UNIVERSIDAD DE SONORA

DIVISION DE INGENIERIA

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

Tarea: K-MEANS Base de datos IRIS

Carrera: Ingeniería en Sistemas de Información

Alumnos: Issam Silverio Jimenez Ortega

Ruddy Miranda Marez

Victor Hugo Garcia Mendez

Materia: Minería de datos

Maestra: Raquel Torres Peralta

Hora: 12:00 – 1:00 PM

Hermosillo Sonora

07 de Abril del 2022

Índice

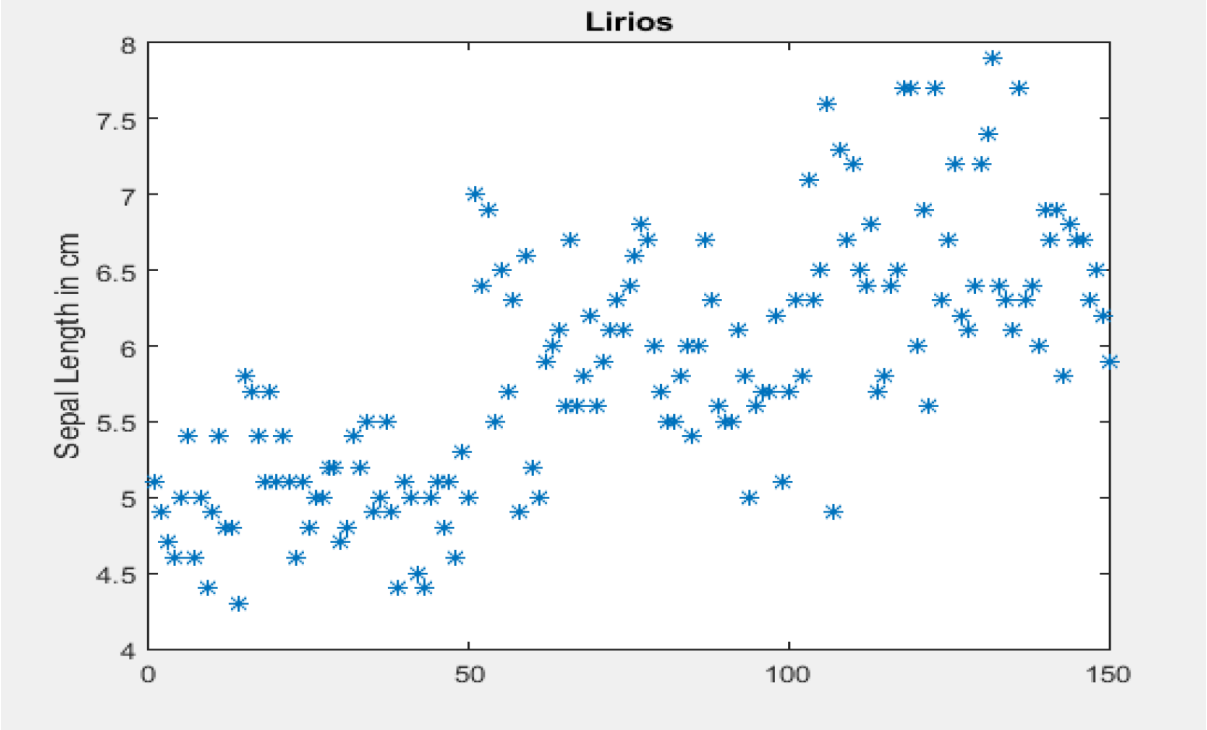
En este trabajo se aplicará el algoritmo k-means a una base de datos de Iris para clasificación.

La base de datos: http://archive.ics.uci.edu/ml/datasets/Iris

Análisis de las variables.

Sepal Length in Cm.

