Instituto Tecnológico y Estudios Superiores de Monterrey

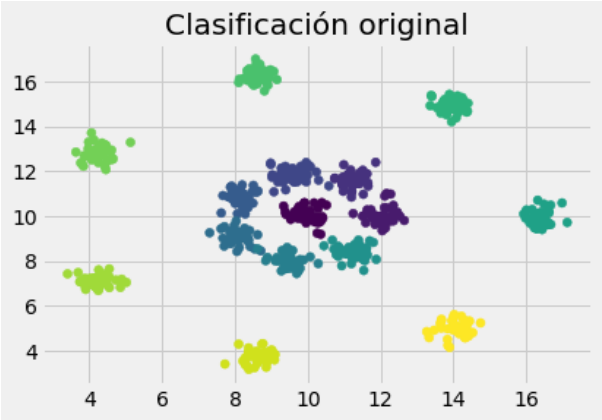
Diana Cañibe Valle A01749422

Materia: Inteligencia artificial avanzada para la ciencia de datos I (Gpo 101)

Fecha: 10 de Septiembre del 2022

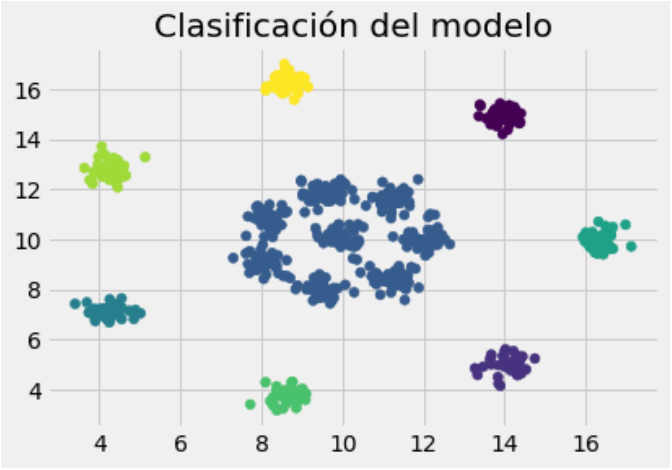
Momento de Retroalimentación: Módulo 2 Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución

El modelo seleccionado para esta entrega es K-means, basado en centroides para hacer la clasificación de clusters, el dataset utilizado es el shape set R15, cuya clasificación original es la siguiente:



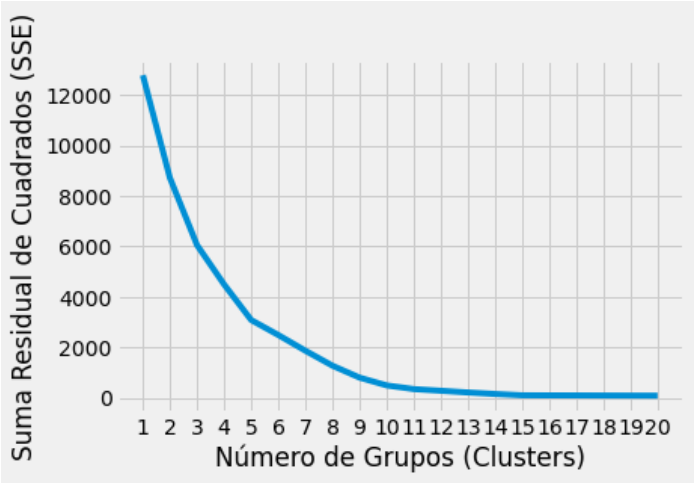
Utilizaré el método del codo para determinar la mejor cantidad de clusters, pero plantearé la corrida inicial con los valores default de la función, los cuales son: (n_clusters =8, init='k-means++',n_init=10,max_iter=300), pero utilizare el argumento de "random_state: 12", para poder hacer comparaciones y visualizar con mayor claridad los resultados de utilizar la gráfica de codo para la siguiente prueba y los cambios en los demás argumentos.

El valor default de clusters en la función es 8, por ello si se envía el modelo sin ningún argumento esta será la cantidad de grupos que intentará clasificar, y como podemos apreciar en la gráfica une grupos que podemos identificar son independientes originalmente.



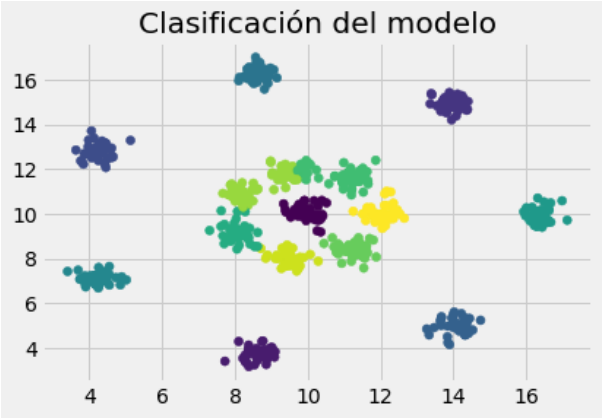

```
#Modelo
model = KMeans(random_state=12)
model.fit(X)
```

La gráfica de codo obtenida es la siguiente:



Como podemos observar el valor se estabiliza a partir de los grupos 14-15, algo razonable considerando que originalmente son 15 grupos.

