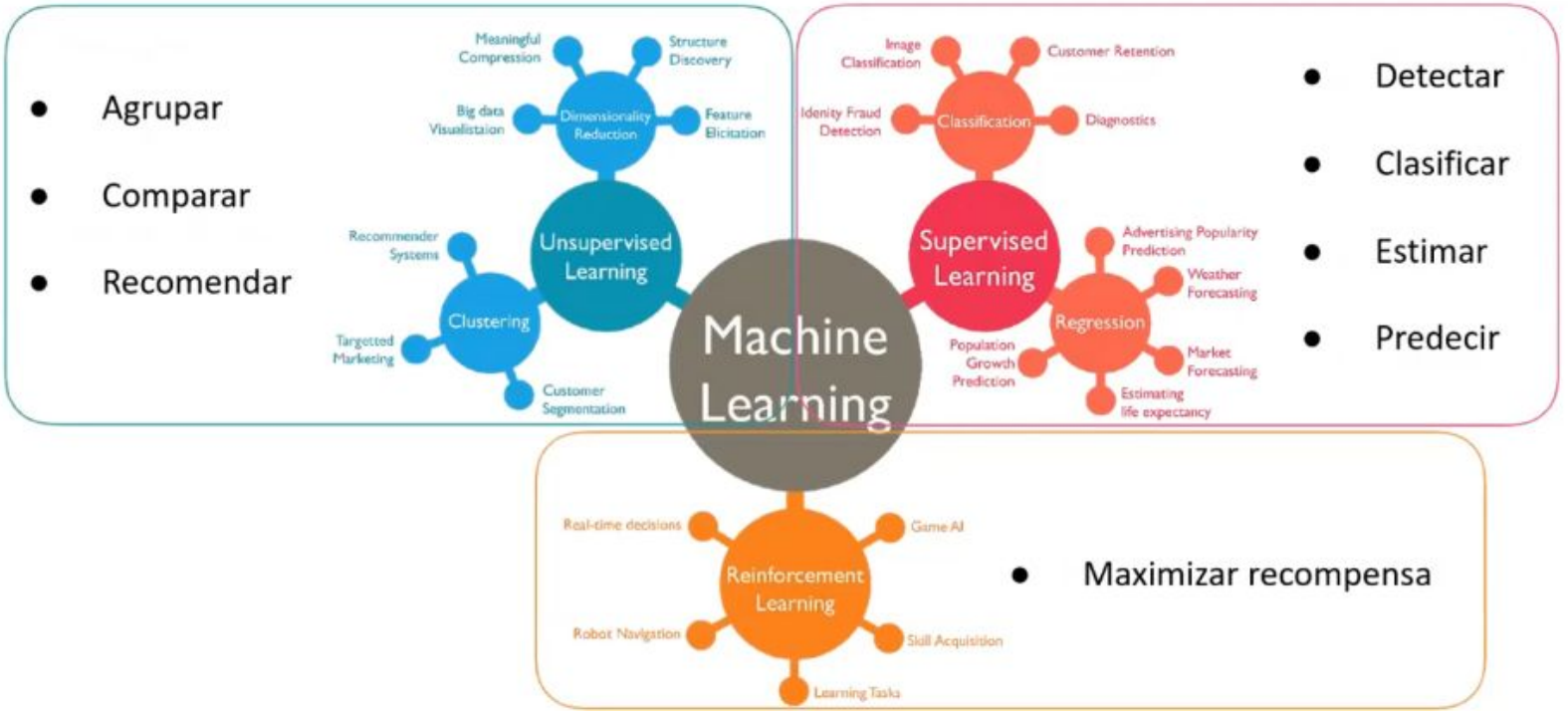
Aprendizaje no Supervisado

Una introducción…

Objetivos

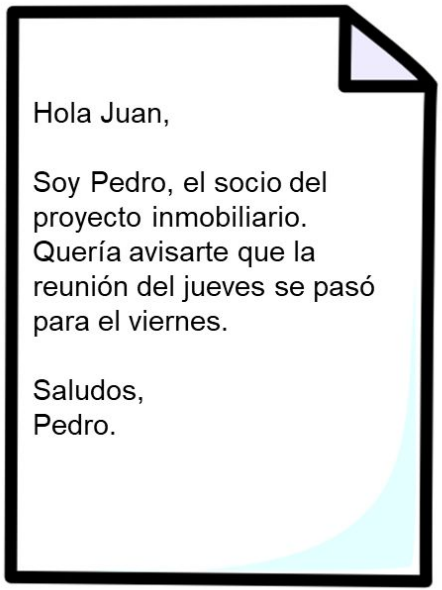
- Entender los conceptos básicos del aprendizaje no supervisado.
- Conocer cómo funciona un modelo de aprendizaje no supervisado.
- Comprender cómo se evalúa el desempeño de un modelo de aprendizaje no supervisado.
- Comprender ejemplos de aplicaciones del aprendizaje no supervisado.

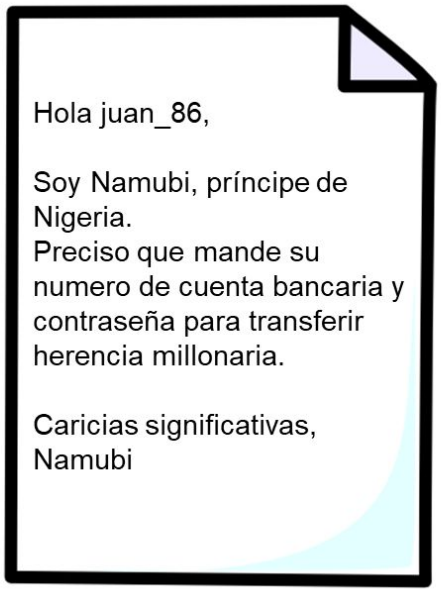
Machine Learning y tipos de aprendizaje



Volvamos al ejemplo del correo Spam

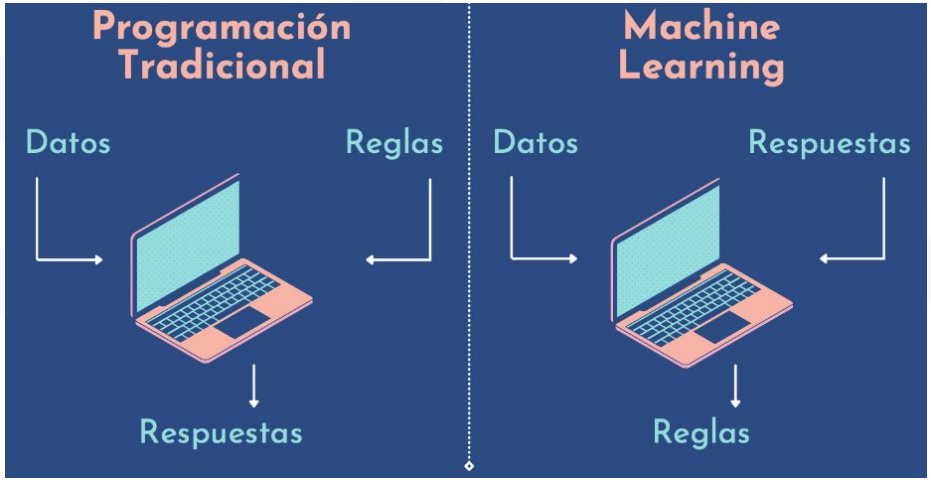
Esta vez con otro enfoque…





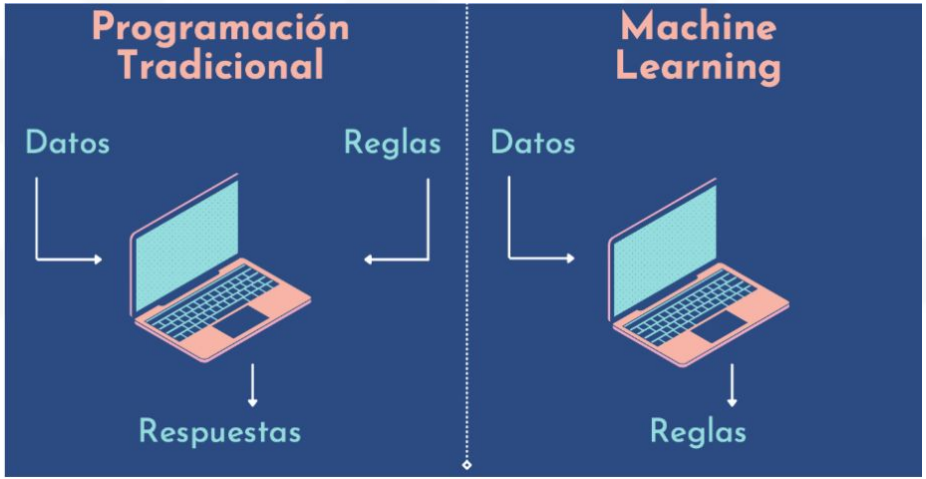
Definición de aprendizaje no supervisado

Llamamos Aprendizaje no Supervisado a los métodos para trabajar con datos (instancias) que no tienen asociados una etiqueta (una clase o un valor).