Rango 4-8

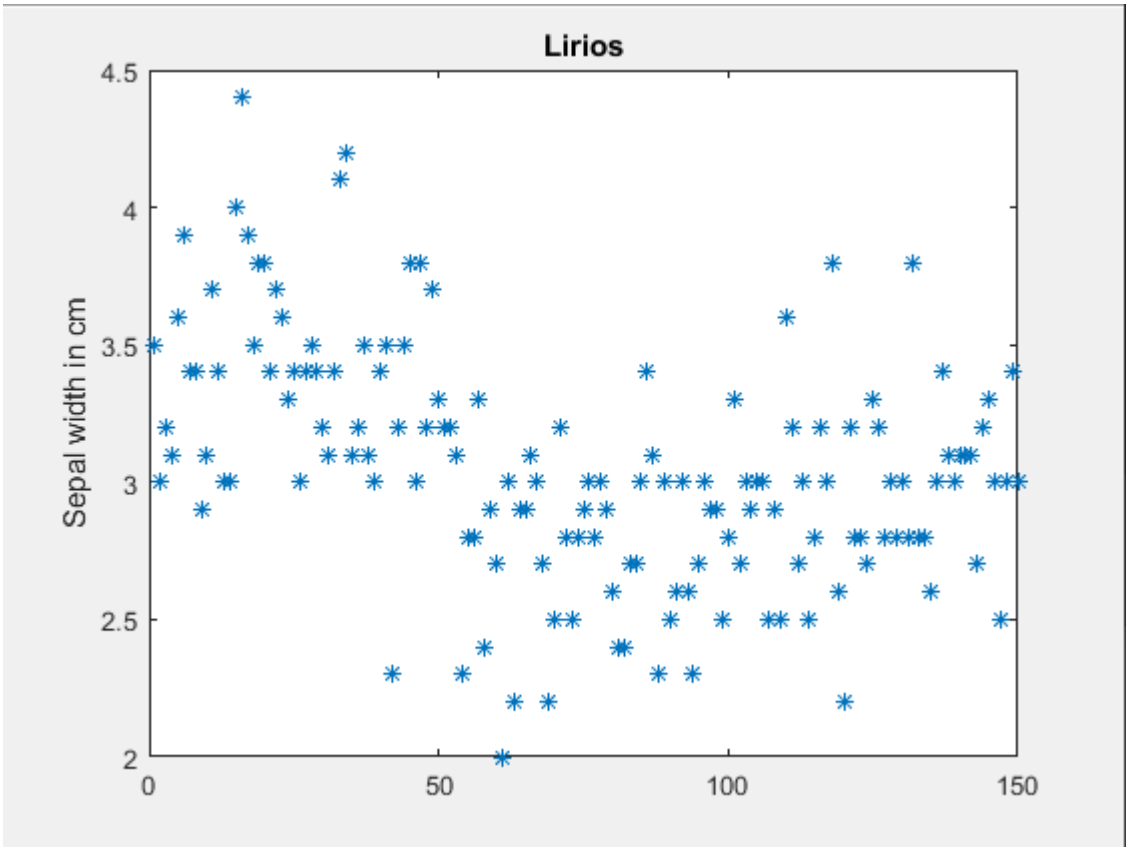


Media	5.8433
Desviación Estándar	.8281

En esta primera característica de los lirios se puede observar que la longitud media se encuentra cerca de 5 cm, aunque llega haber de 8 cm son pocos casos. Es visible tanto en la gráfica como en su media y desviación estándar que la mayoría de las muestras el pétalo del lirio es de tamaño mediano.

