# Appunti di Studio: Teoria e Algoritmi di Regressione

## SEZIONE 1: Teoria Generale della Regressione

### 1. Introduzione alla Regressione

La regressione è uno dei compiti più comuni e fondamentali nel campo del Machine Learning supervisionato.

- **Definizione di Regressione (come task di Machine Learning supervisionato):** La regressione è un tipo di problema di apprendimento supervisionato in cui l'obiettivo è prevedere un valore di output **continuo** (numerico reale) basandosi su una o più variabili di input. Il modello apprende una mappatura tra le variabili di input (feature) e la variabile di output (target) dai dati di addestramento.
- **Scopo principale e casi d'uso tipici:** Lo scopo primario della regressione è quello di modellare la relazione tra le variabili di input e la variabile di output per effettuare previsioni accurate su nuovi dati o per comprendere la forza e la direzione di tale relazione.
  - **Casi d'uso tipici:**
    - Previsione dei prezzi delle case in base a caratteristiche come dimensione, posizione, numero di stanze.
    - Stima del salario in base agli anni di esperienza, istruzione, ruolo.
    - Previsione delle vendite future di un prodotto.
    - Modellazione della temperatura, dell'umidità, dei livelli di inquinamento.
    - Previsione del tempo di consegna di un pacco.
- **Differenza chiave tra regressione e classificazione:**
  - **Regressione:** Prevede un output **continuo** (es. prezzo, temperatura, età).
  - **Classificazione:** Prevede un output **discreto** o **categorico** (es. sì/no, cane/gatto, spam/non-spam).
- **Concetti di variabile dipendente (target) e variabili indipendenti (features/predittori):**
  - **Variabile Dipendente (Target o Output, $y$):** È la variabile che stiamo cercando di prevedere o spiegare. Il suo valore "dipende" dai valori delle altre variabili.
  - **Variabili Indipendenti (Features, Predittori, Attributi, Input, $X$):** Sono le variabili utilizzate per prevedere o spiegare la variabile dipendente. I loro valori sono considerati "indipendenti" nel contesto del modello.

# 2. Concetti Fondamentali e Termini Chiave

Per comprendere appieno gli algoritmi di regressione, è essenziale padroneggiare alcuni concetti chiave.

- **Modelli Lineari vs. Non Lineari:**
  - **Modelli Lineari:** Assumono una relazione lineare tra le variabili indipendenti e la variabile dipendente. Ciò significa che la relazione può essere rappresentata da una linea retta, un piano o un iperpiano nello spazio delle feature. Sono semplici, interpretabili e veloci. (Es. Regressione Lineare, Ridge Regression).
  - **Modelli Non Lineari:** Sono in grado di catturare relazioni più complesse e non lineari tra le variabili. Possono modellare curve e pattern più articolati nei dati. (Es. Decision Tree, Random Forest, XGBoost, Support Vector Machines non lineari).
- **Overfitting e Underfitting:**
  - **Overfitting (Sovraccarico):** Si verifica quando un modello è troppo complesso e si adatta eccessivamente ai dati di addestramento, imparando anche il "rumore" o le fluttuazioni casuali presenti nei dati specifici di addestramento. Questo porta a un'ottima performance sui dati di addestramento, ma a una performance scarsa e una bassa capacità di generalizzazione su nuovi dati non visti (dati di test). Il modello ha un'alta varianza.
  - **Underfitting (Sotto-carico):** Si verifica quando un modello è troppo semplice per catturare la struttura sottostante e le relazioni significative nei dati. Non riesce ad apprendere sufficientemente bene dai dati di addestramento e di conseguenza ha una performance scadente sia sui dati di addestramento che su quelli di test. Il modello ha un alto bias.
  - **Come riconoscerli e strategie generali per mitigarli:**
    - **Riconoscere:**
      - **Overfitting:** Basso errore di addestramento, alto errore di test/validazione.
      - **Underfitting:** Alto errore di addestramento e alto errore di test/validazione.
    - **Mitigare:**
      - **Overfitting:** Aumentare la quantità di dati di addestramento, ridurre la complessità del modello (es. minore `max_depth` in alberi, meno feature, regolarizzazione), usare tecniche di regolarizzazione (L1, L2), cross-validazione, dropout (per reti neurali).
      - **Underfitting:** Aumentare la complessità del modello (es. più feature, modelli non lineari, maggiore `max_depth` ), aggiungere più feature rilevanti, ridurre la regolarizzazione, migliorare la qualità dei dati.
- **Bias-Variance Trade-off:**
  - Questo è un concetto fondamentale per comprendere le prestazioni dei modelli di Machine Learning. L'errore totale di un modello può essere scomposto in tre parti: **Bias, Varianza** e **Errore irriducibile** (rumore intrinseco nei dati).

- **Bias (Errore da Ipotesi Sbagliate):** Rappresenta l'errore che deriva da ipotesi errate o eccessivamente semplificate nel modello, che lo rendono incapace di catturare la vera relazione tra input e output. Un modello con alto bias è tipicamente semplice e soffre di *underfitting*.
- **Varianza (Sensibilità ai Dati Specifici):** Rappresenta l'errore che deriva dalla sensibilità del modello alle piccole fluttuazioni nei dati di addestramento. Un modello con alta varianza è tipicamente complesso e soffre di *overfitting*, performando bene solo sui dati specifici su cui è stato addestrato.
- **Il Compromesso (Trade-off):** C'è una relazione inversa tra bias e varianza.
  - Se si aumenta la complessità del modello, generalmente si riduce il bias (il modello diventa più flessibile e capace di catturare relazioni complesse), ma si aumenta la varianza (il modello diventa più sensibile al rumore e alle specificità del set di addestramento).
  - Se si riduce la complessità del modello, generalmente si aumenta il bias (il modello diventa troppo rigido per catturare tutte le relazioni), ma si riduce la varianza (il modello diventa più stabile e meno sensibile alle variazioni nei dati di addestramento).
- **Importanza:** L'obiettivo è trovare un equilibrio ottimale tra bias e varianza che minimizzi l'errore totale su dati non visti. Questo spesso si traduce nella scelta del modello giusto e nella corretta regolazione degli *hyperparameters*.
- **Come influisce sulle scelte del modello:** Modelli lineari hanno generalmente un bias più alto e una varianza più bassa. Modelli complessi come alberi di decisione profondi o reti neurali hanno un bias più basso ma una varianza più alta. Tecniche di regolarizzazione (Ridge, Lasso) e ensemble (Random Forest, Boosting) sono spesso usate per gestire questo compromesso.

