UNIVERSITÀ
DI PARMA

Dipartimento di Scienze Matematiche, Fisiche e Informatiche

Corso di Laurea Magistrale in Scienze Informatiche

Ricerca delle migliori linee guida per la creazione di modelli di regressione di alta qualità

Finding the best guidelines to build high-quality regression models

Candidato:
Matteo Toma

Relatore:
Prof.ssa Iotti Eleonora

Correlatore:
Prof. Bonnici Vincenzo

Anno Accademico 2023–2024

La fine di questo percorso è per me la prova che anche con poca autostima tutto è possibile.

Indice

Elenco delle figure

Elenco degli algoritmi

Elenco delle tabelle

Introduzione

Questo studio nasce dalla necessità di fornire un insieme di linee guida che possano orientare sviluppatori, ricercatori e qualunque persona che si cimenti nella progettazione di modelli di Machine Learning, in particolare il modello di Regressione.
Lo studio è motivato dal fatto che negli ultimi anni vi è un contesto in cui i modelli di regressione stanno assumendo un ruolo sempre più centrale in vari ambiti applicativi, dalla medicina all'industria, ed è cruciale comprendere come costruire modelli robusti, efficaci e generalizzabili.
Delle linee guida sono necessarie perché tuttora non c'è un vero metodo per costruire un modello di regressione, di solito chi è in questo campo attua delle scelte in base a una conoscenza teorica pregressa e l'esperienza pratica, ma non è possibile adattare le stesse scelte e comportamenti a dataset diversi.
Per costruire delle linee guida generali si è tenuto conto delle sfide comuni nei dataset, come, ad esempio, l'overfitting, la scelta delle giuste feature e la gestione di dati non bilanciati.
Nel primo capitolo verranno trattati gli algoritmi e le procedure utilizzate.
Nel secondo capitolo saranno spiegate una serie di raccomandazioni pratiche, basate sulla letteratura e validazioni sperimentali e una parte in cui verranno spiegati gli iperparametri da cercare per ottimizzare gli algoritmi.
Nel terzo capitolo verranno presentati gli esperimenti condotti e i risultati ottenuti sui dataset analizzati. In particolare, si procederà inizialmente senza applicare le linee guida proposte, per poi implementarle, al fine di dimostrare che la loro adozione contribuisce a un miglioramento significativo delle prestazioni del modello.
Nel quarto capitolo saranno tratte le conclusioni, che evidenzieranno come l'adozione di queste linee guida possa significativamente migliorare le prestazioni dei modelli, facilitando allo stesso tempo un processo decisionale più consapevole durante le fasi di progettazione e sviluppo.

Capitolo 1

Background

Nell'ambito della regressione (ma anche classificazione) esistono varie procedure e algoritmi per l'analisi dei dati. I più comuni ed efficienti algoritmi sono: le Random Forest (RF), le Reti Neurali Artificiali (ANN) , K-nearest neighbors (KNN) e Support Vector Machines (SVM). Tutti questi algoritmi possono essere utilizzati sia per fare classificazione che regressione.

1.1 Algoritmi e Procedure

1.1.1 Algoritmi Utilizzati

Gli algoritmi trattati sono tutti di apprendimento supervisionato, cioè definito dall'uso di set di dati che sono etichettati.

Random Forest (RF): Funziona costruendo una "foresta", cioè un insieme di alberi decisionali, solitamente addestrati con il metodo di bagging. L'idea generale del metodo di bagging è che una combinazione di modelli di apprendimento aumenta il risultato complessivo, cioè crea più alberi decisionali unendoli insieme insieme per ottenere una previsione più accurata e stabile. [2].

Reti Neurali Artificiali (ANN): È un modello ispirato al funzionamento del cervello umano, mimandolo in modo da imparare dall'esperienza e poter prevedere un risultato a partire da determinate situazioni al contorno. Per farlo utilizza dei composti da strati di nodi (neuroni) che sono interconnessi, in grado di apprendere rappresentazioni complesse dai dati. Successivamente ci sono pesi, che rappresentano il funzionamento delle sinapsi, che hanno il ruolo di dire al neurone quali segnali sono rilevanti e quali no.

La rete neurale è formata da 3 strati: input layer, hidden layer, ed output layer. Ogni nodo nell'input layer rappresenta una feature o una variabile del dataset.
L'Hidden Layer elabora i dati attraverso connessioni pesate.
L'output layer produce il risultato finale della rete, che può essere una classe (classificazione) o un valore numerico (regressione).
In Figura 1.1, i nodi gialli a sinistra rappresentano i dati in input. Il nodo rosso a destra rappresenta l'output. Il suo valore può essere continuo, binario [0 o 1] o categorico [rosso, giallo, verde, etc...]. In verde, c'è il neurone: questo trasforma gli stimoli in reazione. Per fare questo, nel modello, il neurone applica un funzione, detta funzione di attivazione, alla somma pesata dei valori in input. La funzione di attivazione è una funzione matematica che determina l'output del neurone sulla base dell'uscita del sommatore.

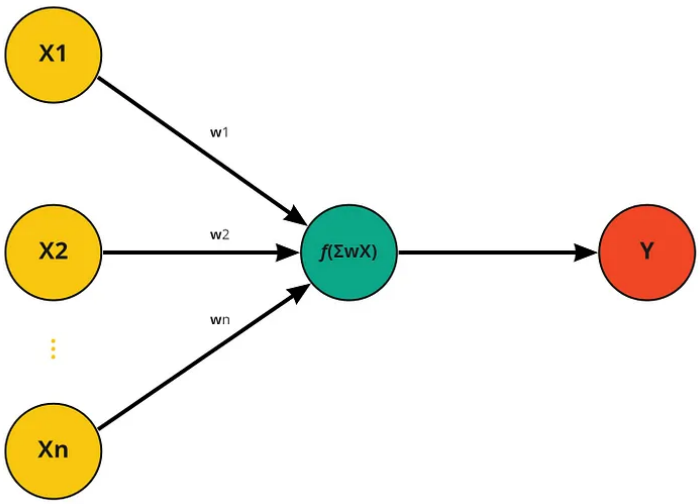

Figura 1.1: Funzionamento della rete neurale artificiale [3]

K-nearest neighbors (KNN): Questo algoritmo di apprendimento supervisionato inizia con la rappresentazione di ogni punto dati come vettore in uno spazio multidimensionale. Per classificare un nuovo punto, l'algoritmo calcola la distanza tra questo punto e tutti gli altri nel dataset. Identifica poi i k punti più vicini, dove il valore di k è cruciale. Nella classificazione, il punto di query riceve l'etichetta della classe più comune tra i vicini, mentre nella regressione si calcola la media dei valori dei k vicini. Quindi ogni punto dati che si avvicina l'uno all'altro rientra nella stessa classe. [4].

Support Vector Machines (SVM): Sono modelli supervisionati che separano i dati in categorie utilizzando un confine definito da un gap massimo. Esistono due tipi di separazione: lineare, per dati separabili con un confine lineare, e non lineare, che usa funzioni kernel per rendere i dati separabili. Le SVM mappano un insieme di dati in uno spazio multidimensionale per poi trovare un iperpiano (una "linea" che è in più dimensioni) che meglio divide in classi i dati. L'iperpiano è selezionato per massimizzare la distanza tra sé e i punti più vicini in ciascuna delle classi, questi punti sono chiamati vettori di supporto. Una volta identificato l'iperpiano, i nuovi dati vengono classificati mappandoli nello stesso spazio e determinando da quale parte dell'iperpiano si trovano[5].

1.1.2 Procedure Utilizzate

K-fold Cross Validation: In generale la Cross Validation è una tecnica per valutare la performance di un modello dividendo i dati in più parti, addestrando il modello su alcune e testandolo su altre. Le parti in cui il dataset è diviso è la parte di test data, training data e validation data.
Il flusso generale è:

1. Addestrare il modello sul training set.
2. Testare le performance della configurazione attuale sul set di validazione.
3. Se e solo si è soddisfatti delle performance sul set di validazione, allora testare sul test set.

La K-fold Cross validation invece ha il seguente flusso:

1. Randomizzare ogni riga del dataset
2. Dividere il dataset in k porzioni
3. Per ogni gruppo:
   (a) Creare una porzione di test
   (b) Allocare il restante all'addestramento
   (c) Addestrare il modello e valutarlo sui menzionati set
   (d) Salvare la performance
4. Valutare le performance generali prendendo la media dei punteggi alla fine del processo

Il test set continuerà ad essere usato per la valutazione finale, mentre le performance del modello verranno valutate sulle porzioni generate dalla cross-validazione. Il valore di k è tipicamente 5 o 10, ma è possibile usare la regola di Sturges [6] per stabilire un numero più preciso di split. La regola di Sturges utilizza la seguente formula per determinare il numero ottimale di intervalli da usare in un istogramma:

$$\text{Numero di Split Ottimali} = 1 + \log_2(N) \quad (1)$$

dove N è Il numero totale di osservazioni nel dataset. Questa procedura si usa per creare modelli di regressione altamente generalizzabili [7].

Grid Search: È una metodologia nella regressione e classificazione usata per individuare gli iperparametri ottimali di un modello. Questo processo prevede la definizione di una gamma di valori per ciascun iperparametro e l'esplorazione sistematica di tutte le combinazioni possibili. Attraverso questa ricerca esaustiva, si valutano le prestazioni del modello su ogni combinazione per selezionare i parametri migliori [8].

Random Search: È una tecnica di ottimizzazione degli iperparametri nella classificazione e regressione che prevede la selezione casuale di un numero definito di combinazioni di iperparametri da distribuzioni specificate. Queste distribuzioni rappresentano intervalli o insiemi di valori possibili per i parametri. Rispetto alla Grid search, esplora in modo più efficiente una porzione dello spazio degli iperparametri, riducendo il tempo e le risorse.
Sebbene non garantisca di trovare la combinazione ottimale, spesso individua configurazioni valide, specialmente in spazi parametrici ampi o complessi [9].

Hyperband: Questo metodo si distingue per la sua capacità di effettuare una valutazione efficiente delle configurazioni, sfruttando strategie di early stopping per modelli di apprendimento profondo che richiedono tempo e risorse considerevoli. È una tecnica di ottimizzazione degli iperparametri nella di regressione e classificazione che prevede la selezione casuale di un numero definito di combinazioni di iperparametri da distribuzioni specificate. Queste distribuzioni rappresentano intervalli o insiemi di valori possibili per i parametri. Il funzionamento si basa su un approccio bandit, dove una porzione di configurazioni di iperparametri viene esplorata in modo casuale. Il processo si articola in diverse fasi in cui si valutano i modelli parzialmente addestrati e si fermano quelli che non mostrano prestazioni promettenti. Questo approccio non solo consente di risparmiare tempo, ma migliora anche la probabilità di individuare configurazioni efficaci in spazi di parametri complessi e di grandi dimensioni [10].

Bayesian Optimization: Questa procedura crea un modello probabilistico della funzione obiettivo e lo utilizza per selezionare gli iperparametri da valutare nella funzione obiettivo reale. L'approccio si basa quindi sull'idea di aggiornare continuamente il modello delle probabilità man mano che si raccolgono nuovi dati, consentendo di esplorare efficacemente lo spazio degli iperparametri e di identificare le configurazioni più promettenti in modo più efficiente rispetto ai metodi tradizionali [11].

Sequential Model-Based Optimization:
È una tecnica Bayesiana che costruisce modelli incrementali per migliorare gradualmente la scelta degli iperparametri.
Il seguente algoritmo è presente in [12]:

Algoritmo 1 Sequential Model-Based Optimization (SMBO)

1: $\mathcal{H} \leftarrow \emptyset$
2: **for** $t \leftarrow 1$ **to** T **do**
3: $\quad x^* \leftarrow \arg\min_x S(x, M_{t-1})$
4: $\quad$ Valuta $f(x^*)$
5: $\quad \mathcal{H} \leftarrow \mathcal{H} \cup (x^*, f(x^*))$
6: $\quad$ Addestra un nuovo modello M_t su $\mathcal{H}$
7: **return** $\mathcal{H}$

- **H** : Storia delle osservazioni di coppie (iperparametro, punteggio)
- **T** : Numero massimo di iterazioni
- **f** : Funzione obiettivo vera
- **M** : Funzione surrogata, che viene aggiornata ogni volta che viene aggiunto un nuovo campione
- **S** : Funzione di acquisizione
- **x*** : Il prossimo iperparametro scelto da valutare

Bayesian Optimization (Tree-Structured Parzen Estimator):
È una variante delle tecniche di ottimizzazione Bayesiana che gestisce spazi di ricerca strutturati ad albero, ovvero spazi che includono parametri condizionali. Utilizza stimatori di densità Parzen (KDE) per modellare la distribuzione $p(y|x, D)$, dividendo le osservazioni in due gruppi: uno "migliore" con punteggi $y \leq y_\gamma$, e uno "peggiore" con punteggi $y > y_\gamma$, dove y_γ rappresenta il valore del quantile superiore γ, aggiornato ad ogni iterazione.