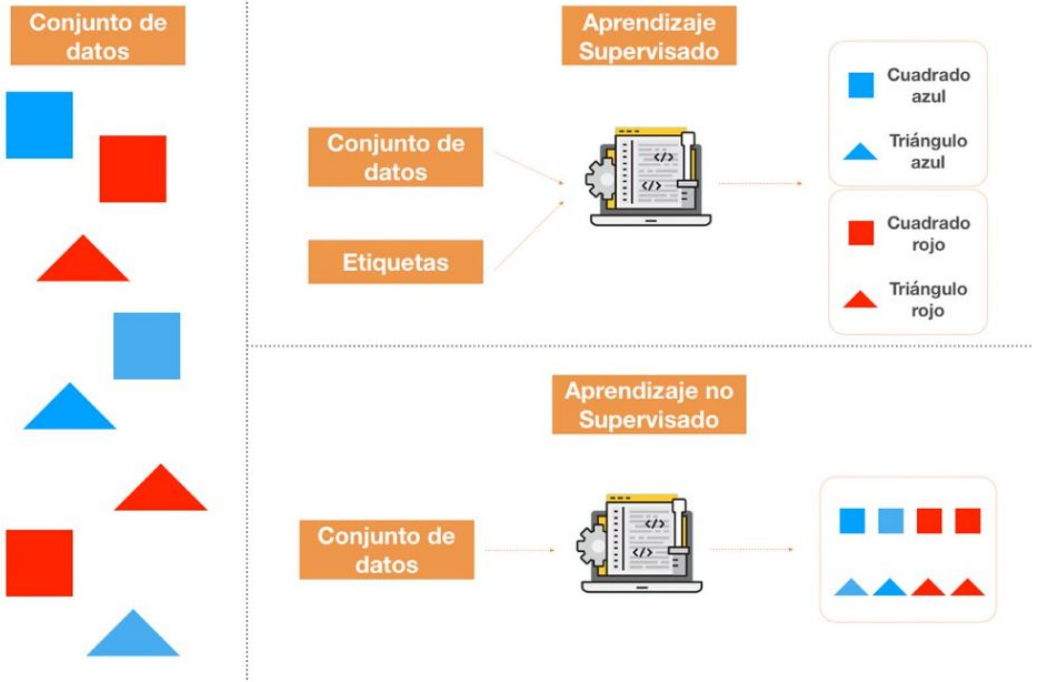
Definición de aprendizaje no supervisado

Llamamos Aprendizaje no Supervisado a los métodos para trabajar con datos (instancias) que no tienen asociados una etiqueta (una clase o un valor).



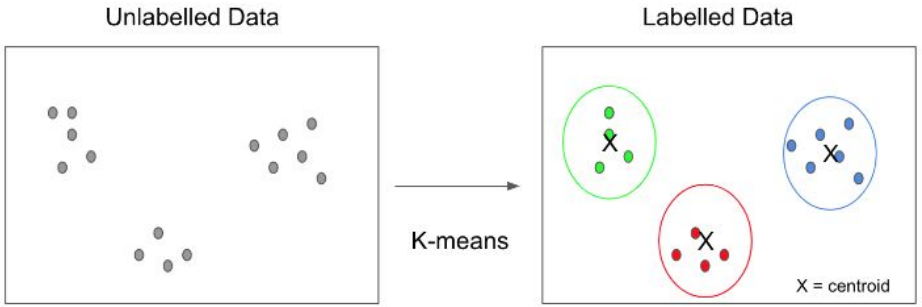
Supervisado vs no-Supervisado

A diferencia del **Aprendizaje Supervisado**, el objetivo ya no pasa por predecir la etiqueta, sino por encontrar **patrones en el set de datos**.

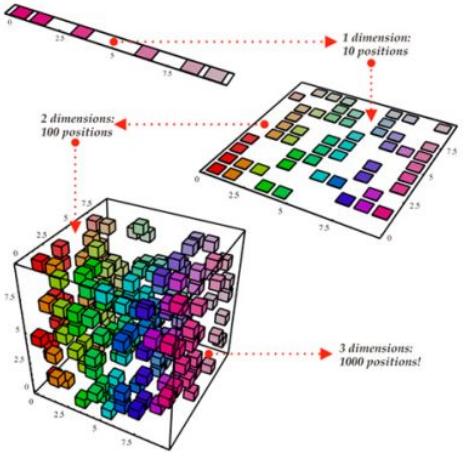


Tipos de aprendizaje no supervisado

Agrupamiento



Reducción de la dimensionalidad



Componentes del Aprendizaje no Supervisado

Ejemplo: En un modelo de segmentación de clientes de una entidad financiera.

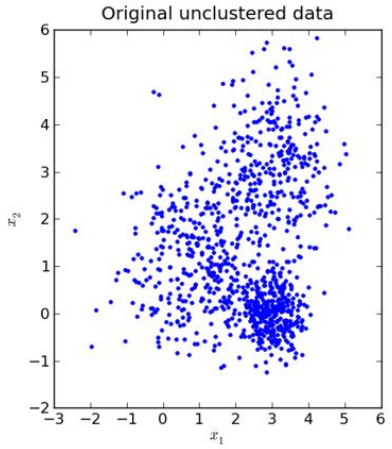
Ejemplos: Algoritmos de agrupación (K-means, DB-Scan) y reducción de la dimensionalidad.

Agrupamiento

Agrupamiento (clustering)

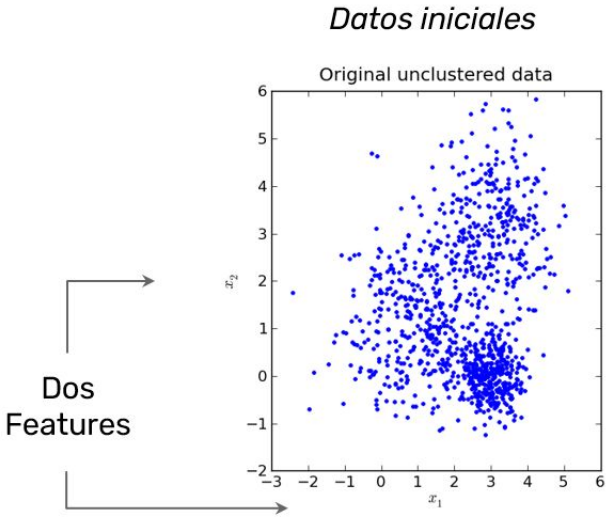
Dado un conjunto de datos, el objetivo del algoritmo es encontrar grupos (clusters) en los cuales las instancias pertenecientes a cada grupo sean parecidas (estén “cerca”).

Datos iniciales



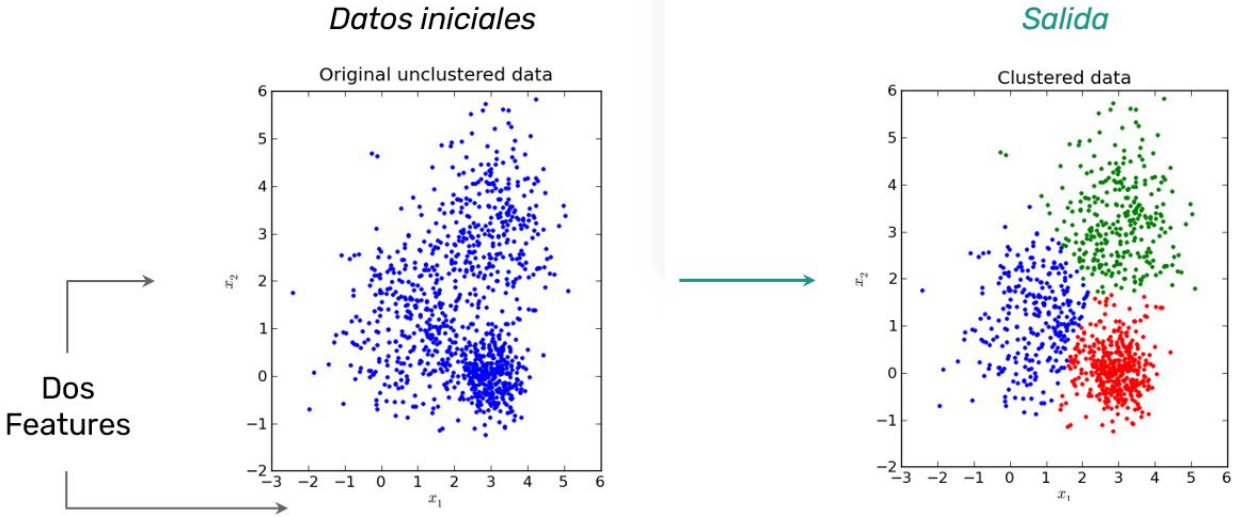
Agrupamiento (clustering)

Dado un conjunto de datos, el objetivo del algoritmo es encontrar grupos (clusters) en los cuales las instancias pertenecientes a cada grupo sean parecidas (estén “cerca”).



Agrupamiento (clustering)

Dado un conjunto de datos, el objetivo del algoritmo es encontrar grupos (clusters) en los cuales las instancias pertenecientes a cada grupo sean parecidas (estén “cerca”).



Agrupamiento (clustering)

¿Para qué sirve?

Encontrar grupos en los datos puede ayudar en problemas de:

- Investigación de mercado
- Sistemas de recomendación
- Medicina
- Biología (genética y especies)
- Muchísimas mas cosas

Agrupamiento (clustering)

¿Para qué sirve?

Encontrar grupos en los datos puede ayudar en problemas de:

- Investigación de mercado
- Sistemas de recomendación
- Medicina
- Biología (genética y especies)
- Muchísimas mas cosas

¿Cómo se hace?

