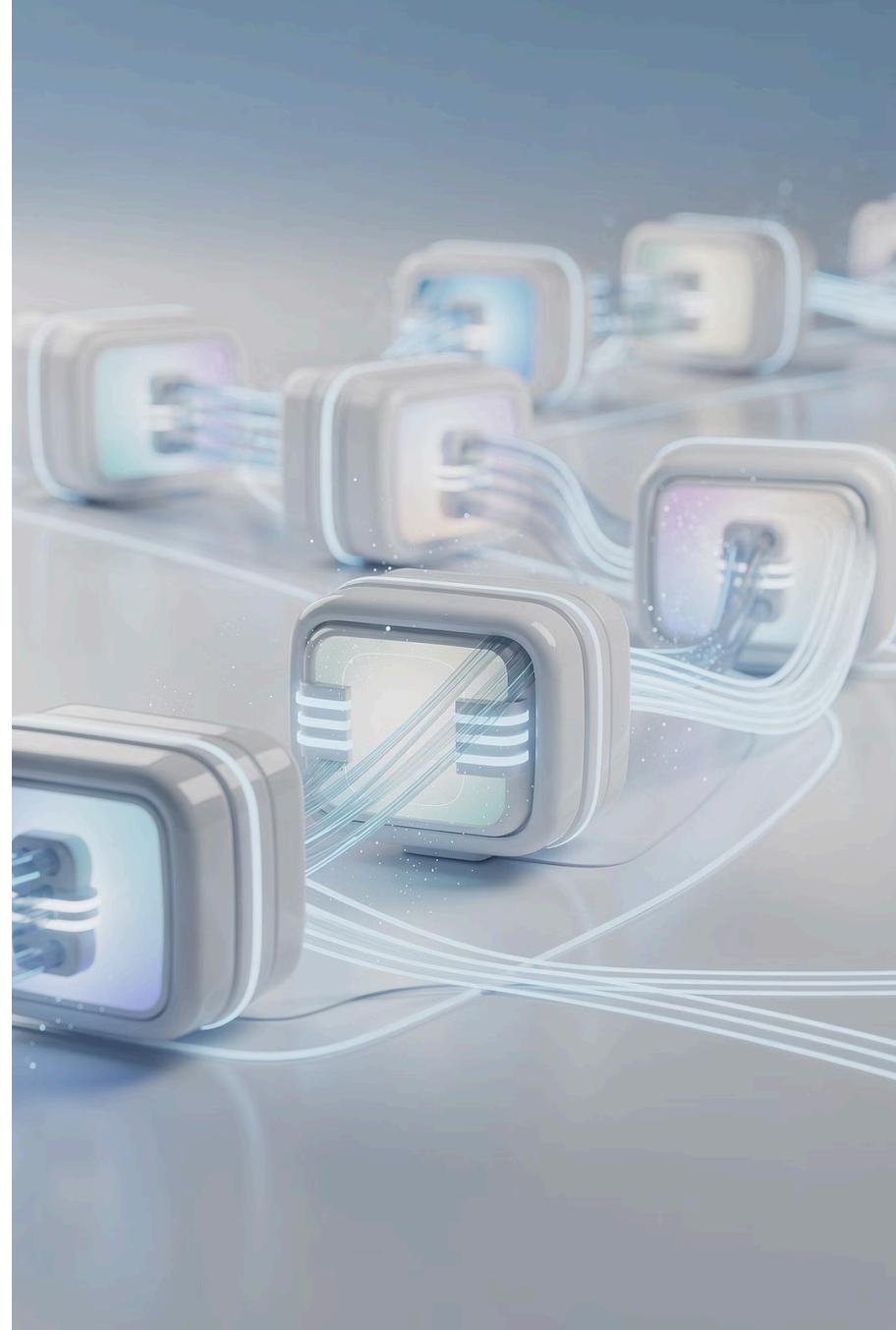


Preprocessing e Preparazione dei Dati per il Machine Learning

Un percorso pratico dalla business intelligence al machine learning: come trasformare informazioni grezze in dati pronti per l'analisi predittiva.





L'Obiettivo del Nostro Percorso

Da dove partiamo

Dataset estratto da un Data Warehouse contenente informazioni sui clienti: ordini, fatturato, date e area geografica. Dati nel formato tipico del business, con stringhe, date e strutture complesse.

Dove vogliamo arrivare

Una matrice numerica pulita e standardizzata, pronta per essere elaborata da algoritmi di Machine Learning. Un formato matematico ottimizzato per l'analisi predittiva.

- ❑ ! In questo notebook **non alleneremo alcun modello**: ci concentreremo esclusivamente sul preprocessing, una fase critica e concettualmente autonoma che determina il successo di ogni progetto di ML.

Le Librerie Fondamentali



Pandas & NumPy

Gestione e manipolazione efficiente dei dati strutturati



Matplotlib & Seaborn

Visualizzazione esplorativa per comprendere la distribuzione dei dati



Scikit-learn

Toolkit completo per preprocessing, pipeline e trasformazioni

Gli strumenti di scikit-learn che utilizzeremo includono **train_test_split** per separare training e test, **Pipeline** e **ColumnTransformer** per costruire flussi di lavoro riproducibili, e trasformatori come **SimpleImputer**, **StandardScaler** e **OneHotEncoder** per il preprocessing vero e proprio.

Idea chiave: Prima di applicare algoritmi complessi, dobbiamo costruire un processo riproducibile e metodologicamente corretto di preparazione dei dati.



Il Dataset: Customer Features

Il file **customer_features.csv** è stato esportato da pgAdmin e rappresenta un tipico dataset di customer analytics. Ogni riga corrisponde a un singolo cliente e contiene informazioni aggregate su tutto il suo percorso di acquisto.

Informazioni anagrafiche

Città e regione di appartenenza del cliente

Informazioni temporali

Data del primo ordine e data dell'ultimo ordine effettuato

Metriche aggregate

Numero totale di ordini, fatturato complessivo, scontrino medio, prodotti acquistati e categorie esplorate

Questo formato è estremamente comune in problemi di **churn prediction**, **customer lifetime value**, **credit scoring** e analisi del rischio. La struttura "una riga per cliente" facilita l'applicazione di algoritmi di classificazione e regressione.

Gestione delle Date e Snapshot Date



01

Evita il leakage temporale

Previene contaminazioni tra passato e futuro nei dati

02

Garantisce coerenza

Tutti i clienti sono confrontati rispetto allo stesso momento temporale

03

Best practice industriale

Approccio standard nei modelli di churn, rischio creditizio e analisi predittiva

Conversione e standardizzazione

Le colonne contenenti date vengono convertite nel formato `datetime` di pandas, permettendo calcoli temporali precisi e operazioni di sottrazione tra date.

Il concetto di Snapshot Date

Definiamo una **data di riferimento comune**, che rappresenta l'"oggi" dal punto di vista del dataset. Viene calcolata come il giorno successivo all'ultimo ordine osservato in tutto il database.

Feature Engineering: Recency e Tenure

Le date in formato calendario non possono essere utilizzate direttamente dai modelli di Machine Learning. È necessario trasformarle in **misure numeriche significative** che catturino l'informazione temporale in modo utilizzabile.

Recency (Giorni dall'ultimo ordine)

Differenza in giorni tra la snapshot date e la data dell'ultimo ordine

- Valori bassi = cliente attivo recentemente
- Valori alti = cliente inattivo da tempo
- Forte predittore di churn

Tenure (Anzianità del cliente)

Differenza in giorni tra la snapshot date e la data del primo ordine

- Valori bassi = cliente nuovo
- Valori alti = cliente storico
- Indicatore di fedeltà e lifetime value

👉 Il modello non lavora con calendari e timestamp, ma con **misure temporali quantitative**. Questa trasformazione è fondamentale per rendere l'informazione temporale utilizzabile matematicamente.

Definizione del Target: Customer Churn

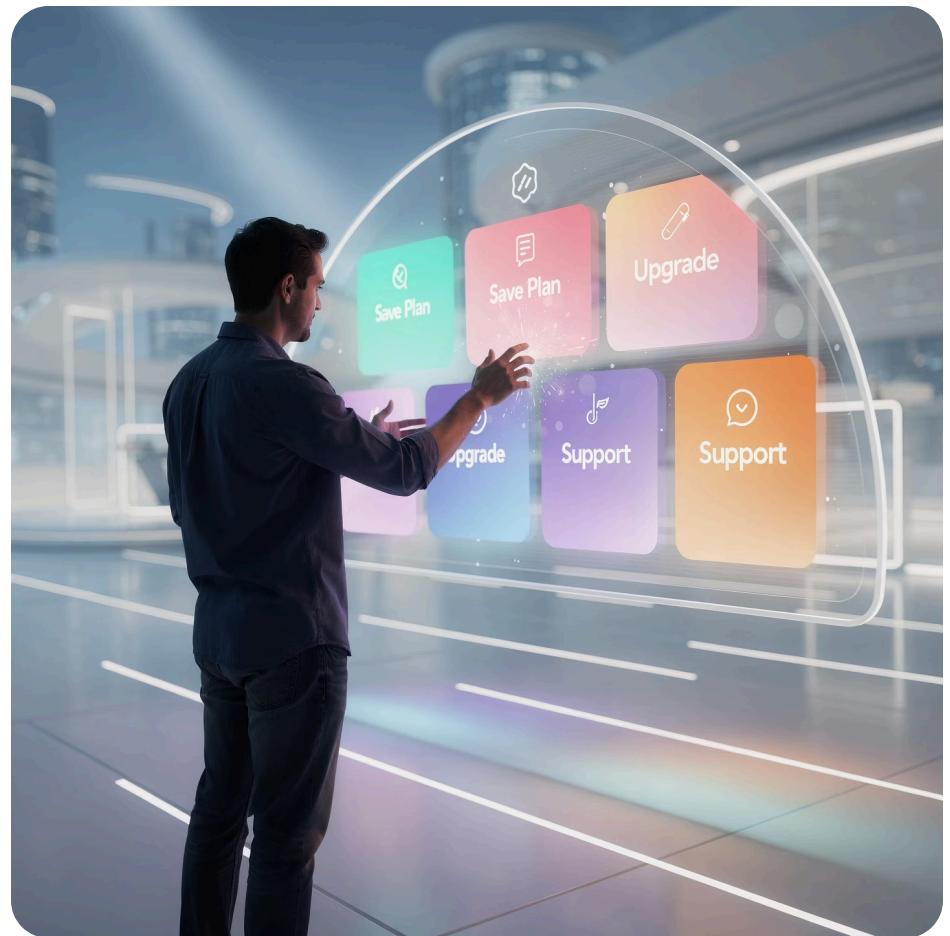
La variabile target `is_churn` viene creata applicando una **regola di business**

esplicita e interpretabile: un cliente viene considerato "churned"

(abbandonato) se non effettua acquisti da più di 90 giorni rispetto alla snapshot date.

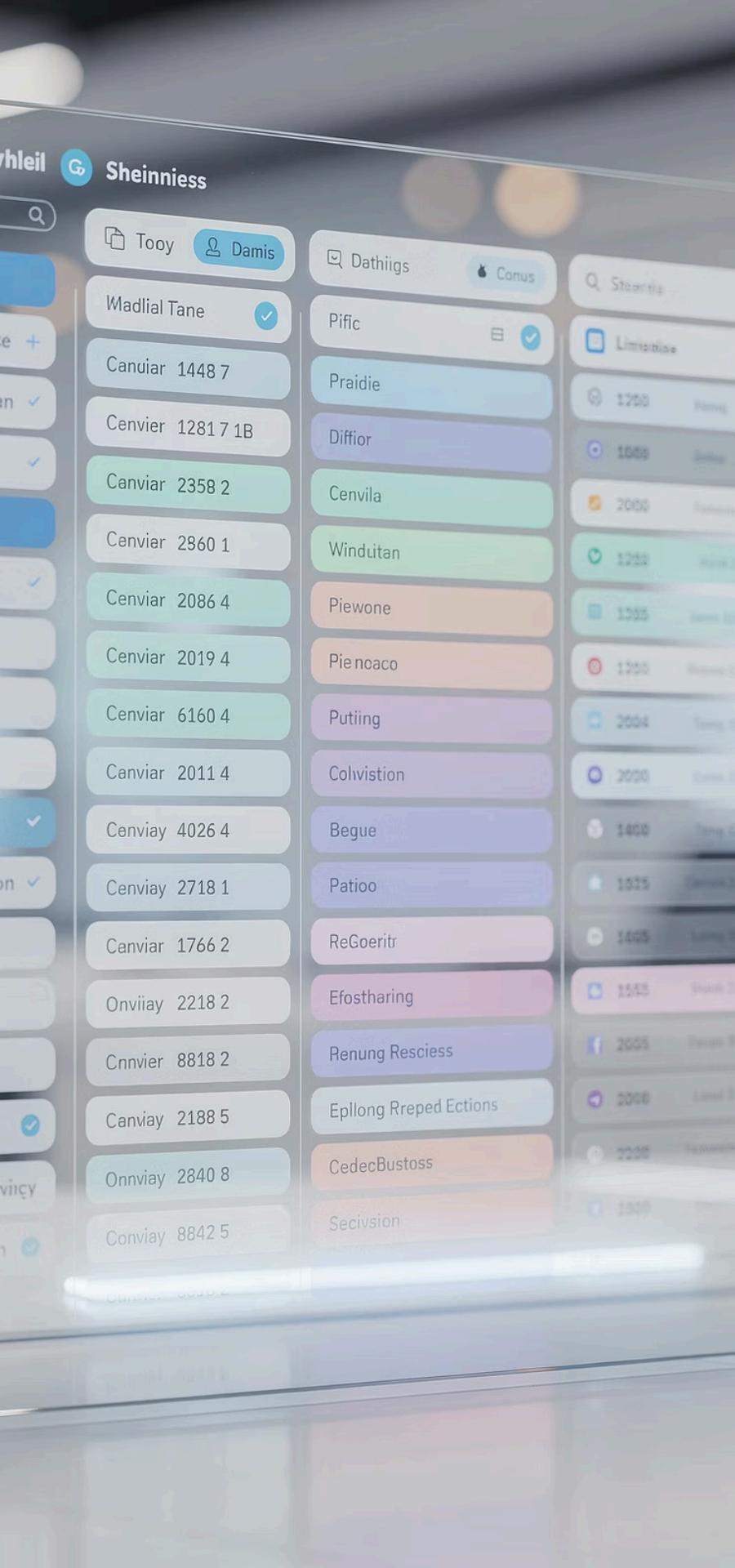
Perché questo approccio?

- **Esplicito**: la definizione è chiara e documentata
- **Interpretabile**: stakeholder e business possono comprenderla facilmente
- **Modificabile**: la soglia dei 90 giorni può essere adattata in base al contesto di business
- **Testabile**: permette di validare ipotesi diverse



❑ **Messaggio didattico:** Il target non è un dato "naturale", ma una [scelta di modellazione](#) che riflette obiettivi di business specifici.

Nei progetti reali, questa soglia viene definita collaborativamente con il business team, analizzando la distribuzione temporale degli acquisti, la stagionalità del prodotto e gli obiettivi strategici dell'azienda.



Selezione e Classificazione delle Feature

Prima di applicare qualsiasi trasformazione, è essenziale **identificare e classificare** le variabili in base alla loro natura. Questa distinzione determina quale preprocessing applicare a ciascuna colonna.

1

Feature Numeriche

- recency (giorni dall'ultimo ordine)
- tenure (anzianità cliente)
- numero_ordini
- fatturato_totale
- scontrino_medio
- totale_prodotti_acquistati
- num_prodotti_distinti
- num_categorie_distinte

2

Feature Categoriche

- regione (variabile geografica con valori discreti)

Le variabili categoriche contengono etichette testuali che devono essere convertite in formato numerico attraverso tecniche di encoding.

Questa distinzione è **fondamentale** perché i due tipi di variabili richiedono trattamenti completamente diversi: le numeriche necessitano di scaling e normalizzazione, mentre le categoriche richiedono encoding.

Split Training e Test Set

I dati vengono suddivisi in due insiemi completamente separati, seguendo best practice consolidate nel Machine Learning. Questa separazione è cruciale per valutare correttamente le performance del modello.



Training Set (80%)

Utilizzato per apprendere pattern, calcolare statistiche e addestrare il modello

Test Set (20%)

Rappresenta dati completamente nuovi, mai visti dal modello durante l'addestramento

Parametri critici dello split

random_state

Garantisce la riproducibilità dello split: eseguendo il codice più volte si ottiene sempre la stessa divisione dei dati

stratify=y

Mantiene la stessa proporzione di clienti churned/non-churned in entrambi i set, evitando sbilanciamenti che comprometterebbero la valutazione

- 💡 Il test set rappresenta il "mondo reale": dati che il modello incontrerà solo dopo essere stato completamente sviluppato. Qualsiasi contaminazione tra training e test invalida completamente la validazione.

Pipeline di Preprocessing

Costruiamo **pipeline separate e specializzate** per gestire i due tipi di feature, garantendo trasformazioni appropriate e riproducibili. Questo approccio modulare è una best practice industriale.

Pipeline Numerica

1. **SimpleImputer (strategy='median')**: I valori mancanti vengono sostituiti con la mediana della colonna, più robusta agli outlier rispetto alla media
2. **StandardScaler**: Ogni colonna viene standardizzata con media 0 e deviazione standard 1, rendendo comparabili feature con scale diverse

Pipeline Categorica

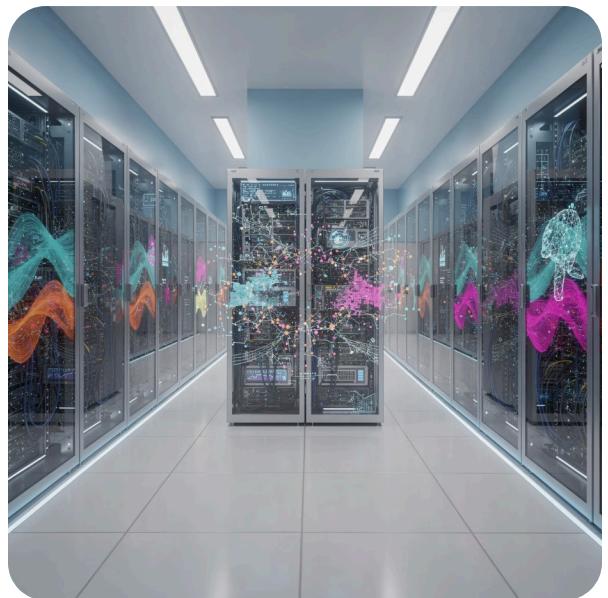
1. **SimpleImputer (strategy='most_frequent')**: I valori mancanti vengono sostituiti con la categoria più frequente
2. **OneHotEncoder**: Ogni categoria viene trasformata in una colonna binaria (0/1), permettendo al modello di processare informazioni categoriche



ColumnTransformer: l'orchestratore

Il ColumnTransformer applica automaticamente la pipeline corretta a ciascun gruppo di colonne e combina i risultati in un'unica matrice. Questo garantisce coerenza, riproducibilità e manutenibilità del codice.

Fit e Transform: La Regola d'Oro



Due operazioni, due significati

La distinzione tra **fit** e **transform** è fondamentale per evitare il data leakage, uno degli errori più gravi nel Machine Learning.

1

Sul Training Set: `fit_transform()`

Fit: Calcola le statistiche (mediane, medie, deviazioni standard, categorie presenti)

Transform: Applica queste statistiche per trasformare i dati

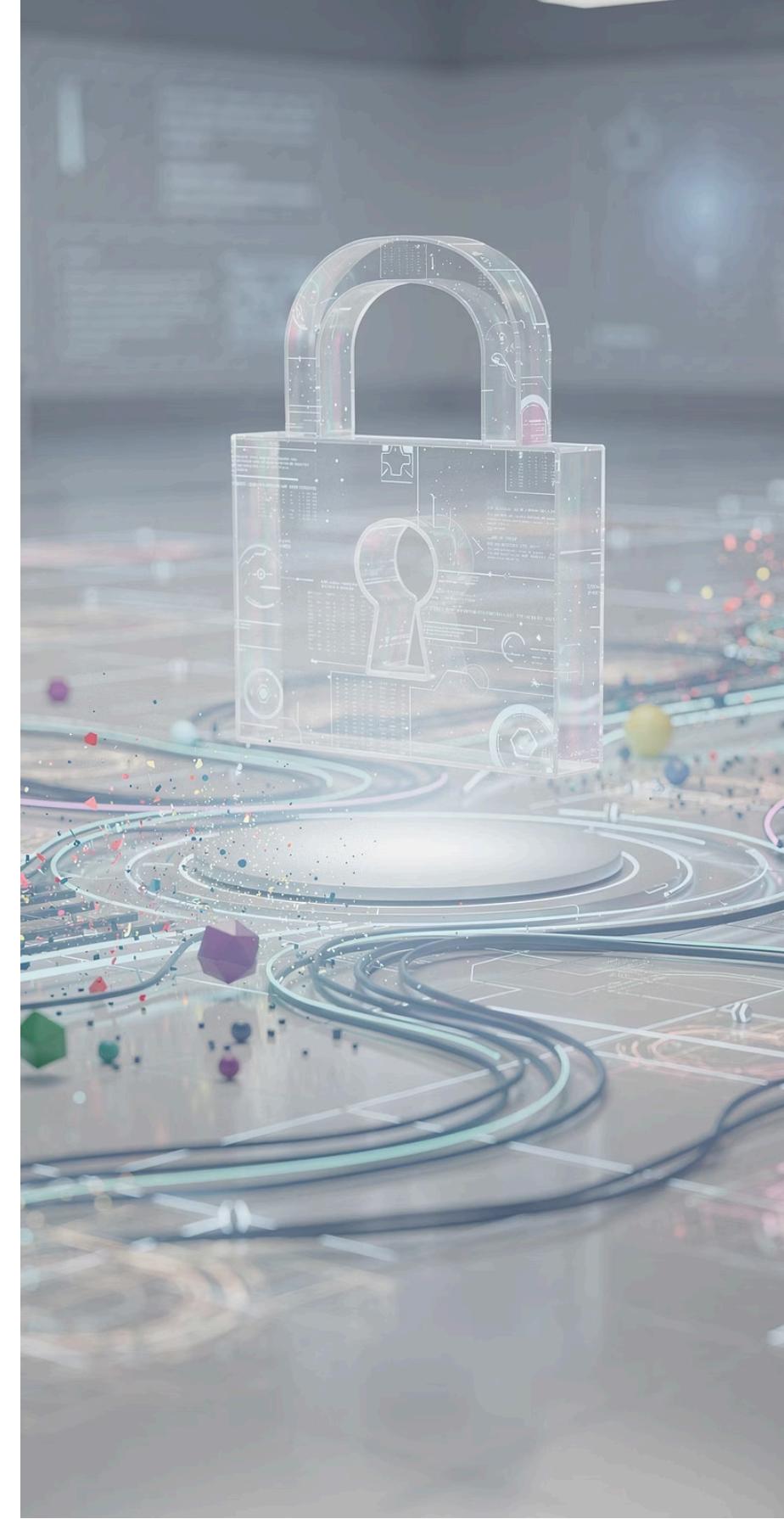
2

Sul Test Set: `transform()` ONLY

Applica le statistiche già calcolate sul training set

Mai calcolare nuove statistiche sul test!

In questo modo il test set rimane realmente "nuovo" e non influenza in alcun modo il processo di apprendimento. Utilizzare `fit_transform()` anche sul test causerebbe **data leakage**, producendo metriche di performance artificialmente ottimistiche e inaffidabili.



Output del Preprocessing: La Matrice ML-Ready

Al termine del processo di preprocessing otteniamo una **matrice numerica pulita**, completamente priva di elementi non numerici e pronta per essere elaborata da qualsiasi algoritmo di Machine Learning.

9

Feature originali

Il numero iniziale di colonne nel dataset

28

Feature finali

Dopo one-hot encoding delle variabili categoriche

100%

Valori numerici

Nessuna stringa, nessuna data, nessun valore mancante

Perché l'aumento del numero di colonne?

La variabile **regione**, originariamente contenente N regioni diverse, viene espansa in N colonne binarie attraverso il one-hot encoding. Ogni colonna rappresenta la presenza (1) o assenza (0) di quella specifica regione per il cliente.

- 💡 28 feature **non sono considerate tante** nel Machine Learning moderno. Molti modelli gestiscono efficacemente centinaia o migliaia di feature. Quello che conta è la qualità e la rilevanza delle informazioni, non solo la quantità.

La matrice risultante ha caratteristiche ottimali: feature standardizzate su scale comparabili, assenza di valori mancanti, encoding appropriato delle variabili categoriche e formato matematicamente compatibile con librerie come scikit-learn, TensorFlow e PyTorch.