Profice

# AI ACADEMY

# Applicare l’Intelligenza Artificiale nello sviluppo software

# AI ACADEMY

# ML unsupervised
# 18/06/2025

# INTRODUZIONE DELL'ISTRUTTORE

## *Tamas Szakacs*

*Formazione*

- Laureato come programmatore matematico
- MBA in management

*Principali esperienze di lavoro*

- Amministratore di sistemi UNIX
- Oracle DBA
- Sviluppatore di Java, Python e di Oracle PL/SQL
- Architetto (solution, enterprise, security, data)
- Ricercatore tecnologico e interdisciplinare di IA

Dedicato alla formazione continua

- Teorie, modelli, framework IA
- Ricerche IA
- Strategie aziendali
- Trasformazione digitale
- Formazione professionale

*email: tamas.szakacs@proficegroup.it*

# MOTIVI E RIASSUNTO DEL CORSO

L'**Intelligenza Artificiale (AI)** è oggi il motore dell'innovazione in ogni settore, grazie alla sua capacità di analizzare dati, automatizzare processi e generare nuove soluzioni. Questo corso offre una panoramica completa e pratica sullo sviluppo di applicazioni AI moderne, guidando i partecipanti dall'ideazione al rilascio in produzione.

Attraverso una **combinazione di teoria chiara ed esercitazioni pratiche**, saranno affrontate le tecniche e gli strumenti più attuali: **machine learning, deep learning, reti neurali, Large Language Models (LLM), Transformers, Retrieval Augmented Generation (RAG)** e progettazione di agenti AI.
Le competenze acquisite saranno applicate in progetti concreti, dallo sviluppo di chatbot all'integrazione di modelli generativi, fino al deploy di soluzioni AI in ambienti reali e collaborativi.

Il percorso è pensato per chi vuole imparare a progettare, valutare e integrare sistemi AI di nuova generazione, con particolare attenzione alle best practice di programmazione, collaborazione in team, sicurezza, valutazione delle performance ed etica dell'AI.

DURATA: 17 GIORNI

# OBIETTIVI

Il percorso formativo è progettato per **giovani consulenti junior**, con una conoscenza base di programmazione, che stanno iniziando un percorso professionale nel settore AI.

**L’obiettivo centrale è fornire una panoramica pratica, completa e operativa sull’intelligenza artificiale moderna**, guidando ogni partecipante attraverso tutte le fasi fondamentali.

# OBIETTIVI

- Allineare conoscenze AI, ML, DL di tutti i partecipanti
- Saper usare e orchestrare modelli LLM (closed e open-weight)
- Costruire pipeline RAG complete (retrieval-augmented generation)
- Progettare agenti AI semplici con strumenti moderni (LangChain, tool calling)
- Capire principi di valutazione, robustezza e sicurezza dei sistemi GenA
- Migliorare la produttività come sviluppatori usando tool GenAI-driven
- Padroneggiare best practice di sviluppo, versioning e deploy AI
- Introdurre i fondamenti di Graph Data Science e Knowledge Graph
- Ottenere capacità di valutazione dei modelli e metriche
- Comprensione dell'etica e dei bias nei modelli di intelligenza artificiale
- Approfondire le normative di riferimento: AI Act, compliance e governance AI

Il corso è **estremamente pratico** (circa il 40% del tempo in esercitazioni hands-on, notebook, challenge e hackathon), con l'utilizzo di Google Colab, GitHub, e tutti gli strumenti necessari per lavorare su progetti reali e simulati.

# STRUTTURA DELLE GIORNATE – PROGRAMMA BREVE

**Tutte le giornate sono di 8 ore (9:00-17:00), con 1 ora di pausa suddivisa (mezz'ora pranzo, due pause da 15 min durante la mattina e il pomeriggio).**

La progettazione sintetica delle giornate:

| Giorno | Tema | Breve descrizione |
|---|---|---|
| 1 | Git & Python clean-code | Collaborazione su progetti reali, versionamento, codice pulito e testato |
| 2 | Machine Learning Supervised | Modelli supervisionati per predizione e classificazione |
| 3 | Machine Learning Unsupervised | Clustering, riduzione dimensionale, scoperta di pattern |
| 4 | Prompt Engineering avanzato | Scrivere e valutare prompt efficaci per modelli generativi |
| 5 | LLM via API (multi-vendor) | Uso pratico di modelli LLM via API, autenticazione, deployment |
| 6 | Come costruire un RAG | Pipeline end-to-end per Retrieval-Augmented Generation |
| 7 | Tool-calling & Agent design | Progettare agenti AI che usano strumenti esterni |
| 8 | Hackathon: Agentic RAG | Challenge pratica: chatbot agentico RAG in team |

# STRUTTURA DELLE GIORNATE – PROGRAMMA BREVE

**Tutte le giornate sono di 8 ore (9:00-17:00), con 1 ora di pausa suddivisa (mezz'ora pranzo, due pause da 15 min durante la mattina e il pomeriggio).**

La progettazione sintetica delle giornate:

| Giorno | Tema | Breve descrizione |
|---|---|---|
| 9 | Hackathon: Rapid Prototyping | Da prototipo a web-app con Streamlit e GitHub |
| 10 | AI Productivity Tools | Workflow con IDE AI-powered, automazione e refactoring assistito |
| 11 | Docker & HF Spaces Deploy | Deployment di app GenAI containerizzate o su HuggingFace Spaces |
| 12 | AI Act & ISO 42001 Compliance | Fondamenti di compliance e governance AI |
| 13 | Knowledge Base & Graph Data Science | Introduzione a Knowledge Graph e query con Neo4j |
| 14 | Model evaluation & osservabilità | Metriche avanzate, explainability, strumenti di valutazione |
| 15 | AI bias, fairness ed etica applicata | Analisi dei rischi, metriche e mitigazione dei bias |
| 16-17 | Project Work & Challenge finale | Lavoro a gruppi, POC/POD, presentazione e votazione progetti |

# METODOLOGIA DEL CORSO

1. Approccio introduttivo ma avanzato

Il corso è introduttivo nei concetti base dell'AI applicata allo sviluppo, ma affronta anche tecnologie, modelli e soluzioni avanzate per garantire un apprendimento completo.

2. Linguaggio adattato

Il linguaggio utilizzato è chiaro e adattato agli studenti, con spiegazioni dettagliate dei termini tecnici per favorirne la comprensione e l'apprendimento graduale.

3. Esercizi pratici

Gli esercizi pratici sono interamente svolti online tramite piattaforme come Google Colab o notebook Python, eliminando la necessità di installare software sul proprio computer.

4. Supporto interattivo

È possibile porre domande in qualsiasi momento durante le lezioni o successivamente via email per garantire una piena comprensione del materiale trattato.

# NOTA

Il corso segue un **approccio laboratoriale**: ogni giornata combina sessioni teoriche chiare e concrete con molte attività pratiche supervisionate, per sviluppare *competenze reali* immediatamente applicabili.

I partecipanti lavoreranno spesso in gruppo, useranno notebook in Colab e versioneranno codice su GitHub, vivendo una vera simulazione del lavoro in azienda AI.

**Nessun prerequisito avanzato richiesto:** si partirà dagli strumenti e flussi fondamentali, con una crescita graduale verso le tecniche più attuali e richieste dal mercato.

# ORARIO TIPICO DELLE GIORNATE

| Orario | Attività | Dettaglio |
|---|---|---|
| 09:00 – 09:30 | Teoria introduttiva | Concetti chiave, schema della giornata |
| 09:30 – 10:30 | Live coding + esercizio guidato | Esempio pratico, notebook Colab |
| *10:30 – 10:45* | *Pausa breve* | |
| 10:45 – 11:30 | Approfondimento teorico | Tecniche, best practice |
| 11:30 – 12:30 | Esercizio hands-on individuale | Sviluppo o completamento di codice |
| 12:30 – 13:00 | Discussione soluzioni + Q&A | Condivisione e correzione |
| *13:00 – 14:00* | *Pausa pranzo* | |
| 13:30 – 14:15 | Teoria avanzata / nuovi tools | Nuovi strumenti, pattern, demo |
| 14:15 – 15:30 | Esercizio a gruppi / challenge | Lavoro di squadra su task reale |
| *15:30 – 15:45* | *Pausa breve* | |
| 15:45 – 16:30 | Sommario teorico e pratico | |
| 16:30 – 17:00 | Discussioni, feedback | Riepilogo, best practice, domande aperte |

# DOMANDE?

Cominciamo!

# OBIETTIVI DELLA GIORNATA

**Obiettivi della giornata**

- Comprendere i concetti e le differenze tra clustering e riduzione di dimensionalità
- Applicare algoritmi di clustering (k-Means, DBSCAN) a dati reali di e-commerce per segmentare i clienti
- Valutare la qualità dei cluster tramite metriche come il silhouette score
- Utilizzare tecniche di dimensionality reduction (PCA, t-SNE, UMAP) per la visualizzazione e l'analisi dei dati ad alta dimensionalità
- Introdurre l'uso degli autoencoder per l'estrazione automatica di feature e la riduzione dimensionale
- Sviluppare pipeline pratiche di clustering e visualizzazione con Python e scikit-learn
- Generare e interpretare i profili di segmenti/clienti ("personas") con il supporto di ChatGPT
- Consolidare le best practice per la segmentazione dei dati e la presentazione dei risultati in ambito business

# AGENDA DELLA GIORNATA

**1. Introduzione a clustering e riduzione di dimensionalità**
- Obiettivi e differenze tra clustering e DR

**2. Analisi ed esplorazione del dataset Olist**
- Preparazione, feature engineering e visualizzazione dati

**3. Algoritmi di clustering**
- K-Means: funzionamento, scelta dei cluster
- DBSCAN: outlier e cluster complessi
- Valutazione dei cluster (silhouette score)

**4. Dimensionality reduction e visualizzazione**
- PCA, t-SNE, UMAP: principi, applicazioni, confronto pratico

**5. Autoencoder per feature extraction**
- Architettura, differenze da PCA, applicazioni su Olist

**6. Esercitazioni pratiche su clustering e DR**
- Applicazione algoritmi, visualizzazione, confronto risultati

**7. Interpretazione dei cluster e generazione personas**
- Uso di ChatGPT per descrivere segmenti e business case

**8. Best practice e casi d'uso reali**
- Errori tipici, suggerimenti per progetti in azienda

**9. Q&A, sintesi e risorse utili**

# RIPASSO GIORNO 2 – ML SUPERVISED

**Cos’è il ML supervisionato**:

- Modelli che apprendono da dati etichettati (input → output noto)
- Obiettivo: prevedere valori (regressione) o categorie (classificazione)
- Tipici algoritmi: regressione lineare, logistic regression, decision tree, random forest, XGBoost, neural network

**Pipeline tipica**:

- Definizione del problema
- Raccolta e preparazione dati
- Split in train/validation/test
- Addestramento modello
- Valutazione con metriche (MAE, accuracy, confusion matrix, ecc.)
- Ottimizzazione/tuning
- Deploy e monitoraggio

**Esercizi svolti**:

- Regressione: previsione prezzi case
- Classificazione: churn prediction clienti
- Uso pratico di Random Forest e XGBoost
- Gestione dati sbilanciati (SMOTE, class_weight)

# COS'È UN MODELLO UNSUPERVISIONATO

**Definizione e logica:**

- Un modello unsupervisionato analizza dati **senza etichette**: non conosce la risposta corretta, ma cerca autonomamente **pattern, gruppi o strutture** nascoste nei dati.
- L'algoritmo esplora i dati, identifica somiglianze/differenze, costruisce raggruppamenti (cluster) o trova nuove rappresentazioni più semplici (riduzione dimensionalità).

**Perché è fondamentale:**

- Permette di scoprire insight sconosciuti, segmentare clienti, ridurre la complessità dei dati e visualizzare fenomeni nascosti **senza bisogno di output noti o supervisione umana**.
- È indispensabile quando non si hanno risposte predefinite, ma si vuole "capire come sono fatti" i dati.

**Quando scegliere l'unsupervised learning:**

- Quando **non esistono etichette** o target da prevedere (es: non sappiamo chi sono i "tipi di clienti")
- Quando serve **esplorare** dati, raggrupparli, trovare anomalie o visualizzarli meglio
- Ideale come fase iniziale in nuovi progetti o per scoprire nuove opportunità di business

**Esempi concreti:**

- E-commerce: raggruppa clienti in segmenti (cluster) in base a comportamenti d'acquisto
- Banche: individua transazioni anomale senza sapere prima quali siano le frodi
- Sanità: trova sottogruppi di pazienti simili in grandi database clinici
- Marketing: identifica "tribù digitali" senza conoscere a priori le loro caratteristiche
- Industria: riduce il numero di sensori/dati da analizzare mantenendo l'informazione chiave

# COS'È IL CLUSTERING

**Definizione:**
Il clustering è una tecnica unsupervisionata che raggruppa i dati in insiemi ("cluster") di elementi simili fra loro secondo certe caratteristiche, senza conoscere le etichette a priori.

**Obiettivo:**
- Trovare segmenti naturali nei dati: clienti con abitudini simili, prodotti simili, anomalie.

**Logica:**
- Ogni punto viene assegnato al cluster più "vicino" (in senso matematico: distanza o somiglianza).
- I cluster emergono in modo autonomo dai dati stessi.

**Applicazioni:**
- Segmentazione clienti
- Analisi di mercato
- Rilevamento anomalie
- Organizzazione automatica di documenti, immagini, prodotti

# COME FUNZIONA UN ALGORITMO DI CLUSTERING

**Processo base:**

1. Si definisce una misura di somiglianza (distanza euclidea, coseno, ecc.)
2. Si sceglie quanti cluster (oppure l'algoritmo li trova)
3. L'algoritmo raggruppa i punti in base alla vicinanza
4. Si valutano e interpretano i gruppi trovati

**Algoritmi principali:**

- **k-Means:** divide i dati in un numero prefissato di gruppi, massimizzando la compattezza interna
- **DBSCAN:** trova cluster di forma arbitraria, identifica outlier (rumore)
- **Gerarchico:** costruisce una gerarchia di cluster (albero)

**Cosa serve:**

- Dati numerici e normalizzati, scelta attenta di "quanti cluster" e delle feature.

# VALUTAZIONE DEI CLUSTER

**Perché valutare la qualità dei cluster?**

- Capire se i gruppi trovati sono realmente distinti
- Scegliere il numero ottimale di cluster
- Evitare “overfitting” o raggruppamenti casuali

**Buone pratiche:**

- **Confronta più metodi:** non fidarti solo di una metrica
- **Guarda i dati:** un buon clustering deve avere senso anche per il dominio del problema
- **Cerca anomalie:** punti con silhouette score negativo meritano attenzione

**Conclusione:**

La valutazione dei cluster aiuta a garantire che i risultati ottenuti abbiano valore reale e siano utili per decisioni successive.

# K-MEANS CLUSTERING

**Definizione:**
k-Means è un algoritmo di clustering che suddivide i dati in *k* gruppi (cluster) sulla base della somiglianza tra i punti.

**Logica di base:**

- L'utente sceglie il numero di cluster (*k*)
- L'algoritmo trova i *centroidi* (centri) di ciascun cluster
- Ogni punto viene assegnato al centro più vicino
- Si ripete il processo finché i gruppi non cambiano più

**Vantaggi:**

- Semplice e veloce
- Scalabile anche per dataset grandi

**Cosa si fa (step-by-step):**

1. Si scelgono casualmente *k* punti iniziali come centroidi
2. Ogni dato viene assegnato al centro più vicino
3. Si ricalcolano i centroidi come media dei punti assegnati a ciascun cluster
4. Si ripetono assegnazione e calcolo finché i cluster si stabilizzano (convergenza)

**Nota:**
Il risultato dipende dalla scelta iniziale dei centroidi; spesso si esegue l'algoritmo più volte per sicurezza.

# QUANDO E COME USARE K-MEANS?

**Quando usarlo:**

- I dati hanno gruppi separati e compatti (es: segmentazione clienti, immagini simili)
- Il numero di cluster *k* è noto o stimabile

**Limiti:**

- Funziona male se i cluster sono di forma irregolare o con densità diverse
- Sensibile ai valori anomali (outlier)
- Richiede dati numerici e normalizzati
- *k* va scelto in modo ragionato (non sempre semplice)

**Metodi comuni per selezionare il valode di *k*:**

- **Elbow method**: si calcola la somma delle distanze interne per diversi valori di *k* e si sceglie dove la "curva" si appiattisce ("gomito")
- **Silhouette score**: valuta quanto ogni punto è vicino al proprio cluster rispetto agli altri (valori più alti = cluster migliori)
- **Validazione con il dominio**: confronto con conoscenze di business o logica del problema

**Best practice:**

- Prova diversi valori di *k* e confronta risultati con metodi grafici e numerici
- Coinvolgi esperti di dominio dove possibile

# VALUTAZIONE TRAMITE ELBOW METHOD

**Definizione dell'Elbow Method:**
È una tecnica grafica per scegliere il **numero ottimale di cluster** nel k-Means.
Si basa sul calcolo della **"inertia"** (somma delle distanze quadratiche dei punti dal centro del loro cluster).

**Come funziona:**
1.Si esegue k-Means per diversi valori di k (numero di cluster).
2.Per ogni k, si calcola la **inertia totale**.
3.Si rappresentano i valori di inertia in un grafico (asse x: k, asse y: inertia).
**4.Si cerca il "gomito" (elbow)**: il punto dopo il quale l'inertia decresce poco.

**Validazione con il dominio (usando Elbow method)**
- **Non sempre il "gomito" matematico coincide con la scelta più utile per il business!**
- Dopo aver trovato il gomito, è importante chiedersi:
  - "Questo numero di cluster rappresenta segmenti sensati per l'azienda?"
  - "Sono davvero distinguibili e utilizzabili?"

**Conclusione**
- **Elbow Method** aiuta a scegliere "quanto complicare" la segmentazione.
- La scelta finale deve avere senso **sia per la metrica tecnica, sia per chi userà i gruppi** nel mondo reale.

# METRICHE - COS'È IL SILHOUETTE SCORE

**Definizione:**
Il *Silhouette Score* è un indice che misura quanto ogni punto di un dataset è ben assegnato al proprio cluster rispetto agli altri.

**Come funziona:**

- Per ogni punto si calcolano:
  - **a:** la distanza media dagli altri punti del suo cluster
  - **b:** la distanza media dai punti del cluster più vicino
- La silhouette per un punto è:

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

- Valori possibili: da -1 (malissimo) a +1 (perfetto)

**Interpretazione:**

- **Vicino a 1:** punto ben raggruppato
- **Vicino a 0:** punto tra due cluster
- **Negativo:** punto probabilmente assegnato al cluster sbagliato

**Utilizzo:**

- *Silhouette score medio* = media di tutti i punti.
- Più il valore medio è vicino a 1, meglio sono separati i cluster.

# SILHOUETTE SCORE: DAL PUNTO DI VISTA DEL DOMINIO

**Cosa misura il Silhouette Score?**

Indica **quanto ogni elemento (o oggetto)** è ben raggruppato con altri simili e **quanto è lontano dai gruppi diversi**.

Un valore alto significa che il cluster rappresenta **un segmento coeso, riconoscibile e ben separato dagli altri**.

- **Collegamento al business:**
  - Se il silhouette score è **alto** (vicino a 1), significa che i segmenti trovati sono **chiari e distinti**: puoi definire offerte, strategie o comunicazione su misura per ognuno.
  - Se il silhouette score è **basso** (vicino a 0), i gruppi si sovrappongono molto: le differenze tra segmenti non sono nette e non conviene trattarli in modo diverso.
  - Se ci sono molti valori **negativi**, alcuni elementi sono "nel cluster sbagliato": attenzione a non costruire strategie su segmenti troppo "confusi"!
- **Come usarlo nel dominio:**
  - **Prima di lanciare campagne o offerte**: assicurati che i segmenti trovati abbiano un silhouette score almeno discreto.
  - **Analizza i dati "al confine"** tra due gruppi: hanno comportamenti misti? Per esempio, può diventare "power user" o "cliente fedele"?
  - **Un buon silhouette score** aumenta la fiducia che i segmenti rispecchino veri comportamenti del business, non solo artefatti matematici.

# VALIDAZIONE CON IL DOMINIO

**Definizione:**
La validazione con il dominio consiste nel confrontare i risultati dei modelli (come i cluster) con la **conoscenza reale del business** o con la logica del problema trattato.
Non basta una buona metrica numerica: bisogna chiedersi **“questi gruppi hanno senso per chi conosce il settore?”**

**Esempio pratico (e-commerce Olist):**
Se il modello divide i clienti in cluster, bisogna chiedersi:

- I cluster individuati corrispondono a segmenti di clienti che già conosciamo?
- Sono davvero utili per strategie di marketing o vendita?
- Ci sono gruppi strani o insensati secondo la realtà aziendale?

# VALIDAZIONE CON IL DOMINIO

**In pratica:**

1. **Coinvolgere esperti di dominio**
   - Presenta i risultati (ad es. caratteristiche medie dei cluster) a chi conosce il business.
   - Chiedi se i gruppi hanno senso (“Abbiamo davvero molti clienti che spendono poco e pochi power buyer?”).
2. **Confronto con segmentazioni note**
   - Esistono già categorie definite dall’azienda? Il modello le ritrova?
   - Ci sono nuove scoperte utili, o solo rumore?

**3.Analisi delle azioni possibili**

- Puoi creare offerte personalizzate sui cluster trovati?
- I gruppi sono utilizzabili per campagne, assistenza, sconti, ecc.?

**4.Controllo di anomalie**

- Ci sono cluster troppo piccoli, troppo simili tra loro, o senza senso pratico?
- Se un cluster raggruppa clienti molto diversi, forse c’è un errore o serve rivedere le feature.

# DATABASE OLIST E-COMMERCE

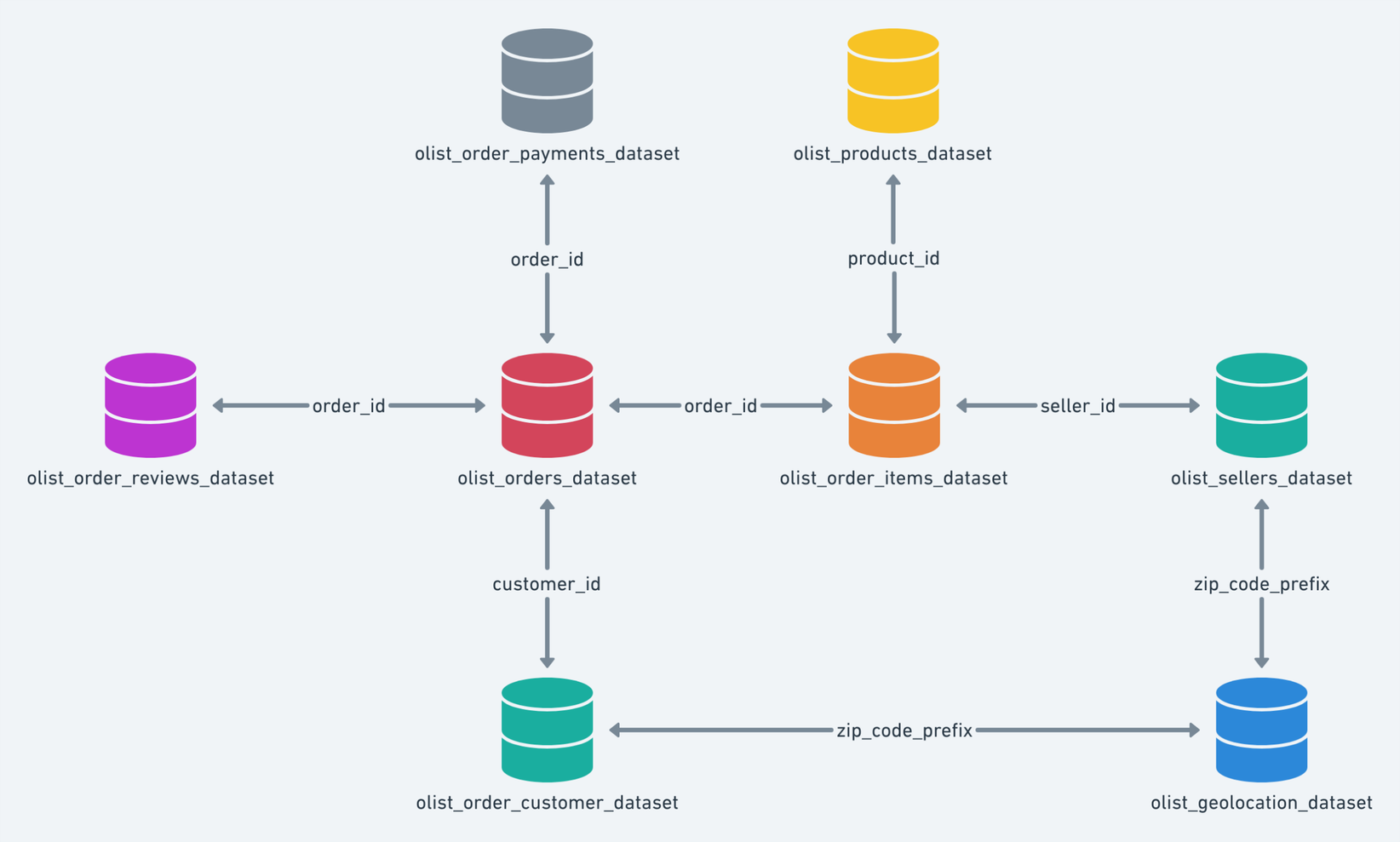
**Cos'è Olist?**

Olist è un dataset pubblico di e-commerce brasiliano che raccoglie **dati reali di acquisti, clienti, prodotti e venditori**. È uno degli esempi più usati per esercizi di machine learning su dati tabellari reali.

**Principali tabelle e cosa contengono**

- **olist_customers_dataset.csv**
  Informazioni sui clienti: ID, città, stato, ZIP code.
- **olist_orders_dataset.csv**
  Dati sugli ordini: ID ordine, ID cliente, data acquisto, stato consegna.
- **olist_order_items_dataset.csv**
  Dettaglio degli articoli per ordine: prodotto, prezzo, quantità, spedizione.
- **olist_products_dataset.csv**
  Dettagli sui prodotti: categoria, nome, descrizione.
- **olist_sellers_dataset.csv**
  Informazioni sui venditori: ID, posizione geografica.
- **olist_geolocation_dataset.csv**
  Dati di posizione geografica (latitudine/longitudine) per analisi spaziali.
- **olist_reviews_dataset.csv**
  Recensioni dei clienti: valutazione, commenti, date.

# DATABASE OLIST E-COMMERCE




**Brazilian E-Commerce Public Dataset by Olist** https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

# DATABASE OLIST E-COMMERCE

**Perché è utile per il machine learning?**

- **Ricco di relazioni**: puoi collegare clienti, ordini, prodotti, venditori.
- **Dati tabellari**: perfetti per esercizi di clustering, classificazione, regressione, visualizzazione.
- **Esempi pratici**:
  - Segmentazione clienti (clustering)
  - Previsione spesa o comportamento di acquisto (regressione)
  - Analisi delle recensioni (sentiment analysis)

**Dimensioni**

- **~100.000 ordini**
- **~99.000 clienti**
- **Decine di migliaia di prodotti e venditori**

**In sintesi:**

Il database Olist offre una base solida e realistica per imparare le principali tecniche di analisi e modellazione dei dati nel mondo reale!

# DOMANDE?

Facciamo gli esercizi!

# INTERPRETAZIONE DEI DATI DI CLUSTERING OLIST

**1. La distribuzione dei cluster**

- Il grafico (es. violin plot) mostra come la **spesa totale** si distribuisce nei vari cluster.
- Ogni cluster rappresenta un “gruppo tipo” di clienti secondo il modello.

**2. Le differenze tra i cluster**

- Cluster “basso”: clienti che spendono poco (magari un solo acquisto o piccoli ordini).
- Cluster “medio”: clienti con spesa intermedia, potenzialmente più fedeli o ricorrenti.
- Cluster “alto”: clienti “power buyer”, cioè chi spende molto (clienti VIP).

**3. Analisi della numerosità dei cluster**

- Se un cluster ha molti clienti ma bassa spesa, è il segmento più comune (tipico nei dati e-commerce).
- Se pochi clienti sono nel cluster ad alta spesa, questi rappresentano il target di valore maggiore.

**4. Uso del silhouette score**

- **Score alto (>0.5):** i cluster sono ben separati, i gruppi hanno senso e possono essere trattati in modo diverso.
- **Score basso (~0):** i cluster si sovrappongono, le differenze sono deboli: meglio ripensare la segmentazione o le feature usate.

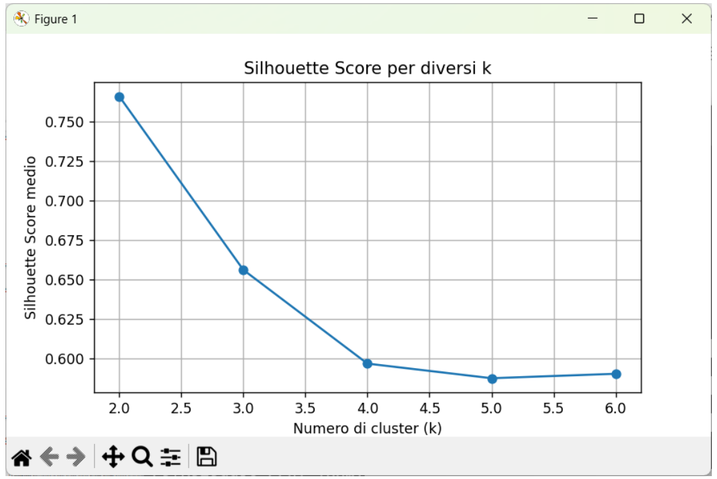
# INTERPRETAZIONE DEI DATI DI CLUSTERING OLIST

**5. Validare con il dominio**

- I gruppi trovati corrispondono a segmenti che l'azienda riconosce? (Es. "clienti occasionali", "clienti medi", "VIP")
- I segmenti sono utili per strategie di marketing, offerte, assistenza?

**6. Prendere azioni**

- Crea offerte personalizzate per i "power buyer".
- Studia come far crescere i clienti dei cluster bassi.
- Monitora i cambiamenti: un cliente può passare da un cluster all'altro nel tempo.




**Interpretazione:**

- La maggior parte dei clienti fa piccoli acquisti (Cluster 0).
- Un piccolo gruppo (Cluster 2) contribuisce in modo sproporzionato al fatturato (power buyer).
- Le strategie possono essere personalizzate per ciascun segmento.

Ottenendo 2 cluster sulla spesa totale:

- **Cluster 0:** spesa media = 30€ (90% clienti)
- **Cluster 1:** spesa media = 120€ (8% clienti)

**In sintesi:**

- Ogni cluster racconta una "storia" di comportamento d'acquisto.
- I cluster aiutano a capire, gestire e valorizzare meglio i clienti.

# COME LEGGERE IL “VIOLINO” (VIOLIN PLOT)

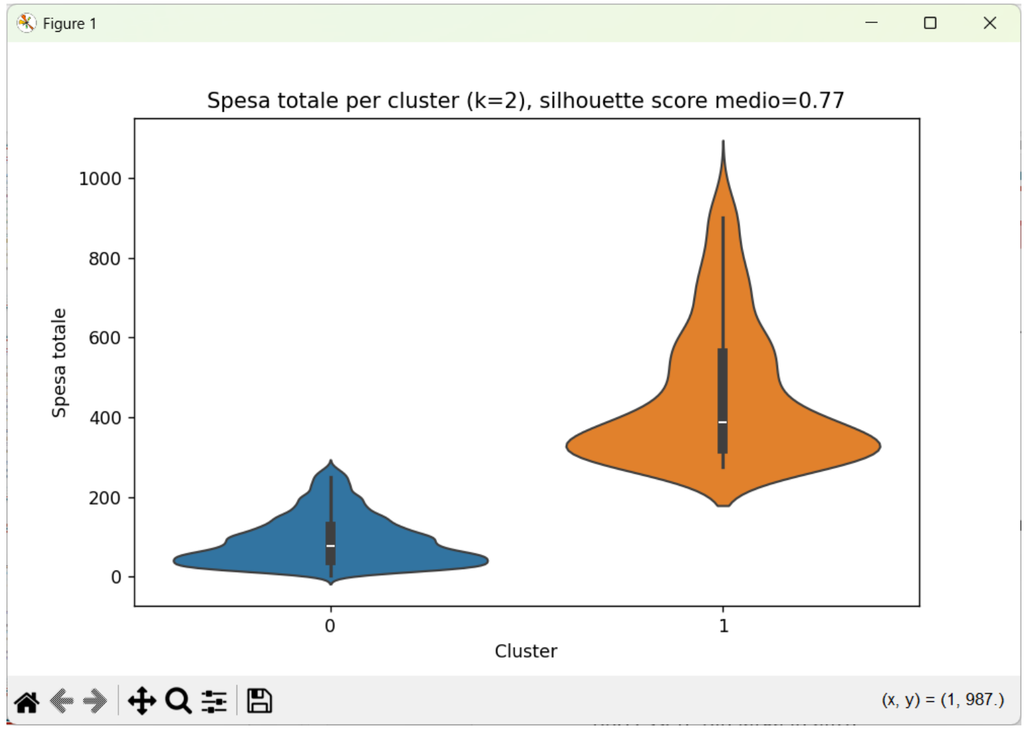
**Definizione**

Un **violin plot** mostra la **distribuzione** di una variabile (es. spesa totale) per ogni cluster trovato.
La **larghezza** del violino in ogni punto indica **quanti clienti** hanno quel livello di spesa:

- Larga = tanti clienti
- Stretta = pochi clienti

**Visualizzazione di spesa totale di Olist**

- **Cluster a sinistra (“base del pino”)**:
  Tanti clienti con **spesa bassa** (la parte larga in basso).
- **Code strette e lunghe verso l’alto**:
  Pochissimi clienti con **spesa alta** (la parte sottile che si allunga).
- **Cluster a destra** (se il modello ne trova):
  Clienti medi o “power buyer” (la distribuzione può essere più larga in alto).



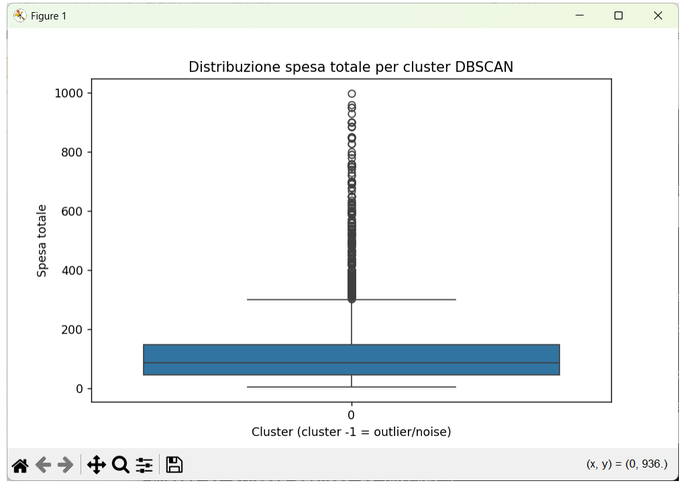
# DBSCAN

**Definizione:**
DBSCAN (Density-Based Spatial Clustering of Applications with Noise) è un algoritmo di clustering che trova gruppi di punti “vicini” tra loro in base alla densità.

**Caratteristica chiave:**
Non serve specificare il numero di cluster a priori; DBSCAN individua automaticamente sia i cluster che gli **outlier** (punti anomali o “noise”).

**Come funziona DBSCAN**

1. **Densità locale:**
   Per ogni punto, conta quanti altri punti si trovano entro una certa distanza (eps).
2. **Core point:**
   Un punto con almeno min_samples vicini è un “core” e dà origine a un cluster.
3. **Espansione:**
   I cluster crescono inglobando tutti i punti densi vicini.
4. **Outlier (noise):**
   I punti che non appartengono a nessun cluster sono etichettati come -1.



# DBSCAN

**Quando usare DBSCAN**

- **Cluster di forma irregolare:**
  Funziona bene anche se i gruppi non sono “rotondi” come in k-Means.
- **Presenza di outlier:**
  Riconosce facilmente i punti anomali, non costringe tutto nei cluster.
- **Non si conosce il numero di cluster:**
  Ideale quando non puoi o non vuoi scegliere il numero di gruppi in partenza.

**In sintesi:**
DBSCAN è una scelta potente quando vuoi trovare gruppi naturali nei dati **senza ipotesi forti sulla forma o sul numero dei cluster** ed è perfetto per scoprire anomalie!

**Parametri principali**

- **eps:**
  Distanza massima per considerare due punti “vicini”.
- **min_samples:**
  Minimo numero di punti per formare un “core” (cioè l’inizio di un cluster).

- **Vantaggi:**
  - Scopre cluster di qualsiasi forma
  - Rileva outlier
  - Non richiede k fisso
- **Limiti:**
  - Sensibile alla scelta di eps e min_samples
  - Non funziona bene se la densità varia molto tra cluster

# CLUSTERING GERARCHICO

**Il clustering gerarchico**

- Algoritmo di clustering che costruisce una **gerarchia di gruppi annidati** (dall'individuo al gruppo più ampio, o viceversa).
- Produce una struttura ad **albero** (dendrogramma) che mostra come i punti/dati si raggruppano a diversi livelli.

**Come funziona**

- **Agglomerativo (bottom-up):**
  Ogni punto inizia come un cluster singolo, poi i cluster più simili vengono fusi ripetutamente fino a formare un solo grande gruppo.
- **Divisivo (top-down):**
  Si parte da tutti i punti in un unico cluster e si divide ricorsivamente.

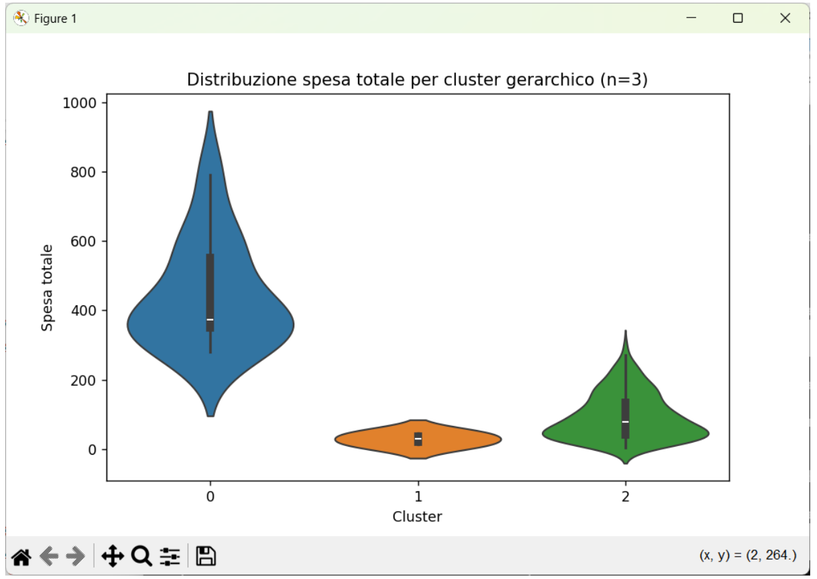
**Caratteristiche principali**

- **Non serve scegliere k** all'inizio: puoi "tagliare" l'albero a diversi livelli, ottenendo quanti cluster vuoi.
- **Visualizzazione:**
  Il dendrogramma aiuta a capire la struttura interna dei dati e i livelli di similarità tra i gruppi.

# CLUSTERING GERARCHICO

**Quando usarlo**

•Quando vuoi **esplorare come cambiano i gruppi** a diversi livelli di dettaglio.
•Per scoprire sottogruppi o relazioni annidate tra dati.
•Per dataset di dimensioni **medio-piccole** (è meno efficiente di k-Means su grandi dati).




**Vantaggi**

- Visualizza la “storia” della formazione dei gruppi.
- Non richiede di fissare il numero di cluster a priori.
- Utile per scoprire sottogruppi “nascosti”.

**Limiti**

- Scalabilità: lento su dataset grandi.
- Sensibile alla metrica di distanza scelta (euclidea, manhattan, ecc.).

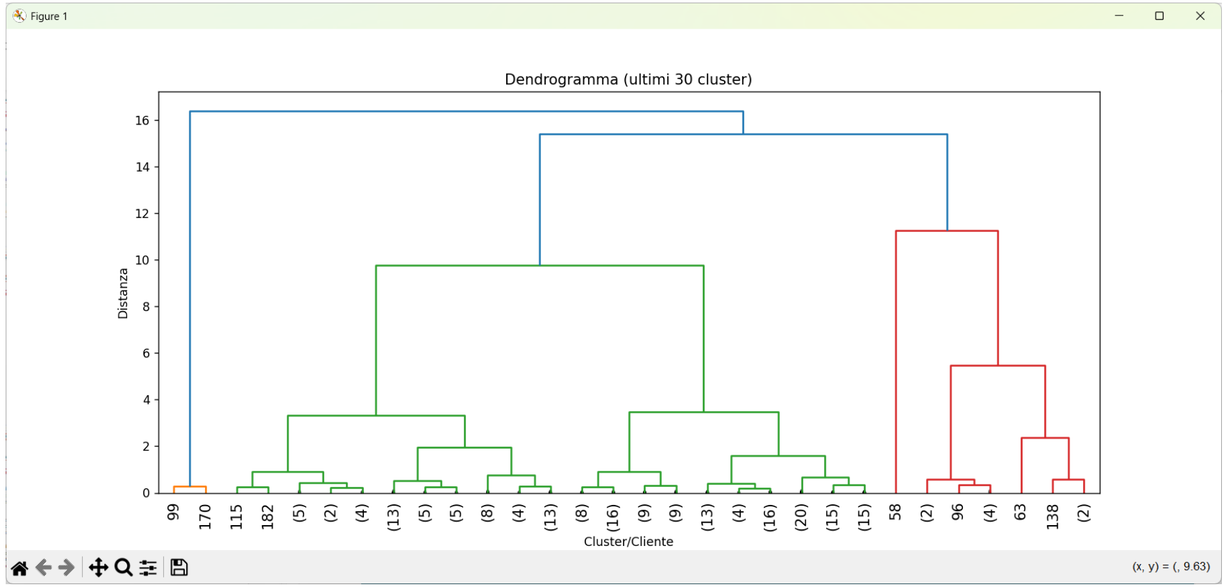
**In sintesi**

Il clustering gerarchico è ideale per **analizzare relazioni a più livelli**, ottenere una panoramica completa dei dati e comunicare la struttura dei gruppi tramite dendrogramma.

# CLUSTERING GERARCHICO

**Dendrogramma (albero delle fusioni tra cluster)**

- **Cosa mostra:**
  Il dendrogramma è una **rappresentazione ad albero** di come i clienti vengono raggruppati man mano che il criterio di similarità diventa più “largo”.
- **Come leggerlo:**
  - **Le foglie** in basso sono i singoli clienti.
  - **I rami** mostrano quando due gruppi si fondono: più il ramo è alto, più sono diversi tra loro i gruppi che si fondono.
  - **Tagliando l’albero** a un certo livello verticale (“altezza” del ramo) scegli quanti cluster ottenere (es: con 3 cluster, taglia dove hai 3 grandi rami).
- **Utilità pratica:**
  Aiuta a vedere se i cluster sono ben separati, se ci sono sottogruppi naturali e a scegliere “a posteriori” il numero ottimale di cluster.



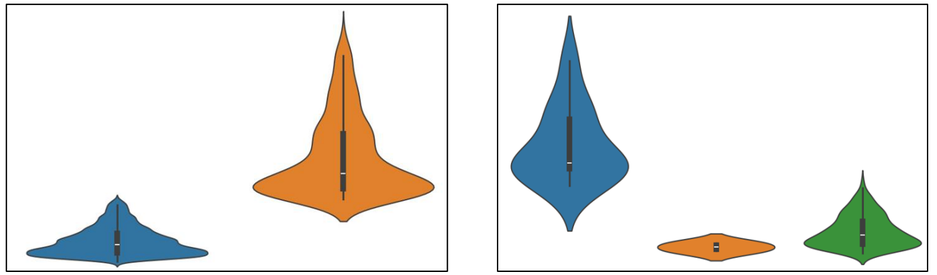
# PERCHÉ SONO DIVERSI K-MEANS E GERARCHICO

**Criteri di raggruppamento diversi**

- **k-Means**:
  Divide i dati in gruppi “rotondi” e di dimensione simile, minimizzando la distanza dai centroidi.
  Funziona meglio se i dati sono “ben separati” e i gruppi sono compatti e omogenei.
- **Gerarchico (agglomerativo)**:
  Unisce i punti e i gruppi **secondo la similarità locale** passo dopo passo.
  I gruppi possono avere **forme e dimensioni molto diverse**, e i confini non sono sempre “rotondi”.

**Sensibilità alla struttura dei dati**

- k-Means “forza” i dati in cluster di grandezza simile anche quando i veri gruppi sono sbilanciati.
- Gerarchico rispetta la distribuzione naturale dei dati:
  - Se c’è un sottogruppo molto compatto, lo lascia a parte.
  - Se c’è un gruppo molto “sparso”, può includerlo in un cluster più grande.



K-means Gerarchico

**In sintesi**
**I cluster sono diversi perché i due algoritmi “vedono” e dividono i dati con logiche molto differenti.**
Per questo i “violini” risultanti possono avere **forme e ampiezze molto diverse** tra k-Means e gerarchico.

# CONFRONTO PRATICO TRA ALGORITMI DI CLUSTERING

| Aspetto | KMeans | DBSCAN | Gerarchico |
|---|---|---|---|
| **Velocità** | Molto veloce su dataset medi e grandi | Lento su dataset grandi, ma scalabile con ottimizzazione | Lento su dataset grandi |
| **Forma dei cluster** | Solo cluster “rotondi” (simili, isotropi) | Qualsiasi forma, anche irregolare | Qualsiasi forma |
| **Numero cluster** | Va scelto a priori (k) | Determinato automaticamente (densità) | Determinato dal taglio nel dendrogramma |
| **Sensibilità outlier** | Sensibile agli outlier | Robusto agli outlier | Può gestire outlier, ma dipende da metodo |
| **Scalabilità** | Eccellente | Buona con dataset medi, meno su dataset enormi | Bassa |
| **Parametri chiave** | k (numero cluster) | epsilon, min_samples | Metodo di linkage, distanza |
| **Interpretabilità** | Alta (centroidi visibili) | Media (definisce solo aree dense) | Buona (albero/dendrogramma) |
| **Quando usarlo** | Cluster “regolari”, dati numerici | Cluster di forma arbitraria, dati rumorosi | Visualizzare gerarchie o similarità |
| **Limiti** | Non gestisce bene cluster allungati o rumore | Difficile scegliere epsilon giusto | Molto lento su dati grandi |

# CLUSTERING – SOMMARIO

**Il clustering**
È una tecnica di machine learning **non supervisionata** che serve a **scoprire gruppi simili** (cluster) all'interno dei dati, senza sapere in anticipo quali siano.

**Serve a**
- Segmentare clienti, prodotti, documenti, ecc.
- Individuare pattern, anomalie o gruppi nascosti nei dati
- Precedere strategie di marketing, personalizzazione, prevenzione frodi, ecc.

**Algoritmi principali**
- **k-Means:** semplice, veloce, richiede di scegliere il numero di cluster k
- **DBSCAN:** trova cluster di forma qualunque e identifica outlier, non richiede k
- **Gerarchici:** costruiscono una gerarchia di gruppi (alberi, dendrogrammi)

# CLUSTERING – RACCOMANDAZIONI

- **Prepara i dati:** scegli e aggrega le feature più rilevanti
- **Riduci la dimensionalità** se necessario (PCA, t-SNE…) per semplificare l'analisi e migliorare la visualizzazione
- **Prova più algoritmi** e confronta i risultati (non esiste "il migliore" a priori)
- **Valida sempre i cluster:** usa silhouette score, elbow method, e verifica con la logica del dominio (il business deve riconoscere i gruppi!)
- **Visualizza i risultati:** violin plot, scatter plot, heatmap… rendono i gruppi interpretabili e comunicabili

**In sintesi**

Il clustering è uno strumento potente per **scoprire il significato nascosto nei dati**, ma ha senso solo se i gruppi trovati sono **chiari e utili per chi deve decidere**!

# DOMANDE?

PAUSA

# RIDUZIONE DI DIMENSIONALITÀ: COS'È E PERCHÉ SERVE

**Definizione:**
La riduzione di dimensionalità (dimensionality reduction) è una tecnica che trasforma i dati ad alta dimensione (molte feature) in uno spazio più piccolo, mantenendo l'informazione essenziale.

**Obiettivo:**

- **Semplificare**: rendere i dati più leggibili e veloci da analizzare
- **Visualizzare**: vedere pattern nascosti in 2D/3D
- **Eliminare ridondanze**: rimuovere feature poco utili o collegate tra loro

**Quando usarla:**

- Quando i dati hanno molte colonne/features
- Prima del clustering, per migliorare i risultati
- Per visualizzare i dati su grafici comprensibili

# PERCHÉ USARE LA RIDUZIONE DIMENSIONALE SU OLIST

**Motivo pratico:**
Il dataset Olist contiene molte variabili (spesa, ordini, categorie, tempi…) che spesso sono correlate tra loro.
Ridurre la dimensionalità (con tecniche come PCA, t-SNE, UMAP):

- **Aiuta a visualizzare** meglio i dati e i cluster (2D/3D)
- **Elimina ridondanze** e semplifica l'analisi
- **Velocizza i calcoli** per algoritmi che soffrono l'"effetto del grande numero di variabili"
- **Migliora il clustering** quando solo alcune combinazioni di variabili separano bene i gruppi

**Motivo didattico:**
Permette di capire quali **caratteristiche contano davvero** per distinguere i clienti, e di **spiegare meglio** i risultati del modello.

# PRINCIPALI TECNICHE DI RIDUZIONE DIMENSIONALE

**PCA (Principal Component Analysis):**

- Trasforma i dati in un nuovo sistema di assi che spiega la maggior parte della “varianza”
- Usa combinazioni lineari delle feature originali
- Ideale per dati numerici e correlati

**t-SNE:**

- Mappa dati complessi in 2D/3D preservando le relazioni di vicinanza
- Ottimo per visualizzare gruppi naturali, ma meno per uso operativo

**UMAP:**

- Alternativa moderna a t-SNE
- Più veloce e mantiene meglio la struttura globale e locale dei dati

**Nota:**

Queste tecniche aiutano sia nell’analisi esplorativa sia nel pre-processamento di clustering.

# COS'È UNO SPAZIO MULTIDIMENSIONALE

**Spazio multidimensionale**
È un ambiente matematico in cui ogni elemento è descritto da più valori, uno per ogni "dimensione".

**Esempio pratico:** in un dataset di clienti, ogni dimensione può rappresentare una caratteristica (es. spesa, ordini, categorie).

**A cosa serve?**
Permette di rappresentare dati complessi (non solo punti su una retta, ma in uno spazio a molte dimensioni) e di calcolare distanze, vicinanze, raggruppamenti.

**Perché è importante nel machine learning?**
La maggior parte degli algoritmi lavora in spazi multidimensionali: ogni esempio (riga del dataset) è un punto in uno spazio con tante dimensioni quanti sono gli attributi/feature.

# COS'È UNO SPAZIO MULTIDIMENSIONALE

**Nota:**
In seguito useremo il concetto di **spazio** per introdurre idee come "vettorizzazione" e "encoding", cioè il modo in cui dati reali (testi, immagini…) vengono trasformati in punti di uno spazio numerico su cui i modelli possono lavorare.

**Esempio pratico:**

- 2D: una mappa (x = latitudine, y = longitudine)
- 3D: spazio fisico (x, y, z)
- Più di 3D: ogni "feature" (colonna del dataset) è una dimensione.

**Esempio:** ogni cliente è un punto nello spazio con coordinate = [num_ordini, spesa_totale, num_categorie, ...].

# MATRICE

**Definizione:**
Una matrice è una tabella di numeri, organizzata in righe e colonne.

**Ogni riga è un vettore (punto nel tuo spazio)**
**Ogni colonna è una feature (una dimensione)**

**Esempio:**

| num_ordini | spesa_totale | num_categorie |
|---|---|---|
| 2 | 100 | 3 |
| 1 | 40 | 1 |
| 5 | 220 | 2 |

**In Python:**

```
import numpy as np
M = np.array([
    [2, 100, 3],
    [1, 40, 1],
    [5, 220, 2]
])
```

Qui, la matrice rappresenta **3 clienti** in uno spazio a **3 dimensioni**.

# TENSORE

**Definizione:**

Un tensore è una **struttura dati multidimensionale**.

- Vettore = tensore di ordine 1 (1D)
- Matrice = tensore di ordine 2 (2D)
- Tensore vero e proprio = 3D, 4D, ... (es: immagini, serie temporali, batch di dati)

**Esempi pratici:**

- **Immagini**: matrice di pixel (2D), più i canali colore (RGB = 3D)
- **Batch di dati**: matrice di vettori (ogni batch = 3D)
- **Video**: sequenza di immagini (4D: altezza × larghezza × canali × tempo)

**In Python:**

```
# Tensore 3D: batch di 10 immagini RGB 32x32
import numpy as np
T = np.random.rand(10, 32, 32, 3)
```

# RIEPILOGO RAPIDO

| Nome | Esempio | Dimensioni | Utilizzo comune |
|---|---|---|---|
| Punto | (2, 100, 3) | 1 | Singolo dato/cliente |
| Vettore | [2, 100, 3] | 1D | Feature di un esempio |
| Matrice | [[2,100,3], ... ] | 2D | Dataset tabellare |
| Tensore | (10,32,32,3) | 3D, 4D, 5D, … | Immagini, video, batch |

- **Lo spazio multidimensionale è il "mondo" astratto in cui i modelli di machine learning lavorano.**
- Ogni dato reale (persona, prodotto, evento) viene rappresentato come un punto in questo spazio, dove **ogni dimensione corrisponde a una caratteristica** (feature) misurata.
- **Tutte le operazioni fondamentali** (calcolare distanze, trovare gruppi, separare categorie, stimare valori) avvengono tra punti in questo spazio.
- **Il compito del modello** è trovare regole e pattern geometrici in questo spazio:
  - Scoprire se i punti si raggruppano (clustering)
  - Trovare le frontiere che separano classi (classificazione)
  - Capire le direzioni principali di variazione (riduzione dimensionale)
- **L'efficacia del modello** dipende da come i dati sono rappresentati in questo spazio e da quanto le informazioni utili risultano "visibili" ai suoi algoritmi.

# PCA (PRINCIPAL COMPONENT ANALYSIS)

**Definizione**
Tecnica di **riduzione dimensionale** che trasforma le feature originali in nuove “componenti principali”, cioè combinazioni lineari che spiegano la massima varianza dei dati.

**A cosa serve:**
- Semplifica dataset complessi
- Permette di **visualizzare dati multidimensionali** (in 2D/3D)
- Migliora il clustering eliminando rumore e ridondanze
- Riduce il rischio di overfitting

**Come funziona:**
- Trova le direzioni (assi) dove i dati “si distribuiscono di più”
- Ordina queste direzioni per importanza (varianza spiegata)
- Puoi scegliere quante componenti mantenere (es: 2 per visualizzare)

**Limiti:**
- Perde interpretabilità delle feature originali
- Funziona solo su variabili numeriche e scalate
- Non cattura relazioni non lineari

**Quando usarla:**
- Prima del clustering su dataset ricchi di variabili
- Per esplorare pattern nascosti o semplificare modelli
- Per accelerare calcoli e visualizzazioni

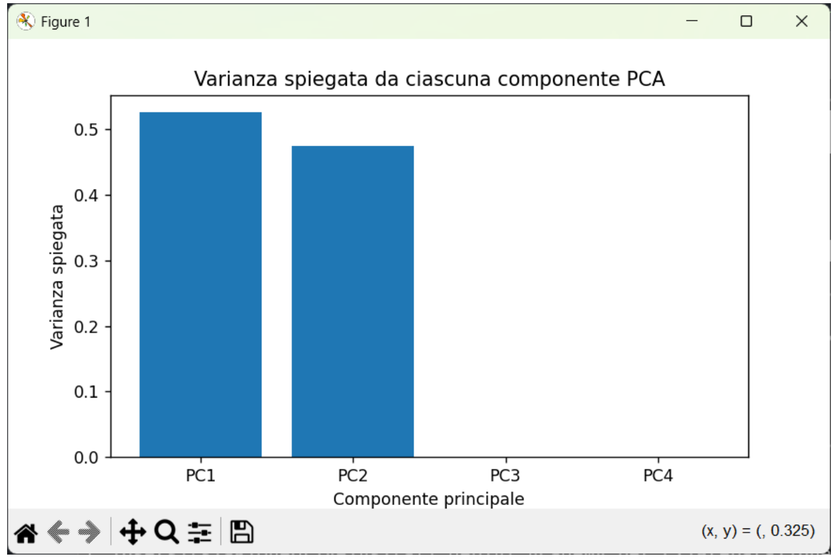
# PCA IN AZIONE

**Quali feature ridurre?**

Le feature più comuni e utili per clustering clienti su Olist sono:

- PC1: total_spent (spesa totale)
- PC2: num_orders (numero ordini)
- PC3: num_categories (categorie diverse acquistate)
- PC4: activity_days (giorni di attività)

Queste feature sono spesso **correlate**: la PCA aiuta a sintetizzarle e capire quali “dimensioni latenti” distinguono davvero i clienti.




**In dettaglio:**

- **PC1 (prima componente principale):**
  - È la direzione che spiega **la maggior parte della varianza** nei dati.
  - I dati “si sparpagliano” di più lungo questa direzione.
- **PC2 (seconda componente):**
  - È la direzione **ortogonale** (indipendente) a PC1 che spiega la maggior parte della varianza **residua**.
- **PC3, PC4, ...**
  - Sono ulteriori direzioni, sempre ortogonali alle precedenti, che spiegano la varianza residua, spesso molto meno rilevante.
  - Se hai solo 4 variabili originali, puoi avere massimo 4 componenti principali.

# T-SNE

**t-SNE (t-distributed Stochastic Neighbor Embedding)**
**Definizione**
Una tecnica avanzata di **riduzione dimensionale** che serve soprattutto per **visualizzare dati complessi e multidimensionali** in 2D o 3D.

**Come funziona:**

- Mantiene le relazioni di vicinanza: punti “vicini” nello spazio originale restano vicini nel nuovo spazio.
- Usa una trasformazione non lineare per evidenziare cluster e pattern nascosti.

**Cosa fa vedere:**
Gruppi, strutture, anomalie e sottogruppi che in altre proiezioni potrebbero essere nascosti.

**Quando usarlo:**
Per **visualizzare** cluster e relazioni tra dati complessi (es: dati clienti, immagini, parole) quando PCA o altre tecniche non bastano.

**Vantaggi:**

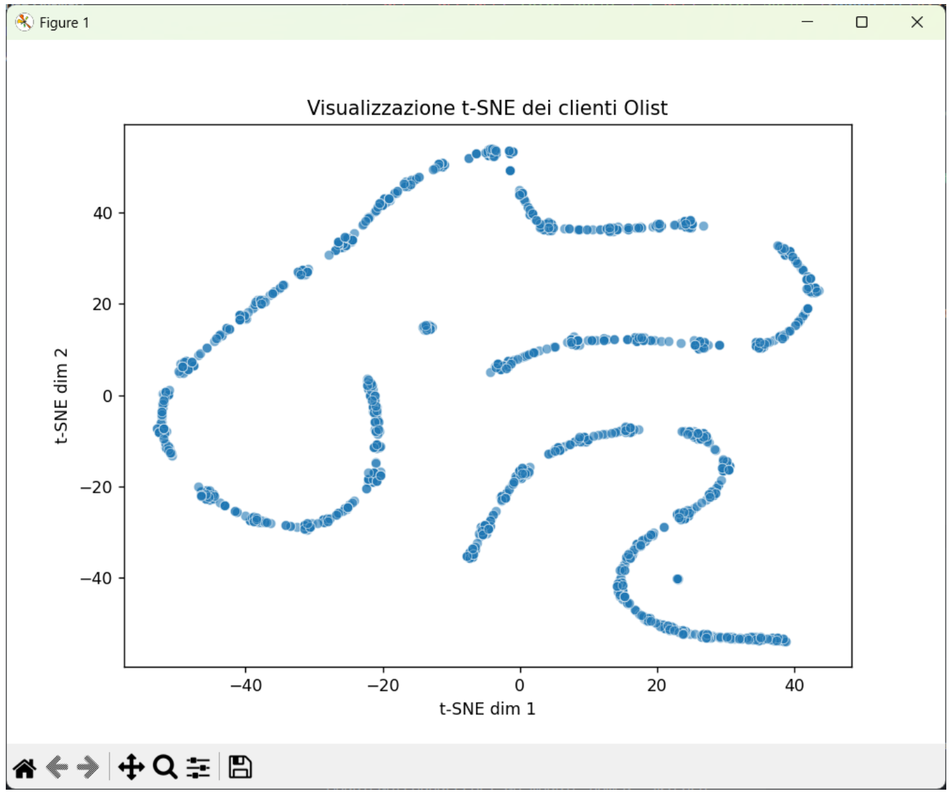
- Ottimo per scoprire visivamente **cluster naturali**
- Non richiede che i dati siano lineari o ben separati

**Limiti:**

- **Solo per visualizzazione**, non per modellazione o predizione
- Lento su dataset grandi (>10.000 punti)
- Parametri sensibili (perplexity, learning rate…)
- L’output cambia ogni volta a meno che non si imposti il seed

# "DIM 1" E "DIM 2" NEI GRAFICI T-SNE

- **"dim 1"** e **"dim 2"** sono le **nuove dimensioni** (“coordinate”) generate dall’algoritmo di riduzione dimensionale.
- Ogni punto del dataset originale (che aveva tante variabili: spesa, ordini, categorie, ecc.)
  viene **“proiettato” in un piano 2D** (due assi) dove:
  - **dim 1**: la prima dimensione calcolata da t-SNE
  - **dim 2**: la seconda dimensione calcolata da t-SNE




**Cosa rappresentano?**

- **Non sono più le feature originali:**
  Sono **combinazioni delle variabili originali** calcolate dall’algoritmo per *preservare* le relazioni di vicinanza/similarità tra i dati.

- **Permettono di visualizzare**:
  - I punti “vicini” nel grafico corrispondono a clienti “simili” nei dati reali.
  - Gruppi o pattern visibili sul grafico riflettono reali cluster o relazioni nascoste nel dataset originale.

# COME LEGGERE E INTERPRETARE LE CURVE DI T-SNE

**I punti**

- **Un punto = un cliente** (nel nostro caso Olist), cioè una riga della tabella aggregata.
- La sua posizione (dim 1, dim 2) è decisa dal valore di tutte le feature originali dopo la trasformazione t-SNE.

**Curve: transizioni graduali**

- Una curva indica che **non ci sono "salti" tra gruppi**, ma che i dati passano "morbidi" da un comportamento a un altro.
- I punti all'inizio della curva rappresentano un tipo di cliente; quelli in fondo, un altro; nel mezzo, i clienti "di transizione".

**Distanza: somiglianza**

- **Punti vicini sulla curva**: clienti con caratteristiche molto simili (es: stessa fascia di spesa, stessi comportamenti d'acquisto).
- **Punti lontani**: clienti sempre meno simili man mano che ti sposti lungo la curva.

**Possibili biforcazioni**

- Alcuni rami si dividono (ramificazione, biforcazione):
  questo indica che dopo un certo livello di comportamento d'acquisto, **i clienti si "specializzano" in almeno due stili diversi**.

# UMAP

**UMAP (Uniform Manifold Approximation and Projection)**

**Definizione**
Un'algoritmo di **riduzione dimensionale** moderno, pensato per visualizzare e analizzare dati multidimensionali, simile a t-SNE ma spesso più veloce ed efficiente.

**A cosa serve:**

- Visualizzare dati complessi (2D/3D)
- Esplorare e scoprire cluster o pattern nascosti
- Preparare dati per altri algoritmi (clustering, classificazione)

**Come funziona:**

- Ricostruisce la "struttura locale" dei dati (chi è vicino a chi)
- Proietta i dati in uno spazio a 2 o 3 dimensioni, preservando sia la vicinanza locale che (meglio di t-SNE) alcune relazioni globali

# UMAP

**Vantaggi:**

- **Molto veloce** anche su dataset grandi (>10.000 punti)
- Permette la **ricostruzione** dei dati originali (funzione inversa)
- Preserva sia la struttura locale che quella globale meglio di t-SNE
- Output più “stabile” rispetto a t-SNE (ripetibile)

**Limiti:**

- Non sempre interpretabile direttamente (assi artificiali)
- Come t-SNE, serve per **visualizzare e capire**, non per predire
- Parametri (n_neighbors, min_dist) possono influenzare molto il risultato

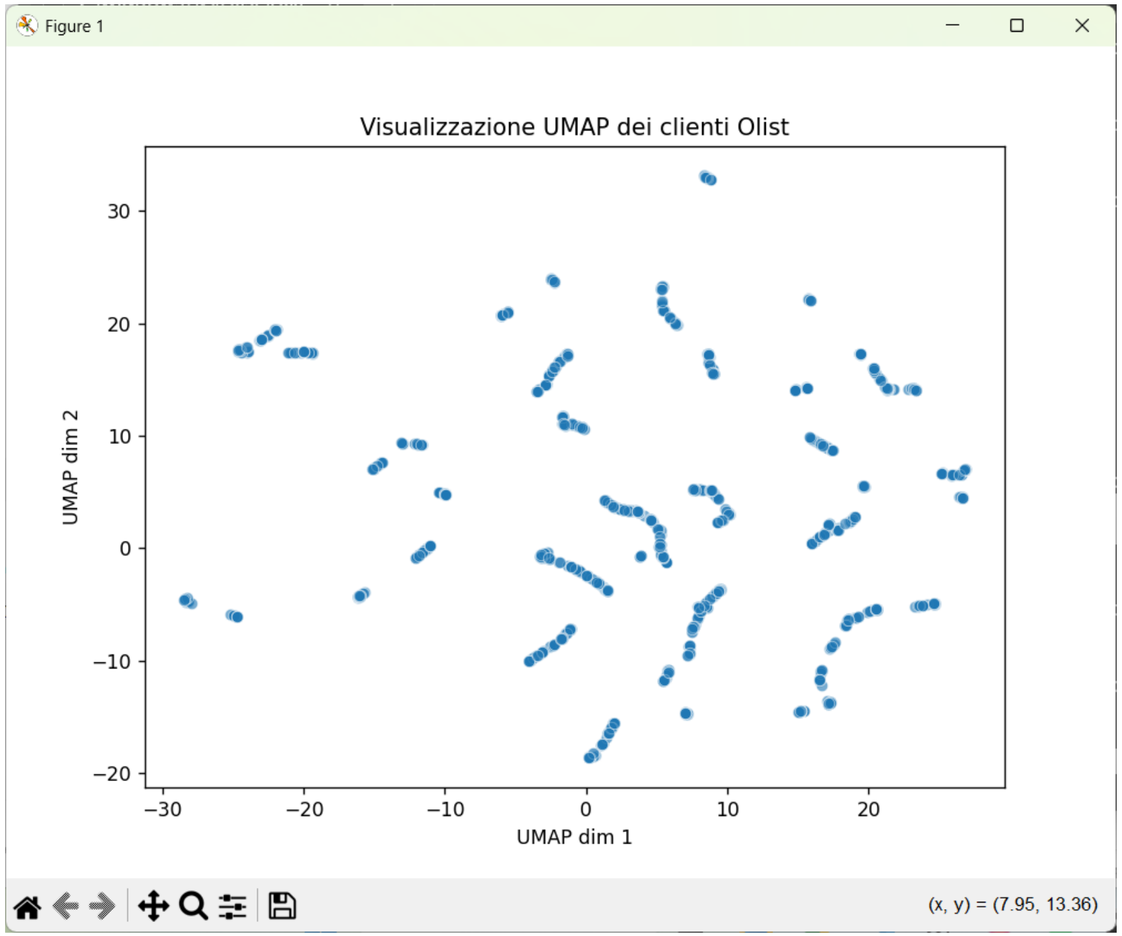
**Quando usarlo:**

- Visualizzazione rapida di grandi dataset
- Analisi esplorativa per scoprire pattern e cluster
- Preprocessing per clustering avanzato

# COME INTERPRETARE IL GRAFICO UMAP

**Cosa rappresenta ogni punto**

- Ogni **punto blu** è un **cliente** Olist, posizionato in base alle sue caratteristiche (spesa totale, numero ordini, categorie, giorni attività), **riassunte** in due dimensioni artificiali (UMAP dim 1 e UMAP dim 2).




**Struttura del grafico**

- **Rispetto a t-SNE**, i punti qui sono più **sparsi e separati**.
- Si osservano diversi **piccoli gruppi**, “rametti” o “filamenti” di punti, senza curve lunghe e continue.
- Manca una divisione netta in cluster grandi, ma si notano **micro-cluster**, gruppetti molto compatti di clienti con profili simili.

# COME INTERPRETARE IL GRAFICO UMAP

- **Distanze e separazione**:
  Se due gruppi sono distanti, significa che sono clienti con **profili nettamente diversi** (magari alcuni sono molto attivi, altri pochissimo, ecc.).
- **Gruppi locali**:
  Dove vedi rametti o piccoli addensamenti, **i clienti lì sono molto simili tra loro**.
  Può trattarsi di profili simili per spesa/ordini/attività.
- **Assenza di "rami continui"**:
  UMAP spesso evidenzia i **micro-cluster** meglio di t-SNE.
  Le transizioni graduali sono meno visibili, ma si può individuare più facilmente "nodi" o "tipi base" di clientela.

**Confronto con t-SNE**

- **t-SNE** mette in evidenza le transizioni continue (curve, archi, rami lunghi).
- **UMAP** evidenzia di più **i piccoli gruppi stabili**, mostrando dove i clienti sono veramente simili e facilmente separabili dagli altri.
- In UMAP il rischio di trovare cluster naturali è **più alto** se vuoi segmentare clienti con pochi dati e differenze nette.

# SOMMARIO - RIDUZIONE DIMENSIONALE

La **riduzione dimensionale** è uno strumento fondamentale dell'analisi dati moderna:

- permette di **trasformare dati complessi e multidimensionali** in una forma più semplice e interpretabile,
- facilitando l'identificazione di pattern, gruppi e relazioni nascoste.

Applicando metodi come **PCA, t-SNE e UMAP** possiamo non solo velocizzare i calcoli e migliorare i risultati dei modelli, ma soprattutto **rendere visibile ciò che prima era invisibile**.

In ogni progetto, la riduzione dimensionale aiuta a vedere i dati da una nuova prospettiva, semplificando le decisioni, la comunicazione e la scoperta di valore reale.

# DIFFERENZA TRA MODELLI

StandardScaler, PCA e simili sono considerati “modelli” nel senso di oggetti che imparano dai dati, ma NON sono modelli predittivi.

**Differenza chiave:**

- **Modelli predittivi:**
  - **Per esempio:** Linear Regression, Random Forest, XGBoost, ecc.
  - Usano `.fit()` per imparare e poi `.predict()` per produrre risultati (previsioni, classificazioni, ecc.)
  - **Imparano dai dati**
  - Si usano per: regressione, classificazione, ecc.
- **Modelli di trasformazione:**
  - Es: StandardScaler, PCA, OneHotEncoder, ecc.
  - Usano `.fit_trasform()` per imparare come trasformare i dati e .transform() per applicare la trasformazione
  - **Trasformano dati.**
  - Si usano per modificare dati.

# CONFRONTO TRA .FIT() E .FIT_TRANSFORM()

| Aspetto | .fit() | .fit_transform() |
|---|---|---|
| **Cosa fa** | Calcola i parametri del modello sull'intero dataset (ad es. media/varianza per StandardScaler, componenti per PCA, ecc.) | Calcola i parametri **e** applica subito la trasformazione ai dati |
| **Output** | Il modello "addestrato" (ma i dati NON sono trasformati) | I dati trasformati (array o DataFrame) |
| **Quando si usa** | Se vuoi solo calcolare i parametri e applicare la trasformazione dopo (es. per train/test) | Se vuoi fare tutto in una volta sola |
| **Esempio pratico** | scaler.fit(X_train) | scaler.fit_transform(X_train) |
| | X_test_std = scaler.transform(X_test) | |
| **Utilità** | Necessario quando separi training e test per evitare "data leakage" | Comodo per la fase esplorativa o quando lavori su tutti i dati assieme |

# DOMANDE?

PAUSA PRANZO

# AUTOENCODER – INTRODUZIONE

**Definizione**

Un **autoencoder** è una **rete neurale** progettata per imparare a comprimere (codificare) i dati in uno spazio più piccolo (rappresentazione compatta) e poi ricostruirli (decodifica) il più fedelmente possibile.

**A cosa servono?**

- **Riduzione dimensionale** automatica e non lineare (alternativa "deep" a PCA/t-SNE/UMAP)
- **Denoising:** rimuovere rumore dai dati
- **Compressione** dati (ad esempio immagini)
- **Pre-elaborazione** per clustering o altre analisi
- **Anomaly detection:** trovano ciò che è "strano" o difficile da ricostruire

**Vantaggi**

- Gestiscono anche relazioni non lineari molto complesse
- Possono essere adattati facilmente a immagini, testi, dati tabellari

**Limiti**

- Possono sovra-adattarsi (overfitting) su dati piccoli
- Richiedono tuning e dati numerici

# AUTOENCODER – PRINCIPI

**Struttura**:

- Rete neurale composta da **due parti**:
  **Encoder** (riduce le dimensioni dei dati) e **Decoder** (ricostruisce i dati originali).
- Punto centrale ("**bottleneck**"): qui il dato è compresso al massimo.

**Funzionamento**:

- L'autoencoder cerca di **"copiare" l'input in output** passando per il bottleneck.
- L'addestramento avviene **minimizzando la differenza tra input e output** (loss = errore di ricostruzione).
- La rappresentazione compressa nel bottleneck contiene le informazioni più importanti dei dati.

**Tipo di apprendimento**:

- **Non supervisionato:**
  non richiede etichette (usa solo i dati "così come sono").

# AUTOENCODER – USO PRATICO

**Riduzione dimensionale avanzata**

- Gli autoencoder imparano una rappresentazione compatta, anche per dati non lineari (oltre i limiti di PCA).
- Puoi usare la codifica del bottleneck come “nuove feature” per clustering o visualizzazione.

**Denoising (pulizia dati)**

- Se addestrati su dati puliti, riescono a ricostruire input “sporchi” ripulendoli (es: immagini con rumore).

**Compressione**

- Le rappresentazioni bottleneck possono essere salvate per comprimere i dati in modo efficiente.

**Anomaly detection**

- Gli autoencoder ricostruiscono bene i dati “normali”, ma male quelli “strani”:
  l’errore di ricostruzione alto segnala un’anomalia.

**Preprocessing**

- Possono preparare dati per altri modelli, creando feature più “essenziali”.

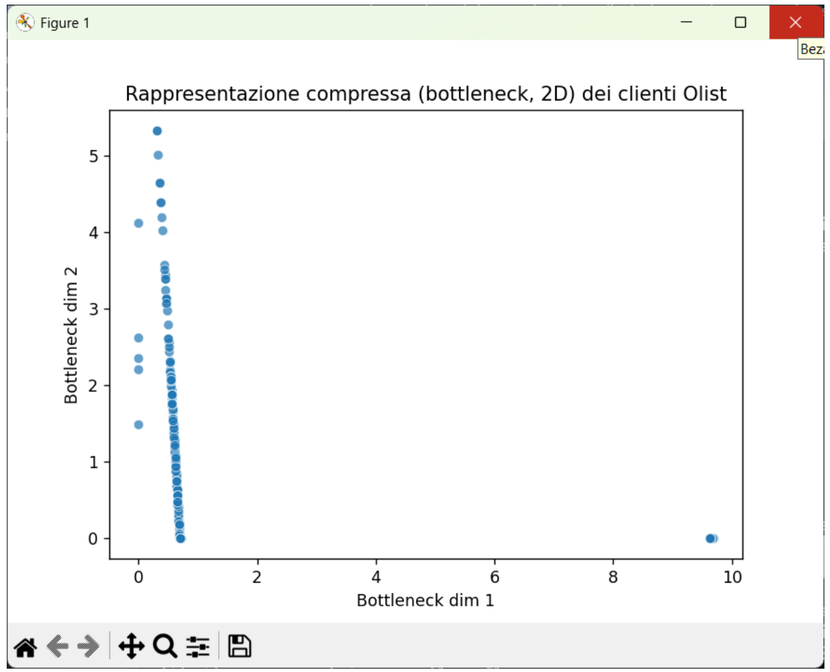
# INTERPRETAZIONE DEL GRAFICO BOTTLENECK (2D)

**Ogni punto = un cliente**

- Ogni punto blu rappresenta un cliente, visto nello spazio compresso a 2 dimensioni appreso dall'autoencoder.

**Distribuzione dei punti**

- **Quasi tutti i clienti** si trovano molto vicini tra loro, con valori di "Bottleneck dim 1" molto bassi (tra 0 e 2) e una dispersione in "Bottleneck dim 2" (da 0 a 5).
- **Un solo punto** ("outlier") si trova molto distante dagli altri (circa a 10 su dim 1, vicino a 0 su dim 2):
  questo è chiaramente un **cliente anomalo** rispetto al resto.




```
32/32 [==============================] - 0s 530us/step
32/32 [==============================] - 0s 797us/step
Errore medio di ricostruzione: 0.233
Dati originali (standardizzati) e ricostruiti dai primi 5 clienti:
 total_spent  num_orders  num_categories  activity_days  total_spent_recon  num_orders_recon
num_categories_recon  activity_days_recon
   0.594296         0.0        0.070888            0.0           0.443122         -0.072948
0.028890              0.011811
  -0.623674         0.0        0.070888            0.0          -0.352666          0.107591
0.066802              0.010833
   0.341924         0.0        0.070888            0.0           0.229645         -0.024517
0.039060              0.011548
   1.100729         0.0        0.070888            0.0           0.871505         -0.170133
0.008482
```

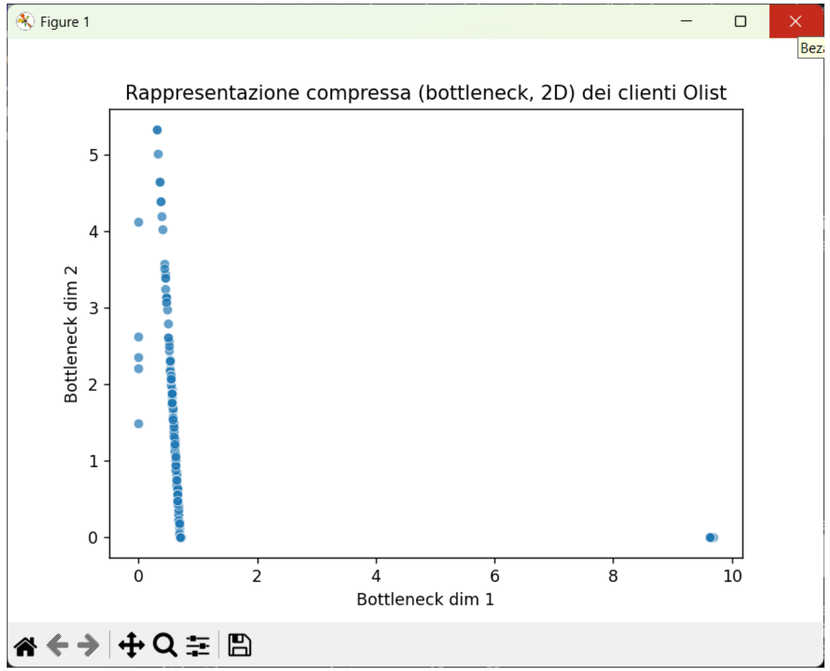
# INTERPRETAZIONE DEL GRAFICO BOTTLENECK (2D)

**Ogni punto = un cliente**

- Ogni punto blu rappresenta un cliente, visto nello spazio compresso a 2 dimensioni appreso dall'autoencoder.

**Distribuzione dei punti**

- **Quasi tutti i clienti** si trovano molto vicini tra loro, con valori di "Bottleneck dim 1" molto bassi (tra 0 e 2) e una dispersione in "Bottleneck dim 2" (da 0 a 5).
- **Un solo punto** ("outlier") si trova molto distante dagli altri (circa a 10 su dim 1, vicino a 0 su dim 2):
  questo è chiaramente un **cliente anomalo** rispetto al resto.



# SSS

**Cosa significa questa forma**

- **Cluster unico molto compatto**:
  l'autoencoder ha trovato che quasi tutti i clienti hanno caratteristiche simili e possono essere compressi in uno stesso "gruppo".
- **Outlier evidente**:
  il punto lontano mostra un cliente i cui dati sono così diversi che il modello lo isola, assegnandogli una rappresentazione "estrema".
- **Dispersione su dim 2**:
  la variabilità tra clienti viene codificata principalmente su una sola dimensione ("Bottleneck dim 2"), mentre la prima ("Bottleneck dim 1") resta quasi costante per quasi tutti,
  segno che il modello usa soprattutto la seconda dimensione per distinguere tra i clienti "normali".

- Bisogna trovare **chi è il cliente outlier**: nel dataset potrebbe essere qualcuno con spesa/ordini molto diversi dalla media.
- Possiamo usare questi due numeri (bottleneck) per clusterizzare, visualizzare o analizzare rapidamente la clientela.
- Se servono cluster più ricchi, possiamo aumentare la dimensione del bottleneck (da 2 a 3 o più).

# INTERPRETAZIONE DEL GRAFICO BOTTLENECK (2D)

**Aumentare la dimensione del bottleneck** nell'autoencoder significa **consentire al modello di comprimere i dati in più numeri** (più informazioni), quindi avere una rappresentazione "più ricca" e meno costretta.

Basta **cambiare il parametro `encoding_dim`** quando costruisci la rete.

```
# Bottleneck a 3, 4, 8... dimensioni
encoding_dim = 4
```

**Cosa cambia aumentando il bottleneck?**

- La **rappresentazione compressa** sarà un vettore con più numeri per ogni cliente (es: [3.2, 1.7, -0.5, 0.9] se hai 4 dimensioni).
- L'autoencoder sarà **meno forzato a comprimere tutto in pochissimi numeri** → l'errore di ricostruzione generalmente diminuisce.

**Quando aumentare il bottleneck?**

- Se l'errore di ricostruzione è troppo alto (modello "strozzato")
- Se vuoi catturare **più sfumature** tra i clienti
- Se hai dati più complessi o molte feature

# AUTOENCODER - CONCLUSIONE

Gli **autoencoder** sono strumenti potenti per la **riduzione automatica e non lineare della dimensionalità**, permettono di comprimere e ricostruire dati complessi imparando direttamente dalle informazioni presenti.

Servono per:

- rappresentare i dati in uno spazio ridotto (bottleneck),
- scoprire anomalie e pattern nascosti,
- preparare dati per clustering e altre analisi avanzate.

**Il risultato**

Un modello che riassume l'essenza dei dati, semplifica l'esplorazione e apre la strada a nuove applicazioni pratiche nell'analisi dei dati e nel machine learning.

# DOMANDE?

PAUSA

# SEGMENTI E PERSONAS: CONCETTI CHIAVE

**Cos'è un segmento?**

- Un **segmento** è un gruppo di clienti o utenti che condividono caratteristiche, bisogni o comportamenti simili.
- I segmenti possono essere creati sulla base di dati (età, area geografica, spesa, attività) oppure su criteri comportamentali (tipi di acquisto, fedeltà, canali preferiti…).

**Obiettivo:** Trattare ogni segmento con offerte, comunicazioni, prodotti o servizi più personalizzati.

**Esempi di segmenti:**

- Clienti ad "alta spesa"
- Utenti "dormienti"
- Acquirenti frequenti vs occasionali
- Clienti che acquistano solo in saldo

**In sintesi**

Ogni segmento raggruppa clienti con **caratteristiche, abitudini o bisogni simili**, permettendo strategie mirate e comunicazione efficace.

# TIPI DI CUSTOMER SEGMENTI

**Segmenti per valore**

- **High value (top spender)**: clienti che spendono molto e generano la maggior parte del fatturato.
- **Low value**: clienti con acquisti occasionali o spesa bassa.

**Segmenti per fedeltà/frequenza**

- **Clienti fedeli**: tornano spesso, fanno acquisti regolari.
- **Clienti nuovi**: appena acquisiti, magari in fase di esplorazione.
- **Clienti dormienti**: non acquistano da tempo.

**Segmenti per bisogni o comportamenti**

- **Ricercatori di offerte**: sensibili a sconti e promozioni.
- **Innovatori**: cercano novità e prodotti nuovi.
- **Convenienza/velocità**: acquistano per comodità, cercano rapidità.
- **Specialisti**: acquistano sempre la stessa categoria/prodotto.

**Segmenti per canale**

- **Online only**: acquistano solo tramite e-commerce.
- **Omnicanale**: usano più canali (es. online e negozio fisico).

**Segmenti per demografia**

- **Giovani, famiglie, senior**: raggruppati per età, composizione familiare, ecc.

**Segmenti per occasione d'acquisto**

- **Regalo**: comprano per fare regali.
- **Personale**: acquistano per sé stessi.

**Segmenti per sensibilità al prezzo**

- **Price sensitive**: fanno attenzione a offerte e prezzi bassi.
- **Premium**: puntano su qualità e sono meno attenti al prezzo.

# SEGMENTI E PERSONAS: CONCETTI CHIAVE

**Cos’è una persona?**

- Una **persona** (o “buyer persona”, “archetipo”) è un **profilo semi-fittizio** che rappresenta un segmento.
- Va oltre i numeri: descrive **età, lavoro, bisogni, motivazioni, paure, abitudini, frasi tipiche**.

**Obiettivo:** Aiutare team di marketing, prodotto, vendita, supporto a visualizzare (quasi come fosse reale) il cliente a cui si rivolgono.

**Perché si usano le personas?**

- **Rendere concreti i dati:** trasforma i numeri in “volti”, storie, esigenze.
- **Guidare strategie:** facilita la scelta di messaggi, prodotti, servizi su misura.
- **Allineare i team:** permette a tutti (marketing, vendite, sviluppo, supporto…) di parlare dello stesso tipo di cliente.
- **Ottimizzare esperienza e comunicazione:** permette di scrivere contenuti, offerte, UX più mirati e coinvolgenti.

# SEGMENTI E PERSONAS: CONCETTI CHIAVE

**Identificazione e creazione della persona: step pratici**

**1. Analisi dati**

- Raccogli dati quantitativi (acquisti, età, canali, frequenza, spesa…)
- Analizza i cluster/segmenti: cerca i gruppi più rilevanti e diversi tra loro.

**2. Sintesi delle caratteristiche**

- Per ogni segmento, identifica le variabili chiave (es. spesa alta/bassa, canali usati, tipologia acquisti, motivazioni).

**3. Arricchimento qualitativo**

- Integra con dati qualitativi (interviste, survey, feedback clienti) dove possibile.

**4. Creazione della scheda persona**

- Dai un **nome** e (opzionale) un'immagine o avatar.
- Indica:
  - Età/media, ruolo/lavoro, situazione familiare
  - Bisogni/motivazioni
  - Abitudini d'acquisto e canali preferiti
  - Barriere, dubbi, "pain points"
  - Frasi tipiche o valori chiave
- Aggiungi una breve storia ("giornata tipo" o scenario d'uso)

**5. Condivisione e uso**

- Usa la scheda persona per guidare decisioni, strategie, brainstorming, campagne, design di prodotto.

# ESEMPIO DI PERSONA

**Nome:** Alessandro, il giovane esploratore digitale
**Età:** 27 anni
**Professione:** Sviluppatore web freelance
**Situazione:** Vive in città, single, molto attivo sui social e nella community tech.

**Motivazioni principali:**

- Vuole scoprire sempre nuovi gadget e servizi.
- Apprezza la personalizzazione e le anteprime esclusive.
- Ama risparmiare tempo grazie a tecnologie smart.

**Abitudini d'acquisto:**

- Compra soprattutto online, spesso di notte.
- Si informa su blog e recensioni, confronta molto prima di acquistare.
- Usa app e cashback, cambia spesso marca se trova una promo interessante.

**Barriere/dubbi:**

- Non ama sentirsi "trattato come tutti".
- Si allontana se riceve offerte troppo generiche o ripetitive.

# CREAZIONE DI PERSONA

**Consigli pratici per la creazione di personas**

- Parti sempre dai dati reali: evita stereotipi non supportati.
- Personalizza tono e dettaglio per l'azienda (più narrativo o più analitico).
- Aggiorna le personas ogni 1-2 anni (i clienti cambiano!).
- Fai emergere emozioni e motivazioni, non solo i numeri.
- Usa le personas per stimolare empatia e creatività nei team.

**Evita dati sensibili e discriminazione**

- **Non usare** (né aggregare/visualizzare) dati sensibili come:
  - etnia, religione, orientamento sessuale, salute, opinioni politiche
  - salvo casi eccezionali previsti dalla legge e con motivazione documentata
- **Non creare personas** basate su criteri che possono causare discriminazione diretta o indiretta (es. "solo uomini", "solo under 30", "esclusi stranieri"…).

**Rispetta il GDPR e la privacy**

- Usa **solo dati raccolti legalmente** e con consenso informato.
- Rendi i dati **anonimi e aggregati**: nessuna persona reale deve essere identificabile nella scheda persona.
- Evita di pubblicare o condividere dati personali non necessari.

# CREAZIONE DI PERSONA

**Attenzione ai minori**

- Segmenti e personas non dovrebbero mai essere costruiti (né campagne mirate) su **minori** senza consenso specifico e conforme alle normative locali.

**Evita stereotipi e generalizzazioni offensive**

- Anche se la persona è “fittizia”, **non utilizzare descrizioni che possano risultare offensive, discriminatorie, o basate su pregiudizi**.
- Non “etichettare” i segmenti con nomi o descrizioni che rischiano di ledere la dignità o la sensibilità delle persone.

**Documenta criteri e finalità**

- Conserva traccia di **come sono stati creati i segmenti e le personas**, con quali dati, per quali fini.
- Assicurati che tutti i collaboratori siano consapevoli delle regole di compliance.

**Restrizioni specifiche per settore**

- Alcuni settori (finanza, salute, assicurazioni) hanno regole più stringenti:
  **verifica sempre la normativa di settore** prima di segmentare/targettizzare.

# CHATGPT PER INTERPRETARE SEGMENTI E PERSONAS

**A cosa serve ChatGPT in questo processo?**

- **Tradurre i cluster numerici** in descrizioni ricche, chiare e “umane”
- **Inventare nomi e storie** per le personas
- **Suggerire strategie marketing/prodotto** su misura per ciascun segmento
- **Rispondere a domande di business** (“Cosa diresti a questo tipo di cliente?”, “Come lo fidelizzeresti?”)

**Tipici utilizzi di ChatGPT**

1. **Descrizione profonda del segmento**
   Fornisci le principali statistiche e ChatGPT produce un testo discorsivo:
   “Clienti che spendono molto, acquistano più categorie, sono fedeli e attivi tutto l’anno...”
2. **Generazione della persona tipo**
   Prompt:
3. “Crea una buyer persona dettagliata con nome, lavoro, motivazioni, abitudini, bisogni, stile di vita…”
3. **Ideazione offerte, messaggi, campagne**
   Prompt:
   “Suggerisci 3 idee di email marketing efficaci per questa persona”
4. **Simulazione di dialogo**
   Prompt:
   “Recita la risposta di questa persona alla proposta di un nuovo prodotto/servizio”

# ESEMPI DI PROMPT PER CHATGPT

**Prompt per descrivere un segmento**

*“Ho un segmento di clienti che fanno almeno 10 ordini all’anno, spendono più di 1000€, acquistano in 5 categorie diverse. Descrivimi la loro persona tipo, includi età stimata, abitudini d’acquisto e cosa li motiva.”*

**Prompt per generare buyer persona**

*“Crea una scheda persona per questo cluster:*

- *Frequenza d’acquisto: bassa*
- *Spesa totale: media*
- *Attività principale: shopping per la famiglia*
- *Preoccupazione: prezzo e affidabilità .”*

**Prompt per idee campagne personalizzate**

*“Suggerisci due campagne promozionali su misura per questa persona, una via email e una sui social.”*

**Prompt per confronto tra segmenti**

*“Confronta la persona A (spesa alta, acquisti su molte categorie) e la persona B (spesa bassa, solo una categoria): bisogni diversi, barriere all’acquisto, messaggi ideali.”*

# SOMMARIO – CHATGPT IN MARKETING

**Vantaggi e raccomandazioni pratiche**

- **ChatGPT accelera brainstorming e reporting**: puoi produrre descrizioni, materiali, suggerimenti in pochi secondi.
- **Migliora la comunicazione tra dati e business**: ciò che il data scientist trova nei numeri, ChatGPT lo rende “vivo”.
- **Personalizza i risultati**: puoi chiedere a ChatGPT di scrivere in tono amichevole, tecnico, creativo o con dettagli più pratici.
- **Crea documentazione, supporti marketing, materiali per workshop e presentazioni**.

**Attenzione e buone pratiche**

- **Usa sempre dati aggregati**: ChatGPT non sostituisce l’analisi numerica, ma la arricchisce.
- **Verifica sempre i risultati**: soprattutto per evitare stereotipi o idee troppo generiche.
- **Coinvolgi i team**: usa ChatGPT come base di lavoro, non come “verità assoluta”.

**Sintesi finale**

Segmenti e personas sono fondamentali per capire il pubblico e personalizzare ogni azione.
Con ChatGPT, la transizione dai dati ai profili concreti diventa veloce, accessibile e creativa.
Un vero assistente per la customer intelligence moderna!

# ESERCIZIO: SEGMENTAZIONE E PERSONAS CON CHATGPT

**Scenario – Il mercato di partenza**

Immagina di lavorare per **EcoSmartHome**, una startup che vende online prodotti innovativi per la casa:

- lampadine smart, sensori di risparmio energetico, termostati intelligenti, prese Wi-Fi, kit domotici “fai da te”.

Negli ultimi mesi, le vendite sono cresciute, ma il team marketing vuole **capire meglio i clienti per lanciare campagne mirate e personalizzare l’offerta**.

**Sono disponibili i seguenti dati aggregati (per ciascun cliente):**

- Frequenza di acquisto (bassa/media/alta)
- Spesa media per ordine (bassa/media/alta)
- Prodotti preferiti (illuminazione, risparmio energetico, automazione)
- Canale di acquisto (solo online / anche negozio fisico)
- Utilizzo app EcoSmartHome (mai / saltuariamente / spesso)

# ESERCIZIO: SEGMENTAZIONE E PERSONAS CON CHATGPT

**Problema**

Il management chiede di:

1. **Identificare almeno 3 segmenti di clienti** a partire dai dati disponibili.
2. **Descrivere per ogni segmento una "buyer persona" dettagliata**, con nome, età stimata, motivazioni, abitudini, bisogni, barriere e frase tipica.
3. **Usare ChatGPT** per arricchire la descrizione delle personas e ottenere suggerimenti per campagne mirate.

# ESERCIZIO: SEGMENTAZIONE E PERSONAS CON CHATGPT

**Cosa fare**

1. **Analisi e segmentazione**
   - Raggruppa i clienti in almeno 3 segmenti diversi, motivando la scelta (es. “high spender”, “tecnofili”, “acquirenti occasionali”, ecc.).
2. **Creazione buyer persona**
   - Per ogni segmento, crea una scheda persona dettagliata seguendo i punti dati sopra.
3. **Interazione con ChatGPT**
   - Scrivi un prompt efficace per ChatGPT (esempio: “ChatGPT, crea una buyer persona per il segmento clienti con acquisto frequente, alta spesa media, preferenza per prodotti di automazione e uso assiduo dell’app. Includi nome, età stimata, motivazioni e una breve storia.”).
   - Chiedi a ChatGPT almeno 2 suggerimenti di campagne marketing personalizzate per ciascuna persona.
4. **Presentazione finale**
   - Presenta il risultato al team marketing:
   - Elenco segmenti con motivazione
   - Schede personas
   - Prompt usati e output di ChatGPT
   - Campagne suggerite

# BEST PRACTICE PER WORKFLOW NON SUPERVISIONATO

Profice

**1. Conoscere bene il dominio**

- Analizzare il **contesto** e la logica del problema prima di applicare qualsiasi algoritmo.
- Confrontare sempre i risultati con le conoscenze di business e la realtà operativa (validazione con il dominio).

**2. Esplorazione e preparazione accurata dei dati**

- **Pulizione dei dati**: eliminare errori, valori anomali non giustificati, ridondanze.
- **Standardizzare** o normalizzare le feature numeriche per confrontarle in modo corretto.
- Valutare quali variabili sono rilevanti, togli quelle inutili o troppo collegate tra loro.

**3. Riduzione della dimensionalità**

Usare tecniche come **PCA, t-SNE, UMAP** per esplorare, visualizzare e accelerare l'analisi su molti dati.
Ricordarsi: ridurre le dimensioni facilita il clustering e la scoperta di pattern.

**4. Scelta dell'algoritmo giusto**

- Prova diversi algoritmi: **KMeans, DBSCAN, clustering gerarchico**, ecc.
- Considera la natura dei dati (distribuzione, densità, numero cluster atteso).

# BEST PRACTICE PER WORKFLOW NON SUPERVISIONATO

**5. Valutazione e interpretazione dei risultati**

- Usare metriche oggettive: **silhouette score, Davies-Bouldin, elbow method**.
- Validare i cluster anche con l'esperienza aziendale (es: i segmenti hanno senso per il business?).
- Visualizzare sempre i risultati (scatterplot, dendrogrammi, heatmap…).

**6. Documentazione e riproducibilità**

- Tenere traccia delle scelte, dei parametri e degli step di preprocessing.
- Condividere notebook/codice per permettere la ripetibilità e il confronto.

**7. Iterazione e confronto**

- Non fermarsi al primo risultato: sperimenta vari approcci e confronta diverse soluzioni.
- Includere sempre una riflessione sui limiti (dati, algoritmi, interpretabilità).

**In sintesi**

L'**ML non supervisionato** richiede curiosità, attenzione ai dettagli, confronto continuo con il contesto reale e spirito critico su tutto il processo, dal dato grezzo all'azione concreta.

# RIASSUNTO DELLA GIORNATA

- Abbiamo visto cosa significa apprendere dai dati **senza etichette**, scoprendo gruppi, pattern e anomalie grazie al machine learning non supervisionato.
- Abbiamo esplorato dataset reali, imparando come **pulire, standardizzare e selezionare** le feature più importanti.
- Abbiamo introdotto le tecniche di **riduzione dimensionale** (PCA, t-SNE, UMAP, autoencoder) per visualizzare e semplificare dati complessi.
- Abbiamo sperimentato diversi algoritmi (KMeans, DBSCAN, clustering gerarchico), imparando come trovare, validare e interpretare i cluster con metriche oggettive e confronto con la logica del business.
- Abbiamo visto come passare **dai cluster ai segmenti** e dalle **personas**: profili concreti che aiutano a rendere i risultati utili per marketing, prodotto e strategia aziendale.
- Abbiamo imparato come usare ChatGPT per **tradurre dati e cluster in profili ricchi**, ideare campagne e arricchire la comprensione dei clienti.
- Best practice: Importanza di validazione, documentazione, iterazione e confronto tra metodi diversi.

# PROSSIMI PASSI E MOTIVAZIONE

**Cosa portiamo a casa**

- Le basi per affrontare un progetto di **Machine Learning non supervisionato** con metodo e spirito critico
- La sicurezza di **scegliere, valutare e confrontare algoritmi di clustering e riduzione dimensionale** grazie a metriche, visualizzazioni e confronto con il dominio
- Una metodologia concreta per **preparare i dati, esplorare pattern nascosti e trasformarli in segmenti e personas utili per il business**

**Come continuare**

- Ripassa i materiali della giornata e **prova a replicare gli esercizi** (clustering, riduzione dimensionale, validazione) su altri dataset
- Non fermarti alla metrica o al grafico: **valida sempre i risultati con la logica del problema e il confronto con colleghi o stakeholder**
- Sperimenta diverse tecniche di clustering e visualizzazione;
  cerca i limiti e i punti di forza di ciascun metodo
- **Documenta ogni passo**: scelte, parametri, risultati.
  Confronta e discuti: il valore nasce sempre dal confronto

**Domande finali? Feedback?**

- Lo spazio è aperto:
  fai tutte le domande che hai, suggerisci temi o strumenti da approfondire nelle prossime giornate!

# GRAZIE PER L'ATTENZIONE