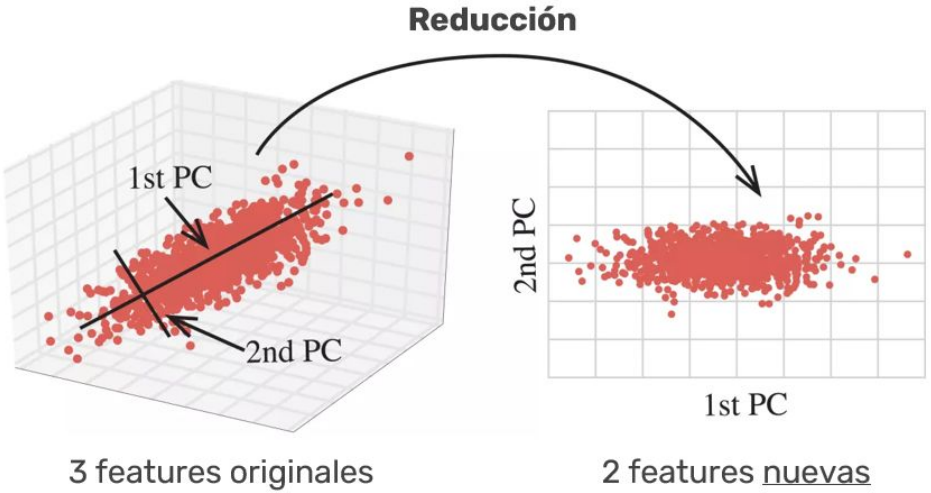
Algunos de los algoritmos para hacer clustering son:

- K-means
- DBSCAN
- Hierarchical Clustering (aglomerativo)
- Fuzzy C-Means (como K-means pero permite overlap)
- GMM: Gaussian Mixture Models (supone distribucion gaussiana)

Reducción de la dimensionalidad

Reducción de la dimensionalidad

Dado un conjunto de datos con “m” features, el objetivo del algoritmo es reducir la dimensionalidad del conjunto de datos a “k” features (k < m), pero reteniendo la mayor cantidad de información posible (varianza explicada).



Reducción de la dimensionalidad

¿Para qué sirve?

Reducir la cantidad de features en un dataset puede servir para:

- Reducir el input en un modelo de regresión o clasificación
- Compresión de archivos
- Visualización
- Detectar features relevantes en datasets
- Muchísimas mas cosas

Reducción de la dimensionalidad

¿Para qué sirve?

Reducir la cantidad de features en un dataset puede servir para:

- Reducir el input en un modelo de regresión o clasificación
- Compresión de archivos
- Visualización
- Detectar features relevantes en datasets
- Muchísimas mas cosas

¿Cómo se hace?

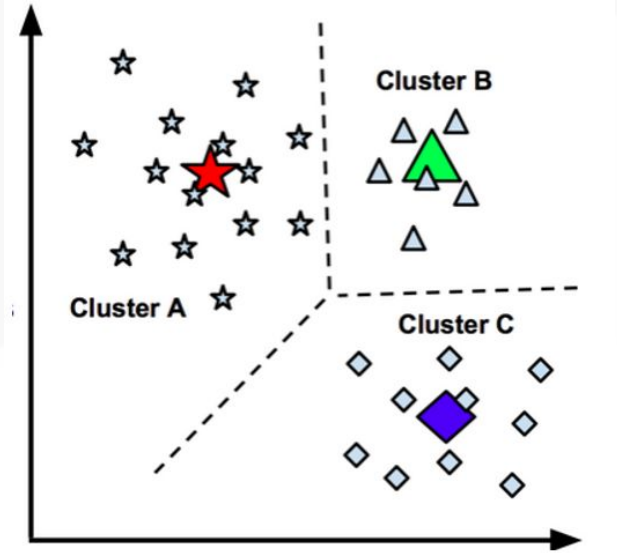
Algunos de los métodos de reducción de dimensionalidad son:

- PCA: Principal Component Analysis (usa SVD)
- MDS: Multidimensional scaling
- t-SNE: t-distributed Stochastic Neighbor Embedding
- Auto-Encoders (Se hace con Redes Neuronales)
- LDA: Linear Discriminant Analysis (si hay etiquetas de clases)

Agrupamiento - Kmeans

K-means

Objetivo: separar los datos en “k” grupos (clusters) ubicando las instancias que estén dentro de una región cercana, dentro de un mismo grupo.



Idea: encontrar un número **k** de centros (centroids), uno por cada cluster, de manera tal que la distancia entre los centros y los datos más cercanos sea la mínima posible.

Luego cada instancia se identifica en el grupo del centroide más cercano.

Prof. Diego Mosquera

K-means

¿Cómo se hace?

K-means sim

Se utiliza un algoritmo iterativo hasta llegar al resultado (convergencia)

1.- Inicializar los “k” centroides
La ubicación inicial puede ser aleatoria o con algún criterio

2.- Encontrar el centroide más cercano
Se asigna a cada instancia al centroide más cercano (el significado de “cercano” puede cambiar, es un hiperparámetro)

3.- Actualizar los centroides
La nueva posición del centroide es el promedio de las posiciones de las instancias en ese cluster (de acá viene el means).

4.- Repetir pasos 2 y 3
Se repiten los updates hasta que la posición del centroide ya no varíe

Ejemplo

Dataset sintético

Librerías Python a utilizar:

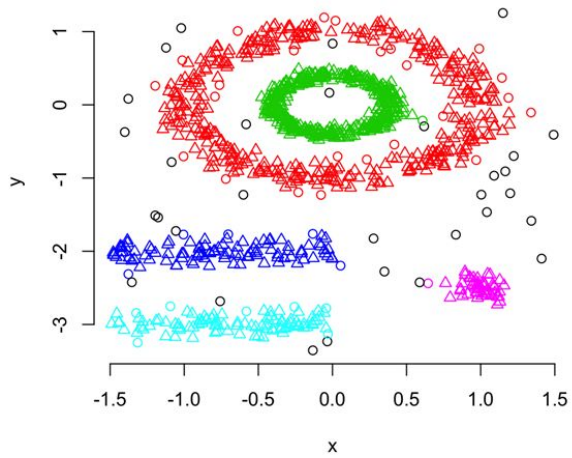
- Sklearn (documentación)

Agrupamiento - DBSCAN

DBSCAN

Density-Based Spatial Clustering of Applications with Noise

Objetivo: Identificar un número arbitrario de clusters. Los clusters estarán definidos por densidad de puntos. Puede haber puntos que no pertenezcan a ningún cluster (noise).



Idea: recorrer todo el dataset e ir identificando las zonas de puntos densamente pobladas como pertenecientes a un mismo cluster.

Los puntos aislados serán reconocidos como ruido.

K-means

¿Cómo se hace?

K-means sim

Se utiliza un algoritmo iterativo hasta llegar al resultado (convergencia)

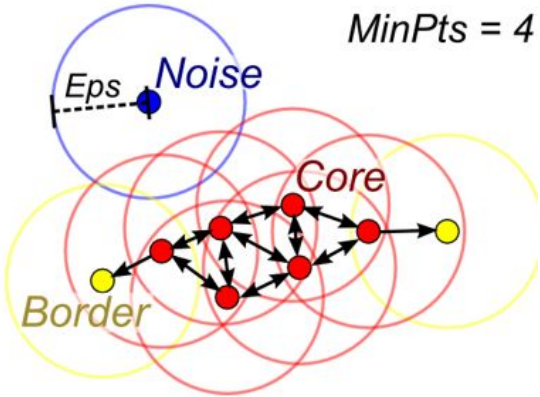
1.- Definir una distancia epsilon (hiperparámetro) como la vecindad de un punto
Se elige un número de puntos mínimos de para considerar un cluster minPoints (hiperparámetro).

2.- Realizar el siguiente proceso sobre todos los puntos del dataset:

2.1.- Se toma un punto no visitado random. Se identifica si el punto es un 'core', es decir, si tiene minPoints en su vecindario. Si no tiene, se lo llama 'noise'. Este punto se marca como visitado.

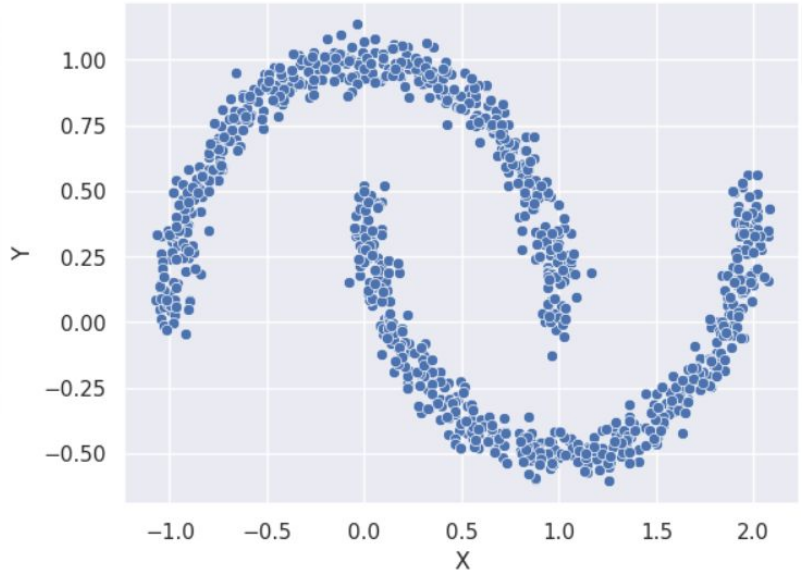
2.2.- Si es un core, se le asigna un nuevo cluster y todos los puntos de su vecindario se consideran dentro de su cluster**. Si alguno de estos puntos también son cores,** este proceso se repite**. A los puntos asignados a un cluster que no son core,** se los llama 'border'. **Todos se marcan como visitados**.