La funzione di acquisizione viene definita come il rapporto $r(x|D) = \frac{p(x|D(l))}{p(x|D(g))}$ (2), dove **D(l)** e **D(g)** rappresentano rispettivamente il gruppo "migliore" e il gruppo "peggiore" all'interno dell'insieme di osservazioni **D**.
Le dimensioni di questi gruppi sono indicate come $\mathbf{N(l)} := |D(l)|$ e $\mathbf{N(g)} := |D(g)|$.
Questo permette all'algoritmo di scegliere i parametri iperottimali campionati dal gruppo migliore. Anche se esistono varianti con garanzie di convergenza globale, TPE può adottare un approccio greedy per limitare il numero di valutazioni necessarie [13].

Particle Swarm Optimization: È un algoritmo ispirato al comportamento collettivo di sciami di uccelli o banchi di pesci. L'idea alla base è che ogni individuo (particella) condivida le proprie scoperte con il gruppo, migliorando le possibilità di trovare la soluzione ottimale. Ogni particella esplora lo spazio delle soluzioni e si aggiorna in base alle informazioni condivise dalle altre. Anche se non vi è garanzia che venga trovata la soluzione ottimale globale, PSO riesce spesso a individuare soluzioni vicine all'ottimo, risultando efficace per molti problemi complessi.
A livello matematico, la PSO è una tecnica usata per trovare il massimo o il minimo di una funzione definita su uno spazio vettoriale multidimensionale. Si assume di avere una funzione $f(X)$ che produce un valore reale da un parametro vettoriale X (ad esempio, le coordinate (x, y) su un piano). Il vettore X può assumere praticamente qualsiasi valore nello spazio (ad esempio, $f(X)$ rappresenta l'altitudine e possiamo trovarla per qualsiasi punto sul piano). L'algoritmo PSO restituirà il parametro X che produce il minimo di $f(X)$ [14].

Algoritmo Genetico: Chiamato anche Genetic Algorithm (GA), è un algoritmo di ottimizzazione ispirato alla selezione naturale, che utilizza un approccio basato sulla popolazione e il concetto di "sopravvivenza del più adatto". Le nuove popolazioni vengono generate mediante l'uso iterativo di operatori genetici sui singoli individui presenti nella popolazione. Gli elementi chiave del GA includono la rappresentazione cromosomica, la selezione, il crossover, la mutazione e il calcolo della funzione di fitness.
Il processo del GA è il seguente: una popolazione Y di n cromosomi viene inizializzata casualmente. Viene calcolata la fitness di ciascun cromosoma in Y. Due cromosomi, detti C_1 e C_2, vengono selezionati dalla popolazione Y in base al valore di fitness. L'operatore di crossover a punto singolo, con probabilità di crossover C_p, viene applicato a C_1 e C_2 per produrre una

progenie O.
Successivamente, viene applicato l'operatore di mutazione uniforme sulla progenie prodotta O con probabilità di mutazione M_p per generare O'. La nuova progenie O' viene inserita nella nuova popolazione. Le operazioni di selezione, crossover e mutazione vengono ripetute sulla popolazione corrente fino al completamento della nuova popolazione.
L'analisi matematica del GA mostra che il processo di ricerca cambia dinamicamente attraverso le probabilità di crossover e mutazione, raggiungendo così una soluzione ottimale. Inoltre, il GA può modificare i geni codificati, valutare più individui e produrre più soluzioni ottimali, offrendo una migliore capacità di ricerca globale.
La progenie prodotta dal crossover dei cromosomi genitori ha la possibilità di eliminare gli schemi genetici pregiati dei cromosomi genitori. La formula per il crossover è definita come:

$$R = G + 2\sqrt{g \cdot p}/3G \quad (3)$$

dove g è il numero di generazioni e G è il numero totale di generazioni evolutive stabilito dalla popolazione. Si osserva da (3) che R cambia dinamicamente e aumenta con l'aumentare del numero di generazioni evolutive. Nella fase iniziale del GA, la somiglianza tra gli individui è molto bassa, mentre al termine dell'evoluzione la somiglianza è molto alta.
Secondo il teorema degli schemi, lo schema originale deve essere sostituito con uno schema modificato per mantenere la diversità nella popolazione.
Alla fine dell'evoluzione, verrà prodotto uno schema appropriato per prevenire la distorsione di schemi genetici eccellenti [15].
TPOT: È uno strumento che automatizza l'intero processo di machine learning attraverso tecniche di ottimizzazione evolutiva, cioè programmazione genetica. Esplora in modo intelligente migliaia di possibili pipeline per trovare quella migliore per i dati.

[1]https://epistasislab.github.io/tpot/

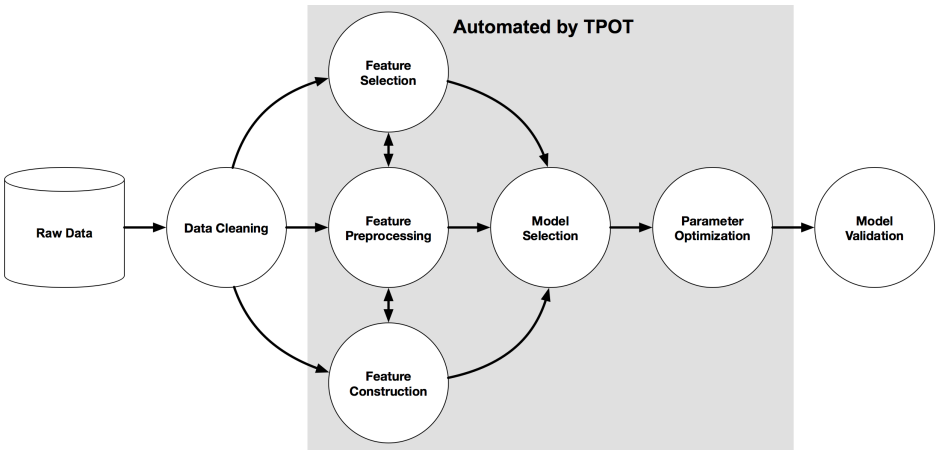

Figura 1.2: Funzionamento di TPOT[1]

1.2 Metriche

Per studiare il comportamento di un modello di machine learning, è fondamentale scegliere bene le metriche, che cambiano se si effettua una regressione o una classificazione. Nel caso della *regressione*, si possono utilizzare vari tipi di metriche [16]:

Mean Absolute Error (MAE): Misura la media degli errori assoluti tra le predizioni e i valori reali.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (4)$$

Mean Squared Error (MSE): Calcola la media dei quadrati degli errori, penalizzando maggiormente gli errori più grandi.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (5)$$

Root Mean Squared Error (RMSE): La radice quadrata della MSE, che fornisce una misura interpretabile della deviazione standard degli errori.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (6)$$

R-squared ($\mathbf{R^2}$): Indica la proporzione della varianza nella variabile dipendente che è spiegata dalle variabili indipendenti.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \quad (7)$$

Adjusted R-squared: Una versione modificata dell'$\mathbf{R^2}$ che tiene conto del numero di predittori nel modello, utile per confrontare modelli con un numero diverso di variabili.

$$\text{Adjusted } R^2 = 1 - \left(\frac{1 - R^2}{n - p - 1} \right) \quad (8)$$

Dove n è il numero di osservazioni e p è il numero di predittori.

Mean Absolute Percentage Error (MAPE): Fornisce una misura dell'errore in percentuale, utile per valutare le prestazioni relative.

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (9)$$

Explained Variance Score: Misura la proporzione della varianza totale spiegata dal modello.

$$\text{Explained Variance} = 1 - \frac{\text{Var}(y - \hat{y})}{\text{Var}(y)} \quad (10)$$

Logarithmic Loss (Log Loss): Utilizzato per i modelli di regressione logistica, misura le prestazioni del modello rispetto alle probabilità previste.

$$\text{Log Loss} = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (11)$$

Nel caso della *classificazione*, si possono utilizzare vari tipi di metriche [17]:

Matrice di confusione: Tabella che riassume le prestazioni del modello mostrando il numero di veri positivi (TP), veri negativi (TN), falsi positivi (FP) e falsi negativi (FN).

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix} \quad (12)$$

Nel caso di una matrice multiclasse la struttura è:

$$\begin{bmatrix} TP_1 & FP_{12} & FP_{13} & \cdots & FP_{1N} \\ FN_{21} & TP_2 & FP_{23} & \cdots & FP_{2N} \\ FN_{31} & FN_{32} & TP_3 & \cdots & FP_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ FN_{N1} & FN_{N2} & FN_{N3} & \cdots & TP_N \end{bmatrix} \quad (13)$$

Dove TP_i rappresenta i veri positivi per la classe i, FP_{ij} indica i falsi positivi per la classe i classificata come classe j, e FN_{ij} indica i falsi negativi per la classe i che sono stati classificati come classe j.

Accuracy: Percentuale di istanze correttamente classificate rispetto al totale delle istanze, derivata dalla matrice di confusione.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (14)$$

Precision: Misura della correttezza delle predizioni positive, calcolata come il numero di veri positivi diviso la somma dei veri positivi e dei falsi positivi, derivata dalla matrice di confusione.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (15)$$

Recall (Sensibilità): Misura della capacità del modello di identificare le istanze positive, calcolata come il numero di veri positivi diviso la somma dei veri positivi e dei falsi negativi, derivata dalla matrice di confusione.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (16)$$

F1 Score: Media armonica tra precision e recall, utile quando si cerca un equilibrio tra le due misure.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (17)$$

Matthew's Coefficient: Una misura dell'accuratezza di una classificazione binaria, che tiene conto dei veri e falsi positivi e negativi, utilizzato anche per classificazioni multi-classe[2].

$$\text{MCC} = \frac{TN \cdot TP - FN \cdot FP}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (18)$$

[2]https://www.voxco.com/blog/matthewss-correlation-coefficient-definition-formula-and-advantages/

Se la previsione restituisce buoni tassi per tutte e quattro queste entità, si dice che è una misura affidabile che produce punteggi elevati.
Per adattarsi alla maggior parte dei coefficienti di correlazione, il coefficiente di Matthew varia anche tra +1 e -1 come:
+1: la migliore concordanza tra i valori previsti e quelli effettivi.
0: nessuna concordanza (la previsione è casuale in base ai valori effettivi).
-1: indica una totale discordanza, dove ogni previsione è contraria al valore reale (ad esempio, ogni elemento classificato come "positivo" è in realtà "negativo" e viceversa).

ROC-AUC (Area Under the Receiver Operating Characteristic Curve): Misura la capacità del modello di discriminare tra le classi, rappresentata da una curva che confronta il tasso di veri positivi e il tasso di falsi positivi.

$$\text{ROC-AUC} = \int_0^1 \text{TPR}(FPR)\, dFPR \quad (19)$$

Dove TPR è il tasso di veri positivi e FPR è il tasso di falsi positivi.

Logarithmic Loss (Log Loss): Misura delle performance del modello in termini di probabilità previste per ciascuna classe.

$$\text{Log Loss} = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \quad (20)$$

Metrica di valutazione multi-classe: Per i modelli a più classi, le metriche possono essere calcolate come medie ponderate delle metriche per ogni classe, utilizzando ad esempio la media macro, micro o pesata.

Cohen's Kappa: Misura l'accordo tra le predizioni del modello e le etichette vere, tenendo conto delle possibilità di accordo casuale.

$$\kappa = \frac{p_o - p_e}{1 - p_e} \quad (21)$$

Dove p_o è la precisione osservata e p_e è la precisione attesa sotto ipotesi di accordo casuale.

