[K-MEANS 2](#_Toc159842851)

[Introduzione 2](#_Toc159842852)

[Applicazioni di K-Means 2](#_Toc159842853)

[Passaggi algoritmo 2](#_Toc159842854)

[Iperparametri e tuning 3](#_Toc159842855)

[Vantaggi e Svantaggi 4](#_Toc159842856)

[Clustering gerarchico 5](#_Toc159842857)

[Introduzione 5](#_Toc159842858)

[Applicazioni del clustering gerarchico 5](#_Toc159842859)

[Passaggi algoritmo 5](#_Toc159842860)

[Tipi di distanza 6](#_Toc159842861)

[Iperparametri e tuning 7](#_Toc159842862)

[Vantaggi e svantaggi 7](#_Toc159842863)

[Density-Based Spatial Clustering of Applications with Noise (DBSCAN) 9](#_Toc159842864)

[Introduzione 9](#_Toc159842865)

[Applicazioni del DBSCAN 10](#_Toc159842866)

[Passaggi algoritmo 10](#_Toc159842867)

[Iperparametri e tuning 11](#_Toc159842868)

[Vantaggi e svantaggi 11](#_Toc159842869)

[BIRCH Clustering: Un'Analisi Approfondita 13](#_Toc159842870)

[Spectral Clustering: Un'Analisi Approfondita 17](#_Toc159842871)

# K-MEANS

## Introduzione

Il clustering K-Means è un algoritmo fondamentale di apprendimento automatico non supervisionato, progettato per suddividere un set di dati in gruppi distinti, basati sulle somiglianze tra i punti dati. Viene ampiamente utilizzato in vari campi, tra cui l'analisi dei dati, l'elaborazione delle immagini, la bioinformatica e il text mining.

L’algoritmo K-Means, basato sul criterio della somma dei quadrati, opera in modo non supervisionato, non basandosi su dati di allenamento etichettati, bensì esplora invece la struttura intrinseca dei dati per formare clusters, con ogni cluster che rappresenta un gruppo distinto dei dati. L'algoritmo cerca di minimizzare la varianza all'interno dei cluster (detta anche *varianza within*) e massimizzare la varianza tra di essi (anche chiamata *varianza between*), cercando fondamentalmente di identificare clusters internamente compatti ed esternamente ben separati.

## Applicazioni di K-Means

K-Means trova applicazioni in diversi campi grazie alla sua versatilità e semplicità. Ecco sei esempi tipici di utilizzo della tecnica di cluster analysis:

* **Segmentazione dei Clienti**: aziende, imprese di e-commerce e spesso utilizzano K-Means come metodo per segmentare i clienti in base al loro comportamento d'acquisto; i clienti con modelli di acquisto simili vengono raggruppati, così che le aziende possano adottare strategie di marketing mirate, ottimizzare le raccomandazioni di prodotti e servizi e migliorare l’esperienza di acquisto del cliente.
* **Compressione delle Immagini**: nell'elaborazione delle immagini, K-Means può essere applicato per ridurre il numero di colori in un'immagine; raggruppando colori simili e rappresentando ciascun cluster con il suo colore di centroide, l'algoritmo permette di comprimere efficacemente l'immagine preservando la qualità visiva.
* **Rilevamento delle Anomalie**: nella sicurezza informatica, K-Means può essere impiegato per il rilevamento delle anomalie per identificare modelli insoliti nel traffico di rete; raggruppando il comportamento medio di utenti online, qualsiasi deviazione da questi cluster può essere segnalata come potenziali minacce alla sicurezza o anomalie.
* **Clustering dei Documenti**: tecniche di text mining ed elaborazione del linguaggio naturale spesso utilizzano K-Means per il clustering di documenti, in modo tale da raggruppare documenti simili, col fine di un efficace recupero delle informazioni, semplificata modellazione dei topic all’interno di una collezione di documenti e riassunto di un corpus di documenti.
* **Analisi dei Dati Genomici**: in bioinformatica, K-Means viene applicato per raggruppare geni in base ai loro modelli di espressione; identificare gruppi di geni con profili di espressione simili può fornire approfondimenti sui processi biologici, contribuendo in modo significativo alla comprensione dei meccanismi genetici.
* **Sistemi di raccomandazioni**: in diversi contesti, come le vendite online, la fruizione di servizi di intrattenimento (come film, serie tv, musica, libri…), l’algoritmo K-Means può essere utilizzato per fornire suggerimenti efficaci, basandosi sugli interessi di altri utenti con comportamenti ed interessi simili all’utente da suggerire.

## Passaggi algoritmo

L'algoritmo K-Means segue un processo iterativo per raffinare iterativamente le assegnazioni dei cluster. I passaggi chiave sono i seguenti:

1. **Normalizzazione dei dati**: il metodo K-Means è molto sensibile agli outliers, per questo motivo la fase di normalizzazione dei dati è fondamentale per ottenere risultati validi.
2. **Scelta del numero di cluster** (e dunque il numero di centroidi): si tratta di un problema fondamentale dell’algoritmo, in quanto il metodo non prevede una scelta automatica del numero di cluster in base a qualche indice o valore, bensì una scelta dell’utente per un numero *K* di cluster.
3. **Definizione dei centroidi**: i centroidi sono i punti definiti come centro di ogni cluster (dunque a K cluster corrispondono *K* centroidi); inizialmente viene scelto casualmente un numero di centroidi pari al numero di cluster ricercati, e i centroidi sono scelti casualmente all’interno dell’insieme di dati, non avendo ancora informazioni approfondite sui dati a disposizione.
4. **Assegnazione dei punti ai clusters**: ogni punto viene assegnato al cluster il cui centroide è più vicino in termini di distanza euclidea, calcolata nel seguente modo:  
   a fase, dunque, viene a formarsi un numero *K* di cluster pari al numero *K* di centroidi.
5. **Aggiornamento dei centroidi**: viene ricalcolata la media (dunque il centroide) di tutti i cluster utilizzando i punti assegnati al cluster stesso:
6. **Ripetizione passaggi 3 e 4 fino alla convergenza dell’algoritmo**: la convergenza avviene quando:
   * i centroidi non cambiano più significativamente
   * viene raggiunto il numero massimo di iterazioni prestabilito all’inizio.
   * i punti non cambiano più il cluster a cui sono assegnati

Dunque gli ultimi centroidi, e quindi anche gli ultimi cluster trovati, sono il risultato finale ottenuto col metodo K-Means applicato.

