UANL

FCFM

CLUSTERING

DATA MINING

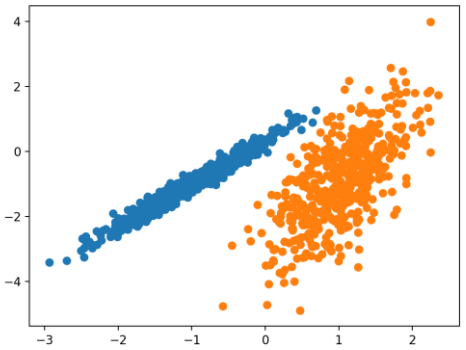
Anahí Alemán Alvarado 1821952
Ricardo Zarek Sánchez Olivares 1795134
Juan Pablo Nasser Benavides 1753367
Eduardo Almaguer Alanís 1741322
Oscar Saúl Vega Macias 1626997

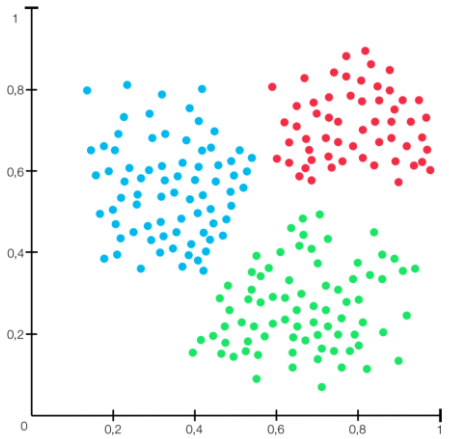
¿Qué es?

También conocido como agrupamiento, es una de las técnicas de minería de datos, el proceso consiste en la división de los datos en grupos de objetos similares.

Las técnicas de Clustering son las que utilizando algoritmos matemáticos se encargan de agrupar objetos. Usando la información que brindan las variables que pertenecen a cada objeto se mide la similitud entre los mismos, y una vez hecho esto se colocan en clases que son muy similares internamente y a la vez diferente entre los miembros de las diferentes clases.

Fuente: https://quantdare.com/quantitative-clustering-with-machine-learning/



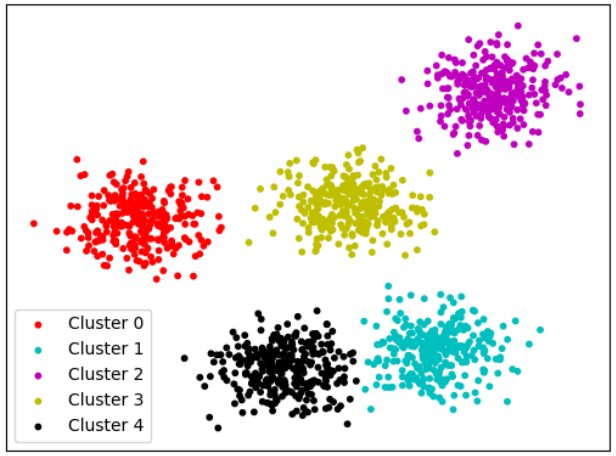


Un **cluster** es una colección de objetos de datos. Similares entre sí dentro del mismo grupo. Disimilar a los objetos en otros grupos.

Análisis de cluster: dado un conjunto de puntos de datos tratar de entender su estructura. Encuentra similitudes entre los datos de acuerdo con las características encontradas en los datos. Es un aprendizaje no supervisado ya que no hay clases predefinidas.

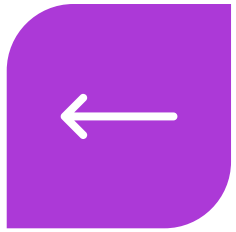
Fuente:https://medium.com/@jaywrkr/miner%C3%ADa-de-datos-4-3bc03e7fc234#:~:text=La%20agrupaci%C3%B3n%20(Clustering)%20busca%20agrupaciones,una%20peque%C3%B1a%20distancia%20entre%20s%C3%AD.

Aplicaciones



Fuente: https://medium.com/@jaywrkr/miner%C3%ADa-de-datos-4-3bc03e7fc234#:~:text=La%20agrupaci%C3%B3n%20(Clustering)%20busca%20agrupaciones,una%20peque%C3%B1a%20distancia%20entre%20s%C3%AD.

Métodos de agrupación

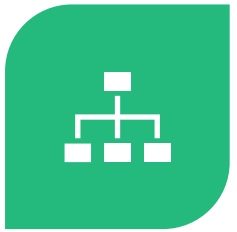


ASIGNACIÓN JERÁRQUICA FRENTE A PUNTO

DATOS NUMÉRICOS Y/O SIMBÓLICOS

DETERMINÍSTICA VS. PROBABILÍSTICA

EXCLUSIVO VS. SUPERPUESTO



JERÁRQUICO VS. PLANO.