- **Funzioni di Costo (Loss Functions):**
  - **Spiegazione generale del loro ruolo nell'ottimizzazione:** Una funzione di costo (o funzione di perdita) quantifica l'errore tra i valori previsti dal modello ($\hat{y}$) e i valori reali (ground truth, $y$). Durante il processo di addestramento, l'algoritmo cerca di minimizzare questa funzione di costo, adattando i parametri del modello. Un valore più basso della funzione di costo indica un modello che sta "imparando" meglio dai dati.
  - **Dettaglio su Mean Squared Error (MSE), Mean Absolute Error (MAE) e Root Mean Squared Error (RMSE):**
    - **Mean Squared Error (MSE):**
      - **Formula:** $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
      - **Significato:** È la media dei quadrati delle differenze tra i valori reali e i valori predetti.
      - **Quando usarla:** È la funzione di costo più comune per la regressione. Penalizza maggiormente gli errori grandi (poiché sono al quadrato), il che la rende sensibile agli *outlier*. È differenziabile, il che la rende adatta per l'ottimizzazione basata sul gradiente.
    - **Mean Absolute Error (MAE):**
      - **Formula:** $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

        - **Significato:** È la media dei valori assoluti delle differenze tra i valori reali e i valori predetti.
        - **Quando usarla:** È più robusta agli *outlier* rispetto al MSE, perché tratta tutti gli errori linearmente. È preferibile quando gli outlier sono presenti e non si vuole che influenzino eccessivamente il modello. Tuttavia, la funzione valore assoluto non è differenziabile in 0, il che può rendere l'ottimizzazione più complessa in alcuni contesti.
    - **Root Mean Squared Error (RMSE):**
        - **Formula:** $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} = \sqrt{MSE}$
        - **Significato:** È la radice quadrata del MSE.
        - **Quando usarla:** Ha lo stesso significato del MSE ma è espressa nella stessa unità di misura della variabile dipendente, rendendola più interpretabile. Se il tuo target è in Euro, il RMSE sarà in Euro.
- **Metriche di Valutazione:**
  Le metriche di valutazione sono usate per quantificare le performance di un modello su dati non visti, dopo che è stato addestrato.
    - **Spiegazione di R-squared (coefficiente di determinazione) e Adjusted R-squared:**
        - **R-squared ($R^2$):**
            - **Formula:** $R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$
                - $SS_{res}$ (Sum of Squares of Residuals): somma dei quadrati degli errori del modello.
                - $SS_{tot}$ (Total Sum of Squares): somma dei quadrati delle differenze tra i valori reali e la media dei valori reali. Rappresenta la varianza totale della variabile dipendente.
            - **Interpretazione:** Indica la proporzione della varianza nella variabile dipendente che è prevedibile dalle variabili indipendenti. Un $R^2$ di 0.75 significa che il 75% della varianza della variabile dipendente è spiegata dal modello. Varia tipicamente tra 0 e 1, ma può essere negativo per modelli molto scarsi (peggiori di un modello che predice sempre la media).
            - **Limitazioni:** $R^2$ ha una limitazione significativa: aumenta o rimane uguale all'aggiunta di nuove variabili indipendenti al modello, anche se queste variabili non sono correlate con la variabile dipendente. Questo può dare un'illusione di miglioramento.
        - **Adjusted R-squared ($R^2_{adj}$):**
            - **Formula:** $R^2_{adj} = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$
                - $n$: numero di osservazioni
                - $p$: numero di predittori (variabili indipendenti)
            - **Interpretazione:** È una versione modificata di $R^2$ che penalizza l'aggiunta di predittori non utili al modello. Aumenta solo se la nuova variabile migliora

significativamente il modello (riducendo l'errore più di quanto ci si aspetterebbe per puro caso). È sempre inferiore o uguale a $R^2$. È preferibile per confrontare modelli con un numero diverso di predittori.
        - **Limitazioni:** Non esiste un valore di "buon" $R^2_{adj}$ universale; dipende dal campo di applicazione.
    - **Accenno ad altre metriche come MAE, MSE, RMSE come metriche di performance:** Le funzioni di costo MSE, MAE e RMSE, oltre ad essere usate per l'ottimizzazione del modello, sono anche le metriche più comunemente usate per valutare le prestazioni del modello su set di dati di test. Un valore più basso indica una migliore performance.

## 3. Assunzioni della Regressione Lineare (introduzione al contesto)

Le assunzioni della regressione lineare sono importanti perché, se rispettate, garantiscono che le stime dei coefficienti siano "BLUE" (Best Linear Unbiased Estimator). La violazione di queste assunzioni non invalida necessariamente il modello, ma può compromettere l'affidabilità delle inferenze statistiche (intervalli di confidenza, p-value).

Breve elenco:

1. **Linearità:** La relazione tra le variabili indipendenti e la variabile dipendente è lineare.
2. **Indipendenza degli Errori:** Gli errori (residui) sono indipendenti l'uno dall'altro. Non c'è correlazione tra i residui.
3. **Omoschedasticità:** La varianza degli errori è costante per tutti i livelli delle variabili indipendenti (varianza costante dei residui).
4. **Normalità dei Residui:** Gli errori sono distribuiti normalmente attorno a una media di zero.
5. **Assenza di Multicollinearità:** Le variabili indipendenti non sono altamente correlate tra loro.

Queste assunzioni saranno esplorate in dettaglio nella sezione dedicata alla Regressione Lineare.

# SEZIONE 2: Metodi di Regressione Specifici

Questa sezione descrive in dettaglio il funzionamento, i pro, i contro e l'uso pratico di algoritmi di regressione chiave.