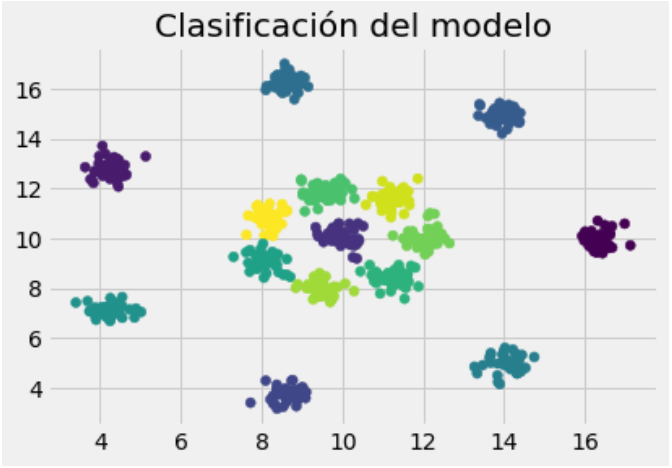
Si no supiéramos el valor exacto podríamos hacer una prueba con 14, lo que nos da la siguiente clasificación:




```
#Modelo
model = KMeans(n_clusters=14,random_state=12)
model.fit(X)
```

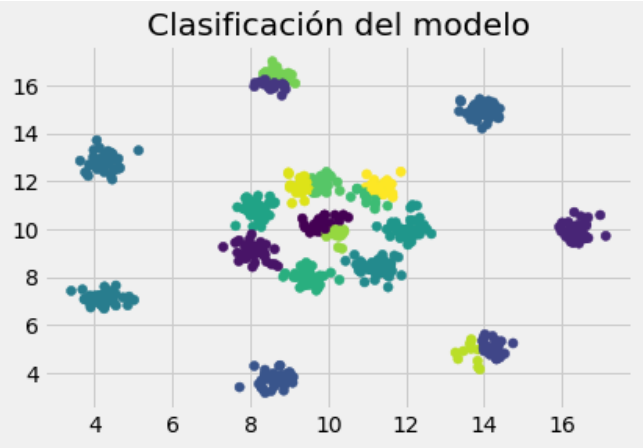
Podemos ver que hay una confusión en un par de los grupos centrales, pero en general los demás grupos han sido clasificados adecuadamente.

Al correr el modelo con los 15 grupos, vemos la clasificación exacta:



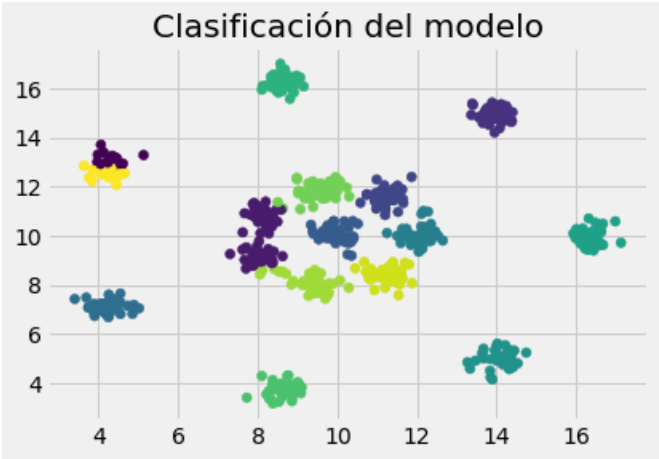

```
#Modelo
model = KMeans(n_clusters=15,random_state=12)
model.fit(X)
```

Si bien ya tenemos la clasificación por mera experimentación cambiaremos los parámetros para identificar el comportamiento del modelo con los mismos.



Al aumentar el número de grupos, el modelo forza la clasificación de grupos en más, lo que en un caso real podría no tener significado, y solo un gasto de recursos, ya que no son significativos los nuevos clusters.

```
#Modelo
model = KMeans(n_clusters=20,random_state=12)
model.fit(X)
```



Si cambiamos el método de inicialización a ‘random’ podemos ver un cambio notable al obtenido con el valor default que usa un método de distribuciones normales a seleccionar un valor aleatorio de los datos para los centroides, lo que causa que incluso con la cantidad correcta de grupos, no sean iguales a la figura original.

```
#Modelo
model = KMeans(n_clusters=15,init="random",random_state=12)
model.fit(X)
```

Podemos alterar los demás valores como la cantidad de iteraciones o los centroides iniciales, pero los resultados permanecen constantes demostrando que tiene una baja varianza y ‘bias’, prevaleciendo una alta precisión.

El modelo de k-means funciona bien con este set de datos, debido a que el tamaño de los grupos es similar, y la implementación del framework proporciona resultados precisos siempre y cuando se determine adecuadamente la cantidad de clusters requeridos.