DE ARRIBA A ABAJO Y DE ABAJO A ARRIBA

Fuente: https://medium.com/@jaywrkr/miner%C3%ADa-de-datos-4-3bc03e7fc234#:~:text=La%20agrupaci%C3%B3n%20(Clustering)%20busca%20agrupaciones,una%20peque%C3%B1a%20distancia%20entre%20s%C3%AD.

ALGORITMOS DE CLUSTERING

Simple K-Means

Este algoritmo debe definir el número de clusters que se desean obtener, así se convierte en un algoritmo voraz para particionar. Pasos:

1. Se determina la cantidad de clusters en los que se quiere agrupar la información, en este caso las simulaciones.

2. Se asume de forma aleatoria los centros por cada clusters. Una vez encontrados los primeros centroides el algoritmo hará los tres pasos siguientes:
   - Determina las coordenadas del centroide.
   - Determina la distancia de cada objeto a los centroides.
   - Agrupa los objetos basados en la menor distancia.

3. Finalmente quedarán agrupados por clusters, los grupos de simulaciones según la cantidad de clusters que el investigador definió en el momento de ejecutar el algoritmo.

Fuente: https://www.ecured.cu/Clustering

X-Means

Este algoritmo es una variante mejorada del **K-Means**.

Su ventaja fundamental está en haber solucionado una de las mayores deficiencias presentadas en K-Means, el hecho de tener que seleccionar a priori el número de clusters que se deseen obtener, a **X-Means** se le define un límite inferior **K-min** (número mínimo de clusters) y un límite superior **K-Max** (número máximo de clusters) y este algoritmo es capaz de obtener en ese rango el número óptimo de clusters, dando de esta manera más flexibilidad al usuario.

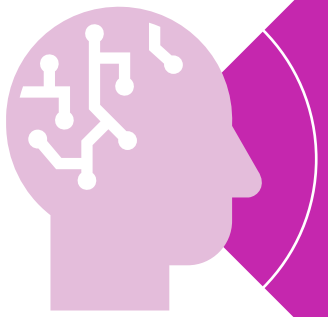
Fuente: https://www.ecured.cu/Clustering

Cobweb

Pertenece a la familia de algoritmos **jerárquicos**. Se caracteriza por la utilización de aprendizaje incremental, esto quiere decir, que realiza las agrupaciones instancia a instancia. Durante la ejecución del algoritmo se forma un árbol (**árbol de clasificación**) donde las hojas representan los segmentos y el nodo raíz engloba por completo el conjunto de datos.

Además, en el algoritmo también hay que tener en cuenta dos parámetros muy importantes:

Acuity: es un parámetro muy necesario, pues la utilidad de categoría está basada en la estimación de la media y la desviación estándar del valor de un atributo para un nodo en particular.



Cut-off: este parámetro es usado para evitar el crecimiento descontrolado de la cantidad de segmentos. Indica el grado de mejoría que se debe producir en la utilidad de categoría

Fuente: https://www.ecured.cu/Clustering

EM

Este algoritmo pertenece a una familia de modelos que se conocen como Finite Mixture Models, los cuales se pueden utilizar para segmentar conjuntos de datos. Está clasificado como un método de particionado y recolocación, o sea, **Clustering Probabilístico**. Se trata de obtener la FDP (Función de Densidad de Probabilidad) desconocida a la que pertenecen el conjunto completo de datos.

El algoritmo EM, procede en dos pasos que se repiten de forma iterativa:

- Expectation: Utiliza los valores de los parámetros, iniciales o proporcionados por el paso Maximization, obteniendo diferentes formas de la FDP buscada.
- Maximization: Obtiene nuevos valores de los parámetros a partir de los datos proporcionados por el paso anterior.

Finalmente se obtendrá un conjunto de clusters que agrupan el conjunto de proyectos original. Cada uno de estos cluster estará definido por los parámetros de una distribución

Fuente: https://www.ecured.cu/Clustering

EJERCICIO

- En este ejercicio se descargan datos de precios para las acciones del S&P 500, calcula sus retornos históricos y volatilidad y luego procede a usar el algoritmo de agrupamiento de K-Means para dividir las acciones en grupos distintos basados en dichos retornos y volatilidades.
- Dividir las acciones en grupos con "características similares" puede ayudar en la construcción de la cartera para asegurar que elegimos un universo de acciones con suficiente diversificación entre ellas.