# 1. Linear Regression (Regressione Lineare)

- **1. Definizione e Obiettivo:**
  La Regressione Lineare è un algoritmo statistico e di Machine Learning supervisionato che modella la relazione tra una variabile dipendente ($y$) e una o più variabili indipendenti ($X$) come una funzione lineare. L'obiettivo è trovare i coefficienti (pesi) che meglio descrivono questa relazione lineare, in modo da poter prevedere la variabile dipendente per nuovi input.
- **2. Teoria e Funzionamento Approfondito:**
  - **Modello Matematico:** Per una regressione lineare multipla con $p$ variabili indipendenti, il modello è:
    $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon$
    - $\hat{y}$: valore previsto della variabile dipendente.
    - $\beta_0$: l'intercetta (il valore di $\hat{y}$ quando tutti gli $x_i$ sono zero).
    - $\beta_1, \ldots, \beta_p$: i coefficienti di regressione (pendenze), che indicano quanto cambia $\hat{y}$ per una variazione unitaria di $x_i$, mantenendo costanti le altre variabili.
    - $x_1, \ldots, x_p$: le variabili indipendenti (features).
    - $\epsilon$: il termine di errore (residuo), che rappresenta la parte di $y$ che non può essere spiegata dal modello.
  - **Metodo dei Minimi Quadrati Ordinari (OLS - Ordinary Least Squares):**
    - **Intuizione:** L'OLS è il metodo più comune per stimare i coefficienti $\beta_0, \ldots, \beta_p$. Il suo obiettivo è trovare la "migliore" retta (o piano/iperpiano) che si adatta ai punti dati, minimizzando la somma dei quadrati delle differenze verticali tra i valori osservati ($y_i$) e i valori predetti ($\hat{y}_i$) dal modello. Queste differenze verticali sono chiamate *residui*.
    - **Funzione di Costo (MSE):** L'OLS minimizza la Mean Squared Error (o Sum of Squared Residuals):
      $Loss(OLS) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - (\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}))^2$
    - **Soluzione Analitica:** Per i modelli lineari, esiste una soluzione in forma chiusa per i coefficienti $\beta$. Nel caso matriciale, la formula è $\hat{\beta} = (X^T X)^{-1} X^T y$. Questa formula deriva dall'impostazione a zero delle derivate parziali della funzione di costo rispetto a ciascun coefficiente e risolvendo il sistema di equazioni.
  - **Concetti di pendenza e intercetta:**
    - **Intercetta ($\beta_0$):** Rappresenta il valore atteso della variabile dipendente quando tutte le variabili indipendenti sono uguali a zero. Il suo significato pratico dipende dal contesto dei dati; a volte può non avere un significato realistico se zero non è un valore plausibile per le feature.
    - **Pendenza ($\beta_j$):** Ogni coefficiente $\beta_j$ rappresenta il cambiamento atteso nella variabile dipendente per ogni aumento unitario della variabile indipendente $x_j$, assumendo che tutte

le altre variabili indipendenti rimangano costanti. È una misura della forza e della direzione della relazione lineare tra $x_j$ e $y$.

- **Assunzioni complete e cosa succede se vengono violate:**
  a. **Linearità:** La relazione tra $X$ e $y$ deve essere lineare.
     - **Violazione:** Se la relazione è non lineare, il modello lineare non la catturerà adeguatamente, portando a previsioni errate e alto bias.
     - **Soluzione:** Trasformazione delle variabili (log, radice quadrata), utilizzo di modelli non lineari.
  b. **Indipendenza degli Errori:** I residui devono essere indipendenti l'uno dall'altro (es. non c'è correlazione seriale in dati temporali).
     - **Violazione:** Errore standard sottostimato, p-values errati.
     - **Soluzione:** Utilizzare modelli time-series specifici, clustering degli errori.
  c. **Omoschedasticità (Varianza Costante degli Errori):** La varianza dei residui deve essere costante per tutti i livelli delle variabili indipendenti.
     - **Violazione (Eteroschedasticità):** Se la varianza degli errori non è costante, gli estimatori OLS sono ancora imparziali ma inefficienti (non hanno la minima varianza), portando a intervalli di confidenza e p-values inaccurati.
     - **Soluzione:** Trasformazione della variabile dipendente, regressione ponderata (WLS), errori standard robusti.
  d. **Normalità dei Residui:** I residui devono essere approssimativamente distribuiti normalmente con media zero.
     - **Violazione:** Influisce sulla validità dei test di ipotesi (p-value, intervalli di confidenza), specialmente per campioni piccoli. Per campioni grandi, il Teorema del Limite Centrale mitiga il problema.
     - **Soluzione:** Trasformazione delle variabili, metodi non parametrici o robusti.
  e. **Assenza di Multicollinearità:** Le variabili indipendenti non devono essere altamente correlate tra loro.
     - **Violazione:** Difficoltà nel distinguere l'effetto individuale di ogni predittore, stime dei coefficienti instabili e ad alta varianza, difficoltà nell'interpretazione dei p-value.
     - **Soluzione:** Rimuovere una delle variabili correlate, combinare le variabili correlate, utilizzare tecniche di regolarizzazione (Ridge Regression), PCA.

- **3. Vantaggi e Svantaggi:**
  - **Vantaggi:**
    - **Semplice e Veloce:** Facile da capire e implementare, computazionalmente efficiente.
    - **Interpretabile:** I coefficienti forniscono una chiara comprensione della direzione e della forza della relazione tra ogni feature e il target.
    - **Baseline:** Spesso usato come modello di base (baseline) per confrontare le prestazioni di modelli più complessi.