La seguente tabella mostra come interpretare il coefficiente[3]:

Valore di Kappa	**Interpretazione**
< 0	Concordanza minore di quella attesa per caso
0	Concordanza uguale a quella attesa per caso
0.01 - 0.20	Scarsa concordanza
0.21 - 0.40	Modesta concordanza
0.41 - 0.60	Moderata concordanza
0.61 - 0.80	Sostanziale concordanza
> 0.80	Quasi perfetta concordanza
1	Accordo perfetto

Tabella 1.1: Interpretazione dei valori del coefficiente di Kappa

Le misure scelte per semplicità e importanza per la regressione sono state: *MSE* (5) e R^2 (7).
Mentre per la classificazione sono state scelte *Accuracy (14), Precision (15), Recall (16), F1 Score (17) e Log Loss (20).*
Un'altra metrica da considerare è sicuramente il tempo impiegato per eseguire gli algoritmi, sia nella classificazione che nella regressione.

1.3 Metodi ottimizzazione

Di solito, quando si affronta un problema di machine learning, si inizia utilizzando alcuni modelli per analizzare e apprendere i dati, senza però fare delle considerazioni a priori. I problemi potenziali sono di solito i seguenti:

1. Se i dati di addestramento insufficienti bisognerebbe provare a ottenere più dati di addestramento.

2. Se si hanno troppe caratteristiche rumorose/irrilevanti allora è possibile provare a ridurre il set di caratteristiche.

3. Se non si hanno abbastanza caratteristiche rilevanti allora è possibile provare un set di caratteristiche più ampio.

4. Se le caratteristiche non sufficientemente potenti allora è possibile provare a cambiare le caratteristiche.

[3]https://paolapozzolo.it/kappa-cohen

5. Se il modello non converge abbastanza velocemente allora è possibile eseguire la discesa del gradiente per più iterazioni.

6. Se il modello non converge allora si possono cambiare gli ottimizzatori.

7. Se gli iperparametri di default non funzionano allora si possono utilizzare iperparametri diversi.

8. Se la scelta del modello è inappropriata è possibile utilizzare un modello diverso.

Per eseguire il debug di modelli di machine learning, è possibile effettuare diverse diagnosi:

Diagnosi bias-varianza
È possibile che un comportamento anomalo sia dovuto a problemi di bias o varianza.
Problemi:

- Sovradattamento (alta varianza)
- Poche caratteristiche (alto bias)

Diagnosi:
È possibile tracciare una curva di apprendimento con la dimensione del set di addestramento sull'asse X (anziché il numero di epoche). Successivamente, è possibile analizzare il comportamento degli errori del set di addestramento e di validazione.
Alta varianza:
Se l'errore sul test continua a diminuire al massimo della dimensione del set di addestramento, o se c'è un grande divario tra l'errore di addestramento e quello di validazione, allora il problema è probabilmente dovuto all'alta varianza. Aumentare la dimensione del set di addestramento può aiutare.
Alto bias:
Se sia l'errore di validazione che quello di addestramento sono grandi (cioè c'è un piccolo divario), allora il problema è probabilmente dovuto ad alto bias. Aumentare la complessità del modello può aiutare.
Possibili soluzioni: Se la varianza è alta è possibile risolvere con:

1. Un set di addestramento più grande

2. Un set di caratteristiche più piccolo

3. Regolarizzando il modello

Il bias alto è possibile risolverlo con:

1. Un set di caratteristiche più ampio
2. Aggiunta di caratteristiche più potenti
3. Prova di un modello più complesso

Diagnosi algoritmo di ottimizzazione

- Se l'algoritmo di ottimizzazione non converge, può essere utile tracciare un grafico della funzione di perdita in funzione del numero di passi di discesa del gradiente, anche se solitamente questi sono difficili da estrapolare.
- Potrebbe capitare che non venga ottimizzata la funzione corretta. Di solito, la discesa del gradiente non ottimizza la funzione esatta di cui ci interessa. Ad esempio, l'accuratezza (pesata) non è differenziabile, quindi si ottimizza la log-Likelihood regolarizzata.

Possibili soluzioni:

1. Eseguire la discesa del gradiente per più iterazioni aiuta a risolvere problemi dell'ottimizzatore.
2. Provare il metodo di Newton aiuta a risolvere problemi dell'ottimizzatore.
3. Utilizzare un valore diverso per λ (il parametro di regolarizzazione) aiuta a risolvere problemi legati alla funzione obiettivo.
4. Provare l'utilizzo di una SVM aiuta a risolvere problemi legati alla funzione obiettivo.

La regolarizzazione viene utilizzata per evitare l'overfitting, un problema in cui il modello si adatta troppo ai dati di addestramento e perde la capacità di generalizzare a dati nuovi. Un valore maggiore di λ aumenta la penalità sulle grandezze dei coefficienti del modello, portando a una maggiore semplificazione del modello.
Esistono due tipi di regolarizzazione principali:
La *Ridge Regression* aiuta a minimizzare la somma dei residui quadratici e dei valori quadrati dei parametri scalati da un fattore (λ o α). Questo termine di

regolarizzazione, λ, controlla la forza del vincolo sui coefficienti e funge da parametro di tuning.
La *Lasso Regression* aggiunge un termine di regolarizzazione alla funzione obiettivo della regressione lineare. La differenza risiede nella funzione di perdita utilizzata: la Lasso Regression utilizza la regolarizzazione L1, che mira a minimizzare la somma dei valori assoluti dei coefficienti moltiplicata per il fattore di penalità λ.
Le formule per la Ridge e la Lasso Regression sono le seguenti:
Ridge Regression:

$$\min_{\mathbf{w}} \left(\sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \mathbf{w})^2 + \lambda \|\mathbf{w}\|_2^2 \right) \quad (20)$$

La prima parte della funzione obiettivo è la somma dei quadrati dei residui, mentre il termine di regolarizzazione è $\lambda \|w\|_2^2$, che penalizza la somma dei quadrati dei coefficienti.

Lasso Regression:

$$\min_{\mathbf{w}} \left(\sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \mathbf{w})^2 + \lambda \|\mathbf{w}\|_1 \right) \quad (21)$$

Simile alla Ridge, ma utilizza la norma L1, $\|w\|_1$, che penalizza la somma dei valori assoluti dei coefficienti.

Debugging delle pipeline di Machine Learning
Per molte applicazioni pratiche, quando non sono disponibili un numero molto elevato di campioni, una pipeline di algoritmi di Machine Learning semplici e/o statici può funzionare molto meglio di un grande modello di Machine Learning end-to-end.
Se questo sistema non funziona, la domanda è: quanto errore è attribuibile a ciascuno dei componenti? Per diagnosticare questo, è possibile provare a collegare il ground-truth per ciascun componente e vedere come cambia l'accuratezza.
L'analisi ablativa è una tecnica utilizzata per comprendere l'importanza relativa di diverse variabili o caratteristiche nel contribuire alle predizioni di un modello [18].
Si valuta l'effetto della rimozione di una variabile alla volta dal modello e si osserva come questo influisce sulle prestazioni del modello stesso. Questo processo aiuta a determinare quali variabili sono più critiche per il modello e quali hanno un impatto meno significativo. Inoltre, può essere utile per

identificare le caratteristiche più rilevanti e semplificare il modello riducendo le caratteristiche meno influenti.
L'**analisi degli errori** cerca di spiegare *la differenza tra le prestazioni attuali e le prestazioni perfette.*
L'**analisi ablativa** cerca di spiegare *la differenza tra alcune prestazioni di base (molto inferiori) e le prestazioni attuali.*

Controlli di ispezione del modello
L'**analisi di sensibilità** permette di comprendere l'influenza di ciascun parametro di input sulle previsioni del modello. Si tratta di una tecnica fondamentale per valutare la robustezza e l'affidabilità del modello, soprattutto in presenza di incertezze o variazioni negli input.
L'approccio prevede la creazione di scenari ipotetici variando i valori degli input e osservando come cambiano le previsioni del modello in risposta a tali variazioni.

1.3.1 Come verificare uno scenario interessante

Bisogna trovare le caratteristiche più importanti. Ad esempio, utilizzando:

- **Partial Dependence Plot (PDP)**
  - Mostra l'effetto marginale di una o due caratteristiche sull'esito del modello.
  - Per un intervallo di valori per una caratteristica fissa, esegui:
    - Fissa il valore della caratteristica di interesse a un valore particolare nell'intervallo.
    - Calcola la media su tutti i campioni, lasciando intatte le altre caratteristiche (complemento) e registra l'esito.
    - Traccia l'esito medio.
    - Più informativo rispetto ai punteggi di importanza: può mostrare la natura della relazione.

Nella figura 1.3 è visibile un Partial Dependence Plot :

[4]https://christophm.github.io/interpretable-ml-book/pdp.html

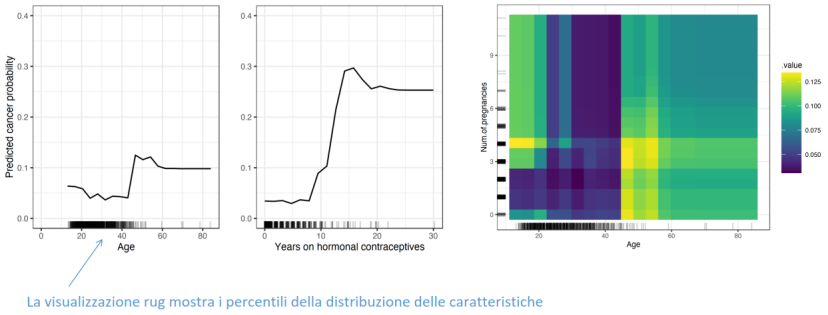

Figura 1.3: Esempio di un Partial Dependence Plot[4]

Vantaggi

- Facile da interpretare
- Facile da implementare

Svantaggi

- Nasconde effetti eterogenei → Possibile soluzione: ICE
- Presuppone indipendenza:
  - Risultati fuorvianti quando si trattano caratteristiche correlate (esempi non realistici) → Possibile soluzione: ALE

- **ALE plots (Accumulated Local Effects)**
  I grafici ALE (Accumulated Local Effects) riducono la complessa funzione di previsione f a una funzione che dipende solo da una (o due) caratteristiche e mediano le variazioni nelle previsioni e le accumulano sulla griglia [5].

- **Individual Conditional Expectation (ICE) Plot**
  I grafici ALE (Accumulated Local Effects) riducono la complessa funzione di previsione f a una funzione che dipende solo da una (o due) caratteristiche e mediano le variazioni nelle previsioni e le accumulano sulla griglia [6].
  Analogamente al Partial Dependence Plot, ma per un singolo caso invece di calcolare la media sui dati: la curva PDP è la media di tutte le curve ICE. **Vantaggi**

[5]https://christophm.github.io/interpretable-ml-book/ale.html
[6]https://christophm.github.io/interpretable-ml-book/ice.html

- Ancora più intuitivo da comprendere rispetto al PDP
- Può rivelare relazioni eterogenee

Svantaggi

- Può visualizzare solo una caratteristica alla volta
- Presenta gli stessi problemi del PDP con caratteristiche correlate
- Il grafico può diventare affollato

- **SHAP values**
  Spiegano la previsione di un'istanza x calcolando il contributo di ciascuna caratteristica alla previsione. Il metodo di spiegazione SHAP calcola i valori di Shapley dalla teoria dei giochi coalizionali. I valori delle caratteristiche di un'istanza di dati agiscono come giocatori in una coalizione. I valori di Shapley ci dicono come distribuire equamente il "payout" (= la previsione) tra le caratteristiche.

  Un giocatore può essere un valore di caratteristica individuale, ad esempio per dati tabulari. Ad esempio, per spiegare un'immagine, i pixel possono essere raggruppati in superpixel e la previsione distribuita tra di essi [7].

Instance x'	Age	Weight	Color
x'	1	1	1

Tabella 1.2: Instance for x'

Instance x	Age	Weight	Color
x	0.5	20	Blue

Tabella 1.3: Feature values for x

Instance z'	Age	Weight	Color
z'	1	0	0

Tabella 1.4: Instance for "absent features" for z'

Instance z	Age	Weight	Color
z	0.5	17	Pink

Tabella 1.5: Feature values for z

La funzione h_x mappa una coalizione a un'istanza valida. Per le caratteristiche presenti (1), h_x mappa ai valori delle caratteristiche di x.
Per le caratteristiche assenti (0), h_x mappa ai valori di un'istanza di dati campionata casualmente.