2.3 Este proceso se repite hasta que todos los puntos hayan sido visitados.



Ejemplo

Dataset sintético



Librerías Python a utilizar:

- Sklearn (documentación)

K-Means vs DBSCAN

K-Means

- Rápido, muy mucho. O(n).
- No tiene parámetros
- Fácil asignar nuevas instancias

- Hay que definir el número de clusters
- Solo funciona bien con clusters tipo esferas
- Sensible a outliers (afectan el promedio)

DBSCAN

- No hay que elegir el número de clusters
- Detecta cualquier forma de clusters
- Determina automáticamente datos outliers

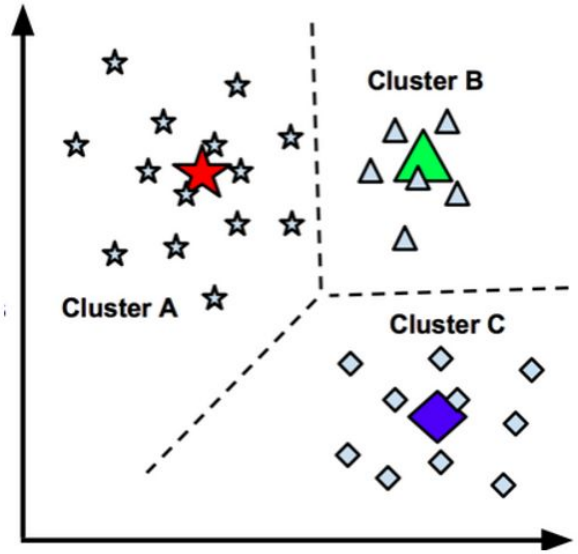
- Hay que elegir bien los parámetros
- No anda bien si hay clusters de diferentes densidades
- Es computacionalmente más costoso (tarda más)

Métricas de calidad para clustering

Métricas de calidad para Clustering

Lo primero que tenemos que saber, es que al no ser un método supervisado **no hay una métrica objetiva** como el accuracy, curva ROC, precisión, sensibilidad, R2, R2 ajustado o MSE, por ejemplo.

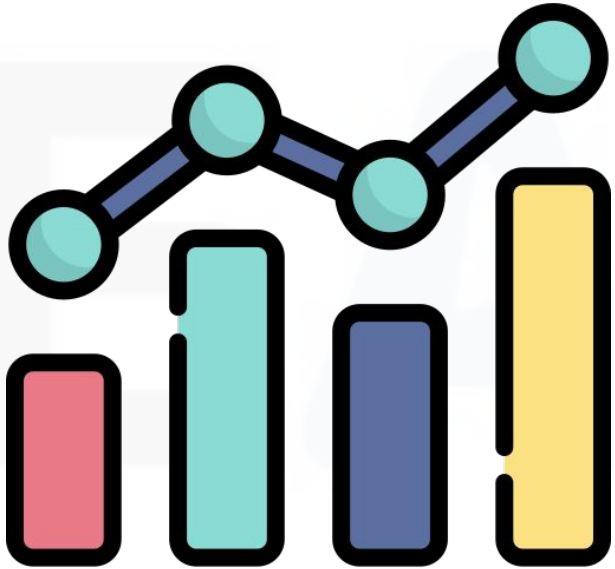
Esto significa, que el data scientist es el principal "*juez*" de un algoritmo de Clustering y dicho juicio se va a encontrar claramente condicionado por el problema a resolver.



Métricas de calidad para Clustering

Existen muchas métricas para evaluar la calidad de los cluster:

1. **Método del codo**
2. **Score de Silhouette**
3. Índice de Rand
4. Índice de Rand Ajustado
5. Criterio de Información Mutua
6. Índice de Calinski-Harabasz
7. Índice de Davies-Bouldin



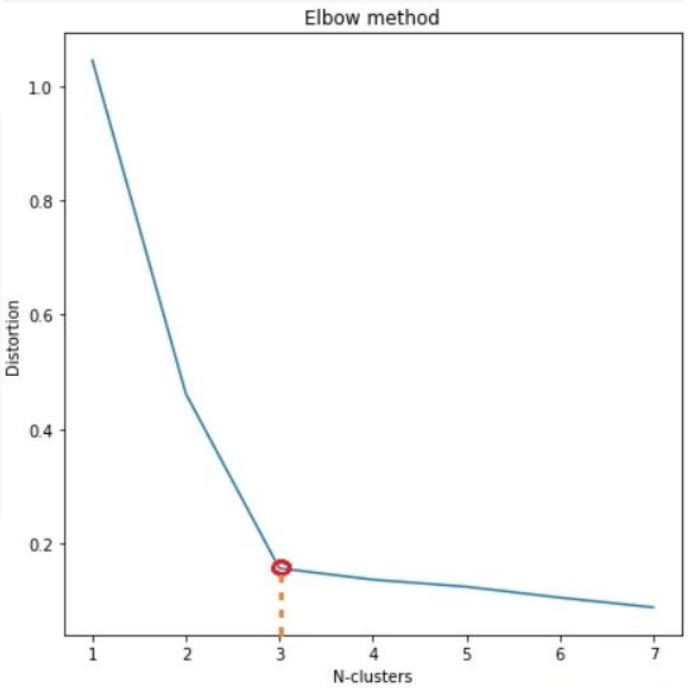
Analizaremos las más comunes y versátiles.

Métricas de calidad para Clustering

Método del codo

Definición:

Este método utiliza la distancia media de las observaciones a su centroide (inercia o distorsión), buscando minimizar la suma de cuadrados dentro del clúster.



Ejemplo

Dataset sintético

Usando make_blobs

Librerías Python a utilizar:

- Sklearn (documentación)

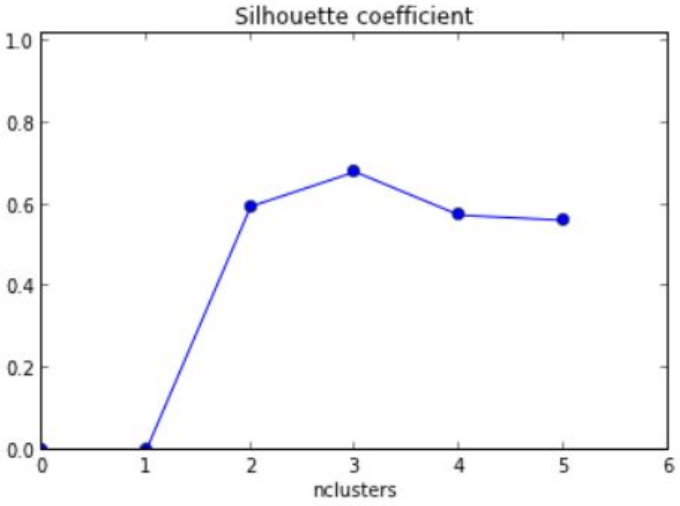
Métricas de calidad para Clustering

Silhouette Score

Definición:

El Silhouette Score proporciona una medida cuantitativa de la calidad de una partición de datos en clusters y puede ayudar a evaluar y comparar diferentes resultados de clustering.

Ejemplo: Si tenemos una partición P1 (k = 3) con un silhouette score de 0.95 y tenemos otra partición P2 (k = 4) con un silhouette score de 0.83, entonces la partición P1 es mejor que la partición P2. Por tanto, el número de clústers debe ser tres.



Métricas de calidad para Clustering

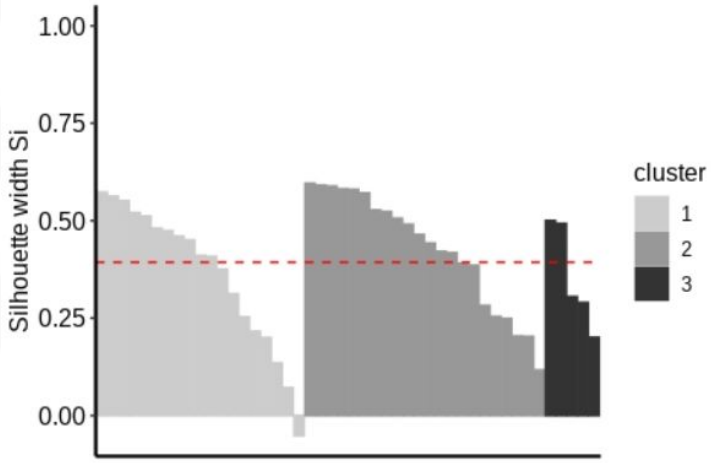
Silhouette Score

Definición:

El silhouette score de una partición es el promedio del valor de silhouette de todos los puntos. Por tanto, para obtener el silhouette score de una partición se debe calcular el valor de silhouette para cada punto.

El valor de silhouette de un punto cuantifica cuán similar es un objeto a su propio grupo (**cohesión**) en comparación con otros grupos cercanos (**separación**).

Evaluación del valor silhouette de un punto
Si el valor Silhouette de un punto es cercano a 1 indica que el objeto está bien agrupado, mientras que un valor cercano a -1 sugiere que el objeto debería estar en un grupo diferente.



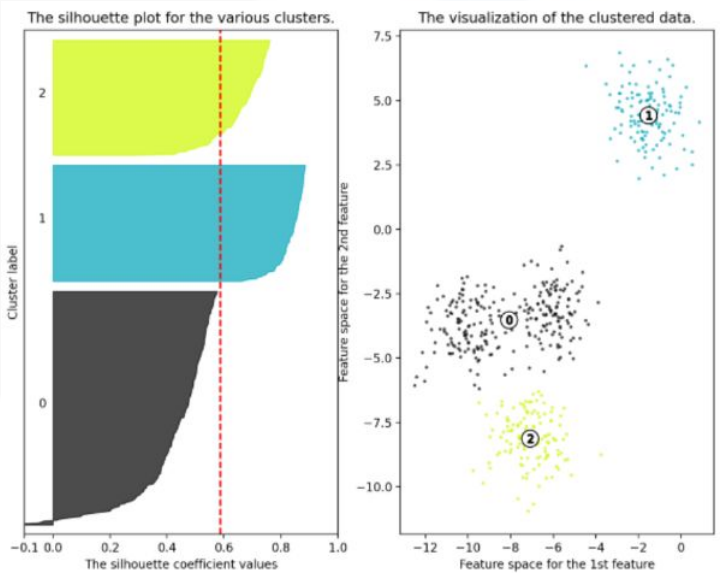
Métricas de calidad para Clustering

Silhouette Score

Evaluación del score de silhouette

El Score de Silhouette se encuentra en el rango de -1 a 1:

- Cerca de 1: Los objetos están bien agrupados.
- Cerca de 0: Los objetos están en o cerca del límite entre dos grupos.
- Cerca de -1: Los objetos pueden estar en grupos incorrectos.



Métricas de calidad para Clustering

Silhouette Score

Proceso de cálculo

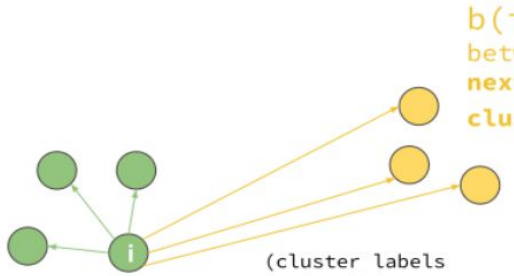
Para calcular el Silhouette Score, se siguen estos pasos para cada punto de datos:

- a(i): La distancia promedio entre el punto de datos i y todos los demás puntos en el mismo cluster. Esto mide la **cohesión**.
- b(i): La distancia promedio entre el punto de datos i y todos los puntos en el cluster más cercano diferente al que pertenece. Esto mide la **separación**.
- Silhouette Score(i): Se calcula para cada punto i utilizando la fórmula de s(i).
- Silhouette Score promedio: Finalmente, calculamos el Silhouette Score promedio para todos los puntos de datos.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Métricas de calidad para Clustering

Silhouette Score



b(i): distance between i and its **next nearest cluster centroid**

(cluster labels assigned from ground-truth classes)

a(i): average distance between i and all of the other points in **its own cluster**

For a single point, i

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

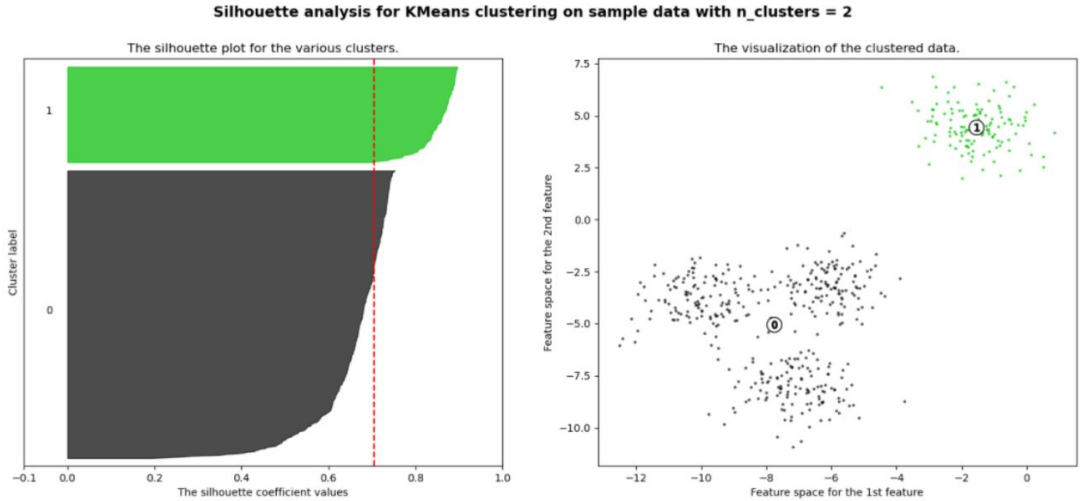
$$-1 \leq s(i) \leq 1$$

Sprawling, overlapped clusters

Tight, well-separated clusters

Métricas de calidad para Clustering

Silhouette Score - Ejemplo

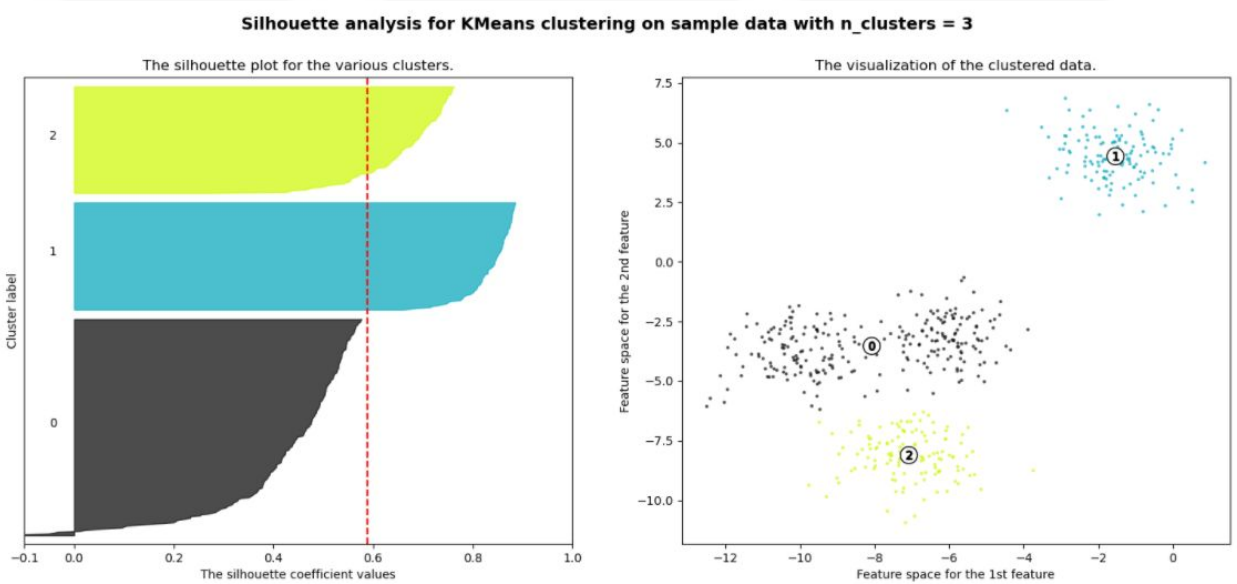


Analicemos el siguiente caso donde tenemos que elegir la cantidad apropiada de grupos. A izquierda el índice de Silhouette para k=2 grupos y a la derecha las observaciones

Prof. Diego Mosquera

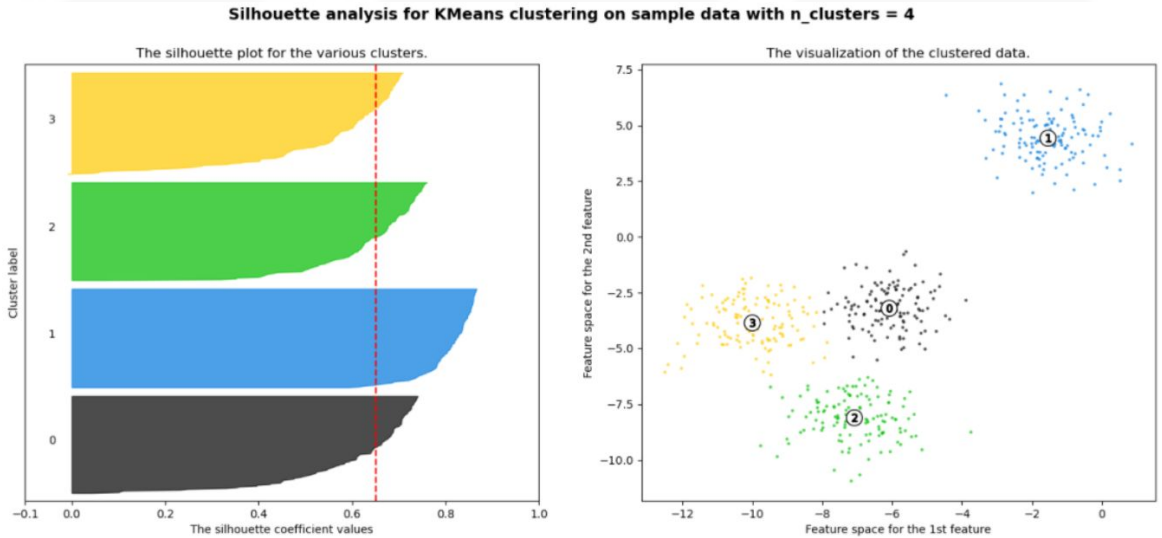
Métricas de calidad para Clustering

Silhouette Score - Ejemplo



Métricas de calidad para Clustering

Silhouette Score - Ejemplo

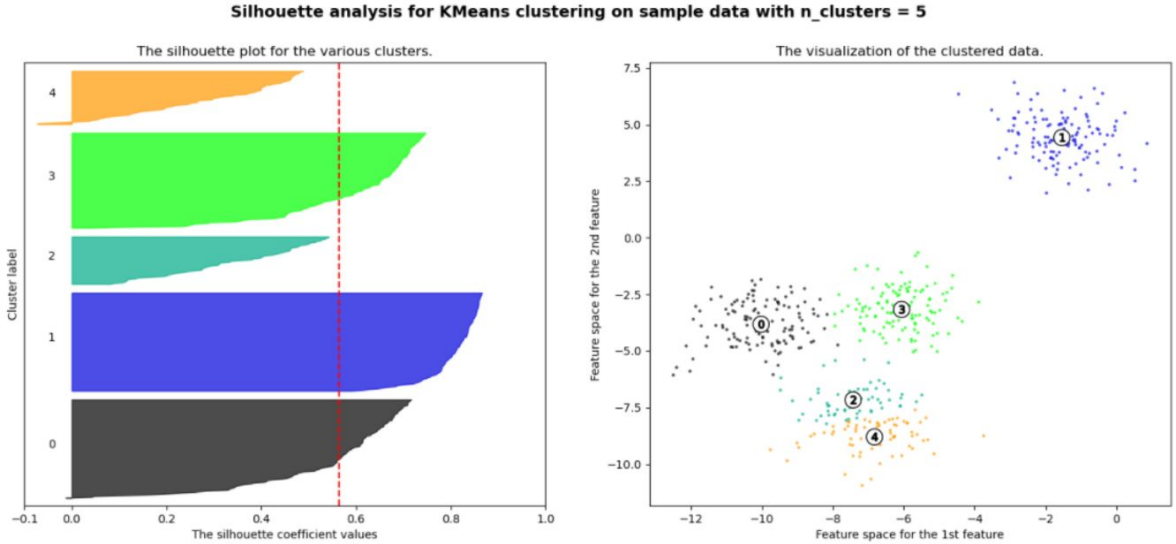


Para el caso de k=4 vamos observando que el valor para el índice de Silhouette se va haciendo más uniforme y obtenemos mejores particiones como se puede observar a la derecha

Prof. Diego Mosquera

Métricas de calidad para Clustering

Silhouette Score - Ejemplo

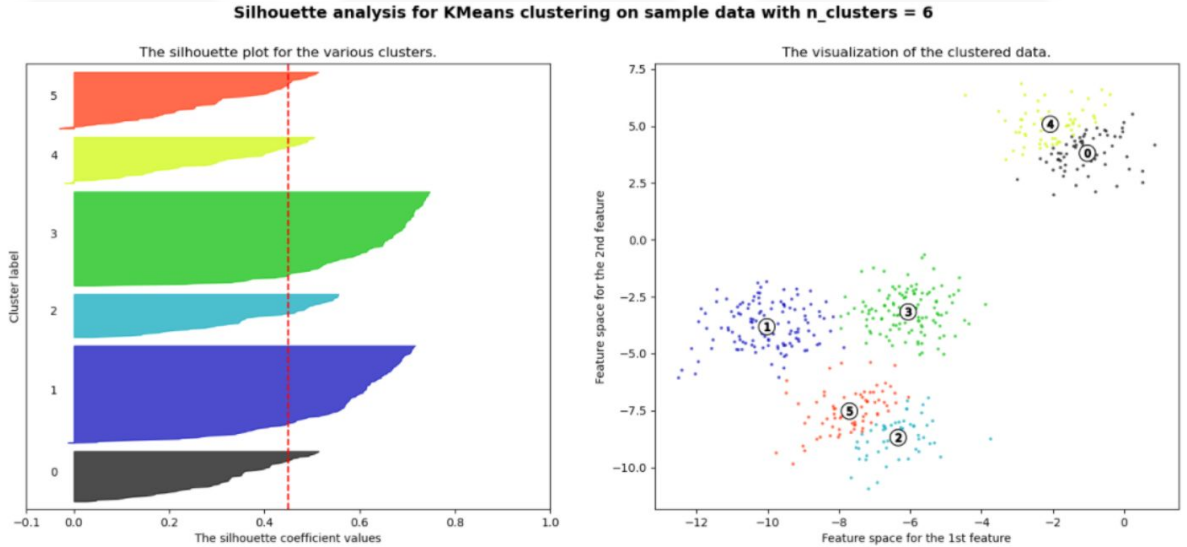


Sin embargo si aumentamos el valor de K aún más como por ejemplo k=5 se observan algunos problemas en la forma de las particiones con grupos desequilibrados

Prof. Diego Mosquera

Métricas de calidad para Clustering

Silhouette Score - Ejemplo

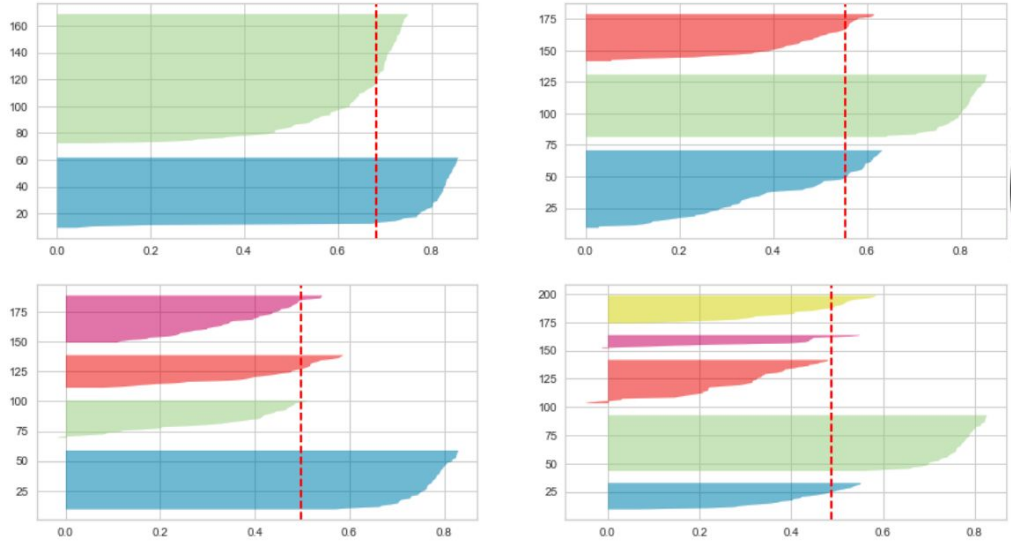


Un caso similar ocurre si seguimos aumentando el número de particiones, en este caso K=6 se observan que se van generando subgrupos que pierden cohesión

Prof. Diego Mosquera

Métricas de calidad para Clustering

Silhouette Score - Ejemplo

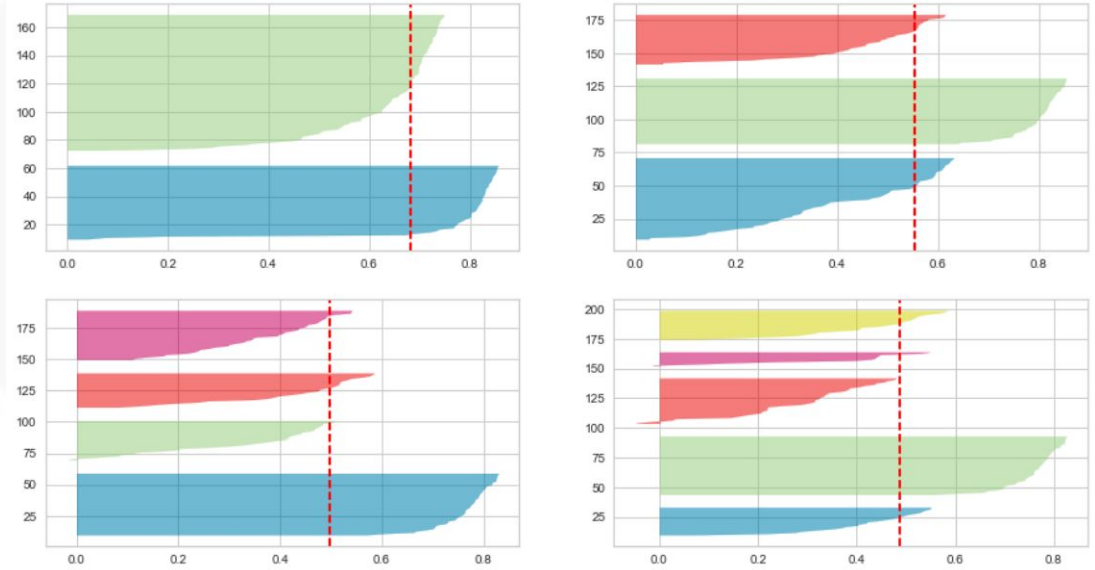


Ejemplo para n=2, 3, 4 y 5 clusters

¿Que valor elegimos?
¿Cómo lo interpretamos?

Métricas de calidad para Clustering

Silhouette Score - Ejemplo

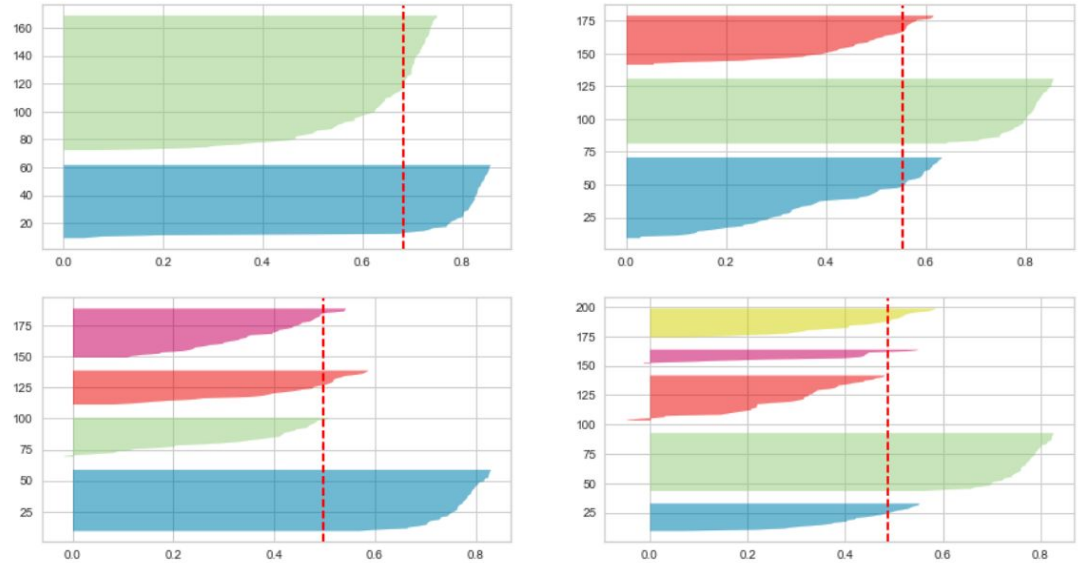


El valor de **n_clusters = 4 y 5 no es muy óptimo** por las siguientes razones:

1. Presencia de grupos con puntuaciones de silueta por debajo del promedio.
2. Amplias fluctuaciones en el tamaño de las parcelas de silueta.

Métricas de calidad para Clustering

Silhouette Score - Ejemplo



1. Entre 2 y 3 parece estar el óptimo. La puntuación de silueta para cada grupo está por encima de la puntuación de silueta promedio.
2. La fluctuación de tamaño es similar.
3. Para el caso n= 3, el grosor es más uniforme que el gráfico con n=2. Por lo tanto, **se puede seleccionar el número óptimo de grupos como 3.**

Ejemplo

Dataset sintético

Usando make_blobs

Librerías Python a utilizar:

- Sklearn (documentación)

Reducción de la Dimensionalidad - PCA

PCA

Principal Component Analysis

Objetivo: Describir un conjunto de datos en términos de nuevas variables (componentes) no correlacionadas. .

	v1	**v2**	**v3**	**v4**
v1	1	-0.1	0.9	0.8
v2	-0.1	1	-0.4	-0.4
v3	0.9	-0.4	1	1
v4	0.8	-0.4	1	1

	v1	**v2**	**v3**	**v4**
v1	1	0	0	0
v2	0	1	0	0
v3	0	0	1	0
v4	0	0	0	1

PCA

Busca encontrar una combinación lineal de las variables originales de modo que los componentes principales sean no correlacionados entre sí.

Procedimiento:

1. Centrado de datos.
2. Cálculo de matriz de covarianza.
3. Descomposición de vectores propios y valores propios.
4. Selección de componentes principales.
5. Combinación lineal (creación de nuevas variables)..

$$PC_i = a_1X_1 + a_2X_2 + \ldots + a_nX_n$$

Donde:

- **PC_i**: es el i-esimo componente principal.
- **X_j**: son las variables originales.
- **a_j**: son los coeficientes que corresponden a los vectores propios

PCA

Ejemplo 1:

Dado el conjunto de datos con las alturas y los pesos de las personas:

Persona	Altura (cm)	Peso (Kg)
A	160	60
B	175	70
C	168	65

PCA

Ejemplo 1:

1.- Centrado de los datos

Altura_est (Persona A) = (160 - 167.67) / 6.36 ≈ -1.21
Peso_est (Persona A) = (60 - 65) / 5 ≈ -1.00

Altura_est (Persona B) = (175 - 167.67) / 6.36 ≈ 1.15
Peso_est (Persona B) = (70 - 65) / 5 ≈ 1.00

Altura_est (Persona C) = (168 - 167.67) / 6.36 ≈ 0.05
Peso_est (Persona C) = (65 - 65) / 5 ≈ 0.00

PCA

Ejemplo 1:

2.- Matriz de covarianza

$$Cov(X,Y) = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{n}$$

Var(Altura_est) = ((-1.21)² + 1.15² + 0.05²) / 3 ≈ 0.93
Var(Peso_est) = ((-1.00)² + 1.00² + 0.00²) / 3 ≈ 0.67

Cov(Altura_est, Peso_est) = ((-1.21 * -1.00) + (1.15 * 1.00) + (0.05 * 0.00)) / 3 ≈ 0.77

Cov(X,Y)	Altura_est	Peso_est
Altura_est	0.93	0.77
Peso_est	0.77	0.67

PCA

Ejemplo 1:

3.- Autovalores y Autovectores

```
import numpy as np

# Matriz de covarianzas
A = np.array([[0.93, 0.77],
              [0.77,  0.67]])

# Calcular autovalores y autovectores
autovalores, autovectores = np.linalg.eig(A)
```

Supongamos que los autovectores resultantes son [0.67, 0.73] y [0.73, -0.67], y los autovalores correspondientes son 1.75 y 0.25.

PCA

Ejemplo 1:

4- Selección de componentes principales

Suponga que estamos reduciendo la dimensionalidad a 1, entonces seleccionamos el autovector correspondiente al autovalor más grande, que es [0.67, 0.73].

PCA

Ejemplo 1:

5- Creación de nuevas variables

$$PC_i = a_1X_1 + a_2X_2 + \ldots + a_nX_n$$

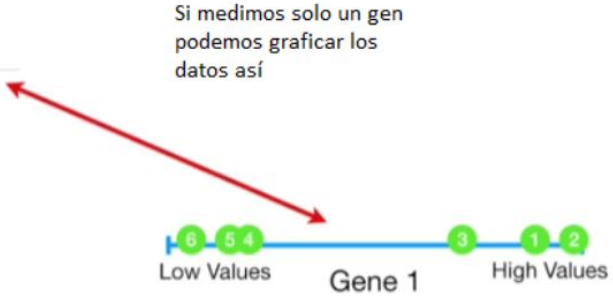
Componente Principal 1 (Persona A) = (0.67 * (-1.21)) + (0.73 * (-1.00)) ≈ -1.62
Componente Principal 1 (Persona B) = (0.67 * 1.15) + (0.73 * 1.00) ≈ 1.63
Componente Principal 1 (Persona C) = (0.67 * 0.05) + (0.73 * 0.00) ≈ 0.03

PCA

Ejemplo 2:

Imaginemos que tenemos el siguiente conjunto de datos que representa dos variables para 6 individuos (ratones). Si queremos representar a los individuos en una dimensión a través de una variable podemos hacer una grafica com la siguiente:

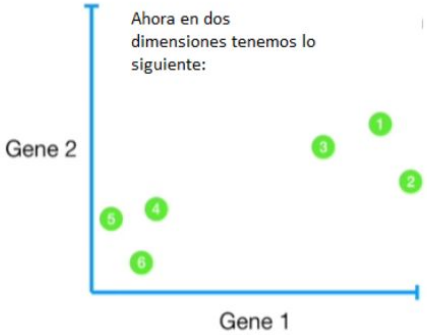
	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1



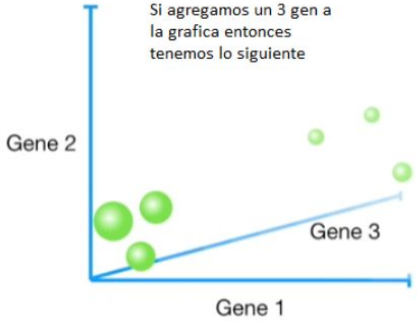
PCA

Ejemplo 2:

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1



	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1
Gene 3	12	9	10	2.5	1.3	2



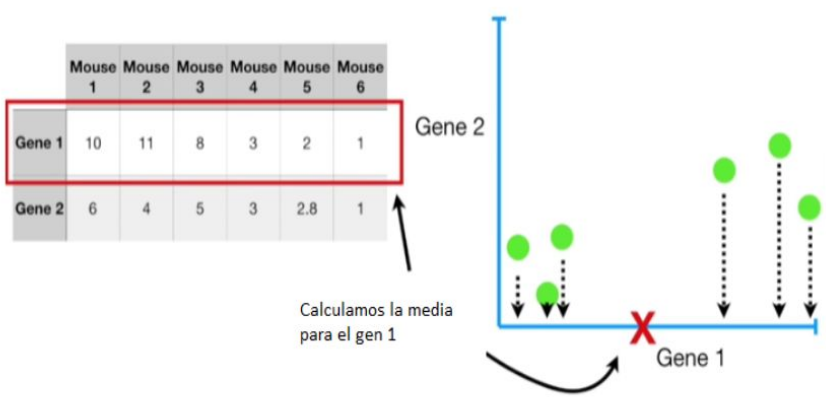
También podemos graficar dos dimensiones como se observa en la figura de arriba generando un diagrama de dispersión

Si agregamos una tercera variable podríamos hacer un gráfico 3D. Aunque si ponemos una cuarta dimensión el análisis ya no es tan sencillo