- **Svantaggi:**
  - **Assunzione di Linearità:** Richiede che la relazione tra le variabili sia lineare, non adatto per relazioni complesse o non lineari.
  - **Sensibile agli Outlier:** La minimizzazione della somma dei quadrati degli errori rende il modello molto sensibile agli outlier, che possono distorcere significativamente i coefficienti.
  - **Multicollinearità:** Può soffrire di problemi di multicollinearità tra le feature, rendendo i coefficienti instabili e difficili da interpretare.
- **4. Hyperparameters Chiave:**
  La Regressione Lineare OLS non ha hyperparameters da ottimizzare nel senso tipico degli algoritmi di ML moderni, poiché la soluzione è deterministica. Si possono considerare aspetti come la gestione delle variabili categoriche (codifica one-hot) o la standardizzazione delle feature, che sono passi di pre-elaborazione dati.
- **5. Casi d'Uso Tipici:**
  - Previsioni semplici (es. prezzi delle case, vendite future) quando la relazione è approssimativamente lineare.
  - Analisi esplorativa dei dati per comprendere la relazione tra le variabili.
  - Come modello di riferimento o baseline prima di applicare modelli più complessi.
  - Per l'inferenza statistica, quando si vuole quantificare l'impatto di specifiche variabili sul target.
- **6. Relazione con altri Modelli:**
  La Regressione Lineare è il fondamento per molti altri modelli lineari. La Ridge Regression e la Lasso Regression (non richiesta in dettaglio, ma da menzionare) sono estensioni della Regressione Lineare che aggiungono termini di regolarizzazione per gestire l'overfitting e la multicollinearità.

# 2. Ridge Regression (Regressione Ridge)

- **1. Definizione e Obiettivo:**
  La Ridge Regression è un'estensione della Regressione Lineare progettata per affrontare i problemi di overfitting e multicollinearità. Lo fa aggiungendo un termine di penalizzazione alla funzione di costo OLS, che "stringe" i coefficienti del modello verso lo zero (ma non esattamente a zero).
- **2. Teoria e Funzionamento Approfondito:**
  - **Concetto di regolarizzazione L2:** La Ridge Regression incorpora la regolarizzazione L2 (anche detta "Tikhonov regularization"). Questo significa che alla funzione di costo OLS (MSE) viene aggiunto un termine di penalizzazione proporzionale alla somma dei quadrati dei coefficienti del modello.
  - **Funzione di Costo (Ridge):**
    $Loss_{Ridge} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

        - Il primo termine è la somma dei quadrati dei residui (come in OLS).
        - Il secondo termine, $\lambda \sum_{j=1}^{p} \beta_j^2$, è il termine di penalizzazione L2.
            - $\lambda$ (lambda, spesso denotato come `alpha` in librerie come scikit-learn) è il *parametro di penalizzazione*.
            - $\sum_{j=1}^{p} \beta_j^2$ è la somma dei quadrati dei coefficienti (esclusa l'intercetta $\beta_0$, che di solito non viene regolarizzata).
    - **Come mitiga l'overfitting e gestisce la multicollinearità:**
        - **Mitigazione Overfitting:** Penalizzando i coefficienti grandi, la Ridge Regression impedisce che il modello si adatti eccessivamente al rumore nei dati di addestramento. Questo porta a una riduzione della varianza del modello, a costo di un piccolo aumento del bias. Invece di permettere ai coefficienti di diventare molto grandi per adattarsi perfettamente a pochi punti dati, li forza a rimanere più piccoli e più stabili.
        - **Gestione Multicollinearità:** Quando due o più variabili predittive sono altamente correlate (multicollinearità), la Regressione Lineare OLS può avere difficoltà a determinare l'effetto unico di ciascuna variabile, portando a coefficienti molto grandi e instabili. La Ridge Regression distribuisce l'effetto tra le variabili correlate, riducendo l'ampiezza dei coefficienti e stabilizzando il modello.
    - **Parametro di penalizzazione ($\lambda$ o alpha): ruolo e impatto:**
        - **Ruolo:** $\lambda$ è l'hyperparameter chiave della Ridge Regression. Controlla l'intensità della penalizzazione.
        - **Impatto:**
            - Se $\lambda = 0$: La Ridge Regression si riduce alla Regressione Lineare OLS (nessuna penalizzazione).
            - Se $\lambda$ è molto grande: I coefficienti $\beta_j$ saranno spinti sempre più verso zero. Il modello diventerà troppo semplice, con alto bias e bassa varianza (potenziale underfitting).
            - Un valore $\lambda$ ottimale (trovato tramite cross-validazione) bilancerà il bias e la varianza per ottenere le migliori prestazioni predittive.
- **3. Vantaggi e Svantaggi:**
    - **Vantaggi:**
        - **Riduce Overfitting:** Aiuta a prevenire l'overfitting e a migliorare la generalizzazione del modello.
        - **Gestisce Multicollinearità:** Migliora la stabilità e l'affidabilità delle stime dei coefficienti in presenza di feature altamente correlate.
        - **Stime Stabili:** Fornisce stime dei coefficienti più stabili rispetto all'OLS quando i dati sono rumorosi o le feature sono multicollineari.
    - **Svantaggi:**
        - **Meno Interpretabile:** I coefficienti sono ridotti (shrunk) verso zero, rendendo l'interpretazione diretta della loro grandezza più complessa rispetto all'OLS.

        - **Non Effettua Selezione delle Feature:** Ridge riduce i coefficienti ma non li azzera mai completamente. Pertanto, non esegue la selezione automatica delle feature (a differenza della Lasso Regression).
        - **Richiede Scaling:** È sensibile alla scala delle feature. È fondamentale standardizzare o normalizzare le variabili indipendenti prima di applicare la Ridge Regression, in modo che la penalizzazione sia applicata equamente a tutti i coefficienti.
- **4. Hyperparameters Chiave:**
    - `alpha` **($\lambda$):** Il parametro di penalizzazione. È l'hyperparameter più importante da regolare. Valori tipici vanno da $0$ a numeri molto grandi. Si esplora tipicamente una scala logaritmica (es. 0.01, 0.1, 1, 10, 100).
- **5. Casi d'Uso Tipici:**
    - Quando si ha un gran numero di predittori e si teme l'overfitting.
    - In presenza di multicollinearità tra le variabili indipendenti.
    - Quando si desidera un modello lineare più robusto e stabile rispetto all'OLS.
    - Come passo preliminare per la selezione di feature, sebbene non le azzeri.
- **6. Relazione con altri Modelli:**
    - **Regressione Lineare:** La Ridge Regression è una generalizzazione della Regressione Lineare. Quando il parametro di penalizzazione $\lambda$ è impostato a zero, la Ridge Regression degenera nella Regressione Lineare OLS.
    - **Lasso Regression:** Simile alla Ridge, ma usa la regolarizzazione L1 ($\lambda \sum |\beta_j|$). Lasso ha la proprietà di azzerare i coefficienti meno importanti, effettuando quindi la selezione delle feature, mentre Ridge li riduce solo. Elastic Net combina L1 e L2.

