## RICERCA SUL CLUSTERING: Cos'è e perché lo stiamo impiegando nel nostro progetto

**Cos'è:**
*clustering* = raggruppamento
Il clustering non è altro che il raggruppamento di dati in una o più categorie pertinenti, per caratteristica.

**Come può essere utile**:
in medicina: vogliamo raggruppare dei tipi di cellule o dei tipi di tumore.
in matematica: classificare delle funzioni che sono iniettive o suriettive
in informatica: raggruppare dei dati

**Caso base:**
punti su una stessa linea (1 sola dimensione)

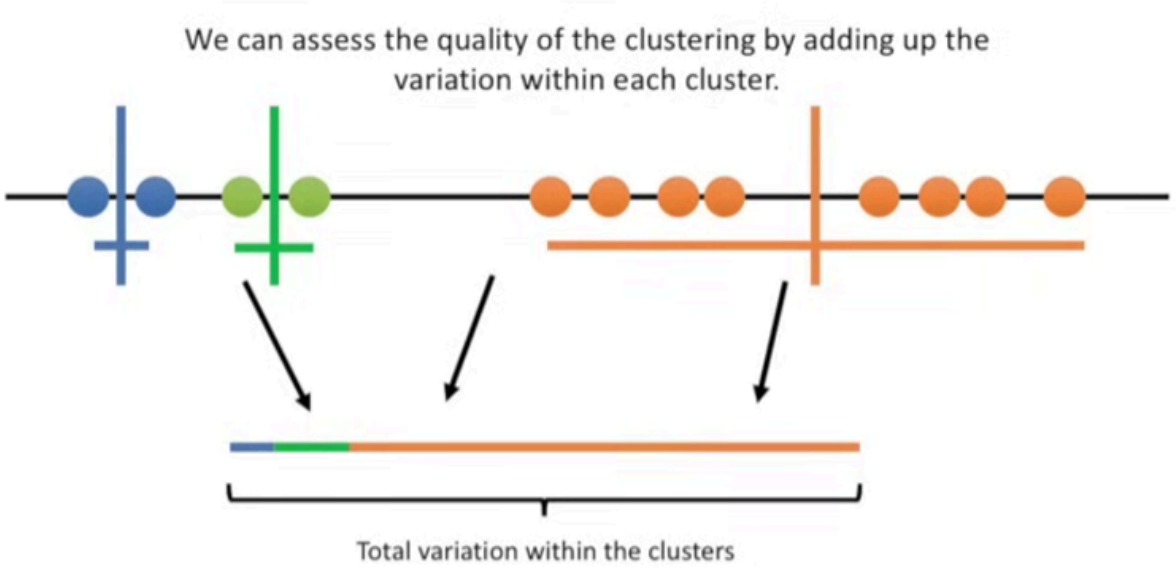
**L'algoritmo :**

1. scegliere K, ovvero il numero dei raggruppamenti

*algoritmo base:*

2. seleziona K punti (dati) distinti per iniziare dello stesso tipo. Essi saranno i raggruppamenti iniziali
   es: numero interazioni dell'attaccante con il server:
   punto 1: A1 attacca 100 volte in un minuto
   punto 2: A2 attacca 2 volte in un minuto
3. Ripetiamo per ogni dato da aggiungere:
   a. aggiungiamo un nuovo dato: A3 attacca 5 volte in un minuto.
   b. calcoliamo la distanza del nuovo dato rispetto al punto medio di ciascun clustering esistente
   c. Inseriamo il dato nel raggruppamento più vicino.