[7]https://christophm.github.io/interpretable-ml-book/shap.html

4.1.2 Trovare esempi avversari: Quando si lavora con dati strutturati (o tabulari), trovare esempi avversariali interpretabili può essere fatto utilizzando un semplice algoritmo euristico. Gli esempi avversariali rendono i modelli di regressione e classificazione vulnerabili agli attacchi: gli esempi avversariali per le immagini sono immagini con pixel intenzionalmente perturbati con l'obiettivo di ingannare il modello durante il periodo di applicazione. Gli esempi dimostrano in modo impressionante quanto facilmente le reti neurali profonde per il riconoscimento degli oggetti possano essere ingannate da immagini che appaiono innocue per gli esseri umani, ecco un esempio:

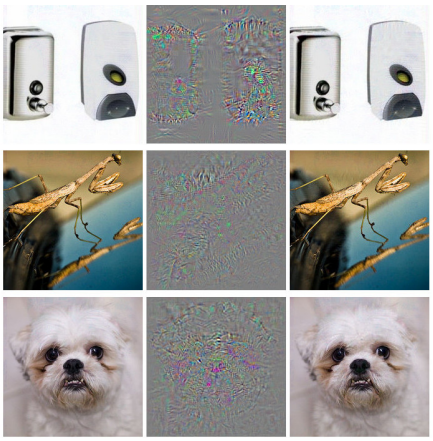

Figura 1.4: Esempi avversari per AlexNet di [1]. Tutte le immagini nella colonna di sinistra sono classificate correttamente.
La colonna centrale mostra l'errore (ingrandito) aggiunto alle immagini per produrre le immagini nella colonna di destra, tutte categorizzate (in modo errato) come "Ostrich".

4.2 Analisi dei residui:

- I residui sono la differenza tra l'esito vero noto e l'esito previsto. L'analisi dei residui del modello (sul set di addestramento o su un set di sviluppo separato) può fornire importanti intuizioni sui suoi modi di fallimento.
- Un grafico dei residui è uno scatterplot con i residui sull'asse Y e una variabile indipendente sull'asse X. La variabile indipendente, in questo caso, può essere anche l'output del modello.
  - Se il problema è di classificazione, i punti sullo scatterplot possono essere colorati per classe, permettendo la visualizzazione dei residui per una variabile indipendente per classe.
  - Nel caso di regressione, può essere utilizzato un gradiente di colore.
- Altre tecniche includono:
  - Creazione di grafici dei residui per caratteristica.
  - Modellazione dei residui.
  - Vincoli di monotonicità.

Capitolo 2

Linee Guida

2.1 Controlli di qualità nel modello di apprendimento automatico

Alcune delle linee guida proposte da [19], riguardano la prototipazione, la valutazione online/offline, le metriche di valutazione e la ricerca degli iperparametri.

Prototipazione: durante questa fase si provano diversi modelli per trovare il migliore (*model selection*). Una volta che si è soddisfatti di un modello prototipato, si distribuisce in produzione, dove verrà ulteriormente testato su dati in tempo reale. Il compito principale durante la fase di prototipazione è selezionare il modello giusto che si adatti ai dati. Il modello deve essere valutato su un set di dati statisticamente indipendente da quello su cui è stato addestrato, poiché le sue prestazioni sul set di addestramento forniscono una stima eccessivamente ottimistica delle sue reali prestazioni su nuovi dati. Il processo di addestramento del modello si è già adattato ai dati di addestramento. Una valutazione più equa misurerebbe le prestazioni del modello su dati che non ha ancora visto. In termini statistici, ciò fornisce una stima dell'*errore di generalizzazione*, che misura quanto bene il modello si generalizza su nuovi dati.
Di solito, per ottenere nuovi dati, la soluzione statistica è suddividerli o riacquisirli, simulando la disponibilità di nuovi dati, oppure è possibile trattenere una parte del set di addestramento e utilizzarla solo per la valutazione: questo è noto come *validazione hold-out*.
Il metodo più generale è noto come *cross-validazione k-fold*, precedentemente discusso a pagina 5.

Ci sono due tipi di valutazione:

Valutazione online: misura le metriche in tempo reale del modello distribuito sui dati live.

Valutazione offline: misura le metriche offline del modello prototipato sui dati storici (e talvolta anche sui dati in tempo reale).
Quindi, ci sono due fonti di dati: storici e live. Molti modelli statistici assumono che la distribuzione dei dati rimanga costante nel tempo (chiamata anche *distribuzione stazionaria*), ma nella pratica ciò che succede è che spesso la distribuzione dei dati cambia nel tempo, talvolta drasticamente.
Questo fenomeno è chiamato *distribution shift*. Un modo per rilevare questa "degradazione" della distribuzione è continuare a monitorare le prestazioni del modello sulla metrica di validazione utilizzando dati live. Se le prestazioni sono comparabili ai risultati di validazione quando il modello è stato costruito, allora il modello si adatta ancora ai dati. Quando le prestazioni iniziano a degradarsi, è probabile che la distribuzione dei dati live si sia sufficientemente discostata dai dati storici, e sarà necessario riaddestrare il modello. Il monitoraggio della distribution shift è spesso eseguito *offline* rispetto all'ambiente di produzione.

Nel capitolo 2 vengono affrontate le metriche di classificazione, di regressione e di ranking e accortezze da avere quando si guarda un dataset. La classificazione riguarda la previsione delle etichette di classe a partire da dati di input.
Nella **classificazione binaria**, ci sono due classi di output possibili.
Nella **classificazione multiclasse**, ci sono più di due classi possibili.
Un esempio di classificazione binaria è il rilevamento dello spam nelle email, dove i dati di input possono includere il testo dell'email e i relativi metadati (come il mittente e l'ora di invio), mentre l'etichetta di output sarà "spam" o "non spam." Di solito, si usano nomi generici per le due classi, come "positivo" e "negativo" o "classe 1" e "classe 0."
Ci sono molti modi per misurare le prestazioni, quelli discussi dall'autrice sono quasi identici a quelli discussi in 1.2.

2.1.1 Dataset/Classi Sbilanciati, Outlier e Dati Rari

Il *data skew* si verifica quando un tipo di dato è molto più raro di altri, o quando ci sono outlier molto grandi o molto piccoli che possono alterare drasticamente la metrica. I dati presentano una distribuzione non uniforme

tra diverse partizioni o nodi nell'elaborazione dati su larga scala.[1]
Classi sbilanciate possono rappresentare un problema nel misurare sia l'accuratezza per classe, sia per tutte le metriche che danno uguale peso a ciascun punto dati.
Ad esempio, se la classe positiva rappresenta solo una piccola porzione dei dati osservati, ad esempio l'1%—una situazione comune nei dataset reali come i tassi di click-through per le pubblicità, i dati di interazione utente-oggetto per i sistemi di raccomandazione, o il rilevamento di malware, ecc, un classificatore di base "stupido" che classifica sempre i dati in arrivo come negativi raggiungerebbe un'accuratezza del 99%. Outlier: un outlier è un valore anomalo che è molto differente dal resto della distribuzione dei dati. La causa di un outlier potrebbe essere di natura umana (inserimento erroneo dei dati) o strumentale.
[2] L'effetto degli outlier di grandi dimensioni durante la valutazione può essere mitigato attraverso:

- Metriche robuste come i quantili degli errori.
- Pulizia attenta dei dati.
- Riformulazione del compito in modo che non sia sensibile a outlier di grandi dimensioni.

2.1.2 Prototyping

Nel capitolo 3 vengono affrontate le fasi della prototipazione, un esempio è in Figura 2.1.

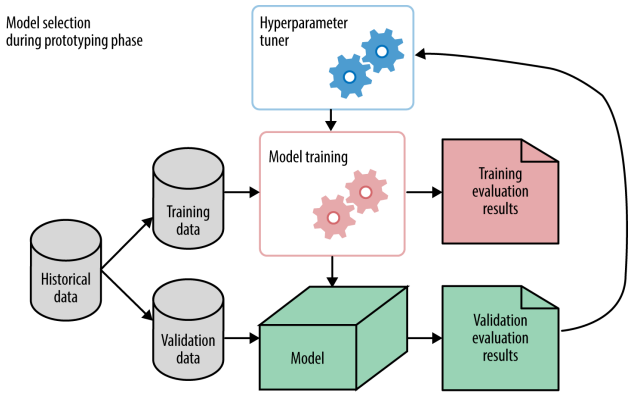

Figura 2.1: Esempio di schema di prototipazione nel costruire un modello di Machine Learning.[19]

[1]https://www.dremio.com/wiki/data-skew/
[2]https://www.yimp.it/outlier-cosa-sono-come-individuarli/

Divisione del Dataset Storico e Processo di Validazione:
Il dataset storico disponibile viene diviso in due parti: **training** e **validation**. Il processo di addestramento del modello riceve i dati di training e produce un modello, che viene valutato sui dati di validazione.
Successivamente i risultati della validazione vengono passati indietro al *hyperparameter tuner*, che regola alcuni parametri e addestra nuovamente il modello.

Stocasticità nella Modellazione Statistica:
Nel mondo della modellazione statistica, tutto viene considerato stocastico. I dati provengono da una distribuzione casuale. Un modello viene appreso dai dati osservati, che sono casuali; di conseguenza, anche il modello appreso è casuale. Il modello appreso viene poi valutato su dataset osservati, anch'essi casuali, quindi i risultati del test sono anch'essi casuali.

Garantire la Correttezza della Valutazione: Per garantire l'equità, i test devono essere eseguiti su un campione di dati che sia statisticamente indipendente da quello utilizzato durante l'addestramento. Il modello deve essere validato su dati che non ha precedentemente visto. Questo fornisce una stima dell'errore di generalizzazione, ovvero quanto bene il modello si generalizza su nuovi dati.

Impostazione Offline:
Nell'ambito offline, si ha solo un dataset storico, quindi bisognare ottenere un altro set indipendente per la validazione. Esistono delle tecniche comuni, illustrate in Figura 2.2.

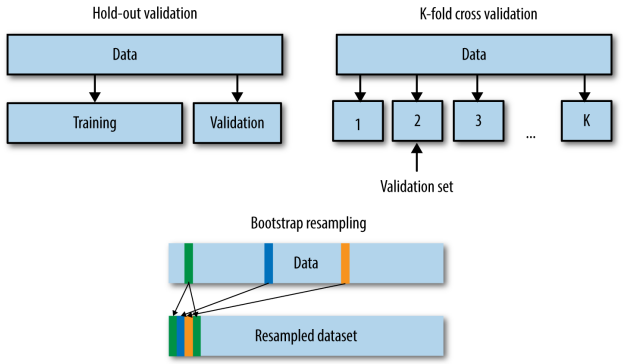

Figura 2.2: Tipi di tecniche di validazione [19]

L'ultimo passaggio della fase di prototipazione dovrebbe essere quello di addestrare un nuovo modello sull'intero set di dati disponibili utilizzando i migliori iperparametri trovati. Ciò dovrebbe includere tutti i dati precedentemente tenuti da parte per la convalida. Questo è il modello finale che dovrebbe essere distribuito in produzione.
Le due regole fondamentali sono:
1) Non mischiare mai dati di addestramento e dati di validazione.
2) L'addestramento, la convalida e il test dovrebbero avvenire su set di dati diversi.

2.1.3 Modelli di Regressione e Iperparametri

Infine, nel capitolo 4 è affrontato il tuning degli iperparametri.
Un modello di regressione ha la formula $w^T x = y$, dove x è un vettore che rappresenta le caratteristiche dei dati e y è una variabile scalare che rappresenta il target (una quantità numerica che si desidera apprendere e prevedere).
Questo modello assume che la relazione tra x e y sia lineare.
La variabile w è un vettore di pesi che rappresenta il vettore normale alla retta; esso specifica l'inclinazione della retta. Questo è noto come *parametro del modello*, che viene appreso durante la fase di addestramento.
"Addestrare un modello" significa usare una procedura di ottimizzazione per determinare il miglior parametro del modello che meglio "si adatta" ai dati.
Gli *iperparametri*, invece, sono valori che devono essere specificati al di fuori della procedura di addestramento. Essi possono controllare la capacità del modello, ovvero quanto è flessibile, quanti gradi di libertà ha nell'adattarsi ai dati. Un controllo adeguato della capacità del modello può prevenire l'overfitting, che si verifica quando il modello è troppo flessibile e il processo di addestramento si adatta troppo ai dati di training, perdendo così precisione predittiva sui nuovi dati di test.
Il modello di regressione standard non ha iperparametri, ma i seguenti modelli ne hanno:

- **Ridge regression**
- **Lasso**
- **Alberi di decisione** (profondità desiderata e numero di foglie nell'albero)
- **Support Vector Machines (SVMs)** (termine di penalità per la misclassificazione)

- **SVM kernelizzati** (parametri del kernel come la larghezza per kernel a base radiale - RBF)

Ottimizzazione degli Iperparametri L'ottimizzazione degli iperparametri è un compito di meta-ottimizzazione. Ogni prova di una particolare impostazione degli iperparametri comporta l'addestramento di un modello, ovvero un processo di ottimizzazione interno. L'esito dell'ottimizzazione degli iperparametri è la migliore impostazione degli iperparametri, mentre l'esito dell'addestramento del modello è la migliore impostazione dei parametri del modello. Gli Algoritmi per l'Ottimizzazione degli Iperparametri sono alcuni di quelli discussi a pagina 9, quindi Grid Search e Random Search, ma c'è anche l'Ottimizzazione Intelligente degli Iperparametri, che formata da:

- L'ottimizzazione senza derivata utilizza euristiche per determinare dove campionare successivamente.