# 3. Decision Tree (Albero di Decisione per la Regressione)

- **1. Definizione e Obiettivo:**
  Un Albero di Decisione (Decision Tree) è un algoritmo di Machine Learning non-parametrico e supervisionato che apprende regole decisionali semplici dai dati e le rappresenta in una struttura ad albero. Per la regressione, l'obiettivo è prevedere un valore continuo segmentando lo spazio delle feature in una serie di regioni rettangolari e assegnando un valore previsto (tipicamente la media) a ciascuna regione.
- **2. Teoria e Funzionamento Approfondito:**
    - **Come avviene la suddivisione (splitting) dei nodi:**
      Un albero di decisione viene costruito in modo ricorsivo, dividendo ripetutamente il set di dati in sottoinsiemi più piccoli e omogenei.
      a. **Nodo Radice:** L'intero set di dati è il nodo radice.

b. **Suddivisione (Splitting):** Ad ogni nodo (non-foglia), l'algoritmo cerca la migliore *feature* e la migliore *soglia di suddivisione* (split point) per quella feature. La "migliore" suddivisione è quella che riduce al massimo l'impurità dei nodi figli risultanti.
c. **Criteri per la Regressione:** Per la regressione, i criteri più comuni per misurare l'impurità (o omogeneità) e guidare la suddivisione sono:
    - **Mean Squared Error (MSE) Reduction:** L'algoritmo sceglie la split point che minimizza il MSE dei campioni nei nodi figli. In altre parole, cerca di rendere i valori target all'interno di ogni nodo figlio il più simili possibile tra loro. Si calcola il MSE del nodo padre, e poi il MSE ponderato dei nodi figli, scegliendo lo split che massimizza la riduzione del MSE (o, equivalentemente, minimizza il MSE post-split).
    - **Mean Absolute Error (MAE) Reduction:** Simile all'MSE, ma minimizza la somma degli errori assoluti. Più robusto agli outlier.
    - **Varianza (Standard Deviation) Reduction:** Simile al MSE, ma basato sulla varianza o deviazione standard dei valori target.
d. **Crescita dell'Albero:** Il processo di splitting continua ricorsivamente per ogni nodo figlio fino a quando non si raggiungono le condizioni di arresto (es. profondità massima, numero minimo di campioni per foglia, nessuna ulteriore riduzione significativa dell'impurità).

- **Concetti di purezza dei nodi:**
    - Un nodo è considerato "puro" (o omogeneo) se tutti i campioni al suo interno hanno valori target molto simili. L'obiettivo delle divisioni è quello di aumentare la purezza dei nodi figli rispetto al nodo padre. Nel contesto della regressione, un nodo "puro" è un nodo in cui la varianza dei valori target è molto bassa.
- **Pre-potatura (Pre-pruning) e Post-potatura (Post-pruning):**
    - **Pre-potatura (Pre-pruning o Early Stopping):** Si impostano dei vincoli sulla crescita dell'albero *durante* la sua costruzione. Questo previene l'overfitting fermando la divisione dei nodi prima che l'albero diventi troppo complesso. Esempi di vincoli: `max_depth` (profondità massima dell'albero), `min_samples_split` (numero minimo di campioni necessari per dividere un nodo), `min_samples_leaf` (numero minimo di campioni che un nodo foglia deve avere), `min_impurity_decrease` (riduzione minima dell'impurità per effettuare una divisione).
    - **Post-potatura (Post-pruning o Cost-Complexity Pruning):** L'albero viene prima costruito completamente (fino a quando non ci sono più divisioni possibili o i nodi sono puri). Successivamente, i rami che non contribuiscono significativamente alla riduzione dell'errore (o che portano ad overfitting) vengono rimossi o "potati" indietro, spesso utilizzando una metrica di costo-complessità e un set di validazione.
- **Sensibilità al rumore e all'overfitting:**
    - Gli alberi di decisione, specialmente se lasciati crescere senza vincoli (non potati), sono altamente sensibili al rumore nei dati di addestramento.

        - Sono molto propensi all'overfitting perché possono creare rami molto specifici per adattarsi perfettamente a ogni punto dato di addestramento, inclusi gli outlier e il rumore. Questo porta a una varianza molto alta.
- **3. Vantaggi e Svantaggi:**
    - **Vantaggi:**
        - **Interpretabilità:** Facili da comprendere e visualizzare (se non troppo profondi). Le regole decisionali sono chiare.
        - **Gestisce Relazioni Non Lineari:** Non assume una relazione lineare tra le feature e il target.
        - **Nessuna Scalatura Richiesta:** Non è sensibile alla scalatura delle feature.
        - **Gestisce Dati Misti:** Può gestire variabili numeriche e categoriche contemporaneamente.
        - **Gestisce Outlier:** Relativamente robusto agli outlier nel dataset, a meno che non influenzino una divisione chiave.
    - **Svantaggi:**
        - **Overfitting:** Molto inclini all'overfitting, specialmente senza potatura, il che porta ad alta varianza.
        - **Instabilità:** Piccoli cambiamenti nei dati di addestramento possono portare a una struttura dell'albero molto diversa (alta varianza).
        - **Bias verso Feature Dominanti:** Possono avere difficoltà quando alcune feature sono molto più informative di altre, tendendo a favorire le prime divisioni basate su queste.
        - **Non Ottimali Globalmente:** La strategia di costruzione è "greedy" (migliore split al passo corrente), non garantisce la struttura ad albero globalmente ottimale.
- **4. Hyperparameters Chiave:**
    - `max_depth`: Profondità massima dell'albero. Limitarla aiuta a prevenire l'overfitting.
    - `min_samples_split`: Numero minimo di campioni che un nodo deve avere per poter essere diviso ulteriormente.
    - `min_samples_leaf`: Numero minimo di campioni che un nodo foglia deve avere.
    - `max_features`: Numero di feature da considerare quando si cerca la migliore divisione. (Importante per Random Forest, meno per un singolo albero).
    - `criterion`: La funzione per misurare la qualità di una divisione (es. "mse", "mae", "poisson").
- **5. Casi d'Uso Tipici:**
    - Quando è richiesta una forte interpretabilità del modello.
    - Come componente di algoritmi ensemble (Random Forest, Gradient Boosting).
    - Esplorazione dei dati per identificare regole decisionali o relazioni chiave.
- **6. Relazione con altri Modelli:**
    - **Random Forest:** I Decision Tree sono i "blocchi fondamentali" del Random Forest. Un Random Forest aggrega le previsioni di molti alberi di decisione per migliorare la robustezza e ridurre l'overfitting.