Sepal width in cm

Rango 2-4

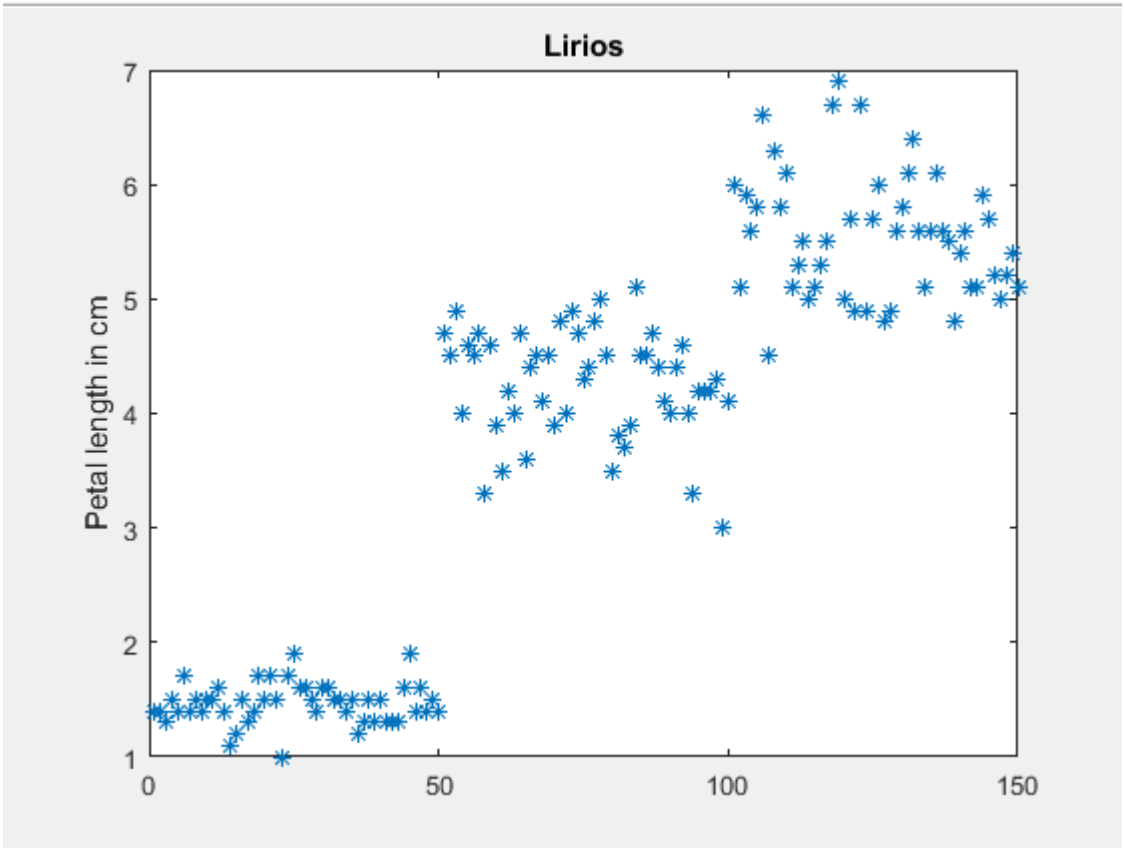


Media	3.0540
Desviación Estándar	.4336

En esta característica se habla del ancho del sépalo en los lirios se puede observar que el ancho se encuentra cerca de 3 cm, aunque llega haber de 4.5 cm son pocos casos. Es visible tanto en la gráfica como en su media y desviación estándar que la mayoría de las muestras el sépalo del lirio es de tamaño mediano.

