Inteligencia Artificial

Proyecto Final IA: Unsupervised Machine Learning

Profesor: Mauricio Alejandro Cabrera Arellano

Alumno: Omar Josue Munguia Camacho

Registro: 21110391

Grupo: 6E2

- Tema seleccionado:

Clustering: Unsupervised Machine Learning

- ¿Por qué elegí este tema?

Me interesó la forma en la que la maquina puede detectar y tomar sus propias decisiones sobre como seleccionar, dividir y organizar un conjunto de datos sin la necesidad de que haya intervención humana en dicha separación de datos. Me agradó la idea de poder generar datos de manera aleatoria y que el programa lograra ajustar dicha información de la manera que más le conviene.

- ¿Es monetizable?

Por supuesto, se puede implementar en muchos aspectos dentro de la vida diaria como en la industria, ejemplos de ello serian:

1. **Segmentación de Clientes:** Permite a las empresas segmentar a sus clientes en grupos basados en comportamientos de compra, demografía y otros datos relevantes para campañas de marketing dirigidas.
2. **Análisis de Sentimientos:** Agrupa comentarios y opiniones de clientes en categorías para análisis de sentimientos, ayudando a las empresas a entender la percepción del cliente y mejorar sus productos o servicios.
3. **Detección de Fraude:** Identifica patrones anómalos en transacciones financieras que podrían indicar actividades fraudulentas, mejorando la seguridad y reducción de pérdidas.
4. **Optimización de la Cadena de Suministro:** Agrupa productos según patrones de demanda, permitiendo una mejor gestión de inventarios y optimización de la cadena de suministro.
5. **Investigación Científica:** Agrupa datos experimentales para descubrir nuevos patrones y relaciones en campos como la biología, la química y la física.

- Ejemplo aplicado de Clustering sin Supervisión

En este caso mostraré dos formas de generar agrupamiento sin supervisión: el Flat Clustering y el Hierarchical Clustering.

- Flat Clustering

El Flat Clustering (también conocido como Semi-supervised Clustering) tiene la cualidad de que el usuario controla manualmente la cantidad de grupos que se quieren formar, de ahí semi-supervisado. Para mostrar su uso, se colocaran puntos aleatorios mediante coordenadas x,y, luego, colocaremos la cantidad de grupos que queremos que se formen y dejaremos que la maquina seleccione por sí misma como agrupar dichos puntos.

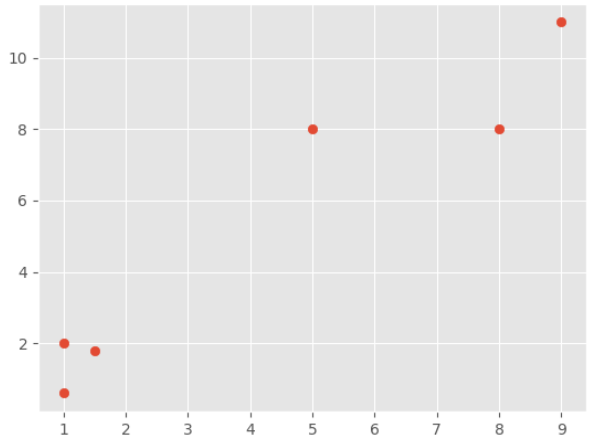
Primero importamos las librerías necesarias para el funcionamiento del programa:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
from sklearn.cluster import KMeans
style.use("ggplot")
```

Una vez colocadas las librerías, comenzamos colocando las coordenadas x,y de los puntos deseados para nuestro agrupamiento y los mostramos en una grafica:

```
x = [1, 5, 1.5, 8, 1, 9]
y = [2, 8, 1.8, 8, 0.6, 11]
plt.scatter(x,y)
plt.show()
```

Una vez hecho esto, obtendremos nuestra gráfica con los puntos



Ahora bien, ya que tenemos los puntos en la gráfica, los colocamos en un array para agrupar la información en un solo lugar, luego utilizando la función KMeans

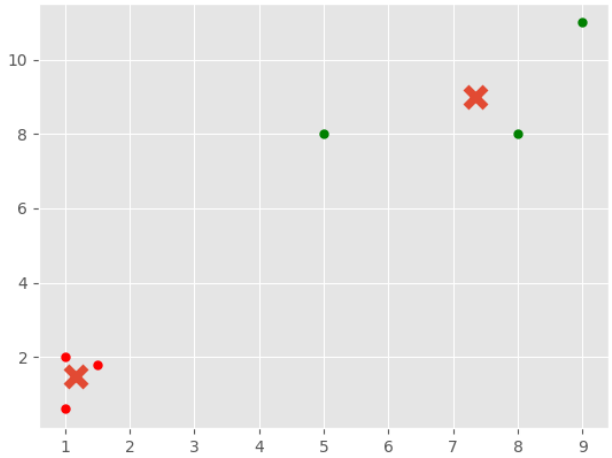
seleccionamos la cantidad de clusters deseados e ingresamos los datos en la función y por último calculamos los centroides de cada cluster y colocamos una variable para separar cada punto en su debido grupo. Ponemos una variable colors, para que pinte de diferente color los puntos (g. = green, r. = red):

```
X = np.array([[1,2],[5,8],[1.5,1.8],[8,8],[1,0.6],[9,11]])
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
centroids = kmeans.cluster_centers_
labels = kmeans.labels_
colors = ["g.","r."]
```

Para poder separar cada punto individualmente colocamos la siguiente función:

```
for i in range(len(X)):
    print("Coordenada:", X[i], "Label:", labels[i])
    plt.plot(X[i][0], X[i][1], colors[labels[i]], markersize = 10)
```

De esta forma, ahora tenemos cada punto seleccionado y separado por el cluster que la maquina considera correcta. Ahora solo queda muestrear nuestra nueva gráfica con los centroides marcados y los puntos separados por color:



De esta forma, la máquina separó de forma automática dos grupos de puntos de forma automática, sin embargo, se tiene que ingresar manualmente la cantidad de grupos que se desea, pero, ¿qué pasa si no quiero que sea así?, para esto se requiere el siguiente método de Clustering.

- Hierarchical Clustering:

A diferencia del Flat Clustering, el Hierarchical Clustering calcula por sus propios algoritmos la cantidad de clusters que se van a dividir, esto significa que no necesita ninguna acción externa por parte de un humano para decidir como separar la información de cada cluster, Sin embargo, esto tiene sus desventajas como por ejemplo, si los datos están muy agrupados, puede llegar a acomodar todo como un solo grupo sin separaciones, es por eso que si no se separan apropiadamente los datos desde un inicio, puede llegar a hacer cosas raras.

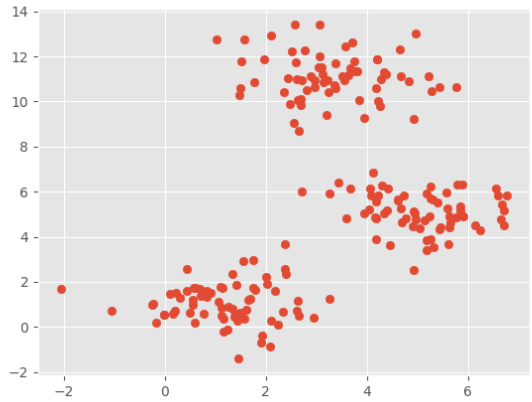
En este caso, podemos utilizar la función “generate_blobs()” para generar un grupo de datos dentro de la gráfica, de manera de que se generen una gran cantidad de puntos sin necesidad de ponerlos manualmente. Para esto se necesitan agregar unas librerías más para este caso.

```
from sklearn.cluster import MeanShift
from sklearn.datasets._samples_generator import make_blobs
ms = MeanShift()
```

Una vez ingresadas nuestras nuevas librerías y comandos, empezamos colocando unos centros en los cuales se agruparán nuestros puntos, personalmente elegí que se colocaran 200 puntos con una desviación estándar de 1, para que se pudiera apreciar la separación de los grupos.