- **Gradient Boosting (XGBoost):** Anche gli algoritmi di boosting utilizzano alberi di decisione come "weak learners" (apprenditori deboli), ma li costruiscono sequenzialmente per correggere gli errori dei modelli precedenti.

# 4. Random Forest (Foreste Casuali per la Regressione)

- **1. Definizione e Obiettivo:**
  Random Forest è un algoritmo di Machine Learning basato su *ensemble learning* che costruisce un "foresta" di alberi di decisione durante l'addestramento e produce la previsione che è la media (per regressione) delle previsioni dei singoli alberi. L'obiettivo è ridurre l'overfitting e migliorare la robustezza e l'accuratezza rispetto a un singolo albero di decisione.
- **2. Teoria e Funzionamento Approfondito:**
  - **Concetto di "Ensemble Learning" e "Bagging" (Bootstrap Aggregating):**
    - **Ensemble Learning:** È una tecnica che combina le previsioni di più modelli per ottenere una previsione complessiva più robusta e accurata di quella che si otterrebbe con un singolo modello.
    - **Bagging (Bootstrap Aggregating):** È un metodo ensemble che riduce la varianza in algoritmi di apprendimento instabili (come gli alberi di decisione). Funziona così:
      a. Si creano più subset del dataset originale mediante *campionamento bootstrap* (campionamento con reintroduzione).
      b. Si addestra un modello separato su ciascun subset.
      c. Le previsioni dei singoli modelli vengono aggregate (per la regressione, di solito si calcola la media delle previsioni).
  - **Come vengono costruiti i singoli alberi nella foresta:**
    Un Random Forest costruisce $N$ alberi di decisione indipendenti (o quasi indipendenti) seguendo questi passaggi per ciascun albero:
    a. **Campionamento delle righe (Bootstrap Sampling):** Viene estratto un sottoinsieme casuale (con reintroduzione) del dataset di addestramento. Questo sottoinsieme ha la stessa dimensione del dataset originale ma conterrà alcune righe duplicate e alcune non presenti.
    b. **Campionamento delle colonne (Feature Randomness):** Ad ogni nodo di ogni albero, invece di considerare tutte le feature per trovare la migliore divisione, viene selezionato solo un *sottoinsieme casuale* delle feature. La migliore divisione viene trovata solo tra questo sottoinsieme.
    c. **Crescita dell'Albero (Spesso non potato):** Ogni albero di decisione viene tipicamente fatto crescere fino alla massima profondità possibile, senza potatura. La riduzione della

varianza si ottiene attraverso la diversità intrinseca degli alberi e l'aggregazione, piuttosto che dalla potatura.

- **Il ruolo della casualità (campionamento delle righe e delle colonne):**
  - **Campionamento delle righe (Bagging):** Introduce diversità tra gli alberi perché ognuno vede un set di dati leggermente diverso. Questo riduce la correlazione tra gli alberi, poiché è meno probabile che tutti commettano gli stessi errori.
  - **Campionamento delle colonne (Feature Randomness):** Questo è il "Random" di Random Forest. Ulteriormente decorrela gli alberi, perché impedisce a una o poche feature molto forti di dominare la parte superiore di tutti gli alberi. Se si avesse una feature dominante, senza questo campionamento delle colonne, tutti gli alberi sarebbero molto simili, e l'ensemble non ridurrebbe significativamente la varianza.
- **Come viene fatta la previsione finale per la regressione (media delle previsioni individuali):**
  Per fare una previsione su un nuovo campione, si fa passare il campione attraverso *tutti* gli alberi della foresta. Ogni albero produce la sua previsione. La previsione finale del Random Forest per la regressione è la **media** delle previsioni prodotte da tutti i singoli alberi. Questo processo di mediazione riduce la varianza complessiva del modello.
- **Vantaggi nel ridurre l'overfitting rispetto a un singolo Decision Tree:**
  Un singolo Decision Tree è instabile e prone all'overfitting (alta varianza). Random Forest mitiga questo problema in due modi principali:
  a. **Riduzione Varianza (Bagging):** Averaging delle previsioni di molti alberi indipendenti (o debolmente correlati) aiuta a ridurre la varianza complessiva del modello. Le previsioni individuali degli alberi, sebbene possano essere rumorose o tendere all'overfitting sui loro specifici subset di dati, vengono mediate, e il rumore o le specificità si "annullano" a vicenda, portando a una previsione più stabile e generalizzabile.
  b. **Decorrelazione degli Alberi (Feature Randomness):** La selezione casuale delle feature ad ogni split node assicura che gli alberi non siano troppo simili tra loro, anche se addestrati su dati simili. Questo è cruciale perché la riduzione della varianza tramite mediazione è più efficace quando i modelli aggregati sono il più possibile indipendenti.