Petal length in cm

Rango 1-7

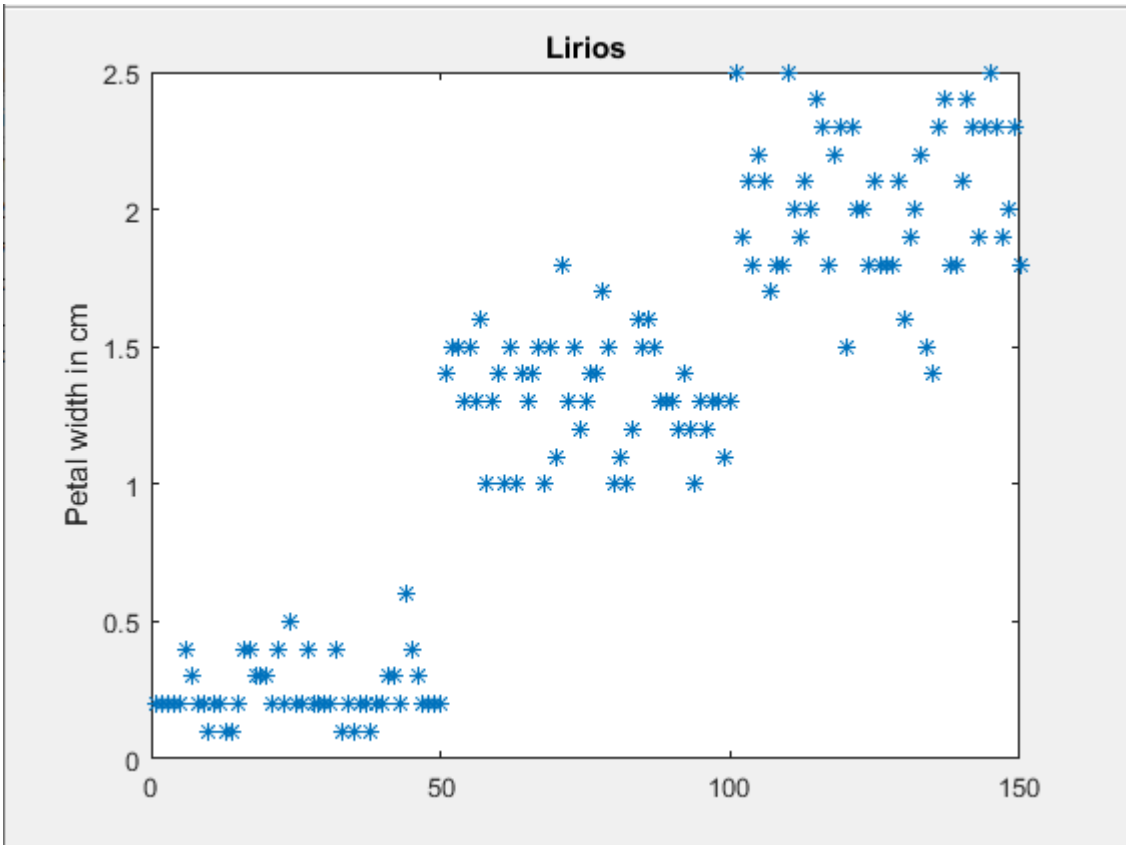


Media	3.7587
Desviación Estándar	1.7644

En esta característica podemos observar la agrupación en 3 medidas distintas, la primera entre 1 y 2 cm con poca variación, el segundo grupo entre 3 y 5 cm con una variación bastante notable y por último el grupo de más de 5 cm hasta los 7 cm de largo con una variación bastante notable en el tamaño de los pétalos.

Petal width in cm

Rango 0- 2.5



Media	1.1987
Desviación Estándar	0.7632

En el atributo de anchura podemos observar la agrupación en 3 medidas distintas, la primera entre 0 y 0.5 con poca variación, el segundo grupo entre 1 y 1.5 cm con una variación bastante notable y por último el grupo de más de 2 cm hasta los 2.5 cm de ancho con una variación bastante notable en el tamaño de los pétalos.

Procedimiento

Primero obtenemos los datos de los centroides iniciales obteniendo un número aleatorio entre el número máximo y mínimo de cada atributo de nuestra base de datos.

```
12
13                          %C E N T R O I D E S   I N I C I A L E S
14 -    k = 3; %Numero de centroides
15 -    Centroides = zeros(k,4); %Tabla de Centroides
16
17 -    for i=1:k %Iteramos por los registros renglon por renglon
18 -        for j=1:4 %Por cada renglon iteramos columna por columna
19
20 -            Max = int16(max(iris(:,j))); %Numero maximo
21 -            Min = int16(min(iris(:,j))); %Numero minimo
22
23 -            Random = randi([Min,Max],1); %un solo valor random entre el minimo y el maximo
24
25 -            Centroides(i,j) = Random; %Asignamos el valor a la casilla de centroide correspondiente
26
27 -        end
28 -    end
```

