Dal Data Warehouse al Machine Learning

Feature Engineering: Il Ponte tra Dati e Intelligenza Artificiale

Cos'è il Feature Engineering?

Il **Feature Engineering** è l'arte e la scienza di utilizzare la conoscenza del dominio per trasformare i dati grezzi in caratteristiche significative (features) che rappresentano meglio il problema sottostante.

Non si tratta di una semplice manipolazione dei dati, ma della creazione di segnali potenti che permettono ai modelli di Machine Learning di **apprendere in modo più efficace**, migliorando drasticamente le loro prestazioni e la loro capacità predittiva.

Cos'è il Feature Engineering?

Il **Feature Engineering** è il processo di trasformazione dei dati grezzi in variabili (feature) che rappresentano al meglio il problema da risolvere, migliorando significativamente le performance dei modelli di Machine Learning. È l'arte di estrarre valore dai dati attraverso la creazione di rappresentazioni significative.

Feature Numeriche

Età, importo, conteggi, medie, statistiche quantitative

Feature Temporali

Giorno della settimana, stagione, trend, ciciclità

Feature Categoriche

Categoria prodotto, regione geografica, segmento cliente

Feature Aggregate

Somme, conteggi, statistiche per gruppo cliente o categoria

80%

Tempo dedicato ai dati

del lavoro in un progetto ML è dedicato alla preparazione dei dati e al Feature Engineering

Estrazione delle Feature dal Data Warehouse

Due approcci complementari per estrarre valore dai dati: SQL per aggregazioni dirette sul database e Python per trasformazioni post-estrazione. La scelta dell'approccio dipende dalla complessità della trasformazione e dalle risorse di calcolo disponibili.

1

Approccio SQL

Estrazione diretta dal Data Warehouse con aggregazioni e join. Ideale per feature RFM, statistiche aggregate e calcoli che possono essere eseguiti direttamente sul database. Sfrutta la potenza computazionale del DWH.

2

Approccio Python

Manipolazione post-estrazione con Pandas e NumPy. Ideale per feature derivate, trasformazioni complesse e operazioni che richiedono logica condizionale o calcoli iterativi non facilmente esprimibili in SQL.

Esempio: Feature RFM (SQL)

```
SELECT
  customer_id,
  MAX(order_date) AS last_order,
  COUNT(*) AS frequency,
  SUM(total_amount) AS monetary
FROM dw.fact_orders
GROUP BY customer_id;
```

Output

customer_id	last_order	frequency	monetary
C001	2024-12-10	28	€3,450
C002	2024-10-15	8	€890
C003	2024-08-20	2	€180

Esempio: Feature Derivate (Python)

```
# Calcolo recency in giorni
df['recency'] = (
  snapshot_date - df['last_order']
).dt.days

# Creazione variabile target
df['is_churn'] = (
  df['recency'] > 90
).astype(int)
```

Output

customer_id	recency	is_churn
C001	5	0 (attivo)
C002	45	0 (attivo)
C003	120	1 (churned)

Feature RFM: Il Cuore dell'Analisi Cliente

Le metriche RFM (Recency, Frequency, Monetary) costituiscono il fondamento dell'analisi del comportamento d'acquisto. Queste tre dimensioni catturano aspetti complementari della relazione con il cliente e sono predittori potenti del valore futuro e del rischio di abbandono.

Recency

Quanto recentemente?

Misura i giorni trascorsi dall'ultimo ordine. Clienti con recency bassa sono più propensi ad acquistare nuovamente.

- **Cliente A:** 5 giorni (molto recente)
- **Cliente B:** 45 giorni (moderato)
- **Cliente C:** 120 giorni (a rischio)

Frequency

Con che frequenza?

Conta il numero totale di ordini effettuati. Indica il livello di engagement e fedeltà del cliente.

- **Cliente A:** 28 ordini (altissima)
- **Cliente B:** 8 ordini (media)
- **Cliente C:** 2 ordini (occasionale)

Monetary

Quanto spende?

Somma l'importo totale speso (Lifetime Value). Identifica i clienti più profittevoli per il business.

- **Cliente A:** €3.450 (alto valore)
- **Cliente B:** €890 (medio valore)
- **Cliente C:** €180 (basso valore)

Insight Chiave

Cliente A con recency di 5 giorni, 28 ordini e €3.450 di spesa rappresenta un Cliente VIP — recente, frequente e ad alto valore. Questo è il profilo ideale da coltivare e proteggere dal churn.

Feature Temporali e Comportamentali

Oltre alle metriche RFM di base, possiamo arricchire il profilo cliente con feature che catturano abitudini, pattern temporali e preferenze specifiche. Queste feature derivate aggiungono profondità all'analisi e migliorano significativamente la capacità predittiva del modello.

Feature Temporali

Catturano ciclicità, stagionalità e abitudini temporali di acquisto. Fondamentali per prevedere *quando* un cliente acquisterà.

- **tenure:** Giorni dal primo acquisto (es: 450 giorni — cliente anziano)
- **avg_days_between_orders:** Intervallo medio tra ordini (es: 28 giorni — ciclo mensile)
- **preferred_shopping_hour:** Ora preferita per acquisti (es: 20:00-22:00 — serale)
- **is_weekend_shopper:** Flag binario: acquista nel weekend? (0/1)

Feature Aggregate e Comportamentali

Descrivono preferenze, abitudini di spesa e statistiche aggregate sul comportamento d'acquisto del cliente.

- **avg_order_value (AOV):** Scontrino medio (es: €125 per ordine)
- **favorite_category:** Categoria più acquistata (es: "Gaming")
- **diversity_score:** N° categorie distinte acquistate (es: 5 — cliente versatile)
- **return_rate:** Tasso di reso: resi / ordini (es: 0.08 = 8%)
- **discount_sensitivity:** % spesa in prodotti scontati (es: 45% — sensibile alle promozioni)

Preprocessing: Gestione dei Valori Mancanti

Prima di alimentare i dati al modello, dobbiamo gestire i valori mancanti. Questi passaggi di preprocessing sono fondamentali per garantire che il modello possa elaborare correttamente tutti i dati.

Gestione dei Valori Mancanti

I valori NULL o mancanti possono causare errori nel modello. La strategia più comune è l'**imputazione** con valori statistici.

Prima (con NULL)

Cliente	Età	Ordini
Mario	35	12
Laura	*NULL*	8
Paolo	42	*NULL*

Dopo (Imputazione)

Cliente	Età	Ordini
Mario	35	12
Laura	38	8
Paolo	42	10

Strategia: Imputazione con mediana per età e media per ordini

Preprocessing: Encoding di Variabili Categoriche

I modelli ML lavorano solo con numeri. Dobbiamo convertire le categorie (es: "Nord", "Sud") in rappresentazioni numeriche attraverso il One-Hot Encoding.

Prima (Categorico)

Cliente	Regione
Mario	Nord
Laura	Sud
Paolo	Centro

Dopo (One-Hot Encoding)

Cliente	Nord	Sud	Centro
Mario	1	0	0
Laura	0	1	0
Paolo	0	0	1

Nota: Ogni categoria diventa una colonna binaria (0/1)

Preprocessing: Scaling e Normalizzazione

Scaling e Normalizzazione sono passaggi cruciali per preparare le feature numeriche ai modelli di Machine Learning. Questi processi assicurano che tutte le feature contribuiscano equamente all'apprendimento del modello, prevenendo che feature con scale più grandi dominino quelle con scale più piccole.

Normalizzazione (Min-Max Scaling)

Scala i valori delle feature in un intervallo predefinito, tipicamente tra 0 e 1. Utile per algoritmi che sono sensibili alla scala degli input, come le reti neurali o gli algoritmi basati sulla distanza.

```
X_norm = (X - X_min) / (X_max - X_min)
```

Standardizzazione (Z-score Scaling)

Trasforma i dati in modo che abbiano una media di 0 e una deviazione standard di 1. Questo rende i dati comparabili e accelera la convergenza per molti algoritmi, inclusi SVM (Support Vector Machine) e regressione lineare.

```
X_std = (X - Media) / Deviazione_Standard
```

Esempio Pratico

Valori Originali

Età	Reddito (€)
25	20000
40	60000
60	35000

Dopo Normalizzazione (0-1)

Età	Reddito (€)
0.00	0.00
0.43	1.00
1.00	0.38

Dopo Standardizzazione (Z-score)

Età	Reddito (€)
-1.24	-0.98
0.08	1.17
1.16	-0.19

Quando scegliere?

Scegli la **normalizzazione** se la distribuzione dei dati non è gaussiana o se l'algoritmo richiede un intervallo fisso. Scegli la **standardizzazione** se l'algoritmo assume una distribuzione gaussiana (es. K-Means, PCA) o se gli outlier sono un problema.

Preprocessing: Scaling e Normalizzazione

Molti algoritmi di Machine Learning sono sensibili alla scala delle feature. Lo **scaling** garantisce che tutte le variabili contribuiscano equamente al modello, evitando che feature con valori grandi dominino l'apprendimento.

Standardizzazione (Z-Score)

Formula: $z = (x - \mu) / \sigma$

Dove **μ** = media e **σ** = deviazione standard. Il risultato ha media = 0 e deviazione standard = 1.

Esempio Pratico

Valore Originale	Standardizzato
100	-1.22
200	0.00
300	+1.22

Quando usarla: Algoritmi basati su distanze (KNN, SVM, Regressione Logistica) o quando le feature hanno scale molto diverse (es: età 20-80 vs reddito 20.000-80.000)

Normalizzazione (Min-Max)

Formula: $x' = (x - x_{min}) / (x_{max} - x_{min})$

Scala i valori nell'intervallo [0, 1], preservando la distribuzione originale dei dati.

Esempio Pratico

Valore Originale	Normalizzato
100	0.00
200	0.50
300	1.00

Quando usarla: Reti neurali, algoritmi che richiedono input in un range specifico, o quando vogliamo preservare le relazioni proporzionali tra i valori

Gestione degli Outlier

Gli **outlier** sono valori anomali che si discostano significativamente dal resto della distribuzione. Possono essere errori di misurazione o valori legittimi ma estremi. La loro gestione richiede una valutazione attenta del contesto business.

Identificazione Visuale

Gli outlier appaiono come punti isolati ben oltre la massa principale dei dati. Nel box plot, sono i punti oltre i "baffi".

Esempio

Stipendio di **€400.000** in un dataset con media di €100.000 e mediana di €95.000. È un outlier genuino (dirigente) o un errore?

01

Sostituzione con Valori Statistici

Rimpiazzare gli outlier con valori limite (percentili o limiti IQR). Esempio: sostituire valori > 99° percentile con il valore del 99° percentile.

02

Trasformazioni Matematiche

Applicare trasformazioni logaritmiche o radice quadrata per comprimere la distribuzione e ridurre l'impatto degli outlier mantenendoli nel dataset.

03

Utilizzo di Modelli Robusti

Scegliere algoritmi naturalmente resistenti agli outlier, come Decision Trees, Random Forest e Gradient Boosting, invece di modelli lineari sensibili.

Attenzione: Non eliminare automaticamente tutti gli outlier! Valuta sempre il contesto business: un CEO con stipendio molto alto è un outlier legittimo, non un errore da correggere.

Target Encoding per Variabili Categoriali

Il **Target Encoding** è un'alternativa efficiente al One-Hot Encoding per gestire variabili categoriche con alta cardinalità (molte categorie diverse). Invece di creare decine o centinaia di colonne binarie, sostituisce ogni categoria con una statistica calcolata sul target.

1

Problema: One-Hot Encoding

Crea una colonna binaria (0/1) per ogni categoria unica. Con molte categorie, il dataset diventa enorme e sparse.

Esempio: Stati USA

Location	CA	TX	NY	...
California	1	0	0	...
Texas	0	1	0	...
California	1	0	0	...

Problema: 50 stati = 50 colonne! Dataset enorme e inefficiente.

2

Soluzione: Target Encoding

Sostituisce ogni categoria con la media (o altra statistica) del target per quella categoria. Una sola colonna!

Esempio: Stati USA

Location	Target (Churn)	Location_Encoded
California	0	0.23
Texas	1	0.41
California	1	0.23

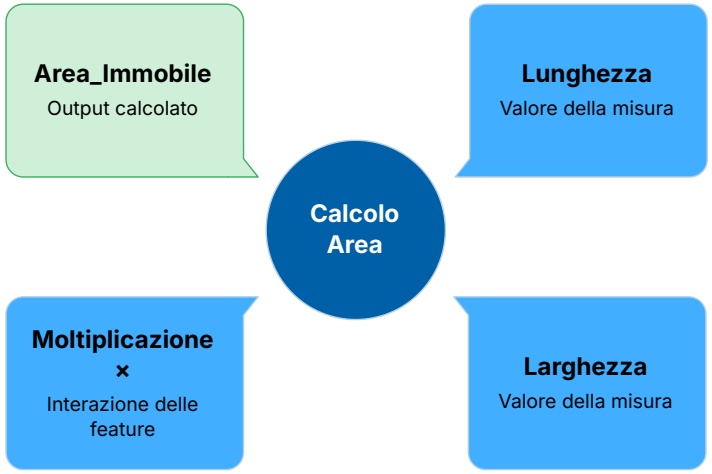
Soluzione: Una sola colonna contenente la % media di churn per ogni stato!

Nota importante: Il Target Encoding può causare data leakage se non implementato correttamente. Calcola sempre gli encoding sul training set e applicali al validation/test set per evitare overfitting.

Feature di Interazione

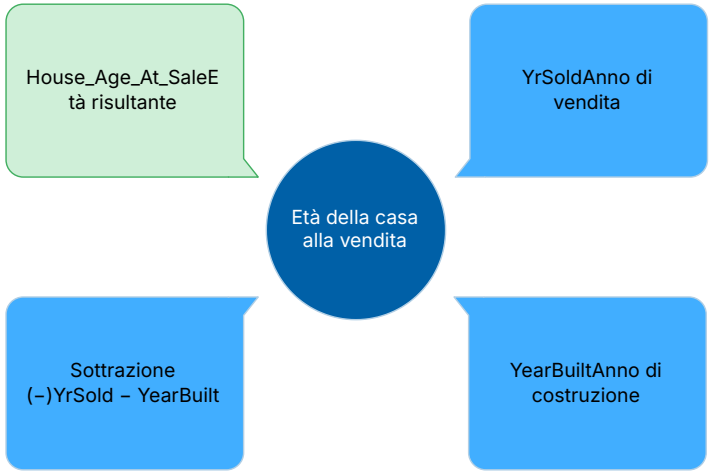
Le **feature di interazione** combinano due o più feature esistenti per catturare relazioni non lineari che il modello potrebbe non scoprire autonomamente. Sono particolarmente potenti con modelli lineari, che non possono apprendere interazioni complesse da soli.

Esempio 1: Area Immobile



Perché funziona: L'area totale è direttamente correlata al prezzo dell'immobile, fornendo un segnale molto più forte delle due misure separate. Cattura la relazione moltiplicativa naturale tra le dimensioni.

Esempio 2: Età della Casa



Perché funziona: L'età della casa al momento della vendita è un indicatore molto più significativo rispetto all'anno di costruzione assoluto. Normalizza il tempo e rende il pattern più chiaro per il modello.

Consiglio Pratico

Inizia con interazioni basate sulla conoscenza del dominio (es: area = lunghezza × larghezza è ovvio per immobili). Poi sperimenta con combinazioni meno ovvie e valuta l'impatto sulla performance del modello.

Trasformazioni Matematiche

Le **trasformazioni matematiche** modificano la distribuzione dei dati per renderla più adatta agli algoritmi di Machine Learning. La trasformazione logaritmica è particolarmente utile per gestire distribuzioni fortemente asimmetriche (skewed) con code lunghe verso destra.

Problema: Distribuzione Skewed

⚠

Distribuzione asimmetrica con pochi valori molto alti che distorcono la media e dominano il modello.

Dati Originali (€)

- €10, €20, €50, €100, €500, €2000

La maggior parte dei valori è bassa, ma alcuni outlier (€2000) distorcono la distribuzione.

Soluzione: Trasformazione Log

✅

Applicando il logaritmo naturale, la distribuzione diventa più simmetrica e normale.

Dati Trasformati (log)

- 2.3, 3.0, 3.9, 4.6, 6.2, 7.6

I valori sono ora più uniformemente distribuiti, riducendo l'impatto degli outlier.

→

Funzione Matematica

```
import numpy as np
feature_log = np.log(feature)
```

Il logaritmo naturale (ln) comprime i valori grandi più dei valori piccoli, normalizzando la distribuzione.

> **Quando usarla:** Variabili con distribuzione esponenziale o log-normale (prezzi, redditi, popolazione), oppure quando c'è una forte asimmetria positiva con outlier estremi. Utile anche per stabilizzare la varianza.

Caso Pratico: Previsione Customer Churn

Mettiamo in pratica tutto ciò che abbiamo imparato con un caso reale: identificare i clienti a rischio di abbandono per attivare campagne di retention mirate e ridurre il churn.

Obiettivo Business

Identificare i clienti a rischio di abbandono **prima** che smettano di acquistare, permettendo interventi di retention tempestivi e mirati.

Variabile Target (Y)

is_churn = variabile binaria (0/1)

```
is_churn = 1
SE giorni_da_ultimo_ordine > 90
ALTRIMENTI 0
```

Definiamo "churned" un cliente che non acquista da oltre 90 giorni.