- Ottimizzazione Bayesiana

- Ottimizzazione intelligente con foreste casuali: sia l'ottimizzazione bayesiana che quella con foreste casuali modellano la superficie di risposta con un'altra funzione, quindi campionano più punti in base a quanto indicato dal modello.

- Validazione Incrociata Annidata: nota come "test di ipotesi statistica" (decide tra un'ipotesi nulla e un'ipotesi alternativa).

2.1.4 Regole per il Test di Ipotesi e la Valutazione di Modelli

Quante osservazioni sono necessarie?

- Scegliendo il valore corretto per la potenza, il livello di significatività e il cambiamento desiderato; è possibile calcolare quante osservazioni sono necessarie in ogni gruppo.

- Non bisogna interrompere il test finché non si sono accumulate tali osservazioni e fin quando non viene rilevata una differenza "significativa", poiché il risultato potrebbe non essere attendibile e non avere ancora potenza statistica sufficiente per decisioni corrette.

La distribuzione della metrica è gaussiana? Di solito, quasi tutto converge a una distribuzione gaussiana grazie al Teorema del Limite Centrale, quando:
La metrica è una media.
La distribuzione dei valori della metrica ha un solo picco.
La metrica è distribuita simmetricamente attorno a questo picco.
Di solito in statistica la regola è che la media di più di 30 osservazioni inizia ad avvicinarsi a una distribuzione gaussiana, anche se per alcuni, negli ultimi anni, è considerata una fallacia [20].
Quando c'è una miscela di popolazioni, però, ci vorrà molto più tempo.

Regole pratiche per mitigare la violazione delle ipotesi del t-test

- Se la metrica è non negativa e ha una lunga coda (ad esempio, un conteggio), bisogna applicare una trasformazione logaritmica.
- Alternativamente, la famiglia delle trasformazioni di potenza tende a stabilizzare la varianza e rendere la distribuzione più simile a quella gaussiana.
- La distribuzione binomiale negativa è più indicata per i conteggi.

Le varianze sono uguali? Le varianze sono uguali per il gruppo di controllo e quello sperimentale?
Se i gruppi sono divisi uniformemente a caso, le varianze probabilmente saranno uguali, ma potrebbero esserci bias sottili nel metodo di divisione dei dati.
Il t-test di Welch è più robusto del t-test di Student e mantiene tassi di errore di tipo I vicini a quelli nominali in caso di varianze e campioni non uguali.
Il t-test di Welch rimane robusto per distribuzioni asimmetriche e campioni grandi, ma l'affidabilità diminuisce per campioni piccoli.

Cosa significa il p-value?

- Un p-value basso non implica un risultato significativo. Un p-value più piccolo non implica un risultato più significativo. Il p-value dipende dalla dimensione del campione, dalla differenza tra le due popolazioni e da quanto bene si possono stimare le medie vere.
  Quindi oltre al test di ipotesi e al calcolo del p-value, si dovrebbe sempre verificare l'intervallo di confidenza delle medie delle due popolazioni.
- Distribuzione gaussiana? Applicare le consuete stime dell'errore standard.

- Distribuzione non gaussiana? Calcolare una stima tramite bootstrap [3].
- Se la distribuzione non è per nulla vicina a quella gaussiana, non utilizzare il t-test; ma scegliere un test non parametrico che non faccia l'assunzione gaussiana, come il test U di Mann-Whitney [4].

Modelli multipli e ipotesi multiple

- Se si vogliono testare più modelli, significa che è un test di ipotesi multiple. È bene prestare attenzione perché più ipotesi aumentano la probabilità complessiva di falsi positivi.
- Test indipendenti? Un test con un tasso di falsi positivi di 0.05 riduce la probabilità che nessuno dei 20 test sia un falso positivo al 36%.
- Test non indipendenti? La probabilità potrebbe essere anche più alta.
- Dati di test in tempo reale? È utile applicare la procedura di Benjamini-Hochberg per controllare il tasso di falsi positivi.[5]

Quanto deve durare il test?

- Dipende dal numero di osservazioni necessarie per raggiungere la potenza statistica desiderata e dall'esperienza dell'utente.
- A volte, eseguire il test troppo a lungo può avere conseguenze etiche per l'utente.
- L'equilibrio tra la necessità di un arresto precoce e una potenza statistica sufficiente ha portato allo studio dell'analisi sequenziale, in cui i punti di arresto precoce vengono determinati a priori all'inizio delle prove.
- Per questo è utile monitorare la metrica offline sui dati in tempo reale oltre ai test online.

Altri autori in letteratura [21], suggeriscono altre linee guida:
Per costruire un modello di qualità, è innanzitutto necessario:

- Definire gli scenari di utilizzo (ad esempio, Come sarà utilizzato il sistema? Da chi? Quali sono le aspettative in termini di qualità? etc.).

[3]https://en.wikipedia.org/wiki/Bootstrapping_(statistics)
[4]https://it.wikipedia.org/wiki/Test_di_Wilcoxon-Mann-Whitney
[5]https://www.statisticshowto.com/benjamini-hochberg-procedure/

- Trasparenza e responsabilità (ad esempio, riproducibilità, interpretabilità e spiegabilità, auditabilità, minimizzazione e report dei possibili impatti negativi).
- Diversità, non discriminazione, equità, così come benessere sociale e ambientale (ad esempio, evitare bias ingiusti, accessibilità e design universale, partecipazione degli stakeholder, sostenibilità e rispetto ambientale).
- Sicurezza, protezione dei dati (ad esempio, rispetto della privacy, qualità e integrità dei dati, accesso ai dati e capacità di gestire dati rumorosi, errati, sconosciuti o avversariali).
- Robustezza tecnica, affidabilità, dipendenza (ad esempio, correttezza dell'output, stima dell'incertezza del modello, robustezza contro input dannosi, errori o situazioni impreviste).
- Agenzia e supervisione umana, aspetti legali ed etici (ad esempio, possibilità di supervisione e agenzia umana, rispetto dei diritti fondamentali).
- Il sistema in studio dovrebbe essere descritto. Questo può essere fatto, ad esempio, ispezionando le caratteristiche dei dati che rappresentano il sistema (ad esempio, distribuzione statistica, struttura topologica).
- Fare una previsione.

Il tipo di compiti eseguito (come la regressione, classificazione, clustering, rilevamento di outlier, riduzione dimensionale, etc.) ha anche un impatto sulla valutazione della qualità.
Ogni tipo di task è accompagnato da corrispondenti misure di qualità.
Ad esempio, per i compiti di classificazione, la bontà dell'adattamento può essere misurata tramite accuratezza, precisione, recall, F-score, etc. Tuttavia, per il clustering, sono necessarie altre misure.
Le misure scelte dipenderanno dal caso d'uso. Ad esempio, nel caso di compiti di classificazione binaria, *il costo di un falso positivo potrebbe non essere lo stesso di un falso negativo.*

2.1.5 Processo di Costruzione di un Modello di Qualità

Definizione del meta-modello di qualità
Per prima cosa, si descrivono le caratteristiche del nostro modello di qualità per il machine learning: la struttura di base che si vuole utilizzare per

documentare tutte le proprietà di qualità di interesse e le misure/metriche per quantificare tali proprietà. Questo ha portato alla definizione di un meta-modello di qualità come base comune per specificare i modelli di qualità.

L'elemento centrale di un modello di qualità è una proprietà di un'entità, ossia un attributo che caratterizza l'entità e che è legato alla qualità dell'entità stessa. Un'entità si riferisce a oggetti concreti che sono importanti per la qualità di un sistema ML (come i dati per l'addestramento dei modelli), mentre le sue proprietà rappresentano determinate caratteristiche di tali oggetti (come la completezza o la coerenza).
Un esempio di una *proprietà astratta* di un'entità potrebbe essere la qualità di un modello. Un esempio di una *proprietà specifica* di un'entità è la correttezza di un modello di classificazione, che può essere misurata utilizzando precisione, recall o misure di F-score.
La valutazione della qualità comprende quattro elementi di base:

- **Misurazione**: consiste nella raccolta dei dati di misurazione per i fattori specificati al livello più basso della gerarchia del modello di qualità, secondo le misure definite nel modello di qualità.
- **Valutazione**: comporta la valutazione del soddisfacimento delle preferenze di qualità associate al fattore.
- **Aggregazione**: comprende la sintesi delle valutazioni ottenute sui singoli fattori figli in modo bottom-up attraverso la gerarchia del modello di qualità, fino a ottenere una valutazione complessiva del sistema in esame.
- **Interpretazione**: è la traduzione delle valutazioni di qualità, potenzialmente astratte, in valutazioni comprensibili (intuitive) per i decisori umani.

Definizione del caso d'uso e del contesto applicativo
Il concetto di "qualità" dipende fortemente dal contesto applicativo e dal caso d'uso concreto. Questo include, ad esempio, la criticità del sistema in fase di sviluppo.
Ad esempio, gli aspetti principali della qualità considerati rilevanti potrebbero essere la correttezza funzionale del sistema (in termini di accuratezza della classificazione), ma anche i costi di sviluppo e operativi (in termini di tempo).

Lo sviluppo di un modello di machine learning può essere suddiviso in diverse fasi relative a

- Comprensione del problema da risolvere
- Raccolta dei dati necessari
- Costruzione del modello di machine learning stesso

Da queste diverse fasi, possiamo derivare differenti requisiti di qualità rilevanti riguardanti gli input e gli output del processo di sviluppo del modello di regressione.

Comprensione del Business
Il contesto aziendale impone assunzioni e vincoli:

- Come vengono progettati e costruiti i componenti (ad esempio, il componente di ML deve essere eseguito su un dispositivo embedded, quindi la complessità temporale e di memoria deve essere limitata)
- Disponibilità dei dati (i.e. quali dati sono stati raccolti o possono essere raccolti e etichettati)
- Requisiti per la privacy, sicurezza, equità, etc.

Comprensione dei Dati
Il tipo di metodo di analisi e le misure di valutazione corrispondenti che possono essere utilizzati dipendono da:

- Il tipo di dati disponibili (ad esempio, dati non strutturati come testi o immagini rispetto a dati strutturati come dati tabellari)
- La disponibilità di ground truth [6]
- La qualità dei dati (ad esempio, la risoluzione, la rappresentatività, la presenza di rumore, outlier o valori mancanti, etc.)
- Come i dati sono stati raccolti

[6]https://en.wikipedia.org/wiki/Ground_truth

Preparazione dei Dati
Diversi artefatti sono implementati per trasformare i dati grezzi in dati che possono essere utilizzati nei metodi di analisi:

- Etichettatura
- Rimozione
- Imputazione di outlier e dati mancanti
- Feature engineering
- Selezione delle feature

Tutte queste operazioni hanno un costo e un impatto sulla qualità dei risultati dell'analisi.

Modellazione
La qualità del modello di regressione è influenzata da diversi aspetti:

- Il tipo di task da risolvere (ad esempio, classificazione, clustering, regressione, rilevamento di anomalie, riduzione dimensionale, etc.)
- Il tipo di modello (rete neurale, albero decisionale, etc.)
- I dati utilizzati per la costruzione (i.e. addestramento)
- Valutazione degli artefatti sviluppati
- Il modo in cui i dati sono suddivisi per l'addestramento e la validazione
- Requisiti sulla complessità di runtime o vincoli di sicurezza
- Ricerca degli iperparametri
- Cross-validation
- Suddivisione indipendente tra training e test

La fase di modellazione include una parte di valutazione che mira a valutare il componente addestrato rispetto ai dati disponibili:

- Misurando le prestazioni tramite metriche come precisione, recall, etc.
- Eseguendo analisi di sensibilità
- Testando contro esempi avversariali

Deployment
La qualità in fase di runtime è influenzata dai dati che fluiscono nel componente.

