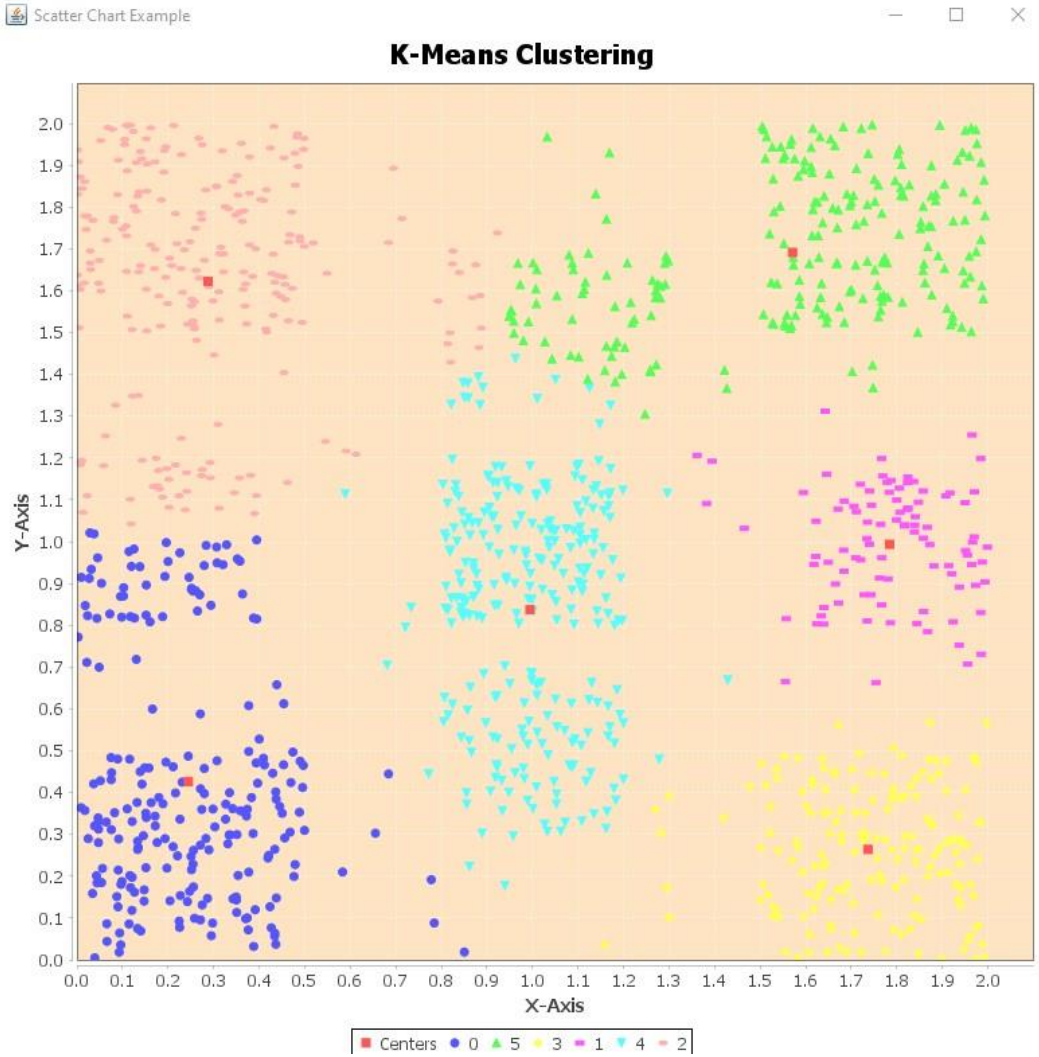
```java
4 usages
public static K_Means update(double[][] SDO_array,K_Means curr,int M){
    for(int i =0 ; i<15 ; i++){
        K_Means new_curr = new K_Means(SDO_array,M);
        if(new_curr.getFull_error()<curr.getFull_error()){
            curr = new_curr;
        }
    }
    return curr;
}
```

Τα αποτελέσματα από τα ScatterPlots:

3

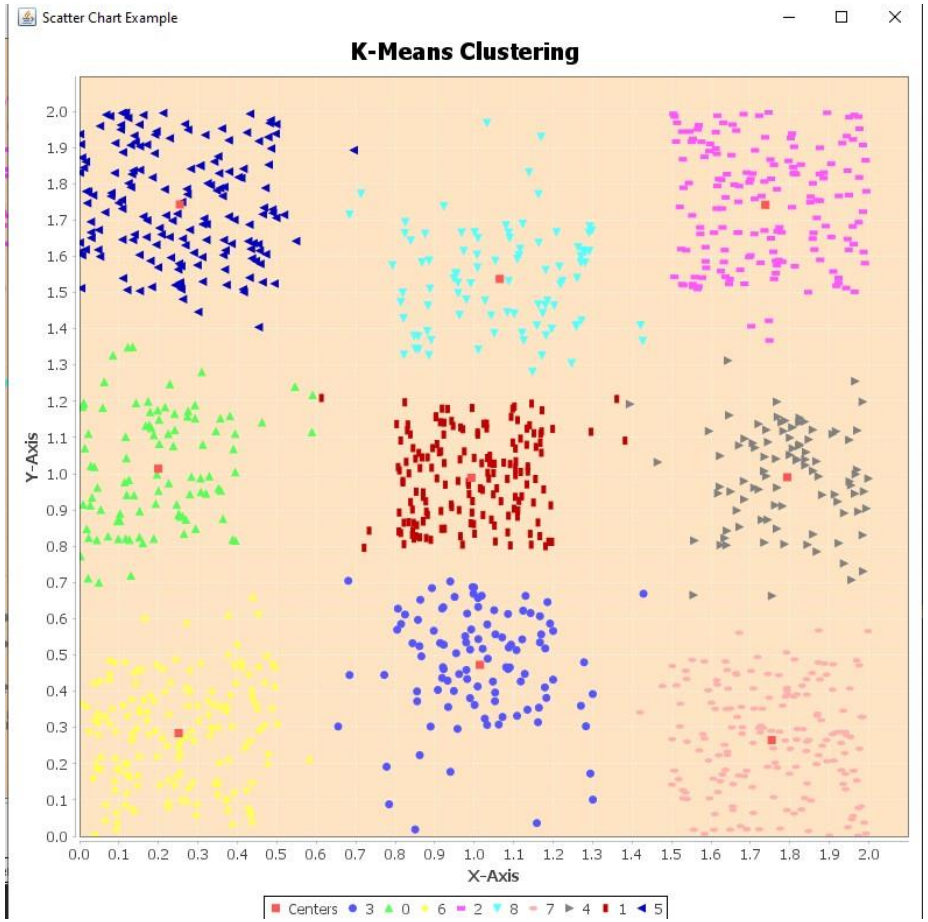
6

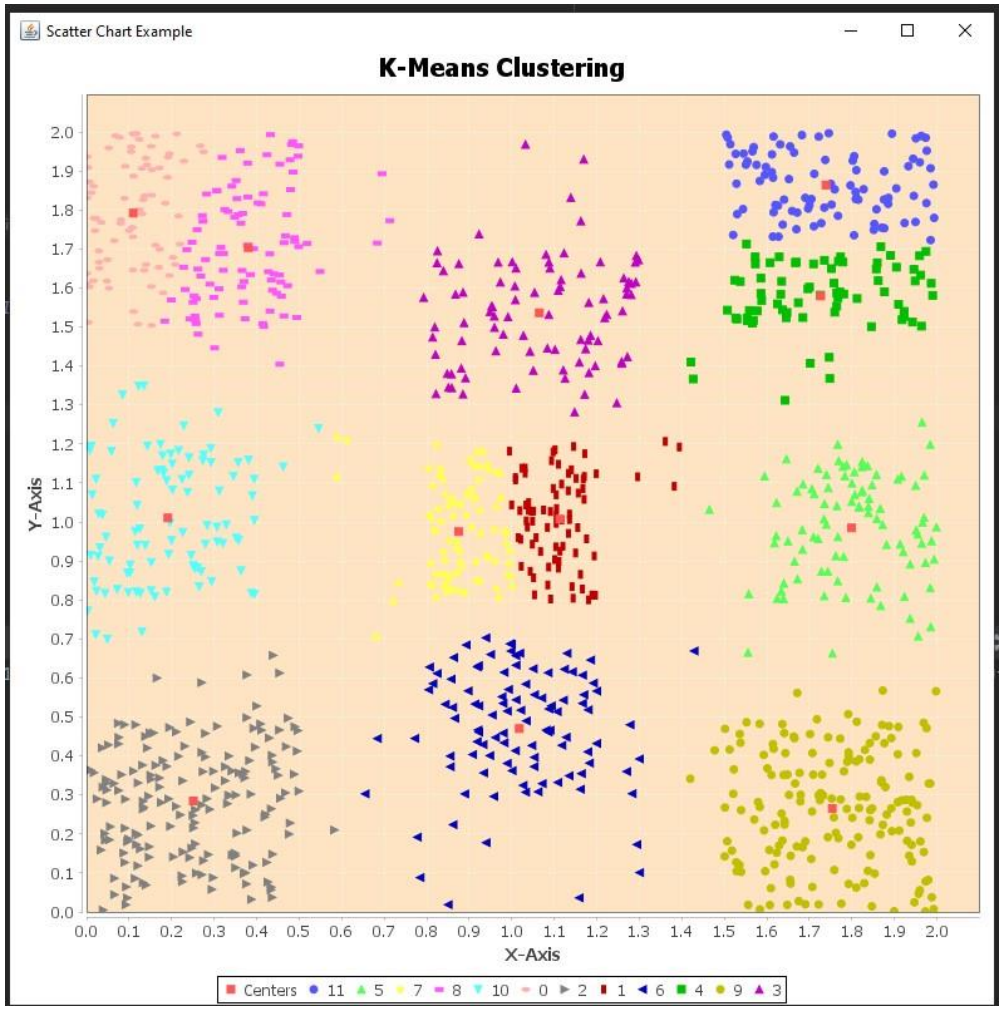

Scatter Chart Example
K-Means Clustering
Y-Axis
X-Axis
Centers 0 1 2


Scatter Chart Example
K-Means Clustering
Y-Axis
X-Axis
Centers 0 5 3 1 4 2

9

12


Scatter Chart Example
K-Means Clustering
Y-Axis
X-Axis
Centers 3 0 6 2 8 7 4 1 5



Τέλος παρατηρούμε από τα σφάλματα ομαδοποίησης ότι έχουμε γόνατο μετά τα 9 κέντρα αρά φτάνουμε στο αποτέλεσμα ότι αυτή είναι η βέλτιστη λύση την οποία δίνεται για το συγκεκριμένο πρόβλημα.

```
620.4007664315346
322.4616091283716
225.06263533931067
201.8219855185549
```
3 κέντρα
6 κέντρα
9 κέντρα
12 κέντρα