## Iperparametri e tuning

L'algoritmo K-Means coinvolge una serie di iperparametri che influenzano significativamente le sue prestazioni e i risultati ottenuti (i seguenti parametri riportati sono quelli modificabili con l’algoritmo *KMeans*, della libreria *sklearn*, utilizzato su *Python*):

* **n\_clusters (default*: 8*)**: definisce il numero *K* di clusters che l'algoritmo mira a creare. Determinare il valore ottimale di *K* è cruciale, e esistono varie tecniche a questo scopo. Un metodo comune è il metodo del gomito (elbow method), in cui l'algoritmo viene eseguito con valori diversi di K, e il punto in cui la riduzione della varianza rallenta (formando una curva a gomito) viene considerato come il numero di clusters *K* ottimale. In generale, la letteratura e studi pregressi in ambito biomedico, riportano un utilizzo più ricorrente di metodo K-Means applicato con un numero di clusters compreso tra due e cinque.
* **init(default: ‘*kmeans++*’)**: permette di definire il metodo con cui sono scelti i centroidi iniziali da cui l’algoritmo inizia ad eseguire la procedura:
* ‘*k-means++*’: tecnica che seleziona un insieme di centroidi iniziali basato su una distribuzione empirica del contributo dei punti sull’inerzia complessiva dei punti; questo metodo permette di arrivare alla convergenza più velocemente.
* ‘*random’*: seleziona i centroidi in maniera random tra i punti presenti
* Può essere anche definito un array con dei punti (esistenti o creati artificialmente) da definire come centroidi iniziali.
* **n\_init(default: ‘auto’)**: definisce il numero di volte che viene rieseguito l’algoritmo con un seed diverso; se ‘*auto’*, il valore dipende dal parametro “init”.
* **max\_iter(default: *300*)**: numero massimo di iterazioni che può eseguire l’algoritmo per arrivare alla convergenza.
* **tol(default: *0.0001*)**: tolleranza relativa alla norma di Frobenius della differenza tra due centroidi in due iterazioni consecutive, per dichiarare la convergenza raggiunta.
* **algorithm(default: ‘*lloyd’*)**: tipo di algoritmo K-Means da usare; il default è ‘*lloyd’*, altrimenti esiste l’opzione ‘*elkan’* più indicata (ma anche più computazionalmente dispendiosa) per datasets con cluster ben definiti.

## Vantaggi e Svantaggi

Tra i vantaggi dell’algoritmo K-Means abbiamo:

* **Semplicità ed efficienza**: l’algoritmo K-Means è diretto e computazionalmente efficiente (a livello di requisiti di memoria, tempo richiesto e ridotta capacità computazionale richiesta dai dispositivi utilizzati per eseguirlo), rendendolo adatto per set di dati di grandi dimensioni.
* **Scalabilità**: l'algoritmo K-Means si adatta bene alle dimensioni del set di dati, rendendolo applicabile a set di dati con un elevato numero di punti dati.
* **Interpretabilità**: i risultati ottenuti sono semplici da interpretare, poiché i cluster sono formati sulla base della media dei punti dati al loro interno.
* **Versatilità**: l’algoritmo K-Means può essere applicato a svariate tipologie di dati, e non è limitato a distribuzioni specifiche dei dati.

Tuttavia, l’algoritmo K-Means presenta comunque una serie di svantaggi, tra cui:

* **Sensibilità ai centroidi iniziali**: le prestazioni dell'algoritmo K-Means possono essere molto sensibili alla selezione iniziale dei centroidi, portando potenzialmente a soluzioni subottimali in caso di scelte iniziali poco indicate.
* **Ottimi locali**: K-Means può convergere a ottimi locali, e il risultato finale dipende dalle condizioni iniziali, influenzando la qualità del clustering (due centroidi iniziali molto vicini possono condizionare il risultato finale, non rappresentando realmente la corretta divisione dei dati in clusters).
* **Assunzione di clusters sferici**: l’algoritmo K-Means assume che i cluster siano sferici e di dimensioni uguali, rendendolo meno efficace in alcuni contesti in cui i clusters sarebbero non convessi, o comunque di dimensioni irregolari.
* **Dipendenza dalla scelta di *K* clusters/centroidi**: l'algoritmo K-Means richiede la specifica del numero di cluster, scelta non sempre semplice e scontata, specialmente in contesti in cui il numero ottimale non è noto a priori né di facile definizione.
* **Sensibilità agli outliers**: gli outliers possono influenzare significativamente il calcolo della media, e dunque il calcolo dei centroidi ad ogni aggiornamento nell’esecuzione dell’algoritmo, influenzando il risultato complessivo del clustering. Per questo motivo, è fondamentale la fase di normalizzazione dei dati.
* **Convergenza lenta dei dati**: in caso di dataset molto grandi, la convergenza può essere più lenta, e dunque bisogna prestare particolare attenzione alla scelta di alcuni parametri, come il *max\_iter*.

# Clustering gerarchico

## Introduzione

Il Clustering Gerarchico è una potente tecnica nell'apprendimento automatico non supervisionato, progettata principalmente per organizzare i dati in una struttura gerarchica di cluster nidificati. A differenza dei metodi di partizionamento come il K-Means, il clustering gerarchico crea una struttura a forma di albero, nota come dendrogramma, che rappresenta le relazioni tra i punti dati. Questa analisi completa mira a chiarire le complessità scientifiche del clustering gerarchico, approfondendone lo scopo, i parametri, le strategie di regolazione, le applicazioni, i passaggi algoritmici e i vantaggi e gli svantaggi associati.

Il principale scopo del clustering gerarchico è organizzare i dati in una gerarchia di cluster, consentendo un'esplorazione dettagliata delle somiglianze e delle differenze tra i dati. A differenza degli algoritmi di partizionamento, il clustering gerarchico fornisce una rappresentazione ricca della struttura dei dati, offrendo approfondimenti sulle relazioni globali e locali. L'algoritmo raggiunge questo obiettivo fondamentalmente unendo o dividendo iterativamente i cluster, formando un dendrogramma che racchiude l'organizzazione gerarchica dei dati.

Il clustering gerarchico si divide in due tipologie differenti: il clustering gerarchico agglomerativo (basato su un approccio *bottom-up*, in cui si inizia con cluster formati da una sola osservazione e uniti a due a due fino ad avere un unico cluster contenente tutti i dati) e il cluster gerarchico divisivo (con approccio *top-down*, in cui tutti gli elementi fanno parte di un unico cluster inizialmente, che viene diviso in sottocluster ad ogni passaggio fino ad ottenere un cluster per ogni osservazione).

Applicazioni del clustering gerarchico

Il clustering gerarchico trova applicazioni in diversi campi grazie alla sua capacità di rivelare relazioni gerarchiche nei dati. Ecco cinque esempi notevoli:

1. **Tassonomia in biologia**: il clustering gerarchico può essere utilizzato per organizzare le specie in base alle somiglianze genetiche; utile per creare tassonomie, comprendere le relazioni evolutive e classificare le specie in base alle caratteristiche genetiche.
2. **Segmentazione dei clienti nel marketing**: gli esperti di marketing utilizzano spesso il clustering gerarchico per suddividere i clienti in base a varie caratteristiche come comportamento d'acquisto, informazioni demografiche e preferenze; in questo modo, è facile personalizzare strategie di marketing per segmenti specifici di clienti, ottimizzando al massimo la pubblicità e le offerte personalizzate.
3. **Classificazione dei documenti nell'elaborazione del linguaggio naturale**: il clustering gerarchico permette di organizzare un ampio corpus di documenti di testo in base alle somiglianze di contenuto e topics; aiuta nella classificazione dei documenti, nella modellazione dei temi e nella sintesi raggruppando documenti simili.
4. **Analisi delle immagini in Artificial Vision**: il clustering gerarchico può essere impiegato per organizzare le immagini in base alle caratteristiche visive; aiuta nel recupero delle immagini basato sul contenuto, consentendo il recupero efficiente e rapido di immagini simili.
5. **Analisi dei dati genomici in bioinformatica**: il clustering gerarchico è applicato per analizzare i profili di espressione genica in diverse condizioni; aiuta ad identificare gruppi di geni con modelli di espressione simili, fornendo approfondimenti su processi biologici e vie metaboliche.

## Passaggi algoritmo

Il clustering gerarchico agglomerativo segue un processo ricorsivo per costruire un dendrogramma. I passaggi chiave sono i seguenti:

1. **Normalizzazione dei dati**: il clustering gerarchico è molto sensibile agli outliers, per questo motivo la fase di normalizzazione dei dati è fondamentale per ottenere risultati validi.
2. **Inizializzazione**: in questa fase iniziale, ogni punto dati viene considerato come un cluster singolo.
3. **Calcolo delle distanze tra coppie**: viene calcolata la distanza tra ogni coppia di cluster utilizzando la metrica di distanza scelta, e i parametri definiti.
4. **Unione/Divisione**: una volta identifica la coppia di cluster con la distanza più piccola (in base al criterio di collegamento scelto), la coppia di cluster viene unita in un nuovo cluster.
5. **Aggiornamento della matrice delle distanze**: a questo punto viene ricalcolata la matrice di distanze, considerando il nuovo cluster appena creato e tutti gli altri cluster.
6. **Ripetizione passaggi dal 3 al 5**: vengono ripetuti iterativamente i passaggi 3,4 e 5 fino a quando rimane un singolo cluster, che comprende tutti i punti dati.

In parallelo al processo viene anche costruito il dendrogramma relativo al risultato della procedura di clustering.

## Tipi di distanza

Ci sono diverse tipologie di distanza che possono essere utilizzate nell’algoritmo:

* **Distanza euclidea**: detta anche “distanza l2”, è calcolata come norma euclidea della differenza tra due punti (o unità statistiche):
* **Distanza di Manhattan**: detta anche “distanza della città a blocchi”, o “metrica del taxi”, o “distanza l1”, è pari alla somma dei cateti che uniscono due punti (o unità statistiche), pensando a due cateti paralleli agli assi:
* **Distanza di Minkowski**: si tratta praticamente di una formula generale, da cui è possibile ricavare tutte le altre distanze (con k=2 è la distanza euclidea, con k=1 è la distanza di Manhattan…)
* **Distanza coseno:** misuraquanto sono similidue vettori:

Inoltre, è importante ricordare alcune proprietà legate alla famiglia di distanze di Minkowski:

1. La metrica di Minkowski è una funzione non crescente dell’indice k, per cui valgono le seguenti disuguaglianze:
2. Le distanze di Minkowski sono invarianti per translazione delle variabili:  
   dove c è un vettore p-dimensionale di costanti.
3. Le distanze di Minkowski sono invarianti per trasformazioni ortogonali (rotazioni) delle variabili:  
   dove *T* è una matrice *p x p* tale che *T/T=I*.

## Iperparametri e tuning

Gli algoritmi di clustering agglomerativo coinvolgono una serie di iperparametri, che influenzano significativamente le loro prestazioni ed i risultati ottenuti (i seguenti parametri riportati sono quelli modificabili con l’algoritmo *AgglomerativeClustering*, della libreria *sklearn*, utilizzato su *Python*):

* **n\_clusters (default: *2*)**: definisce il numero di clusters che l'algoritmo mira a creare. Dev’essere posto pari a ‘None’ se il parametro “distance\_threshold” ha un valore definito.
* **linkage (default: ‘*ward’*):** il criterio di collegamento utilizzato, che determina quale distanza utilizzare per dividere/unire i clusters; può essere:
  + ‘***ward****’* (o metodo di ward), definito come la minimizzazione dell'incremento di varianza per i due clusters uniti:  
    si tratta di un metodo utile con cluster che hanno molto rumore, in quanto tende a separare comunque bene i clusters ottenuti; tuttavia, si tratta di un metodo che tende a riprodurre clusters globulari.
  + ‘***single****’* (o metodo del legame singolo): definito come la distanza minima tra i punti in due clusters diversi:  
    si tratta di un metodo utile in caso di forme non ellittiche dei clusters, e funziona bene a patto che la distanza tra cluster non sia piccola; infatti, se c’è rumore tra i due cluster, la distanza singola tende a separare male clusters sovrapposti.
  + ‘***average****’* (o metodo del legame medio): definito come la distanza media tra i punti in clusters diversi:  
    il metodo del legame medio tende a separare bene clusters che tendono a sovrapporsi, e in generale clusters con rumore; tuttavia, si tratta di un metodo che solitamente riproduce clusters globulari.
  + ‘***complete****’* (o metodo del legame completo): definito come la distanza massima tra i punti in due clusters diversi:  
    il metodo del legame completo tende a separare bene clusters che tendono a sovrapporsi, e in generale clusters con molto rumore; tuttavia si tratta di un metodo che tende a “rompere” grandi clusters (dividendoli in due o più clusters), e, come il metodo del legame medio, che tende a riprodurre cluster globulari.
* **metric(default: ‘*euclidean’*)**: la metrica utilizzata per calcolare la distanza; di default, viene definita come ‘*euclidean’* (uguale a ‘l2’), mentre altre opzioni utilizzabili sono ‘*manhattan’* (o ‘*l1’*), ‘*cosine’* o ‘*precomputed’* (solo se viene data una matrice di distanze come metodo di input dell’algoritmo; inoltre, se il metodo di linkage scelto è ‘*ward’*, solo la metrica ‘*euclidean’* è valida.
* **distance\_threshold(default: *‘None’*)**: definisce il limite che porta l’algoritmo alla convergenza, dunque a fermarsi ad un certo numero di clusters trovati; se pari a ‘*None’*, anche “n\_clusters” dev’essere ‘*None’*.

## Vantaggi e svantaggi

Tra i vantaggi del clustering gerarchico abbiamo:

1. **Rappresentazione gerarchica**: l’algoritmo fornisce una rappresentazione gerarchica delle relazioni tra i dati, consentendo un'esplorazione dettagliata delle strutture dei cluster.
2. **Nessuna assunzione sulla forma dei clusters**: a differenza di alcuni algoritmi (ad esempio, il K-Means), il clustering gerarchico non presuppone l’assunzione di una forma specifica da parte dei cluster, rendendolo molto più versatile.
3. **Nessuna necessità di specificare il numero di cluster a priori**: il clustering gerarchico non richiede necessariamente la specifica preventiva del numero di cluster (come altri metodi come il K-Means), poiché produce una struttura gerarchica.
4. **Applicabilità a diverse metriche di distanza**: l’algoritmo può utilizzare diverse metriche di distanza in base alla natura dei dati.
5. **Interpretabilità**: la struttura ad albero del dendrogramma è intuitiva, e facilita l'interpretazione delle relazioni tra i dati.

Tuttavia, l’algoritmo presenta comunque una serie di svantaggi, tra cui:

1. **Complessità computazionale**: a livello temporale e di complessità dell’esecuzione l'algoritmo può essere dispendioso dal punto di vista computazionale, specialmente per set di dati di grandi dimensioni, rendendolo meno efficiente rispetto ad altri metodi di partizionamento.
2. **Sensibilità a rumore ed outliers**: il clustering gerarchico può essere molto sensibile a rumore ed outliers, influenzando facilmente la stabilità dei cluster ottenuti; per questo motivo, è fondamentale il passaggio di normalizzazione dei dati.
3. **Irreversibilità delle unioni**: una volta che i cluster vengono uniti, il processo è irreversibile, e la scelta del punto di unione può influenzare significativamente il risultato finale (a differenza di un procedimento come il K-Means, dove l’algoritmo rielabora ad ogni passaggio i risultati ottenuti nel precedente).
4. **Difficoltà nel gestire set di dati voluminosi**: i requisiti computazionali e l'uso della memoria aumentano parecchio con le dimensioni del set di dati, rendendolo di difficile utilizzo per set di dati molto grandi.
5. **Dipendenza dalla metrica di distanza**: la scelta della metrica di distanza può influenzare sensibilmente il risultato finale del clustering, e la selezione di una metrica appropriata richiede conoscenze di dominio.

# Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

## Introduzione

DBSCAN, acronimo di Density-Based Spatial Clustering of Applications with Noise, è un algoritmo di clustering robusto e ampiamente utilizzato nell'apprendimento automatico. Sviluppato da Martin Ester, Hans-Peter Kriegel, Jörg Sander e Xiaowei Xu nel 1996, DBSCAN è particolarmente efficace nell'identificare cluster con forme variabili e gestire gli outlier all'interno dei dataset. Questa analisi completa mira a fornire una comprensione approfondita di DBSCAN, elucidando il suo scopo, i parametri, i meccanismi di regolazione, le applicazioni, le fasi algoritmiche e i vantaggi e limiti associati.

Il principale scopo di DBSCAN è scoprire cluster nei dati basandosi sulla densità dei punti. A differenza degli algoritmi di clustering tradizionali come il K-Means, DBSCAN non assume un numero predefinito di cluster e può identificare cluster con forme irregolari. È particolarmente abile nel gestire cluster di dimensioni e forme diverse, rendendolo adatto a dataset in cui i cluster possono avere densità differenti.

Il concetto chiave di DBSCAN ruota attorno alla definizione di cluster come regioni dense di dati separate da regioni più sparse. Categorizza i punti come punti centrali, punti di bordo o punti di rumore, a seconda della loro densità nel dataset. L'algoritmo cerca di raggruppare i punti centrali insieme e identificare i cluster come regioni di elevata densità di punti.

\*Mecanismi di Regolazione:\*

La regolazione di DBSCAN comporta la selezione di valori appropriati per epsilon (ε) e il numero minimo di punti (MinPts). Ecco alcune strategie per la regolazione:

1. \*\*Ispezione Visiva e Conoscenza del Dominio:\*\* Visualizzare il dataset e utilizzare la conoscenza del dominio per stimare valori ragionevoli per epsilon e MinPts. Regolare questi parametri in base alle caratteristiche dei dati.

2. \*\*Grafico di Raggiungibilità:\*\* Tracciare la distanza di raggiungibilità dei punti dati rispetto al loro ordine di elaborazione può aiutare a identificare un valore adatto per epsilon.

3. \*\*Identificazione del Punto di Ginocchio:\*\* Nel grafico di raggiungibilità, il punto di ginocchio indica spesso una transizione da punti centrali a punti di rumore. Scegliere epsilon in questo punto può essere efficace.

4. \*\*Punteggio di Silhouette:\*\* Valutare le prestazioni di clustering utilizzando il punteggio di silhouette per valori diversi dei parametri. Il punteggio di silhouette misura quanto sono ben separati i cluster.

## Applicazioni del DBSCAN

DBSCAN trova applicazioni in vari settori grazie alla sua flessibilità e capacità di gestire cluster di diverse forme e dimensioni. Ecco cinque esempi significativi:

1. **Rilevamento di anomalie**: rilevamento di comportamenti anomali nel traffico di una rete; infatti, DBSCAN può identificare cluster in cui sono presenti di comportamenti normali, e segnalare gli outlier come potenziali anomalie nei dati di rete.
2. **Clustering di dati geospaziali**: clustering di posizioni geografiche in base alla densità dei punti, in modo da identificare cluster di dati basati sulla posizione, come punti caldi di criminalità, aree con elevata attività di clienti, aree molto trafficate...
3. **Segmentazione di immagini**: suddivisione di immagini in regioni significative; DBSCAN può essere applicato per raggruppare pixel con caratteristiche simili, facilitando la segmentazione delle immagini per le applicazioni di artificial vision.
4. **Sistemi di raccomandazione**: clustering di utenti in base alle loro preferenze e comportamenti; DBSCAN può identificare gruppi di utenti con preferenze e comportamenti simili, agevolando la creazione di sistemi di raccomandazione personalizzati.
5. **Biologia e genomica**: clustering di geni basato sui modelli di espressione; utile per identificare gruppi di geni con modelli di espressione simili, in modo da fornire intuizioni sui processi e le vie biologiche.

## Passaggi algoritmo

DBSCAN segue un insieme diretto ma potente di fasi per identificare cluster e punti di rumore:

1. \*\*Inizializzazione:\*\*

- Partire da un punto dati arbitrario che non è stato visitato.

2. \*\*Identificazione dei Punti Centrali:\*\*

- Determinare se il vicinato intorno al punto dati corrente contiene almeno MinPts punti dati. Se sì, il punto è contrassegnato come punto centrale.

3. \*\*Espansione del Cluster:\*\*

- Se viene trovato un punto centrale, avvia un nuovo cluster e aggiungi tutti i punti centrali collegati al cluster. Esplora il vicinato di ogni punto centrale per trovare punti centrali aggiuntivi.

4. \*\*Ripeti:\*\*

- Ripeti i passaggi 2 e 3 fino a quando tutti i punti centrali e i loro punti raggiungibili sono assegnati a cluster.

5. \*\*Assegnazione dei Punti di Bordo:\*\*

- Identifica i punti di bordo raggiungibili da un punto centrale ma che non hanno abbastanza vicini per essere considerati punti centrali. Assegna questi punti di bordo al cluster.

6. \*\*Identificazione dei Punti di Rumore:\*\*

- Assegna i punti rimanenti non visitati come punti di rumore, poiché non fanno parte di alcun cluster.

## Iperparametri e tuning

Il DBSCAN clustering utilizza alcuni iperparametri che influenzano significativamente le prestazioni ed i risultati ottenibili (i seguenti parametri riportati sono quelli modificabili con l’algoritmo *DBSCAN*, della libreria *sklearn*, utilizzato su *Python*):

* **eps (default: *0.5*)**: definisce la massima distanza tra due campioni per essere considerati uno vicino dell’altro. Si tratta del parametro più importante, fondamentale per i risultati dell’algoritmo.
* **min\_samples(default: *5*)**: si tratta del numero di campioni nel vicinato di un punto perché questo possa essere considerato “core point”.
* **algorithm(default: *‘auto’*)**: definisce il tipo di algoritmo utilizzato dal modulo NearestNeighbors per calcolare la distanza e trovare i vicini per ogni punto; può essere:
  + ‘*auto’*: l’algoritmo cerca di determinare il miglior approccio in base ai dati a disposizione.
  + ‘*ball\_tree’*: miglioramento rispetto al ‘*kd\_tree*’ per quanto riguarda il comportamento con dati di grandi dimensioni; questo metodo divide i dati in serie di iper-sfere annidate, rendendolo più costoso di un ‘*kd\_tree*’ ma efficiente su dati di grandi dimensioni e molto strutturati.
  + ‘*kd\_tree’*: miglioramento rispetto al ‘*brute*’, è una struttura ad albero binaria che divide le regioni dell’albero in regioni annidate, in modo da avere una costruzione dell’albero molto veloce; tuttavia, si tratta di un metodo molto efficiente con low-dimensional data ma che tende a diventare inefficiente con dataset con molte (>20) dimensioni.
  + ‘*brute*’: molto usata per piccoli dataset, poco efficiente e utile per grandi dataset.
* **metric(default: ‘*euclidean’*)**: la metrica utilizzata per calcolare la distanza; di default, viene definita come ‘*euclidean’* (uguale a ‘l2’), mentre altre opzioni utilizzabili sono ‘*manhattan’* (o ‘*l1’*), ‘*cosine’* o ‘*precomputed’* (solo se viene data una matrice di distanze come metodo di input dell’algoritmo; inoltre, se il metodo di linkage scelto è ‘*ward’*, solo la metrica ‘*euclidean’* è valida.
* **leaf\_size(default: *30*)**: questo parametro indica la dimensione delle “foglie” per i tipi di albero indicati con il parametro ‘*algorithm*’;
* **p(default: *‘None’*)**: la potenza della distanza di Minkowski utilizzata per calcolare la distanza tra punti (se ‘*None’*, ovvero il default, è posta uguale a 2, ovvero la distanza euclidea);
* **n\_jobs(default: *‘None’*)**: questo parametro indica il numero di operazioni parallele da eseguire (se ‘*None’*, ovvero il default, è posta uguale a 1, ovvero la distanza euclidea);

## Vantaggi e svantaggi

\*\*Vantaggi:\*\*

1. \*\*Flessibilità nella Forma dei Cluster:\*\* DBSCAN può identificare cluster con forme arbitrarie, rendendolo adatto a dataset in cui i cluster hanno forme irregolari.

2. \*\*Rilevamento Automatico degli Outlier:\*\* L'algoritmo identifica e etichetta automaticamente i punti di rumore come outlier, rendendolo robusto agli outlier nel dataset.

3. \*\*Nessuna Necessità di Specificare il Numero di Cluster:\*\* DBSCAN non richiede all'utente di specificare a priori il numero di cluster, poiché può adattarsi alla struttura di densità intrinseca dei dati.

4. \*\*Gestisce Cluster con Diverse Densità:\*\* DBSCAN è efficace nell'identificare cluster con densità variabili, adattandosi a regioni con densità di punti dati sia alta che bassa.

5. \*\*Robusto alle Variazioni dei Parametri:\*\* DBSCAN è relativamente robusto alle variazioni dei parametri, e piccoli cambiamenti in epsilon o MinPts spesso non portano a cambiamenti significativi nei risultati.

\*\*Svantaggi:\*\*

1. \*\*Sensibilità alle Impostazioni dei Parametri:\*\* Le prestazioni dell'algoritmo possono essere sensibili alla scelta dei parametri, e trovare valori adatti può richiedere conoscenze di dominio o sperimentazione.

2. \*\*Difficoltà con Densità Variabile:\*\* DBSCAN può avere difficoltà con dataset in cui i cluster mostrano densità significativamente diverse, poiché la scelta di un epsilon universale può essere difficile.

3. \*\*Difficoltà con Dati ad Alta Dimensionalità:\*\* In spazi ad alta dimensionalità, il concetto di distanza diventa meno intuitivo e l'efficacia del clustering basato sulla densità può diminuire.

4. \*\*Sensibilità ai Punti di Bordo:\*\* L'assegnazione dei punti di bordo può essere sensibile a piccoli cambiamenti nel dataset, portando a variazioni potenziali nelle assegnazioni dei cluster.

5. \*\*Intensivo in Memoria:\*\* Conservare la matrice delle distanze di raggiungibilità per tutti i punti dati può essere intensivo in termini di memoria, rendendo DBSCAN meno adatto per dataset molto grandi.

# BIRCH Clustering: Un'Analisi Approfondita

\*Introduzione:\*

BIRCH, acronimo di Balanced Iterative Reducing and Clustering using Hierarchies, è un algoritmo di clustering progettato per set di dati di grandi dimensioni. Sviluppato da Tian Zhang, Raghu Ramakrishnan e Miron Livny nel 1996, BIRCH è particolarmente adatto per situazioni in cui l'intero dataset non può essere contenuto in memoria, rendendolo una scelta efficiente e scalabile per il clustering di dati. Questa analisi approfondita mira a fornire una comprensione dettagliata di BIRCH, chiarificando il suo scopo, i parametri, i meccanismi di taratura, le applicazioni, le fasi algoritmiche e i vantaggi e le limitazioni associati.

\*Scopo dell'Algoritmo:\*

Lo scopo principale di BIRCH è eseguire il clustering su grandi dataset in modo efficiente. È progettato specificamente per gestire set di dati troppo estesi per adattarsi in memoria, rendendolo adatto a scenari in cui gli algoritmi di clustering tradizionali potrebbero incontrare limitazioni. BIRCH si concentra sulla creazione di una rappresentazione compatta del dataset, chiamata albero di feature di clustering (CF), che consente un clustering efficiente e minimizza la necessità di memorizzare l'intero dataset in memoria.

BIRCH impiega un processo in due fasi: innanzitutto, costruisce un albero CF mediante la fusione e la compressione iterativa dei punti dati e, successivamente, applica un algoritmo di clustering tradizionale alla rappresentazione condensata per formare i cluster finali. Questo approccio consente a BIRCH di scalare bene con grandi dataset e fornisce un equilibrio tra l'uso della memoria e l'accuratezza del clustering.

\*Parametri e Come Tararli:\*

BIRCH coinvolge diversi parametri che possono essere regolati per ottimizzarne le prestazioni. Ecco i principali parametri e le strategie per la taratura:

1. \*\*Fattore di Branched (B):\*\* Il fattore di branched definisce il numero massimo di sotto-cluster che un nodo non foglia nell'albero CF può avere. Un fattore di branched più alto può portare a un albero più bilanciato ma potrebbe richiedere più memoria.

2. \*\*Soglia (T):\*\* La soglia specifica il numero massimo di punti dati che un nodo foglia nell'albero CF può contenere prima di attivare una suddivisione. La regolazione di questa soglia influenza la granularità della struttura dell'albero.

\*Meccanismi di Taratura:\*

1. \*\*Validazione ed Valutazione:\*\* Valutare i risultati del clustering utilizzando metriche di validazione interne o esterne per diverse configurazioni dei parametri. Scegliere la configurazione che si allinea meglio con i requisiti specifici del compito di clustering.

2. \*\*Considerazione dell'Uso della Memoria:\*\* In base alla memoria disponibile, regolare il fattore di branched e la soglia per ottenere un equilibrio tra la rappresentazione compatta e l'accuratezza del clustering.

\*Applicazioni di BIRCH:\*

BIRCH trova applicazioni in vari settori, specialmente dove è necessario elaborare efficientemente set di dati di grandi dimensioni. Ecco cinque esempi significativi:

1. \*\*Rilevamento di Intrusione di Rete:\*\*

- \*Scenario:\* Analisi dei dati del traffico di rete per identificare pattern di intrusione.

- \*Applicazione:\* BIRCH può effettuare il clustering efficiente dei dati di traffico di rete per rilevare pattern insoliti, contribuendo all'identificazione precoce di potenziali intrusioni.

2. \*\*Segmentazione di Clienti nell'E-commerce:\*\*

- \*Scenario:\* Clustering dei clienti in base al loro comportamento d'acquisto e alle preferenze.

- \*Applicazione:\* BIRCH può gestire grandi dataset di transazioni, consentendo alle imprese di e-commerce di segmentare i clienti per marketing mirato e raccomandazioni personalizzate.

3. \*\*Clustering di Documenti nel Text Mining:\*\*

- \*Scenario:\* Clustering di grandi corpora di documenti per il topic modeling.

- \*Applicazione:\* BIRCH può elaborare ed eseguire il clustering efficiente di dati testuali, contribuendo all'organizzazione dei documenti, alla sintesi e alla scoperta di argomenti.

4. \*\*Analisi di Immagini in Computer Vision:\*\*

- \*Scenario:\* Clustering di feature di immagini per il riconoscimento degli oggetti.

- \*Applicazione:\* La scalabilità di BIRCH lo rende adatto per elaborare grandi dataset di immagini, contribuendo a compiti come la segmentazione delle immagini e il recupero di immagini basato sul contenuto.

5. \*\*Analisi di Dati Genomici:\*\*

- \*Scenario:\* Clustering di geni in base ai profili di espressione.

- \*Applicazione:\* BIRCH può gestire vaste quantità di dati genomici, contribuendo all'identificazione di gruppi di geni con pattern di espressione simili e contribuendo alla comprensione dei processi biologici.

\*Fasi Algoritmiche:\*

BIRCH segue un processo in due fasi per creare una rappresentazione compatta del dataset e eseguire il clustering:

1. \*\*Costruzione dell'Albero CF:\*\*

- \*Inizializzazione:\* Inizia con un albero CF vuoto.

- \*Inserimento:\* Itera attraverso i punti dati e inserisci ciascun punto nell'albero CF.

- Se un nodo foglia può contenere il punto senza superare la soglia, inseriscilo nel nodo.

- Se la soglia viene superata, suddividi il nodo foglia e ridistribuisci i punti dati.

- \*Fusione:\* Esegue la fusione iterativa dei nodi nell'albero CF per mantenere una rappresentazione compatta.

2. \*\*Applicazione dell'Algoritmo di Clustering:\*\*

- Dopo aver costruito l'albero CF, applica un algoritmo di clustering tradizionale (ad esempio, K-Means) sui nodi foglia dell'albero CF.

- Utilizza i cluster risultanti per analisi e interpretazione.

\*Vantaggi e Svantaggi:\*

\*\*Vantaggi:\*\*

1. \*\*Scalabilità:\*\* BIRCH è progettato per la scalabilità, rendendolo adatto a grandi dataset che non possono essere contenuti in memoria.

2. \*\*Efficienza:\*\* Creando una rappresentazione compatta (albero CF) del dataset, BIRCH riduce al minimo la necessità di memorizzare l'intero dataset in memoria, migliorandone l'efficienza.

3. \*\*Adattabilità:\*\* BIRCH è adattabile a diversi tipi di algoritmi di clustering, consentendo agli utenti di scegliere un metodo adatto per il compito di clustering specifico.

4. \*\*Uso della Memoria:\*\* L'algoritmo gestisce efficientemente l'uso della memoria creando una struttura gerarchica, consentendo l'elaborazione di grandi dataset con memoria limitata.

5. \*\*Gestione dei Dati Rumorosi:\*\* BIRCH è robusto ai dati rumorosi, e la sua struttura gerarchica aiuta a catturare i modelli complessivi nei dati.

\*\*Svantaggi:\*\*

1. \*\*Sensibilità ai Parametri:\*\* Le prestazioni di BIRCH possono essere sensibili alla scelta dei parametri, come il fattore di branched e la soglia, richiedendo una taratura attenta.

2. \*\*Limitato a Cluster Sferici:\*\* BIRCH potrebbe essere meno efficace nel trattare cluster con forme non sferiche, poiché assume cluster sferici durante la fase di clustering.

3. \*\*Dipendenza dall'Inizializzazione:\*\* La qualità dei risultati del clustering può dipendere dall'inizializzazione dell'albero CF e dalla scelta dell'algoritmo di clustering applicato nella seconda fase.

4. \*\*Complessità dell'Implementazione:\*\* Implementare BIRCH può essere più complesso rispetto ad alcuni altri algoritmi di clustering, richiedendo potenzialmente una comprensione più approfondita delle sue complessità.

5. \*\*Limitato a Certi Tipi di Dataset:\*\* Sebbene efficiente per dataset di grandi dimensioni, BIRCH potrebbe non essere la scelta ottimale per dataset con cluster ben separati e distinti.

\*Conclusione:\*

In conclusione, BIRCH si configura come una soluzione preziosa per il clustering di dataset di grandi dimensioni in modo efficiente. La sua capacità di creare una rappresentazione compatta del dataset attraverso un albero CF gerarchico, insieme alla sua adattabilità a diversi algoritmi di clustering, lo rende adatto a varie applicazioni. Tuttavia, gli utenti dovrebbero essere consapevoli della sensibilità alle impostazioni dei parametri e dell'assunzione di cluster sferici durante la fase di clustering. Una taratura attenta dei parametri, la considerazione delle caratteristiche dei dati e una scelta ponderata degli algoritmi di clustering sono cruciali per massimizzare le prestazioni di BIRCH in scenari del mondo reale. Con l'avanzare del campo dell'apprendimento automatico, BIRCH rimane uno strumento rilevante ed efficace per ricercatori e professionisti che affrontano le sfide del clustering di grandi dataset.

# Spectral Clustering: Un'Analisi Approfondita

\*Introduzione:\*

La spectral clustering è una tecnica potente nell'ambito dell'apprendimento automatico che sfrutta le proprietà spettrali dei dati per eseguire il clustering. A differenza dei metodi di clustering tradizionali che operano nello spazio di input, la spectral clustering trasforma i dati in uno spazio diverso, spesso gli autovettori di una matrice di similarità, e esegue il clustering in questo spazio trasformato. Questa analisi approfondita mira a fornire una comprensione dettagliata della spectral clustering, elucidando il suo scopo, i parametri, i meccanismi di regolazione, le applicazioni, i passaggi algoritmici e i vantaggi e le limitazioni associati.

\*Scopo dell'Algoritmo:\*

Lo scopo principale della spectral clustering è identificare cluster significativi all'interno di un dataset basandosi sulle relazioni tra i punti dati. È particolarmente efficace in scenari in cui i dati presentano strutture complesse, separabilità non lineare o quando gli algoritmi di clustering tradizionali potrebbero avere difficoltà. La spectral clustering raggiunge questo obiettivo catturando la struttura sottostante dei dati attraverso la loro rappresentazione spettrale, consentendo assegnazioni di cluster più flessibili e sfumate.

L'idea fondamentale dell'algoritmo ruota attorno alla rappresentazione dei dati come un grafo di similarità, in cui i nodi corrispondono ai punti dati e gli archi indicano similarità tra coppie di punti. Analizzando le proprietà spettrali di questo grafo, la spectral clustering svela schemi e strutture nascoste, facilitando la formazione di cluster in uno spazio trasformato.

\*Parametri e Come Regolarli:\*

La spectral clustering coinvolge alcuni parametri chiave, e la loro regolazione appropriata è cruciale per ottenere cluster significativi e accurati:

1. \*\*Numero di Cluster (k):\*\* Il parametro più fondamentale è il numero di cluster che si intende identificare. Determinare il valore corretto di k dipende spesso dall'applicazione, e varie tecniche, come il metodo del gomito o l'analisi della silhouette, possono essere impiegate per trovare un valore ottimale.

2. \*\*Matrice di Similarità:\*\* La scelta della misura di similarità e come essa viene codificata nella matrice di similarità è cruciale. Le scelte comuni includono il kernel gaussiano (RBF), i k-nn o grafi completamente connessi. La selezione dipende dalla natura dei dati e dall'esito di clustering desiderato.

3. \*\*Costruzione della Matrice di Affinità:\*\* La matrice di affinità determina le relazioni tra coppie di punti dati. Viene costruita in base alla misura di similarità scelta e può essere normalizzata ulteriormente per migliorare la stabilità dell'algoritmo.

\*Mecanismi di Regolazione:\*

1. \*\*Metriche di Validazione:\*\* Utilizzare metriche di validazione interne o esterne, come punteggi di silhouette o indice di Rand corretto, per valutare la qualità del clustering per diverse configurazioni dei parametri. Selezionare la configurazione che ottimizza la metrica scelta.

2. \*\*Analisi Esplorativa dei Dati:\*\* Condurre un'analisi esplorativa dei dati per ottenere intuizioni sulla struttura dei dati. Tecniche di visualizzazione, come scatter plot o metodi di riduzione della dimensionalità, possono aiutare a comprendere i cluster intrinseci e guidare la regolazione dei parametri.

3. \*\*Cross-Validazione:\*\* Eseguire la cross-validazione per valutare le prestazioni di generalizzazione dell'algoritmo di spectral clustering. Ciò aiuta a garantire che i cluster identificati non siano specifici per il set di dati di addestramento.

\*Applicazioni della Spectral Clustering:\*

La spectral clustering trova applicazioni in vari settori grazie alla sua capacità di gestire strutture complesse e relazioni non lineari. Ecco cinque esempi notevoli:

1. \*\*Segmentazione di Immagini in Computer Vision:\*\*

- \*Scenario:\* Suddividere un'immagine in regioni semanticamente significative.

- \*Applicazione:\* La spectral clustering può essere applicata per rappresentare le similarità tra pixel, consentendo la segmentazione delle immagini in regioni coerenti basate su colore, texture o altre caratteristiche visive.

2. \*\*Analisi di Reti Sociali:\*\*

- \*Scenario:\* Identificare comunità o gruppi all'interno di una rete sociale.

- \*Applicazione:\* La spectral clustering può rivelare strutture nascoste nelle reti sociali catturando i modelli di connettività tra gli utenti, contribuendo alla rilevazione delle comunità o alla pubblicità mirata.

3. \*\*Clustering di Dati Genomici in Bioinformatica:\*\*

- \*Scenario:\* Clustering di geni basato sui profili di espressione.

- \*Applicazione:\* La spectral clustering aiuta a identificare gruppi di geni con pattern di espressione simili, contribuendo alla comprensione dei meccanismi di regolazione genetica e delle vie biologiche.

4. \*\*Clustering di Documenti in Elaborazione del Linguaggio Naturale:\*\*

- \*Scenario:\* Raggruppamento di documenti basato su similarità semantica.

- \*Applicazione:\* La spectral clustering può elaborare matrici documento-termine o altre rappresentazioni per identificare cluster tematici in grandi collezioni di documenti, facilitando la modellizzazione dei temi e l'organizzazione documentale.

5. \*\*Rilevamento di Anomalie in Sist

emi di Rilevamento Intrusione:\*\*

- \*Scenario:\* Rilevare modelli insoliti nel traffico di rete.

- \*Applicazione:\* La spectral clustering può individuare anomalie identificando i punti dati che si discostano dai modelli normali nel traffico di rete, contribuendo alla rilevazione precoce di potenziali minacce alla sicurezza.

\*Passaggi Algoritmici:\*

La spectral clustering segue una serie di passaggi per identificare cluster nel dominio spettrale:

1. \*\*Costruzione del Grafo di Similarità:\*\*

- Dato un dataset con n punti dati, costruire un grafo di similarità in cui ogni nodo rappresenta un punto dati e gli archi codificano le similarità tra coppie di punti. Misure di similarità comuni includono il kernel gaussiano (RBF) o i k-nn.

2. \*\*Calcolo della Matrice di Affinità:\*\*

- In base alla misura di similarità scelta, calcolare una matrice di affinità che cattura le relazioni tra i punti dati. Questa matrice riflette la forza delle connessioni tra i punti nel grafo di similarità.

3. \*\*Matrice dei Gradi e Matrice Laplaciana:\*\*

- Calcolare la matrice dei gradi, che contiene informazioni sul grado (peso totale degli archi) di ciascun nodo nel grafo di similarità. Formare la matrice laplaciana sottraendo la matrice di affinità dalla matrice dei gradi.

4. \*\*Eigendecomposizione:\*\*

- Eseguire l'eigendecomposizione della matrice laplaciana per ottenere i suoi autovalori e gli autovettori corrispondenti. Ordinare gli autovettori in base agli autovalori corrispondenti.

5. \*\*Riduzione della Dimensionalità:\*\*

- Selezionare i k autovettori corrispondenti ai k autovalori più piccoli per formare una nuova matrice. Questo passaggio riduce efficacemente la dimensionalità dei dati a k dimensioni.

6. \*\*Clustering nella Dimensione Ridotta:\*\*

- Applicare un algoritmo di clustering tradizionale, come K-Means, allo spazio ridotto formato dagli autovettori selezionati. Il numero di cluster è determinato dal parametro k.

7. \*\*Assegnazione dei Cluster:\*\*

- Assegnare ciascun punto dati a uno dei cluster identificati in base alla sua rappresentazione nello spazio ridotto.

\*Vantaggi e Svantaggi:\*

\*\*Vantaggi:\*\*

1. \*\*Flessibilità nelle Forme dei Cluster:\*\* La spectral clustering può identificare cluster con forme arbitrarie, rendendola adatta a dataset con strutture complesse.

2. \*\*Gestione delle Relazioni Non Lineari:\*\* L'algoritmo è in grado di catturare relazioni non lineari e può identificare cluster che i metodi tradizionali potrebbero trascurare.

3. \*\*Efficacia in Spazi di Alta Dimensionalità:\*\* La spectral clustering può gestire dati ad alta dimensionalità operando in uno spazio spettrale a dimensionalità ridotta.

4. \*\*Capacità di Catturare la Struttura Globale:\*\* Esaminando le relazioni globali tra i punti dati, la spectral clustering è efficace nel catturare la struttura complessiva dei dati.

5. \*\*Meno Sensibile all'Inizializzazione:\*\* La spectral clustering è meno sensibile all'inizializzazione rispetto ad alcuni algoritmi di clustering tradizionali, contribuendo alla sua stabilità.

\*\*Svantaggi:\*\*

1. \*\*Sensibilità al Parametro k:\*\* La scelta del numero di cluster (k) può influenzare significativamente i risultati del clustering, e la selezione di un valore inappropriato può portare a partizioni subottimali.

2. \*\*Scalabilità:\*\* La spectral clustering potrebbe incontrare sfide di scalabilità per dataset molto grandi a causa del calcolo dell'eigendecomposizione.

3. \*\*Dipendenza dalle Misure di Similarità:\*\* Le prestazioni della spectral clustering sono influenzate dalla scelta delle misure di similarità e dei metodi di costruzione della matrice di affinità, richiedendo conoscenze di dominio.

4. \*\*Limitata a Rumore Basso e Moderato:\*\* La spectral clustering potrebbe avere difficoltà con dataset che contengono livelli elevati di rumore, poiché presume un certo livello di coerenza tra i punti dati all'interno dei cluster