Así nos quedaron los centroides iniciales

iris | Centroides

3x4 double

	1	2	3	4
1	6	2	6	2
2	7	4	1	2
3	5	2	3	1

Teniendo nuestros centroides podemos calcular las distancias que hay entre cada atributo de la tabla y el centroide para poder ver cual es el que le queda más cerca y asignar cada registro para formar los clusters iniciales.

```
41    %Con este bucle calculamos las distancias de cada atributo con cada centroide y le asignamos el centroide mas sercano
42 -  while (finalizado(1,1) == 0)
43 -      for b=1:k
44 -          for a=1:T
45                %En Distancias guardamos las distancias de las muestras con los centroides
46 -              Distancias(a,b) = (((iris(a,1)-Centroides(b,1))^2)+((iris(a,2)-Centroides(b,2))^2)+((iris(a,3)-Centroides(b,3))^2)+((i
47                %Con estos if asignamos el centroide mas sercano a cada muestra para formar los clusters
48 -              if Distancias(a,1)<Distancias(a,2)&&Distancias(a,1)<Distancias(a,3)
49 -              Clusters1(a,1) = 1;
50 -              elseif Distancias(a,2)<Distancias(a,1)&&Distancias(a,2)<Distancias(a,3)
51 -              Clusters1(a,1) = 2;
52 -              elseif Distancias(a,3)<Distancias(a,1)&&Distancias(a,3)<Distancias(a,2)
53 -              Clusters1(a,1) = 3;
54 -              end;
55 -          end;
56 -      end;
```

Así quedó nuestra tabla de distancias y nuestra tabla de asignación de cluster y su representación visual tomando como referencia los atributos 3 y 4.

iris | Centroides | Distancias

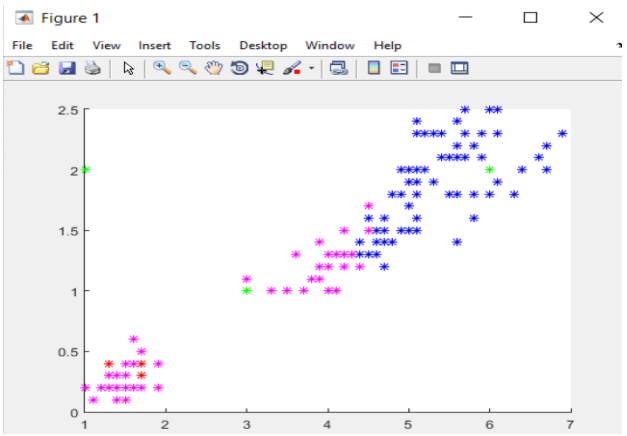
150x3 double

	1	2	3
1	27.4600	7.2600	5.4600
2	26.6100	8.8100	4.2100
3	28.4600	9.2600	5.0600
4	26.6600	10.0600	4.2600
5	27.9600	7.5600	5.7600
6	25.0200	5.6200	5.8200
7	27.9700	9.1700	5.1700
8	26.4500	7.8500	4.8500
9	27.7700	11.3700	4.3700
10	26.2800	9.0800	4.2800
11	26.7400	6.1400	5.9400
12	26.0000	8.8000	4.6000
13	27.2100	9.6100	4.4100
14	31.5100	11.9100	5.9100
15	30.3200	4.7200	8.5200
16	28.6600	4.6600	8.8600
17	28.6200	5.2200	7.0200
18	27.1100	6.9100	5.3100
19	24.7100	5.1100	5.9100
20	27.1900	6.7900	5.9900

Clusters

150x1 double

	1
1	3
2	3
3	3
4	3
5	3
6	2
7	3
8	3
9	3
10	3
11	3
12	3
13	3
14	3
15	2
16	2
17	2
18	3
19	2
20	3