Lo primero es lo primero, necesitamos recopilar los datos: ejecutemos nuestras importaciones y creemos un script de descarga de datos simple que raspe la web para recopilar los *tickers* de todas las acciones individuales dentro del S&P 500.

```
from pylab import plot,show
from numpy import vstack,array
from numpy.random import rand
import numpy as np
from scipy.cluster.vq import kmeans,vq
import pandas as pd
import pandas_datareader as dr
from math import sqrt
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt


sp500_url = 'https://en.wikipedia.org/wiki/List_of_S%26P_500_companies'

#read in the url and scrape ticker data
data_table = pd.read_html(sp500_url)

tickers = data_table[0][1:][0].tolist()
prices_list = []
for ticker in tickers:
    try:
        prices = dr.DataReader(ticker,'yahoo','01/01/2017')['Adj Close']
        prices = pd.DataFrame(prices)
        prices.columns = [ticker]
        prices_list.append(prices)
    except:
        pass
    prices_df = pd.concat(prices_list,axis=1)

prices_df.sort_index(inplace=True)

prices_df.head()
```

Esto genera algo parecido a esto:

Date	MMM	ABT	ABBV	ACN	ATVI	AYI	AAP	AES	AET	AMG	...	WY	WHR	WLTW
2017-01-03	173.964478	37.982937	59.784031	114.158501	36.417038	233.349442	170.341583	11.042577	120.999954	145.098404	...	29.256657	178.437256	121.45404
2017-01-04	174.228271	38.284462	60.626999	114.432961	37.132656	239.068832	171.739441	10.947872	122.122879	148.701248	...	29.516838	180.866104	122.95201
2017-01-05	173.632263	38.615170	61.086800	112.717552	37.709126	237.813354	171.619629	10.805815	122.644943	146.879929	...	29.825212	181.412354	124.13462
2017-01-06	174.140320	39.665661	61.105961	114.001671	37.679310	236.508057	169.373047	11.194105	122.398682	146.551498	...	29.728846	181.529419	124.94274
2017-01-09	173.202347	39.626755	61.508286	112.727356	37.470589	201.783142	169.273209	10.919462	121.334862	142.749603	...	29.844486	177.149704	124.18390

5 rows × 430 columns

Nuestra primera decisión es elegir en cuántos grupos queremos separar los datos. En lugar de tomar una decisión arbitraria, podemos usar una “curva de codo” para resaltar la relación entre cuántos conglomerados elegimos y la suma de errores al cuadrado (SSE) resultante del uso de ese número de conglomerados.
Ejecutemos el código para nuestro gráfico de curva de codo.

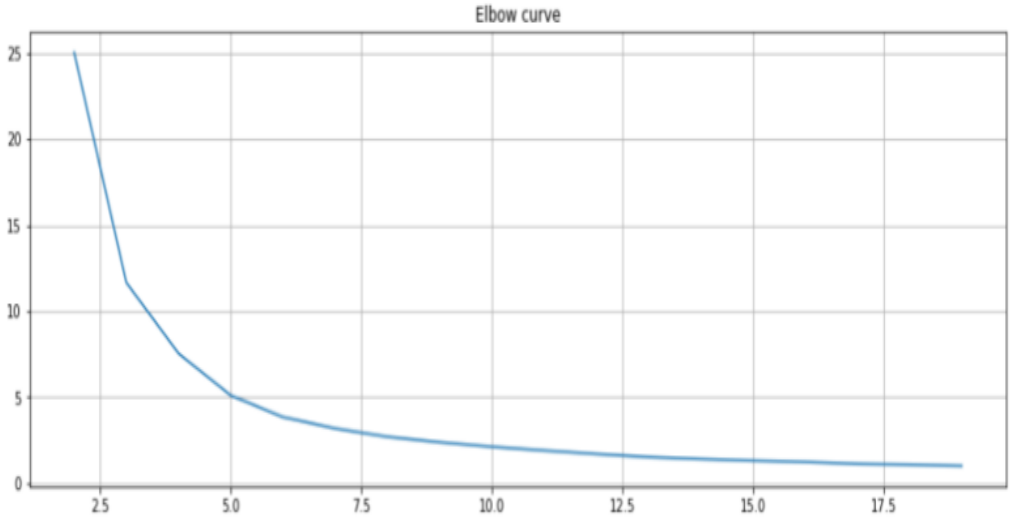
```
#Calculate average annual percentage return and volatilities over a theoretical
one year period
returns = prices_df.pct_change().mean() * 252
returns = pd.DataFrame(returns)
returns.columns = ['Returns']
returns['Volatility'] = prices_df.pct_change().std() * sqrt(252)

#format the data as a numpy array to feed into the K-Means algorithm
data =
np.asarray([np.asarray(returns['Returns']),np.asarray(returns['Volatility'])]).

X = data
distorsions = []
for k in range(2, 20):
    k_means = KMeans(n_clusters=k)
    k_means.fit(X)
    distorsions.append(k_means.inertia_)

fig = plt.figure(figsize=(15, 5))
plt.plot(range(2, 20), distorsions)
plt.grid(True)
plt.title('Elbow curve')
```