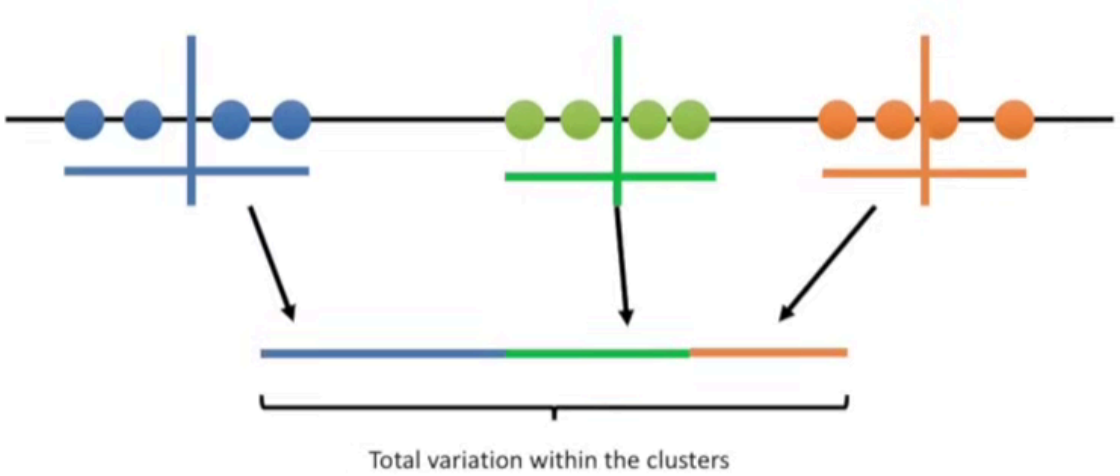
Abbiamo a questo punto inserito tutti i dati, ma il clustering ottenuto è estremamente lontano dall'essere buono.




**La varietà:**
*varietà:* diversità o range di punti dentro un singolo cluster.
Nell'algoritmo base non viene tenuto conto di essa.
Una varietà troppo bassa non ci permette di trovare casi particolari o ci darebbe falsi negativi o positivi.
Peggio ancora, in caso di raggruppamenti multipli, metterebbe molti dati che sono chiaramente in contrasto tra loro nello stesso gruppo.

L'algoritmo diventa quindi:

1. scegliere  K, ovvero il numero dei raggruppamenti
2. ripetizione dell'*algoritmo base* (dalla scelta dei punti iniziali fino al raggruppamento di tutti i dati) finché il clustering ottenuto non cambia più (quiescenza)