Monitoraggio

- Flusso di dati
- Il contesto applicativo
- Il comportamento del componente di ML

L'architettura di un sistema di machine learning potrebbe dover essere distribuita in un ambiente completamente diverso (ad esempio, in un dispositivo mobile utilizzando un linguaggio di programmazione diverso). In questo caso, il modo in cui viene gestita la distribuzione (ad esempio, utilizzando formati di scambio come ONNX o PMML, utilizzando strumenti di versioning e CI/CD automatico, etc.) è cruciale.

Identificazione delle Entità Rilevanti in un Modello di regressione (o classificazione)
È importante analizzare tutte le entità rilevanti che potrebbero entrare in gioco, come ad esempio:

- I dati
- Il modello stesso
- L'infrastruttura su cui il modello è eseguito
- L'ambiente di esecuzione dell'intero sistema

Vista del Modello
Un componente di ML di solito consiste in diversi sottocomponenti organizzati in un grafo aciclico diretto (noto anche come pipeline). Nella vista del modello, [21] hanno fatto una distinzione tra ciò che chiamiamo tipo di modello (ad esempio, albero decisionale, rete neurale, etc.) e un modello addestrato (ad esempio, un'istanza specifica di una rete neurale addestrata su un dataset specifico utilizzando un algoritmo di addestramento specifico). Questa distinzione è stata effettuata per separare le proprietà di qualità relative a un'istanza di entità specifica da quelle relative al tipo di entità.

Vista dei Dati
Il processo di addestramento di un componente di ML richiede di suddividere i dati di sviluppo in diversi sottoinsiemi:

- **Addestramento**: utilizzato per determinare i parametri del modello durante l'addestramento.
- **Validazione**: utilizzato per la sintonizzazione degli iperparametri (ad esempio, la profondità massima di un albero decisionale).
- **Test**: utilizzato per fornire una valutazione imparziale del modello addestrato (si suppone che sia indipendente dai sottoinsiemi di addestramento e validazione).

Vista del Sistema
Un componente di ML è solitamente organizzato in una pipeline di compiti. Il suo sviluppo è per natura sperimentale. Una data pipeline può essere addestrata più volte con diversi tipi di modelli, algoritmi di addestramento o dataset per trovare le migliori combinazioni (noto anche come problema di ottimizzazione combinata del modello e degli iperparametri, o CASH problem) [22].

1. Il modo in cui viene effettuata la ricerca e il modo in cui i sottocomponenti sono connessi ha un impatto sulla qualità.
2. Una volta che un modello è stato sviluppato (cioè, addestrato), è necessario che venga messa in atto una specifica architettura per esportare i modelli addestrati.
3. Il componente distribuito è a sua volta parte di un sistema più grande, cioè consuma dati da una o più fonti e interagisce con altri componenti basati su ML o classici.
4. Le proprietà di qualità tipiche relative alla vista del sistema includono dipendenze dai dati e cicli di feedback.

Vista dell'Infrastruttura
In questa vista, l'attenzione è maggiormente focalizzata sulle proprietà di qualità relative a come il sistema è concretamente implementato (ad esempio, hardware, librerie di addestramento).
Una qualità qui è l'efficienza degli algoritmi di addestramento ed esecuzione, ma anche l'idoneità dell'infrastruttura sia per l'addestramento che per l'esecuzione dei componenti. (I modelli di deep learning addestrati attualmente

utilizzati per l'elaborazione del linguaggio naturale sono di diversi gigabyte e richiedono diversi giorni (o settimane) di addestramento su macchine hardware dedicate (cluster GPU), quindi il modello addestrato non può essere eseguito su dispositivi embedded a causa delle limitazioni computazionali e di storage.)

Vista Ambientale
Diversi aspetti ambientali possono avere un'influenza diretta sulla qualità. Questi includono, ad esempio, aspetti che causano deficit di qualità nei dati. Questo è fortemente legato alla nozione di *concept drift*, il che significa che un'evoluzione dei dati invalida il modello dei dati.
La tabella 2.1, proposta da [21], propone una buona panoramica degli elementi necessari per ogni tipo di vista, entità, proprietà ed esempi per costruire un modello di qualità.

Tabella 2.1: Valutazione di qualità e misure dei Modelli di Regressione

Vista	**Entità**	**Proprietà**	**Esempi di Misure di Qualità e Checklists**
Modello	Tipo di modello	Appropriatezza: Grado in cui il tipo di modello è appropriato per il compito attuale e può gestire il tipo di dato attuale (es. numerico, categorico).	Prerequisiti per il tipo di modello
	Modello addestrato	Correttezza nello sviluppo: capacità del modello di eseguire il compito sul dataset di sviluppo.	Precisione, Richiamo, F-score per l'addestramento
		Correttezza in esecuzione (adeguatezza del modello): stesso come sopra, misurato sul dataset in esecuzione.	Precisione, Richiamo, F-score in esecuzione

		Rilevanza (compromesso bias-varianza): equilibrio tra underfitting e overfitting.	Varianza della bontà di adattamento della validazione incrociata
		Robustezza: capacità di gestire rumore o dati mancanti mantenendo previsioni accurate.	Perdita di Accuratezza Equalizzata (ELA)
		Stabilità: coerenza dei risultati quando addestrato su diversi sottoinsiemi di dati.	Stabilità della validazione incrociata "leave-one-out"
		Equità: capacità di fornire decisioni imparziali.	Probabilità equalizzate
		Interpretabilità: quanto facilmente il modello può essere compreso dagli esseri umani.	Misure di complessità (es. numero di parametri, profondità)
		Utilizzo delle risorse: risorse utilizzate dal modello addestrato.	Spazio di archiviazione richiesto
Dati	Dati di sviluppo	Rappresentatività: grado in cui i dati sono rappresentativi della popolazione statistica.	Test statistici (es. t-test)
		Correttezza: grado in cui i dati sono privi di errori.	Misure di rilevamento degli outlier (es. Z-score)
		Completezza: grado in cui i dati sono privi di valori mancanti.	Numero di valori mancanti

		Attualità: quanto sono aggiornati i dati per il compito attuale.	Età dei dati
		Intra-coerenza: coerenza all'interno di un dataset (es. nessuna contraddizione).	Intervalli di valore, conteggio parole
		Indipendenza Train/-Test: indipendenza tra i dataset di addestramento e test.	Test statistici (es. t-test)
		Bilanciamento: grado in cui tutte le classi sono rappresentate in modo uguale.	Rapporto delle classi
		Assenza di bias: i dati sono privi di bias contro gruppi.	Rapporti di gruppo
		Inter-coerenza: coerenza tra diversi dataset (es. formattazione).	Intervalli di valore, rilevamento incrociato degli outlier
Ambiente	Processo di addestramento	Impatto ambientale: impatto del processo di addestramento sull'ambiente.	Consumo energetico
	Società	Impatto sociale: Impatto del componente ML sulla società.	Impatto sui dipendenti
	Scopo	Conformità allo scopo: conformità dell'applicazione ML con il suo scopo previsto.	Misure di rilevamento delle novità
Sistema	Supervisione dell'output	Efficacia: grado in cui l'algoritmo di supervisione dell'output rileva risultati falsi del componente ML.	Tasso di rilevamento dei falsi positivi/-negativi

		Sovraccarico/efficienza della supervisione: risorse utilizzate per monitorare un dato componente ML.	Tempo, memoria utilizzata, etc.
	Supervisione dello scopo	Efficacia: grado in cui l'algoritmo di supervisione dello scopo rileva cambiamenti di contesto.	Numero di casi fuori scopo
		Sovraccarico/efficienza della supervisione: risorse utilizzate per monitorare l'ambito dell'applicazione.	Tempo, memoria utilizzata, etc.
	Altri componenti non-ML	Proprietà di qualità della norma ISO/IEC 25010 non elencate qui per motivi di spazio.	–
Infrastruttura	Infrastruttura	Idoneità dell'infrastruttura: grado in cui l'infrastruttura corrisponde alle esigenze del componente ML.	Capacità computazionali e di archiviazione
	Algoritmo di addestramento	Efficienza dell'addestramento: risorse utilizzate per addestrare un dato modello.	Tempo, memoria utilizzata, etc.
	Algoritmo di esecuzione	Efficienza dell'esecuzione: risorse utilizzate per eseguire un dato modello addestrato.	Tempo, memoria utilizzata, etc.

Il processo riassunto utilizzato per costruire un modello di Machine Learning di alta qualità è visibile in figura 2.3:

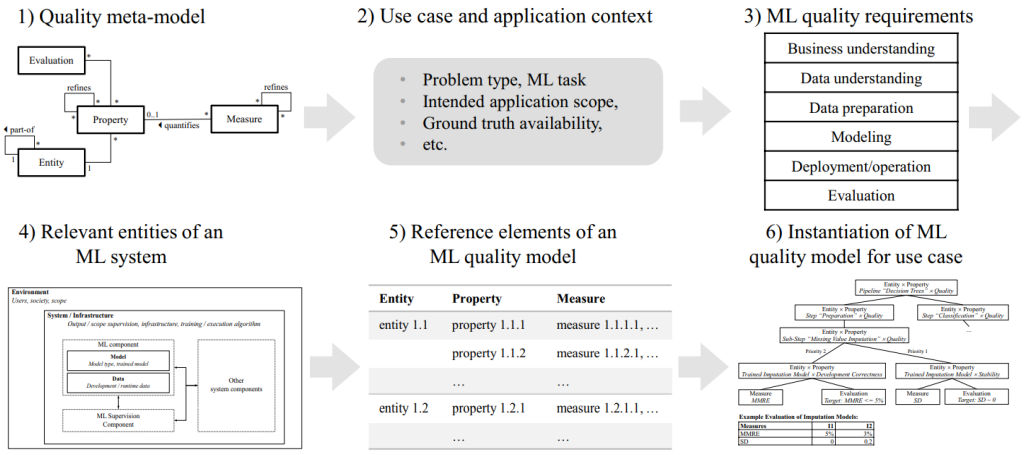

Figura 2.3: Esempio di schema di prototipazione nel costruire un modello di Machine Learning [21].

2.1.6 CRISP-ML(Q)

Altri autori in letteratura, hanno studiato un sistema chiamato **CRISP-ML(Q)**[23] che è un modello di processo di apprendimento automatico con metodologia di garanzia della qualità
A causa della mancanza di un modello di processo specifico per applicazioni di Machine Learning (ML),
molte organizzazioni si affidano a modelli alternativi, come il modello Cross-Industry Standard Process per il Data Mining (CRISP-DM)[7].
Questo modello è basato sull'esperienza industriale nel data mining e viene considerato il più adatto per progetti industriali tra i modelli di processo correlati.
Un modello di ML deve essere adattabile a un ambiente in continua evoluzione, altrimenti le prestazioni del modello si degraderanno nel tempo. Pertanto, è necessaria una costante attività di monitoraggio e manutenzione del modello ML dopo la sua esecuzione. La qualità non è definita solo dall'idoneità del prodotto allo scopo, ma anche dalla qualità delle esecuzioni dei task in ogni fase dello sviluppo di un'applicazione ML.

[7]`https://www.kde.cs.uni-kassel.de/wp-content/uploads/lehre/ws2012-13/kdd/files/CRISPWP-0800.pdf`

In questo modo gli errori verranno rilevati il prima possibile, minimizzando i costi nelle fasi successive del processo di sviluppo.