El gráfico resultante con los datos anteriores es el siguiente:

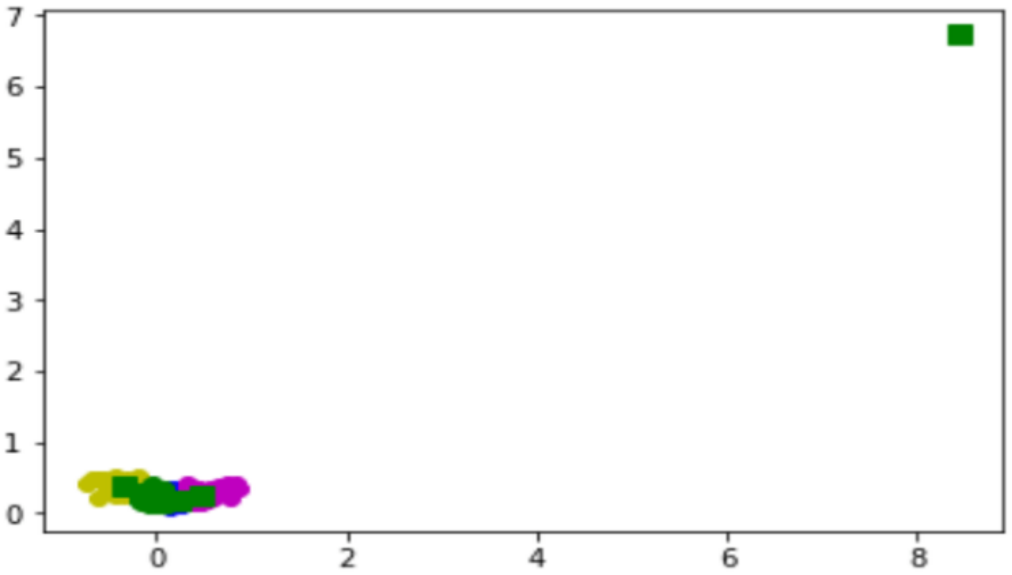


Entonces, podemos ver que una vez que el número de grupos llega a 5 (en el eje inferior), la reducción en el SSE comienza a ralentizarse por cada aumento en el número de grupos. Esto me llevaría a creer que el número óptimo de grupos para este ejercicio se encuentra alrededor de la marca 5, así que usemos 5.

```
# computing K-Means with K = 5 (5 clusters)
centroids,_ = kmeans(data,5)
# assign each sample to a cluster
idx,_ = vq(data,centroids)

# some plotting using numpy's logical indexing
plot(data[idx==0,0],data[idx==0,1],'ob',
     data[idx==1,0],data[idx==1,1],'oy',
     data[idx==2,0],data[idx==2,1],'or',
     data[idx==3,0],data[idx==3,1],'og',
     data[idx==4,0],data[idx==4,1],'om')
plot(centroids[:,0],centroids[:,1],'sg',markersize=8)
show()
```

Esto nos da la salida:



Bien, parece que tenemos un valor atípico en los datos que está sesgando los resultados y dificultando ver realmente lo que está sucediendo con todas las demás acciones. Tomemos la ruta fácil y eliminemos el valor atípico de nuestro conjunto de datos y ejecutemos esto nuevamente.

```
#identify the outlier
print(returns.idxmax())
```

```
#drop the relevant stock from our data
returns.drop('BHF',inplace=True)

#recreate data to feed into the algorithm
data =
np.asarray([np.asarray(returns['Returns']),np.asarray(returns['Volatility'])]).T
```

```
# computing K-Means with K = 5 (5 clusters)
centroids,_ = kmeans(data,5)
# assign each sample to a cluster
idx,_ = vq(data,centroids)

# some plotting using numpy's logical indexing
plot(data[idx==0,0],data[idx==0,1],'ob',
     data[idx==1,0],data[idx==1,1],'oy',
     data[idx==2,0],data[idx==2,1],'or',
     data[idx==3,0],data[idx==3,1],'og',
     data[idx==4,0],data[idx==4,1],'om')
plot(centroids[:,0],centroids[:,1],'sg',markersize=8)
show()
```

Finalmente, esto nos da una representación visual mucho más clara de los grupos de la siguiente manera: