

Imparare dai Dati: La Regressione Lineare

Capire e Prevedere il Futuro con i Numeri

Corso di Python Avanzato

24 novembre 2025

Concetti Chiave

- Come insegnare al computer a trovare tendenze.
- La matematica dietro la "linea migliore".
- Gestire dati complessi e curve.

Pratica

- Preparare i dati (Train/Test Split).
- Evitare l'apprendimento a memoria (Overfitting).
- Interpretare i risultati correttamente.

Cos'è un Dataset?

La Tabella dei Dati

- **Righe (Esempi):** Ogni riga è una storia (es. una casa venduta).
- **Colonne (Feature):** Le caratteristiche (es. mq, zona).
- **Target (Obiettivo):** Quello che vogliamo indovinare (es. prezzo).

Esempio Immobiliare

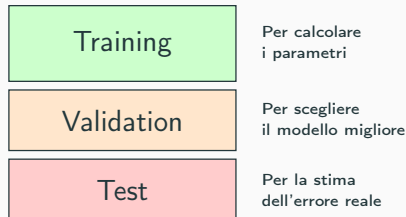
Grandezza (x_1)	Stanze (x_2)	Prezzo (y)
50m ²	2	150k
80m ²	3	220k
120m ²	4	300k

Prima di Iniziare: La Regola dell'Esame

Non possiamo valutare lo studente sulle stesse domande che ha studiato a casa.

Dividiamo i dati in tre parti:

1. **Training Set (60-70%):** I "libri di testo" per imparare.
2. **Validation Set (15-20%):** Le "simulazioni d'esame" per calibrare il modello mentre impara.
3. **Test Set (15-20%):** L'"esame finale". Dati mai visti, usati solo alla fine per il voto reale.

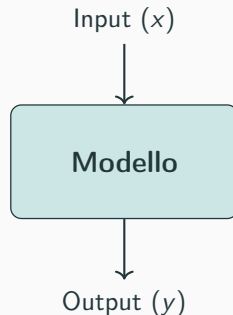


L'Obiettivo: Una "Scatola Magica"

Regressione Lineare Multipla

Vogliamo trovare una formula che combini tutti gli input per darci l'output.

Immaginiamo che ogni caratteristica abbia un "peso" specifico.



Sotto il Cofano: La Formula Matematica

Il modello è un'equazione somma pesata:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$$

- β_0 : Il punto di partenza (Intercetta).
- β_n : Quanto è importante la caratteristica n .

Come si trovano i β perfetti? Esiste una formula esatta ("Forma Chiusa" o Equazione Normale) che calcola tutto in un colpo solo:

$$\beta = (X^T X)^{-1} X^T y$$

Cosa significa?

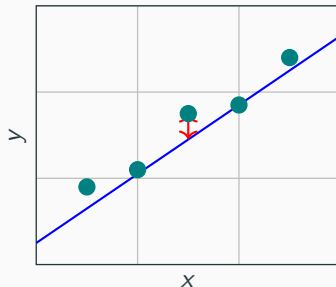
La formula usa l'algebra lineare per minimizzare l'errore istantaneamente.

Nota: Se i dati sono troppi (milioni), questa formula è lenta. In quel caso il computer va per tentativi ("Discesa del Gradiente").

Visualizzare l'Errore: Quanto Sbagliamo?

I Residui

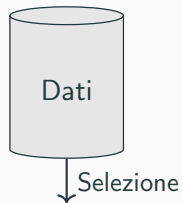
- La distanza tra il punto vero e la nostra previsione è l'**errore**.
- Il modello cerca i valori di β che rendono la somma dei quadrati di questi errori la più piccola possibile.



Scegliere gli Ingredienti Giusti (Features)

Non tutti i dati aiutano. Bisogna scegliere le feature correlate con l'obiettivo.

- **Segnale:** Metri quadri, Numero Bagni.
- **Rumore:** Colore della porta, Nome del venditore.



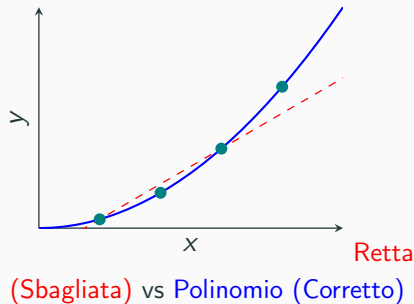
Quando la Retta non Basta: Feature Engineering

Se i dati fanno una curva, la retta sbaglia.

Il Trucco: Possiamo creare "finte" nuove colonne elevando al quadrato i dati originali.

Input originale: $[x]$ \rightarrow Input trasformato: $[x, x^2]$

Il modello lineare ora ha due "manopole" ($\beta_1 x + \beta_2 x^2$) e può disegnare curve (parabole), pur rimanendo matematicamente "lineare" nei parametri.



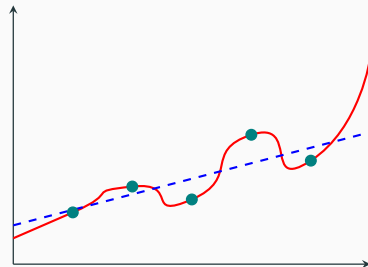
Attenzione: Il Troppo Stroppia (Overfitting)

Se aggiungiamo troppe feature trasformate (x^3, x^4, \dots, x^{10}), il modello diventa troppo flessibile.

Inizia a "unire i puntini" imparando a memoria il rumore dei dati invece della regola generale.

Risultato: Errore bassissimo sul Training Set, ma previsioni disastrose sui nuovi dati (Test Set).

Overfitting (Troppo complesso)



Overfitting vs Buon Modello

Come Farlo in Python

Usiamo scikit-learn per fare i calcoli pesanti.

1. **Importa** lo strumento.
2. **Addestra** (fit) sui dati storici.
3. **Prevedi** (predict) sui nuovi dati.

```
from sklearn.linear_model
    import LinearRegression

# 1. Crea il modello vuoto
modello = LinearRegression()

# 2. Addestra (Trova i Beta)
# X = [x1, x2...], y =
#     Prezzo
modello.fit(X_train, y_train
            )

# 3. Prevedi il futuro
prezzo = modello.predict(
    X_test)
```

Grazie per l'attenzione!

Domande?