Metriche di Successo per i modelli di Regressione e Classificazione
Non esiste una metrica che funzioni meglio per tutte le applicazioni ML, e la scelta di una metrica può privilegiare un modello rispetto a un altro. Tuttavia, è possibile definire:

- **Criteri di successo per il ML**: questi traducono l'obiettivo aziendale in criteri di successo per il ML come misura dei rischi tecnici.
- **Criteri di successo economico**: Nella pratica aziendale e nella produzione, è buona norma aggiungere un criterio di successo economico sotto forma di Key Performance Indicator (KPI)[8] al progetto. Un KPI è una misura economica per valutare la rilevanza attuale e futura di un'applicazione, ed è essenziale per la definizione di obiettivi aziendali concreti. Ad esempio:
  Un'applicazione di ML è in fase di sviluppo per il controllo qualità nella produzione. L'obiettivo è superare il tasso di fallimento manuale attuale del 3%.
- Il criterio di successo aziendale viene fissato nel raggiungimento di un tasso di fallimento inferiore al 3%.
- Il criterio minimo di successo per l'applicazione ML è definito nel raggiungimento di un'accuratezza superiore al 97%.
  Grazie ai KPI, quando un rischio non è fattibile (o si verifica un rischio), vengono implementate adeguate misure di QA (Quality Assurance) .

Fattibilità
Un test di fattibilità dovrebbe valutare la situazione e decidere se continuare lo sviluppo. È cruciale che l'analisi includa la disponibilità, la dimensione e la qualità del set di dati di addestramento. Un campione troppo piccolo comporta il rischio di basse prestazioni sui dati del Test Set, rischio che può essere mitigato **aumentando la qualità dei dati o aggiungendo conoscenza del dominio**.

Raccolta dei Dati
La raccolta dei dati è una task importante che richiede tempo e costi. Se non vengono raccolti abbastanza dati, il progetto ML dovrebbe essere sospeso o ritardato. È importante anche la gestione delle versioni dei dati, poiché la raccolta dei dati è un'attività iterativa, non statica.

[8]https://it.wikipedia.org/wiki/Indicatore_chiave_di_prestazione

Verifica della Qualità dei Dati
È fondamentale avere o creare una descrizione dei dati: i dati devono essere descritti a livello meta e secondo le loro proprietà statistiche. La conoscenza del dominio è necessaria per comprendere i dati e le loro caratteristiche. Tra i requisiti sui dati troviamo:

- i valori attesi delle feature (un intervallo per feature continue o una lista per feature discrete);
- il formato dei dati;
- il numero massimo di valori mancanti.

I dati che non soddisfano le condizioni attese possono essere contrassegnati come anomalie e valutati manualmente o automaticamente. Per evitare errori di bias, è consigliato discutere i requisiti con un esperto del dominio, mentre per mitigare il rischio di una rappresentazione insufficiente di casi estremi, è buona norma utilizzare tecniche di esplorazione dei dati per investigare la distribuzione del campione[9].

Selezione dei Dati Scartare feature sotto-utilizzate che non apportano benefici al modello, poiché possono creare falle per potenziali errori.
Il fenomeno della *curse of dimensionality* (Maledizione della dimensionalità) implica che, più feature vengono selezionate, più campioni sono necessari: un numero di campioni esponenzialmente crescente è necessario per prevenire la dispersione dei dati nello spazio delle feature:

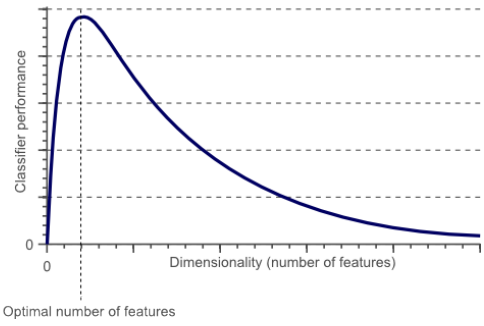

Figura 2.4: Visualizzazione del fenomeno della maledizione delle dimensionalità.[10]

[9]https://www.heavy.ai/learn/data-exploration
[10]`https://www.visiondummy.com/wp-content/uploads/2014/04/dimensionality_vs_performance.png`

2.1.7 Selezione delle Feature e Bias

La selezione delle feature potrebbe comportare il rischio di *selection bias*, che può essere ridotto quando eseguita all'interno della *cross-validation* del modello.

- La selezione delle feature dovrebbe essere analizzata da esperti del dominio per evitare potenziali bias.
- I campioni che non soddisfano la verifica di qualità dei dati e non sono plausibili dovrebbero essere rimossi dal dataset.
- Classi sbilanciate (troppi valori positivi o negativi in una classe rispetto ad un'altra): è suggerito fare *over-sampling* della classe minoritaria e/o *under-sampling* della classe maggioritaria. L'*over-sampling* aumenta l'importanza della classe minoritaria ma potrebbe comportare *overfitting* su di essa. L'*under-sampling*, invece, riduce i punti dati della classe maggioritaria e va fatto con attenzione per mantenere le caratteristiche dei dati e ridurre la probabilità di introdurre bias.
- Evitare di rimuovere punti vicini alla *decision boundary* o più punti dallo stesso cluster.
- *Clean data*: filtri di elaborazione del segnale possono essere usati per rimuovere segnali irrilevanti dai dati e migliorare il rapporto segnale-rumore, ma il processo di filtraggio deve essere documentato per non rimuovere parti importanti del segnale nei dati.
- I valori mancanti, NaN (Not a Number) e valori speciali devono essere interpretati/convertiti in valori leggibili dal modello.
- Nuove feature: nuove feature possono essere derivate da quelle esistenti basandosi su conoscenza del dominio, ad esempio:
  - trasformando le feature dal dominio del tempo al dominio delle frequenze,
  - discretizzando le feature continue in *bin*,
  - aumentando le feature con ulteriori basate su quelle esistenti,
  - clustering,

  - metodi di riduzione dimensionale come Kernel-PCA,
  - *auto-encoders*.

- Le feature nominali e le etichette dovrebbero essere trasformate in *one-hot encoding*.
- Le feature sotto-utilizzate dovrebbero essere rimosse.
- *Data augmentation*: processo di generazione artificiale di nuovi dati da quelli esistenti, conoscendo le invarianti nei dati.
- Standardizzare i dati: ISO 8000 raccomanda l'uso di unità SI secondo il Sistema Internazionale delle Quantità. Definire un insieme fisso di standard e unità aiuta a evitare errori nel processo di fusione e nella rilevazione di dati errati.
- Normalizzare le feature per portare a tassi di convergenza più rapidi, ad esempio nelle reti neurali. Il processo di normalizzazione deve essere applicato sia al *test set* che al *training set* utilizzando gli stessi parametri di normalizzazione.
- Definire misure di qualità del modello: almeno cinque proprietà complementari di un modello dovrebbero essere valutate: Robustezza, Scalabilità, Spiegabilità, Complessità del modello, Domanda di risorse. Quando c'è un'applicazione pratica, la spiegabilità o la robustezza potrebbero essere considerate più importanti dell'accuratezza.

2.1.8 Selezione del modello

Non esiste un modello che offra le migliori prestazioni per tutte le classi di problemi, ma in generale è buona pratica:

- Iniziare con modelli di bassa capacità e aumentare gradualmente la capacità.
- L'aggiunta di conoscenze del dominio dovrebbe sempre migliorare la qualità del modello; altrimenti, è preferibile rimuoverle per evitare bias errati.

2.1.9 Addestramento del Modello

Il problema di apprendimento include: un obiettivo, un ottimizzatore, una regolarizzazione e una convalida incrociata [24].

Obiettivo
È un sostituto per valutare le prestazioni del modello.

Ottimizzatore
Definisce la strategia di apprendimento e come adattare i parametri del modello per migliorare l'obiettivo.

Regolarizzazione
Riduce il rischio di *overfitting* e può aiutare a trovare soluzioni uniche.

Cross-validation
Viene eseguita per la selezione delle feature, l'ottimizzazione degli iperparametri e per testare la proprietà di generalizzazione su dati non visti. Si basa sulla suddivisione dei dati storici in *training*, *validation* e *test set*, dove quest'ultimo viene utilizzato come proxy (sostituto) per l'ambiente target.

Utilizzare dati non etichettati
Durante il processo di addestramento (ad esempio, eseguendo pre-addestramento non supervisionato e algoritmi di apprendimento semi-supervisionato).

Transfer learning
Può essere utilizzato per pre-addestrare la rete su un dataset proxy (ad esempio, da simulazioni) che somiglia ai dati originali per estrarre caratteristiche comuni.

Metodi di compressione o potatura
Possono essere utilizzati per ottenere un modello più compatto. Nei metodi a kernel, l'approssimazione a rango ridotto della matrice del kernel è uno strumento essenziale per affrontare problemi di apprendimento su larga scala (ad esempio, nelle reti neurali è comune potare i pesi della rete).

Metodi di Ensemble
Addestrano più modelli per prendere decisioni basate sulle decisioni aggregate dei modelli individuali; tecniche come *Boosting, Bagging o Mixture of Experts* sono tecniche consolidate per aggregare le decisioni di più modelli [25]:

Bagging
Comporta l'addestramento di più istanze di un modello su diversi sottoinsiemi dei dati di addestramento, creati tramite bootstrapping (campionamento casuale con sostituzione). Un esempio è l'algoritmo Random Forest.

Boosting
Combina le previsioni di modelli deboli in sequenza. Ogni nuovo modello corregge gli errori commessi dai suoi predecessori. Esempi sono AdaBoost e Gradient Boosting.

Stacking
Addestramento di più modelli (studenti di base come Regressione regolarizzata, Random Forest, Gradient Boosting e XGBoost) e combinazione delle loro previsioni utilizzando un meta-modello.

2.1.10 Riproducibilità

I modelli di machine learning (ML) sono difficili da riprodurre a causa delle procedure di addestramento per lo più non convesse e stocastiche e della suddivisione casuale dei dati. Nella letteratura, è stato proposto di distinguere la riproducibilità su due livelli differenti. Prima di tutto, è necessario assicurarsi che il metodo stesso sia riproducibile e, in secondo luogo, i suoi risultati. [11]

- Il modello deve essere riproducibile: l'algoritmo utilizzato, il set di dati, gli iperparametri e l'ambiente di runtime (ad esempio, versioni software, hardware e semi casuali).
- Validare le prestazioni medie e valutare la varianza del modello su diversi semi casuali.
- Tenere traccia delle prestazioni del modello modificato e delle sue cause attraverso le modifiche del modello precedente.
- La qualità deve affrontare la riproducibilità del modello tracciando e memorizzando informazioni pertinenti sul modello come codice sorgente ML per addestrare il modello, set di dati e ambiente di calcolo.
- Il set di test deve coprire l'intera distribuzione degli input e considerare tutte le invarianti, ad esempio, trasformazioni dell'input che non cambiano l'etichetta, nei dati.
- Il modello deve essere robusto: se i dati vengono perturbati, ad esempio, rumorosi o errati, o manipolati in anticipo per ingannare il sistema (ad esempio, attacchi avversari), la robustezza del metodo diminuirà.

[11] https://www.cs.mcgill.ca/ jpineau/ReproducibilityChecklist.pdf

Un approccio consiste nell'aggiungere diversi tipi di input rumoroso o falsificato ai dati o variare gli iperparametri per caratterizzare la capacità di generalizzazione del modello e poi verificare le metriche di qualità per la robustezza.

- Aumentare l'interpretabilità per il praticante di ML e per l'utente finale: l'interpretabilità di un modello aiuta a trovare errori e consente strategie, ad esempio, arricchendo il set di dati, per migliorare le prestazioni complessive.

La principale ragione per cui un modello può deteriorarsi nel tempo è radicata nella violazione dell'assunto che i dati di addestramento e i dati di input per l'inferenza provengano dalla stessa distribuzione.

Principali cause delle violazioni

Distribuzione dei dati non stazionaria
Le distribuzioni dei dati cambiano nel tempo e portano a un set di addestramento obsoleto.
È possibile una variazione nelle caratteristiche e/o nelle etichette.

Degradazione dell'hardware
L'hardware su cui il modello è implementato e l'hardware dei sensori invecchieranno nel tempo. I sensori diventano più rumorosi o si guastano nel tempo.
Ciò sposterà il dominio del sistema e dovrà essere adattato dal modello o ripristinato.

Aggiornamenti del sistema
Gli aggiornamenti sul software o sull'hardware del sistema possono causare uno spostamento nell'ambiente (ad esempio, le unità di un segnale potrebbero essere cambiate durante un aggiornamento e, senza notifiche, il modello utilizzerà questo input scalato per fare previsioni errate).