- **3. Vantaggi e Svantaggi:**
  - **Vantaggi:**
    - **Alta Accuratezza e Robustezza:** Generalmente fornisce ottime prestazioni ed è molto robusto al rumore e agli outlier.
    - **Riduzione Overfitting:** Eccellente nel mitigare l'overfitting grazie all'ensemble e alla casualità.
    - **Gestisce Alta Dimensionalità:** Può gestire dataset con un gran numero di feature senza problemi di multicollinearità.

        - **Importanza delle Feature:** Permette di calcolare l'importanza delle feature, indicando quali variabili sono più rilevanti per la previsione.
        - **Gestisce Dati Misti e Mancanti:** Non richiede scalatura delle feature e può gestire variabili categoriche e numeriche, oltre a valori mancanti.
    - **Svantaggi:**
        - **Meno Interpretabile:** Sebbene i singoli alberi siano interpretabili, l'insieme di centinaia di alberi rende il modello "black box" e meno interpretabile nel suo complesso rispetto a un singolo albero o una regressione lineare.
        - **Intensivo Computazionalmente:** Può essere lento da addestrare e prevedere su dataset molto grandi o con un numero molto elevato di alberi.
        - **Più Lento del Boosting:** Generalmente più lento degli algoritmi di boosting per ottenere risultati simili.
- **4. Hyperparameters Chiave:**
    - `n_estimators`: Il numero di alberi nella foresta. Un numero maggiore generalmente migliora le prestazioni ma aumenta il tempo di addestramento.
    - `max_features`: Il numero di feature da considerare ad ogni split node. Controlla la casualità e la decorrelazione degli alberi. Un valore comune è `sqrt(n_features)` o `log2(n_features)`.
    - `max_depth`: La profondità massima degli alberi. Spesso gli alberi vengono fatti crescere profondamente (`None` in scikit-learn per profondità illimitata) per catturare pattern complessi, affidandosi all'ensemble per ridurre l'overfitting.
    - `min_samples_split` / `min_samples_leaf`: Controllano la dimensione minima dei nodi e delle foglie degli alberi individuali.
- **5. Casi d'Uso Tipici:**
    - Problemi di regressione ad alta prestazione e robustezza.
    - Quando l'interpretabilità esatta del singolo coefficiente non è la priorità assoluta.
    - In presenza di un gran numero di feature o di interazioni complesse tra di esse.
    - Quando si desidera una stima dell'importanza delle feature.
- **6. Relazione con altri Modelli:**
    - **Decision Tree:** È l'elemento costitutivo fondamentale di Random Forest. Random Forest aggrega molti alberi per superare i limiti di un singolo albero (instabilità, overfitting).
    - **Bagging:** Random Forest è un'applicazione specifica del concetto di Bagging, con l'aggiunta della casualità delle feature.
    - **Gradient Boosting (XGBoost):** Mentre Random Forest utilizza il "bagging" (addestramento parallelo e indipendenti) per ridurre la varianza, gli algoritmi di boosting (come XGBoost) costruiscono alberi sequenzialmente per ridurre il bias. Sono entrambi potenti algoritmi ensemble, ma con approcci diversi.

# 5. XGBoost (eXtreme Gradient Boosting)

- **1. Definizione e Obiettivo:**
  XGBoost (eXtreme Gradient Boosting) è una libreria ottimizzata e scalabile che implementa gli algoritmi di Gradient Boosting. È diventato estremamente popolare per la sua velocità, flessibilità e capacità di produrre risultati all'avanguardia in una vasta gamma di problemi di Machine Learning, inclusa la regressione. Il suo obiettivo è fornire un framework efficiente e robusto per l'implementazione del gradient boosting, con miglioramenti significativi in termini di prestazioni e regolarizzazione.
- **2. Teoria e Funzionamento Approfondito:**
  - **Concetto di "Boosting" e "Gradient Boosting":**
    - **Boosting:** È una tecnica di ensemble learning che mira a combinare più "apprenditori deboli" (weak learners, tipicamente alberi di decisione poco profondi) in un unico "apprenditore forte". A differenza del bagging (che addestra i modelli in parallelo), il boosting addestra i modelli in *sequenza*. Ogni nuovo modello cerca di correggere gli errori commessi dai modelli precedenti.
    - **Gradient Boosting:** Specificamente, il Gradient Boosting costruisce i nuovi alberi non sui dati originali, ma sui *residui* (errori) dei modelli precedenti. Formalmente, ogni nuovo albero cerca di predire il *gradiente negativo* della funzione di costo rispetto alla previsione corrente. In pratica, è come dire che ogni nuovo albero cerca di apprendere ciò che il modello combinato finora non è riuscito a imparare.
  - **Come XGBoost costruisce gli alberi sequenzialmente per correggere gli errori dei modelli precedenti:**
    XGBoost opera iterativamente:
    a. Inizia con una previsione iniziale (es. la media del target per la regressione).
    b. Ad ogni iterazione, calcola i residui (errori) tra i valori reali e le previsioni cumulative del modello corrente.
    c. Addestra un nuovo albero di decisione (un "weak learner") per predire questi residui (o più precisamente, per ottimizzare una funzione di perdita basata sul gradiente).
    d. La previsione del nuovo albero viene aggiunta alla previsione cumulativa del modello, ma viene scalata da un "learning rate" (o `eta`) per rallentare il processo di apprendimento e migliorare la robustezza.
    e. Questo processo continua per un numero predefinito di iterazioni o finché le prestazioni non migliorano più.
  - **Concetti di funzioni obiettivo con termini di regolarizzazione:**
    Una delle innovazioni chiave di XGBoost è l'aggiunta di termini di regolarizzazione alla funzione di costo standard del Gradient Boosting. La funzione obiettivo che XGBoost cerca di minimizzare ha la forma:

$Obj^{(t)} = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$

Dove:

- $\sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$ è il termine di errore (loss function) tra il target reale ($y_i$) e la previsione cumulativa fino all'iterazione precedente ($\hat{y}_i^{(t-1)}$) più il contributo del nuovo albero ($f_t(x_i)$).
- $\Omega(f_t)$ è il termine di regolarizzazione per il $t$-esimo albero. Questo termine penalizza la complessità dell'albero, promuovendo modelli più semplici e prevenendo l'overfitting. Tipicamente include penalità per il numero di foglie dell'albero e per la grandezza dei pesi sulle foglie (simile a L1/L2 sui coefficienti della regressione lineare, ma applicato alla struttura e ai valori delle foglie dell'albero).
  Questa regolarizzazione intrinseca rende XGBoost meno soggetto all'overfitting rispetto ad altre implementazioni di Gradient Boosting.

- **Importanza dell'ottimizzazione del gradiente:**
  XGBoost utilizza un'ottimizzazione basata sul gradiente (e le derivate seconde, gli Hessiani) per la costruzione degli alberi. Questo gli consente di:
  - **Generalizzazione:** Il modello si adatta ai residui in modo più preciso e efficiente.
  - **Flessibilità:** Può utilizzare una varietà di funzioni di costo (MSE, MAE, log-loss, ecc.) perché l'ottimizzazione si basa sulle derivate di queste funzioni.
  - **Accuratezza:** La considerazione delle derivate seconde (informazioni di curvatura) nella funzione obiettivo fornisce una stima più precisa del "miglior" passo da compiere, contribuendo a una maggiore accuratezza.
- **Perché è considerato altamente performante e robusto:**
  La sua alta performance e robustezza derivano da una combinazione di fattori:
  - **Boosting:** L'approccio iterativo di correzione degli errori è intrinsecamente potente.
  - **Regolarizzazione:** I termini di regolarizzazione L1 e L2 sulla funzione obiettivo riducono il rischio di overfitting.
  - **Gestione Valori Mancanti:** Ha una gestione intelligente e incorporata dei valori mancanti.
  - **Ottimizzazioni Algoritmiche:**
    - **Parallelizzazione:** La costruzione degli alberi è parallelizzata.
    - **Cache-Aware Access:** Ottimizzazione dell'accesso alla memoria per ridurre il tempo di esecuzione.
    - **Out-of-Core Computing:** Gestisce dataset che non entrano completamente in memoria.
    - **Tree Pruning (Potatura):** A differenza di altri GBDT (Gradient Boosting Decision Tree) che usano greedy split, XGBoost costruisce l'albero fino a una `max_depth` e poi pota i rami "non performanti" (basato su `gamma`), migliorando la generalizzazione.
    - **Shrinkage (Learning Rate):** Rallentare l'apprendimento con `eta` previene l'overfitting.

- **3. Vantaggi e Svantaggi:**
  - **Vantaggi:**
    - **Accuratezza Elevata:** Spesso fornisce prestazioni all'avanguardia nelle competizioni di Machine Learning su dati tabulari.
    - **Velocità ed Efficienza:** Altamente ottimizzato e scalabile, supporta il calcolo parallelo.
    - **Flessibilità:** Può gestire vari tipi di dati e funzioni di perdita.
    - **Robusto all'Overfitting:** Grazie alla regolarizzazione e al meccanismo di shrinkage.
    - **Gestisce Valori Mancanti:** Capace di apprendere il modo migliore per gestire i valori mancanti.
  - **Svantaggi:**
    - **Complessità di Tuning:** Ha molti hyperparameters che richiedono un'attenta ottimizzazione per ottenere le migliori prestazioni.
    - **Meno Interpretabile:** Come Random Forest, è un modello "black box" e non facilmente interpretabile nel suo complesso.
    - **Sensibile agli Outlier:** Essendo un modello che cerca di correggere i residui, può essere sensibile agli outlier, anche se la regolarizzazione può mitigare questo aspetto.
    - **Tempi di Addestramento:** Anche se ottimizzato, per dataset molto grandi e un gran numero di `n_estimators`, l'addestramento può comunque richiedere tempo.
- **4. Hyperparameters Chiave:**
  - `n_estimators` **(num_round):** Il numero di alberi (iterazioni di boosting).
  - `learning_rate` **(eta):** La grandezza del passo di shrinkage, che riduce il contributo di ciascun albero. Valori più piccoli (`0.01` a `0.3`) sono comuni e richiedono più `n_estimators`.
  - `max_depth`**:** La profondità massima di ciascun albero debole. Controlla la complessità degli alberi individuali.
  - `subsample`**:** La frazione di campioni del dataset da utilizzare per addestrare ogni albero. Riduce la varianza (simile al bagging).
  - `colsample_bytree`**:** La frazione di feature da campionare per ogni albero. Riduce la varianza e previene l'overfitting.
  - `gamma` **(min_split_loss):** Il guadagno minimo di perdita richiesto per effettuare un'ulteriore partizione su un nodo foglia dell'albero. Controlla la potatura (pruning).
  - `lambda` **(reg_lambda, L2 regularization term) /** `alpha` **(reg_alpha, L1 regularization term):** Termini di regolarizzazione sui pesi delle foglie.
  - `objective`**:** La funzione di perdita da ottimizzare (es. `reg:squarederror` per MSE, `reg:absoluteerror` per MAE).
- **5. Casi d'Uso Tipici:**
  - Competizioni di Machine Learning (Kaggle).
  - Applicazioni dove è richiesta l'accuratezza predittiva più elevata.
  - Problemi su dati tabulari strutturati.

    - Previsione finanziaria, previsione della domanda, analisi del rischio.
- **6. Relazione con altri Modelli:**
    - **Gradient Boosting:** XGBoost è un'implementazione avanzata e ottimizzata di Gradient Boosting, distinguendosi per le sue ottimizzazioni ingegneristiche e l'aggiunta di termini di regolarizzazione espliciti.
    - **Decision Tree:** Utilizza alberi di decisione come apprenditori deboli, ma li costruisce in modo sequenziale per correggere gli errori, a differenza di Random Forest che li costruisce in parallelo.
    - **Random Forest:** Entrambi sono algoritmi ensemble basati su alberi, ma utilizzano strategie diverse: Random Forest (Bagging) riduce la varianza addestrando alberi indipendenti, mentre XGBoost (Boosting) riduce il bias addestrando alberi sequenzialmente sui residui.