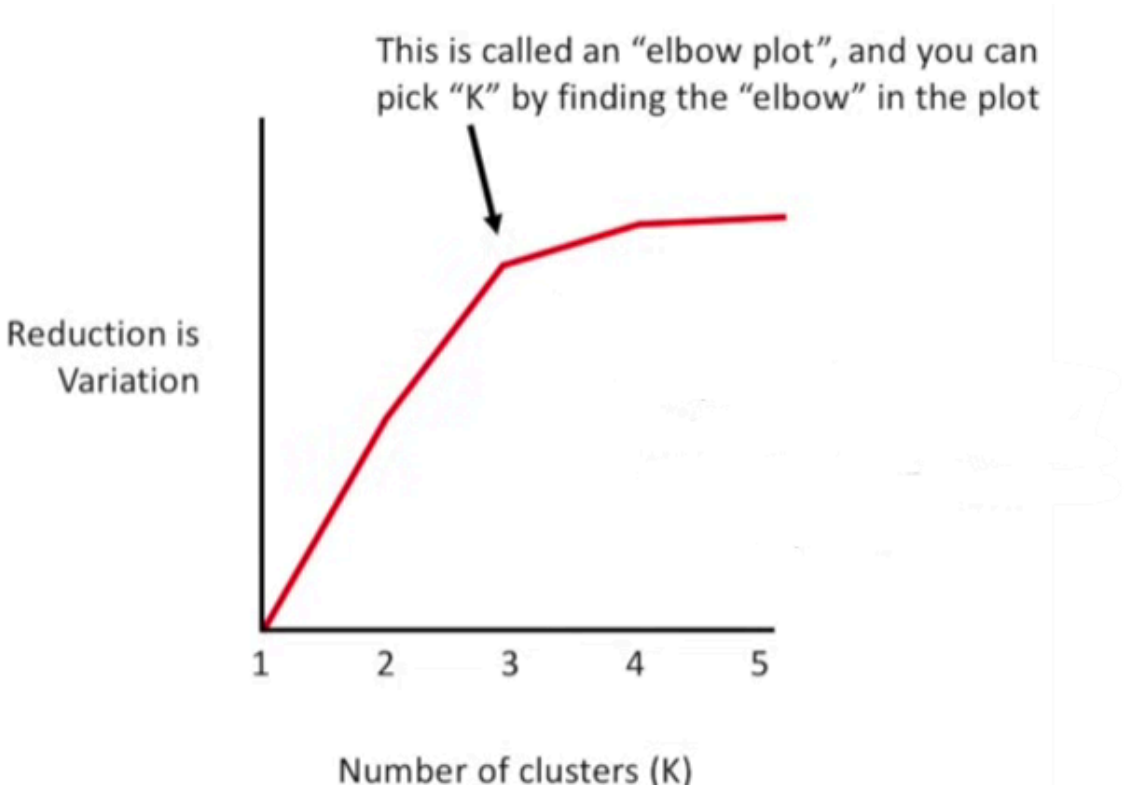



**Il miglior clustering:**
Anche aggiungendo la varietà, non possiamo dire che il clustering ottenuto sia il migliore possibile con certezza matematica.

Qui entra in gioco la scelta di K.
Studiamo pertanto K:
Con K = 1 non possiamo differenziare i dati. Vanno tutti nello stesso gruppo -> è il **caso peggiore**, poiché la **varietà = 0.**

Con K = 2 differenziamo i dati. La varietà aumenta fino a raggiungere un massimo.
…
Aumentando K però notiamo qualcosa: ogni volta che aggiungiamo un'unità a K, ovvero, aggiungiamo un gruppo, la varietà in ciascun gruppo smette di crescere, anzi, diminuisce, fino a raggiungere il valore 0, ovvero, **tornando** al **caso peggiore**.

I matematici hanno quindi deciso di creare un grafico che legasse la varietà e il numero dei gruppi.

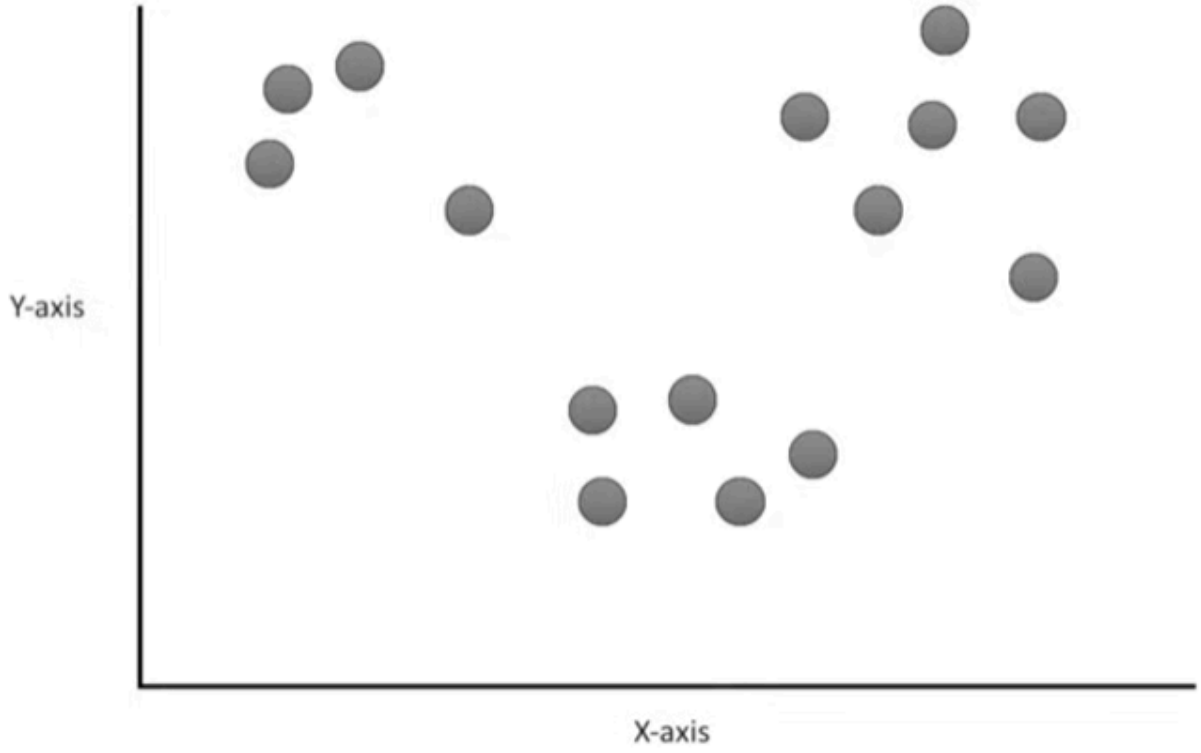



Trovando l'*elbow*, possiamo scegliere il K corretto e ottenere il clustering migliore.

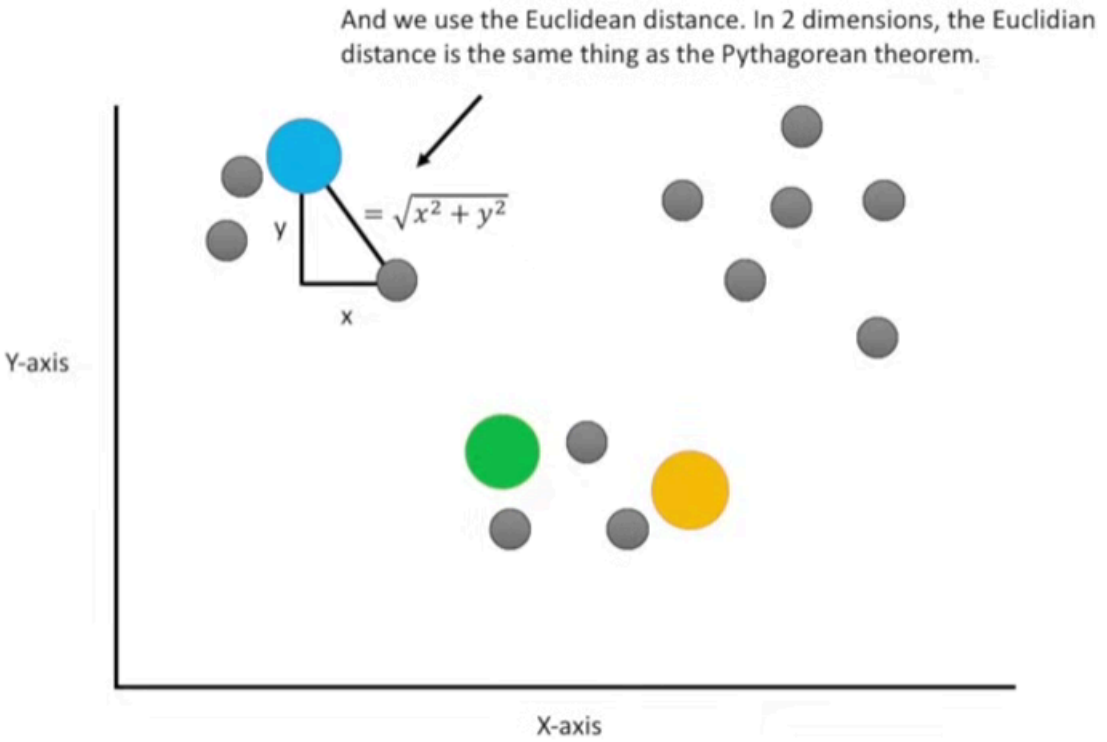
**Clustering alternativo**:
Abbiamo visto finora il clustering di tipo K-means, in cui bisogna scegliere manualmente K.
L'alternativa è il Clustering Gerarchico.

Il Clustering Gerarchico riesce a capire per ciascuna coppia di  elementi, quali sono più simili tra loro.