PCA

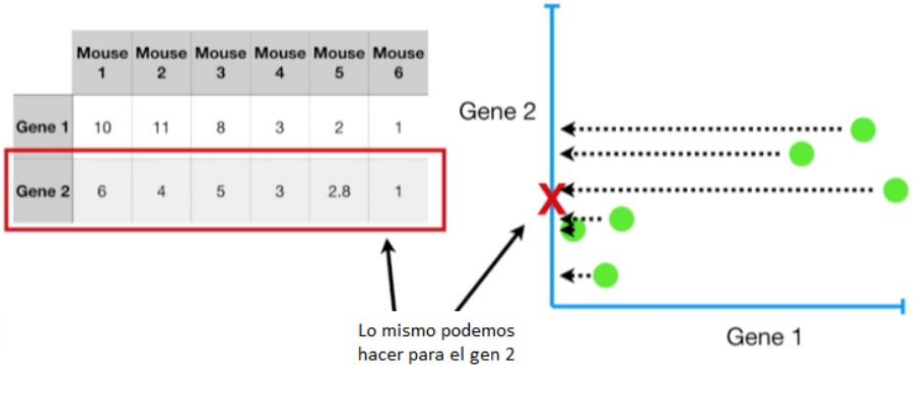
Ejemplo 2:

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1



Si calculamos la media de la primera variable tenemos lo siguiente

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1

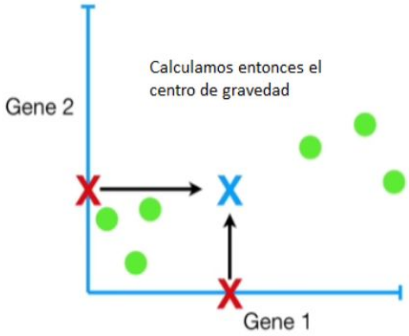


De igual forma si calculamos la media de la segunda variable tenemos lo siguiente

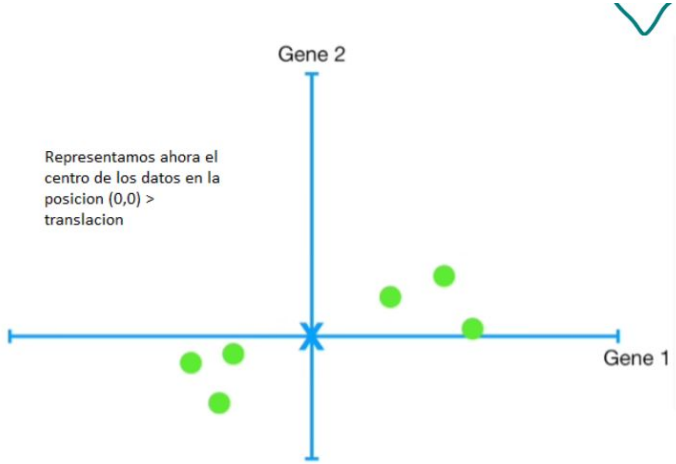
PCA

Ejemplo 2:

	Mouse 1	Mouse 2	Mouse 3	Mouse 4	Mouse 5	Mouse 6
Gene 1	10	11	8	3	2	1
Gene 2	6	4	5	3	2.8	1



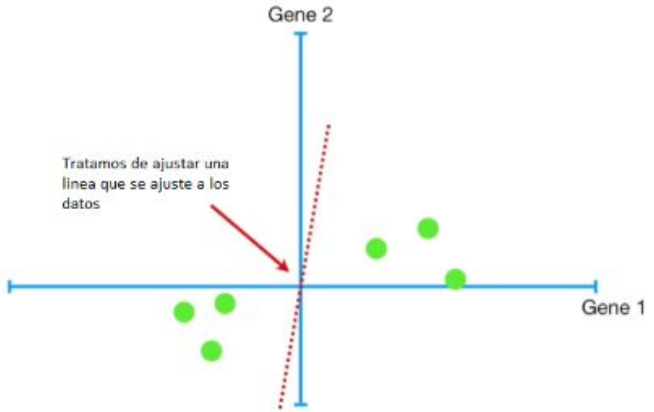
Si juntamos los dos punto tenemos el centro de gravedad o la media global de los individuos



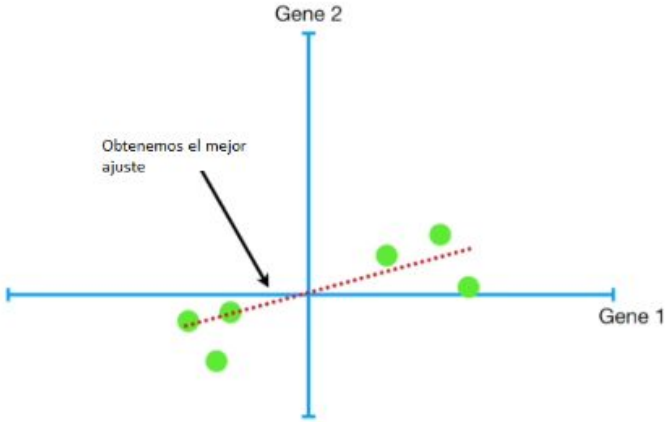
Si hacemos que este punto sea el nuevo centro (0,0) no afecta la posición relativa de los puntos

PCA

Ejemplo 2:



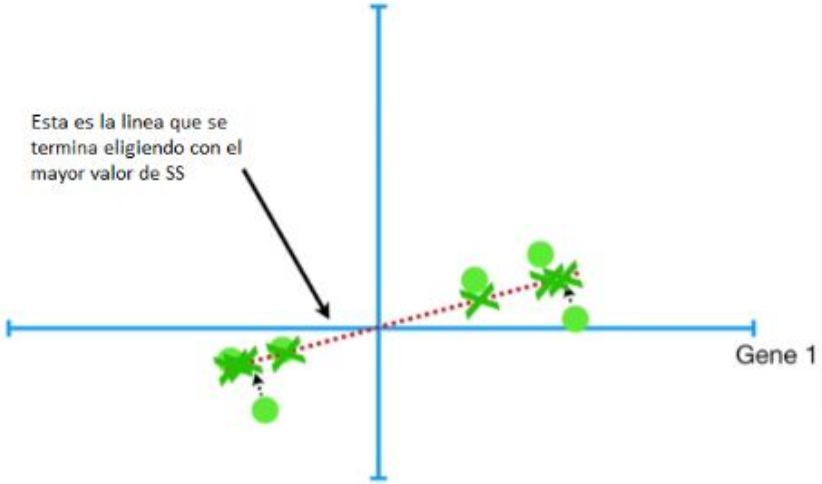
Podemos proceder a ajustar una línea que mejor se ajuste a los datos como en regresión



Y el resultado luego del ajuste es el siguiente

PCA

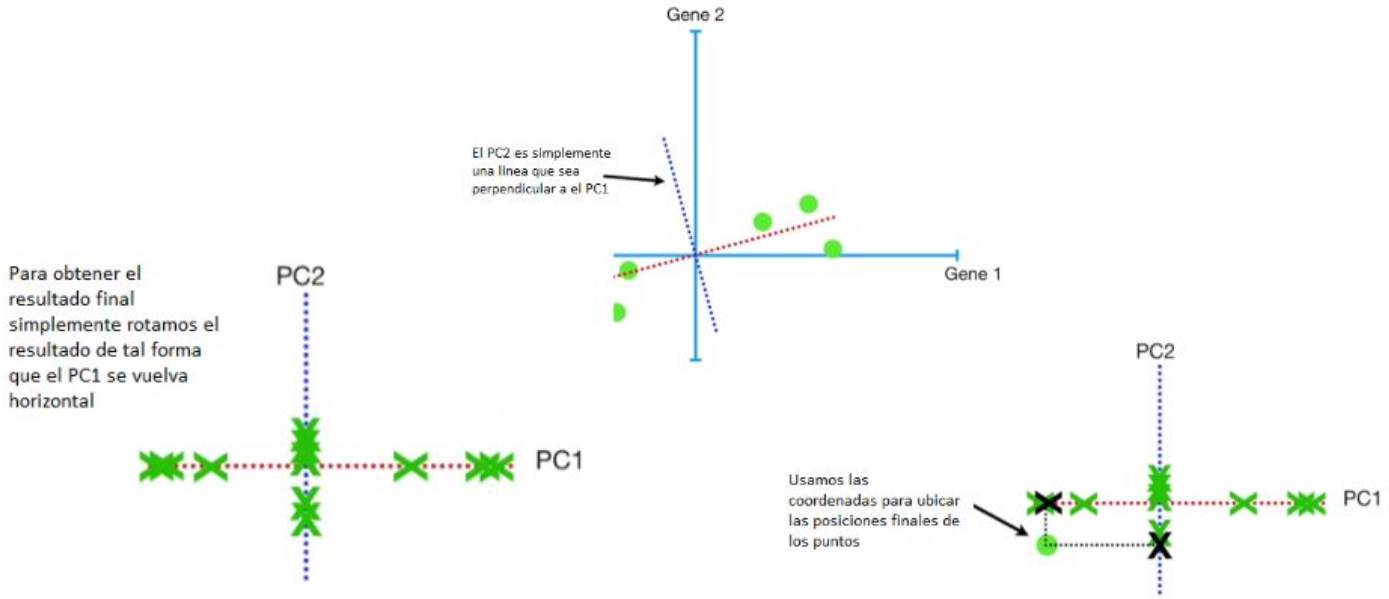
Ejemplo 2:



Nótese que los datos se proyectan sobre la línea de ajuste

PCA

Ejemplo 2:



Ejemplo

Dataset de Iris

Librerías Python a utilizar:

- Sklearn (documentación)

Ejemplo

Dataset de ecommerce

Librerías Python a utilizar:

- Sklearn (documentación)
- yellowbrick (documentación)

Preguntas…