Feature Estratte (X) - Input per il Modello

- **Recency (giorni)**

  Giorni trascorsi dall'ultimo acquisto — *feature temporale critica*

- **Frequency (conteggio)**

  Numero totale di ordini effettuati — *indica engagement*

- **Monetary (€)**

  Importo totale speso (Lifetime Value) — *valore del cliente*

- **Scontrino Medio (€)**

  Valore medio per ordine (AOV) — *comportamento di spesa*

- **Anzianità (giorni)**

  Giorni dal primo acquisto (tenure) — *fedeltà storica*

- **Regione**

  Area geografica (categorica) — *contesto territoriale*

Nelle prossime unità del corso (IA.5-IA.7) imparerete a scegliere gli algoritmi più adatti, ottimizzare i modelli ML e validare le predizioni per questo problema di classificazione binaria.

Best Practice del Feature Engineering

Il Feature Engineering efficace richiede un approccio metodico e disciplinato. Seguire queste best practice garantisce risultati robusti, riproducibili e di qualità nel tempo.

Inizia Sempre dall'Analisi Esplorativa dei Dati (EDA)

Prima di creare qualsiasi feature, dedica tempo a comprendere profondamente i tuoi dati. Utilizza visualizzazioni (istogrammi, scatter plot, heatmap di correlazione) e statistiche descrittive per scoprire pattern, distribuzioni, outlier e relazioni tra variabili. L'EDA guida tutte le decisioni successive.

Scegli il Modello in Anticipo (se possibile)

La scelta dell'algoritmo guida il tipo di preprocessing necessario. I modelli lineari (Regressione Logistica, SVM) richiedono scaling e normalizzazione, mentre gli alberi decisionali (Random Forest, XGBoost) sono robusti alle diverse scale. Sapere quale modello userai ti fa risparmiare tempo.

Crea Feature di Interazione per Scoprire Pattern Nascosti

Non limitarti alle feature esistenti. Combina variabili attraverso operazioni matematiche (rapporti, prodotti, differenze) per catturare relazioni non lineari che il modello potrebbe non scoprire autonomamente. Esempio: area = lunghezza × larghezza, o spesa_per_giorno = monetary / tenure.

Valuta l'Impatto di Ogni Nuova Feature

Non tutte le feature migliorano il modello. Usa tecniche come Permutation Importance, SHAP values o validazione incrociata per verificare se una nuova feature aumenta effettivamente la performance. Elimina le feature ridondanti o dannose per mantenere il modello semplice ed efficiente.

Documenta e Versiona le Tue Feature

Tratta le feature come codice. Mantieni un **feature registry** con descrizione, logica di calcolo, data di creazione e versione. Usa notebook o script versionati su Git. Questo garantisce riproducibilità, facilita la collaborazione e permette di rollback in caso di problemi.

> #### Principio Fondamentale
>
> Il Feature Engineering non è un processo lineare ma **iterativo**. Sperimenta, misura, apprendi e raffina continuamente. Le migliori feature emergono dalla combinazione di conoscenza del dominio business e sperimentazione guidata dai dati.

Riepilogo e Prossimi Passi

Abbiamo esplorato il cuore del Machine Learning applicato: il Feature Engineering. Ricapitolare i concetti chiave ci aiuta a consolidare l'apprendimento e a prepararci per le prossime fasi del percorso.

Concetti Chiave Appresi

1 DWH come Fondamenta

Il Data Warehouse fornisce dati puliti, storicizzati e integrati che costituiscono la base per il Machine Learning aziendale

2 Feature Engineering

Trasformare i dati grezzi in feature significative è **la base** di ogni progetto ML di successo — vale l'80% del lavoro

3 Pipeline di Preprocessing

Gestire missing values, encoding categorico e scaling in modo sistematico garantisce riproducibilità e qualità

Best Practices da Ricordare

Inizia dalle Feature

Dedica tempo alla creazione di feature di qualità **prima** di scegliere o ottimizzare il modello

Evita il Data Leakage

Non usare informazioni dal futuro. Le feature devono essere calcolabili al momento della predizione reale

Continua ad Imparare

Questa è solo l'introduzione. Le unità IA.5-IA.7 approfondiranno modelli avanzati e tecniche di ottimizzazione

Il Feature Engineering è dove l'esperienza di dominio incontra la scienza dei dati

La conoscenza del business unita alla capacità tecnica crea feature potenti che trasformano dati in valore

Unità IA.5

Modelli di Machine Learning: Regressione, Classificazione, Ensemble Methods

Unità IA.6

Reti Neurali e Deep Learning: Architetture, Training, Applicazioni

Unità IA.7

Clustering e Serie Temporali: Segmentazione, Forecasting, Pattern Recognition