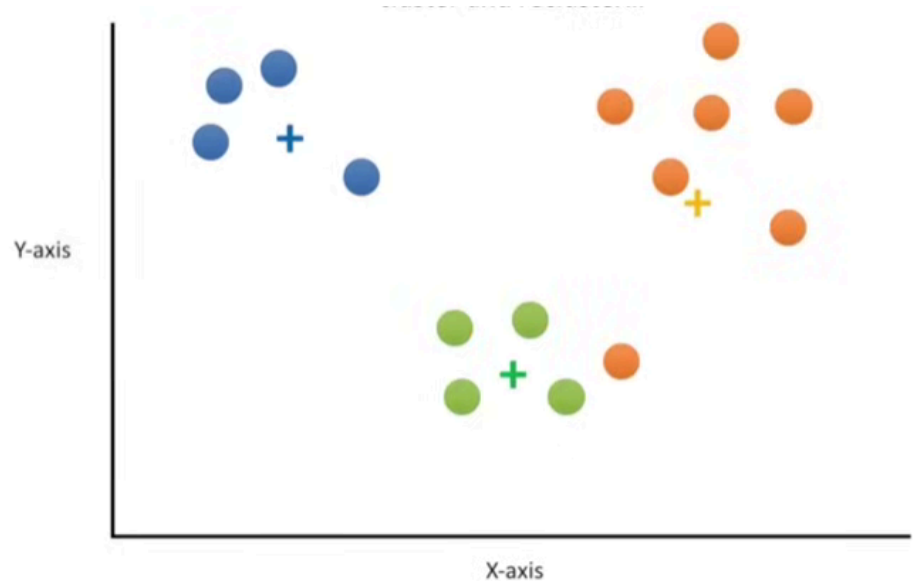
**Aggiungiamo una dimensione:**




Facciamo esattamente come prima, solo che la difficoltà e il peso delle operazioni aumentano.
Prendiamo, per K=3, 3 punti distinti casuali e calcoliamo la distanza euclidea secondo il teorema di Pitagora e assegnamo i punti ai cluster più vicini.




Lo faccio per tutti i valori e alla fine calcoliamo il centro di ciascun cluster e ricalcoliamo.




Sembra facile, ma la macchina computazionale non può saperlo finché non svolge il clustering più volte.

*Appunti presi da un video di Josh Starmer*

**La libreria Python che useremo**: **sklearn.cluster**

è il modulo di scikit-learn che contiene gli algoritmi di apprendimento non supervisionato più frequenti per *raggruppare dati simili senza sapere prima le classi* (clustering)

Inizialmente si hanno :

- **N oggetti** che sono **vettori numerici**.

I '**vettori ML**' sono array o arraylist o liste di valori numerici (*non* stringhe o date o classi).

Il clustering vuole rispondere alla domanda "*quali sessioni si assomigliano di più?"*

Tutti gli algoritmi di sklearn.cluster seguono lo stesso schema:

```
from sklearn.cluster import Algoritmo

model = Algoritmo(parametri...)
labels = model.fit_predict(X)
```

X = matrice m x n  m = ; n = ;
labels[i] = cluster assegnato all'elemento i

**SCALING:**
se non si normalizzano i valori, il più grosso prevarrà sempre sull'altro.
es:
duration = 1000

success_ratio = 0.2

duration >>>> success_ratio !

come risolvere:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

**GLI ALGORITMI PIU' FAMOSI:**

1. KMeans :
   spiegato meglio sopra, ma concettualmente:
   - si scelgono K centroidi
   - si assegna ogni punto al centro più vicino
   - si spostano i centroidi
   - si ripete finché non converge

   Da usare solo quando si **deve** sapere **prima** quanti cluster si possono ottenere.

```
from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=4, random_state=42)
labels = kmeans.fit_predict(X_scaled)
```

2. DBSCAN : trova gruppi + outlier :
   - non si sceglie il numero di cluster a priori
   - osserva i gruppi di punti densi, quelli isolati vengono visti come 'rumore/outlier'

   Si usa quando si devono individuare molti 'casi particolari'.

```
from sklearn.cluster import DBSCAN

dbscan = DBSCAN(eps=1.2, min_samples=5)
labels = dbscan.fit_predict(X_scaled)
```

   ATTENZIONE : se labels = -1 c'è un comportamento anomalo

3. AgglomerativeClustering : clustering gerarchico
   - ogni punto è un cluster
   - unisce i punti più vicini tra loro

- costruisce una gerarchia

```
from sklearn.cluster import AgglomerativeClustering

agg = AgglomerativeClustering(n_clusters=4)
labels = agg.fit_predict(X_scaled)
```

Si usa quando si vogliono visualizzare dendrogrammi o spiegare bene i risultati.

4. MeanShift, SpectralClustering, ecc…