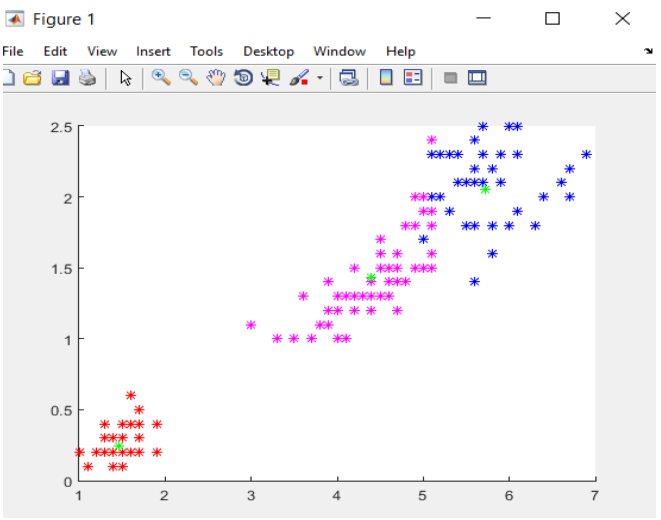


Re-asignamos cada centroide con la media de los atributos que quedaron en el centroide correspondiente, esto lo hacemos para acomodar el centroide cada vez más al centro de nuestros clusters y así podemos recalcular las distancias para ver los cambios en la asignación de clusters.
Finalmente nos queda repetir el proceso hasta que nuestra tabla de asignaciones de cluster no cambie, en nuestro caso con ayuda de un contador nos dimos cuenta que el algoritmo dio 13 vueltas para llegar a la respuesta final.

```
57          %Reasignamos los Centroides con la media de cada atributo
58 -        c1map = (Clusters1==1);
59 -        c1 = iris(c1map,:);
60 -        Centroides(1,:) = mean(c1);
61
62 -        c2map = (Clusters1==2);
63 -        c2 = iris(c2map,:);
64 -        Centroides(2,:) = mean(c2);
65
66 -        c3map = (Clusters1==3);
67 -        c3 = iris(c3map,:);
68 -        Centroides(3,:) = mean(c3);
69
70 -        if (Clusters1 == Clusters)
71 -            finalizado(1,1) = 1;
72 -        else
73 -            Contador = Contador+1;
74 -            Clusters = Clusters1;
75 -            Clusters1(:,1) = 0;
76 -        end;
77 -    end;
```

Nuestra tabla de distancias, asignación de cluster y gráfica tomando los atributos 3 y 4.

	1	2	3		1
1	25.3143	0.0216	11.6452	1	2
2	25.8827	0.1920	11.4897	2	2
3	27.5866	0.1700	12.6744	3	2
4	26.2866	0.2692	11.6439	4	2
5	25.7696	0.0392	11.9738	5	2
6	21.6496	0.4676	9.8754	6	2
7	26.8827	0.1724	12.3002	7	2
8	24.7473	0.0036	11.0824	8	2
9	28.1120	0.6416	12.6815	9	2
10	25.4050	0.1344	11.2206	10	2
11	23.7081	0.2380	11.0210	11	2
12	24.6958	0.0632	10.9082	12	2
13	26.6643	0.2420	11.9533	13	2
14	31.6473	0.8264	15.1700	14	2
15	25.7881	1.0408	13.2826	15	2
16	23.5866	1.4716	12.1997	16	2
17	25.0220	0.4388	12.1862	17	2
18	24.9535	0.0228	11.4083	18	2
19	21.0535	0.6864	9.6703	19	2
20	24.4443	0.1592	11.3661	20	2



Con esto nos podemos dar cuenta que la clase de lirios 1 (Iris-Setosa) la representa el cluster número 2 representada en la gráfica con el color rojo.
La clase de lirios 2 (Iris-Versicolor) la representa el cluster número 3 representada en la gráfica con el color rosa.
La clase de lirios 3 (Iris-Virginica) la representa el cluster número 1 representada en la gráfica con el color azul.
También podemos darnos cuenta que el algoritmo junto correctamente casi todos los registros de nuestra base de datos.