```
centers = [[1,1],[5,5],[3,11]]
Z, _ = make_blobs(n_samples = 200, centers = centers, cluster_std
= 1)
plt.scatter(Z[:,0], Z[:,1])
plt.show()
```

De esta manera se muestra la siguiente gráfica:



Ahora bien, mandamos a llamar la función MeanShift para colocar los puntos dentro de ella, generamos las marcas para los grupos, hacemos una variable para que detecte los centros de cada grupo y por último generamos una variable que servirá para que el programa decida de forma automática cuantos grupos desea generar:

```
ms.fit(Z)
labelsZ = ms.labels_
cluster_centers = ms.cluster_centers_
n_clusters_ = len(np.unique(labelsZ))
colorsZ = 10*['r.','g.','c.','b.','k.','y.','m.']
```

En esta parte, la función unique() sirve para que decida de forma automática cuantos clusters se van a generar independientemente.

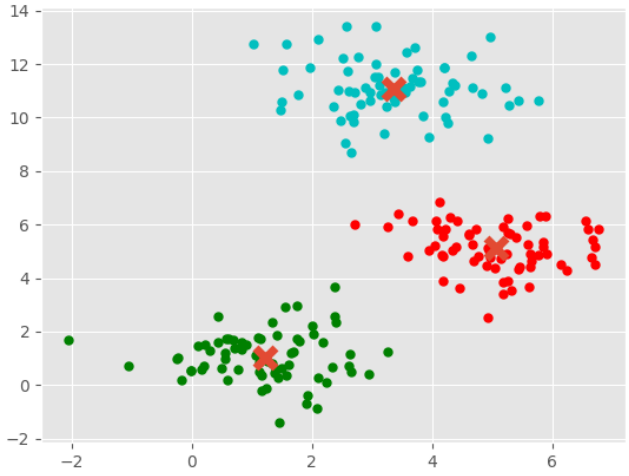
Nota: Podemos imprimir el número de clusters generados y sus centros de la siguiente manera, esto para visualizar de mejor manera como se ajustaron los valores de nuestros grupos y verificar que los datos que ingresamos son válidos:

```
print("Numero estimado de Clusters:", n_clusters_)
print("Centros de los Clusters:", cluster_centers)
```

Por último, al igual que el Flat Clustering, seleccionamos cada punto generado y lo ponemos en una función para separar cada grupo y mostramos nuestro resultado:

```
for v in range(len(Z)):
    plt.plot(Z[v][0], Z[v][1], colorsZ[labelsZ[v]], markersize =
10)
plt.scatter(cluster_centers[:,0], cluster_centers[:,1], marker =
"x", s = 150, linewidths = 5, zorder = 10)
plt.show()
```

Dándonos así, la siguiente gráfica:



En esta se puede apreciar como se colocaron los clusters de diferentes colores y con su respectivo centro, los cuales (dentro de la consola) dieron los siguientes resultados:

```
Numero estimado de Clusters: 3
Centros de los Clusters:
 [[ 5.05974356  5.11975618]
 [ 1.22859087  1.01744042]
 [ 3.36472595 11.07473561]]
```

Los cuales, se aproximan bastante a nuestros iniciales [1,1], [5,5] y [3,11].

De esta forma probamos de primera mano el funcionamiento de el Agrupamiento Sin Supervisión, en el cuál, no requerimos dar ningún tipo de parámetro para generar los grupos, simplemente ingresando nuestros datos a la función y dejando que la máquina haga su trabajo.

En conclusión:

El Unsupervised Machine Learning nos permite acumular datos de manera eficiente y sin intervención externa. Lo que llevaría horas realizar a mano ahora se puede automatizar de manera casi instantánea mediante un algoritmo, lo que nos ayuda a procesar grandes cantidades de información de manera compacta y sencilla.

Referencias:

- *Python programming tutorials*. (n.d.). https://pythonprogramming.net/flat-clustering-machine-learning-python-scikit-learn/

Repositorio de GitHub:

- https://github.com/OmarMunguiaC/3P_InteligenciaArtificial_21110391/tree/946e1ceff032aaf8cfe20189e7954875415d8437/Proyecto%20Final

Videotutorial en YouTube:

- https://youtu.be/qz4MShcX_xc?si=7lPlP9p8VilMmIYt

Video de TikTok:

- https://vm.tiktok.com/ZMrFs7Wao/