Métricas

Primero agregamos los números de cluster correspondientes a los registros de la base de datos, después separamos los datos por clase para poder comparar los datos reales con los obtenidos con el algoritmo y así poder medir el número de aciertos y errores obtenidos por cada clase.

```
89 -    iris(1:50,5)= 2; %Agregamos el numero del cluster correspondiente de la clase en la base de datos
90 -    Setosa = iris(1:50,:); %Guardamos solo la primera clase
91 -    TPositive = sum(iris(1:50,5) == Clusters(1:50)); %True positive para la primera clase
92 -    FalseNegative1 = size(Setosa,1)-TPositive; %False Negative para la primer clase
93
94
95 -    iris(51:100,5)= 3; %Agregamos el numero 3 que es el cluster correspondiente
96 -    Versicolor = iris(51:100,:); %Separamos la segunda clase
97 -    TPositive2 = sum(iris(51:100,5)== Clusters(51:100));%Verdaderos positivos para esta clase
98 -    FalseNegative2 = size(Versicolor,1)-TPositive2; %Falsos negativos para esta clase
99 -    PorcentajeVersicolor = TPositive2/size(Versicolor,1); %Porcentaje de acierto para esta clase
100
101 -   iris(101:150,5)= 1;
102 -   Virginica = iris(101:150,:);
103 -   TPositive3 = sum(iris(101:150,5) == Clusters(101:150));
104 -   FalseNegative3 = size(Virginica,1)-TPositive3;
105 -   PorcentajeVirginica = TPositive3/size(Virginica,1);
```

Primer clase Iris-Setosa

```
108     %Metricas para la primer clase
109 -   PorcentajeSetosa = TPositive/size(Setosa,1); %Porcentaje de acierto para la primera clase.
110 -   Precision1= TPositive/TPositive + (FalseNegative2+FalseNegative3);
111 -   Recall1= TPositive/TPositive-FalseNegative1;
112 -   FScore1= 2*Precision1 * Recall1 / (Precision1 + Recall1);
```

Porcentaje de aciertos para esta clase: 100%

Precisión: 18

Recall: 1

F Score: 1.8947

Segunda clase Iris-Versicolor

```
%Metricas para la segunda clase
PorcentajeVersicolor = TPositive2/size(Versicolor,1); %Porcentaje de acierto para esta clase
Precision2= TPositive2/TPositive2 + (FalseNegative3); %Los falsos negativos de las otras clases son los falsos positivos de esta
Recall2= TPositive2/TPositive2+FalseNegative2;
FScore2= 2*Precision2 * Recall2 / (Precision2 + Recall2);
```

Porcentaje de aciertos para esta clase: 94%

Precisión: 15

Recall: 4

F Score: 6.3158

Tercer clase Iris-Virginica

```
%Metricas para la tercera clase
PorcentajeVirginica = TPositive3/size(Virginica,1);
Precision3= TPositive3/TPositive3 + (FalseNegative2); %Los falsos negativos de las otras clases son los falsos positivos de esta
Recall3= TPositive3/TPositive3+FalseNegative3;
FScore3= 2*Precision3 * Recall3 / (Precision3 + Recall3);
```

Porcentaje de aciertos para esta clase: 72%

Precisión: 4

Recall: 15

F Score: 6.3158

Métricas Generales

Sumamos lo obtenido en las métricas individuales y las dividimos por 3 que es el número de clases.

Porcentaje de aciertos: 88.66%

Precisión: 12.33

Recall: 20

F Score: 4.8421

Conclusión

Ante los resultados obtenidos, hemos obtenido los resultados que dividen entre las clases de lirios, asignándoles como clústeres de forma aleatoria asignando el proceso reajustando la ubicación de los clusters para finalmente tener una ubicación media adecuada que funcione para la separación en grupos, las cuales podemos asignar la clase de lirio que tiene cada una de estos grupos.