2.2 Il problema del bias e della varianza

Il bias è considerato un errore sistematico che si verifica nel modello di apprendimento automatico stesso a causa di assunzioni errate nel processo di ML. È l'errore tra la previsione media del modello e la verità di fondo.
Alto bias implica:

- Fallimento nel catturare le tendenze corrette dei dati
- Potenziale di underfitting
- Modelli più generalizzati/over-semplificati
- Alta percentuale di errore

La varianza è il cambiamento del modello quando si utilizzano porzioni diverse del set di dati di addestramento.
La varianza implica:

- Rumore nel set di dati
- Potenziale di overfitting
- Modelli complessi
- Tentativo di avvicinare tutti i punti dati il più possibile

Tabella degli Algoritmi
La tabella 2.2 contiene un elenco di algoritmi comuni e il loro comportamento atteso riguardo al bias e alla varianza:

Tabella 2.2: Comportamento di algoritmi comuni rispetto a Bias e Varianza

Algoritmo	**Bias**	**Varianza**
Regressione Lineare	Alto Bias	Bassa Varianza
Albero Decisionale	Basso Bias	Alta Varianza
Bagging	Basso Bias	Alta Varianza (minore rispetto all'Albero Decisionale)
Random Forest	Basso Bias	Alta Varianza (minore rispetto all'Albero Decisionale e Bagging)

2.3 Definizione di Bias-Varianza

Consideriamo un set di dati $D = (X, y)$ composto da N coppie di variabili indipendenti e dipendenti. Supponiamo che i dati veri siano generati da un modello rumoroso $y = f(x) + \epsilon$, dove ϵ è normalmente distribuito con media zero e deviazione standard σ_ϵ.
Si assume di avere una procedura statistica (ad es. regressione ai minimi quadrati) per formare un predittore $f(x; \hat{\theta})$ che fornisce la previsione del nostro modello per un nuovo punto dati x. Questo stimatore è scelto minimizzando una funzione di costo che si considera essere l'errore quadratico:

$$C(y, f(X; \theta)) = \sum_i (y_i - f(x_i; \theta))^2$$

Pertanto, le stime per i parametri sono:

$$\hat{\theta}_{\mathcal{D}} = \arg_\theta \min \mathcal{C}(\mathbf{y}, f(\mathbf{X}; \boldsymbol{\theta})).$$

sono una funzione del set di dati D. Otterremmo un errore diverso:

$$\mathcal{C}\left(\boldsymbol{y}_j, f\left(\boldsymbol{x}_j; \hat{\boldsymbol{\theta}}_{\mathcal{D}_j}\right)\right)$$

$$\text{per ogni set di dati } D_j = (y_j, X_j),$$

in un universo di possibili set di dati ottenuti estraendo N campioni dalla vera distribuzione dei dati. Denotiamo un valore atteso su tutti questi set di dati come E_D.
Inoltre, sarebbe meglio anche fare la media su diverse istanze del "rumore" ϵ e si denota il valore atteso sul rumore con E_ϵ. Pertanto, è possibile decomporre l'errore di generalizzazione atteso come:

$$E_{(D,\epsilon)}[C(y, f(X; \hat{\theta}_D))] = E_{(D,\epsilon)}\left[\sum_i (y_i - f(x_i; \hat{\theta}_D))^2\right]$$

Dopo aver decomposto il secondo termine otteniamo due termini: il primo è chiamato "Bias"$\hat{2}$:

$$\text{Bias}^2 = \sum_i \left(f(\mathbf{x}_i) - \mathrm{E}_{\mathcal{D}}\left[f(\mathbf{x}_i; \hat{\boldsymbol{\theta}}_{\mathcal{D}})\right]\right)^2$$

Il bias misura la deviazione del valore atteso del nostro stimatore (cioè il

valore asintotico del nostro stimatore nel limite di dati infiniti) dal valore vero. E il secondo termine è chiamato varianza:

$$\text{Var} = \sum_i \text{E}_{\mathcal{D}} \left[\left(f\left(\mathbf{x}_i; \widehat{\boldsymbol{\theta}}_{\mathcal{D}}\right) - \text{E}_{\mathcal{D}} \left[f\left(\mathbf{x}_i; \widehat{\boldsymbol{\theta}}_{\mathcal{D}}\right) \right] \right)^2 \right]$$

La varianza misura quanto lo stimatore scelto fluttua a causa degli effetti del campione finito. Combinando queste espressioni, si vede che l'errore atteso fuori campione,

$$E_{\text{out}} := \mathbb{E}_{\mathcal{D},\epsilon} \left[\mathcal{C}\left(\mathbf{y}, f\left(\mathbf{X}; \hat{\theta}_{\mathcal{D}}\right)\right) \right],$$

Può essere decomposto in:

$$E_{\text{out}} = \text{Bias}^2 + \text{Var} + \text{Noise} \quad (\text{con Noise} = \sum_i \sigma_\epsilon^2)$$

Quindi, il trade-off bias-varianza riassume la tensione fondamentale nell'apprendimento automatico, in particolare nell'apprendimento supervisionato, tra la complessità di un modello e la quantità di dati di addestramento necessari per addestrarlo. A causa della limitata disponibilità di dati, è spesso utile utilizzare un modello meno complesso con un bias maggiore (un modello il cui rendimento asintotico è peggiore di un altro modello) perché è più facile da addestrare e meno sensibile al rumore di campionamento derivante dall'avere un set di dati di addestramento di dimensioni finite (varianza più piccola). Tutto questo può essere visualizzato in figura 2.5:

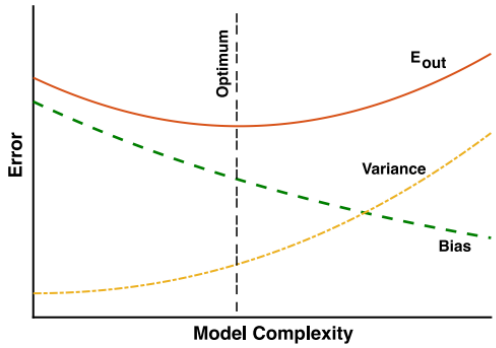

Figura 2.5: Si nota come il bias diminuisca sempre con la complessità del modello, ma la varianza aumenta con la complessità del modello. Quindi, le prestazioni ottimali vengono raggiunte a livelli intermedi di complessità del modello [37].

In generale, a seconda della quantità di dati di addestramento, potrebbe essere più favorevole utilizzare un modello meno complesso e ad alto bias per fare previsioni.

2.3.1 Come ridurre il bias

Esistono alcuni metodi in letteratura, come l'uso dell'algoritmo Fair-SMOTE [38], che utilizza un modello più complicato (ma come visto in precedenza, a un certo punto il modello diventa troppo complesso per la quantità di dati di addestramento e l'errore di generalizzazione diventa grande a causa dell'elevata varianza).
Oppure, uno dei metodi per ridurre la varianza è mantenere fissa la dimensione dell'ensemble, facendo la media delle previsioni dei modelli non correlati. Per fare questo le tecniche comuni sono Bagging, Boosting, Stacking.

Altre tecniche sono esposte nella tabella 2.3 mostrata in [39]:

Tabella 2.3: Approcci in letteratura per la Gestione del Bias

Approccio	**Pro**	**Contro**	**Metodi**	**Descrizione**
Pre-processing	Prevenzione del bias tramite pre-processing sofisticato	Possibile distorsione dei dati che causa perdita di informazione	Rietichettatura	Trasforma l'etichetta dei dati di training in base a una specifica condizione
			Generazione	Genera campioni per il training set o identifica gruppi svantaggiati
			Rappr. Equa	Impara una nuova rappresentazione senza bias che mantiene l'informazione
In-processing	Dati completi applicabili indipendentemente dal set	Instabilità dovuta ai vincoli nel modello	Ottim. Vincolata	Aggiunge un termine di equità nella funzione obiettivo
			Regolariz.	Aggiunge penalità legate all'equità durante l'addestramento
Post-processing	Raggiungimento di bias zero possibile su tutti i modelli	Intervento utente necessario per analisi risultati	Calibrazione	Assicura probabilità di previsioni positive pari agli esempi positivi
			Thresholding	Imposta soglie decisionali con minimo intervento

Mentre altre tecniche sono state illustrate in figura 2.6:

CRISP-DM Phase / Bias	Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment
Social Bias			Rapid Prototyping Reweighing Data Massaging Disparate Impact Remover Learning Fair Representation Optimized Pre-Processing	Prejudice Remover Adversarial Debiasing Equalized Odds Multiple Models Latent Variable Model Model Interpretability		
Measurement Bias	Diversity in Teams Exchange with Domain Experts	Proxy Estimation	Rapid Prototyping			
Representation Bias	Diversity in Teams	Data Plotting Exchange with Domain Experts	Reweighing Data Augmentation	Model Interpretability		
Label Bias		Exchange with Domain Experts	Data Massaging			
Algorithmic Bias			Rapid Prototyping	Exchange with Domain Experts Resampling Model Interpretability Multitask Learning		
Evaluation Bias				Resampling	Representative Benchmark Subgroup Validity Data Augmentation	
Deployment Bias	Diversity in Teams Consequences in Context		Rapid Prototyping			Monitoring Plan Human Supervision
Feedback Bias						Human Supervision Randomness

Figura 2.6: Matrice CRISP-DM dei Bias e delle Fasi del Data Mining [40].

2.4 Iperparametri

Gli iperparametri sono delle impostazioni di configurazione che determinano la struttura generale e il comportamento di un modello di machine learning prima dell'inizio del processo di addestramento. In letteratura esistono diversi studi in cui c'è una ricerca e una dimostrazione di quanto migliorino le prestazioni dei modelli andando a fare una ricerca completa degli iperparametri che fanno performare al meglio un modello di Machine Learning.

Uno di questi è di [26], in cui si discute che una delle principali difficoltà nel lavorare con i problemi di Machine Learning è la regolazione degli iperparametri.

La figura 2.7 mostra un esempio dei diversi tipi di regolazione:

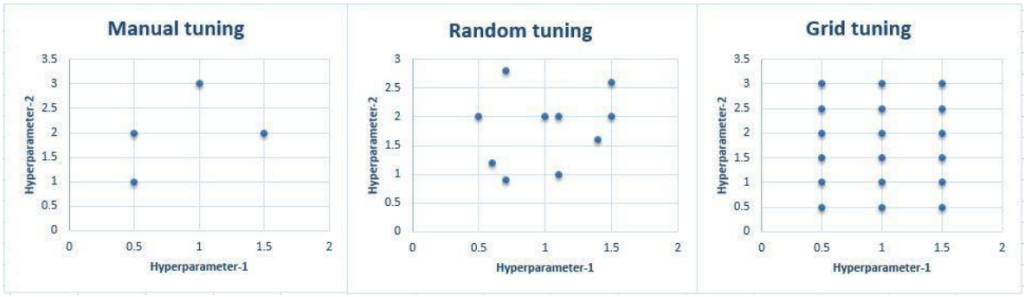

Figura 2.7: Diversi tipi di tuning [26].

Gli algoritmi di regolazione degli iperparametri sono o senza modello o basati su modello.
Gli algoritmi senza modello non utilizzano conoscenze sullo spazio delle soluzioni estratte durante l'ottimizzazione; esempi sono:

- Manual Search: assumiamo i valori dei parametri basandoci sulla nostra esperienza precedente. Esempi dei principali parametri utilizzati dai classificatori Random Forest sono criterion, max_depth, n_estimators, min_samples_split, etc.

- Random Search: combinazioni casuali di iperparametri; richiede più tempo di elaborazione, ma solitamente porta a migliori prestazioni [27].

  È stato dimostrato che fare la ricerca di iperparametri fornisce risultati migliori grazie a diversi vantaggi:

  1. Il budget può essere impostato in modo indipendente secondo la distribuzione dello spazio di ricerca, quindi può funzionare meglio, soprattutto quando più iperparametri non sono distribuiti uniformemente [28].
  2. Poiché ogni valutazione è indipendente, è facile da parallelizzare e allocare risorse.
  3. Maggiore consumo di tempo porterà a una maggiore probabilità di trovare il miglior set di iperparametri.

- Grid Search: l'utente imposta una griglia di iperparametri e addestra il modello in base a ciascuna possibile combinazione. La ricerca a griglia funziona bene quando le migliori combinazioni sono già determinate,

ma in generale l'intera procedura richiede molto tempo. È più pratica quando il numero totale di parametri nel modello è ridotto, ad esempio $M < 10$. La griglia è M-dimensionale, quindi il numero di soluzioni di test è proporzionale a L^M, dove L è il numero di soluzioni di test lungo ciascuna dimensione della griglia. Può ridurre il tempo complessivo di consumo, ma il principale svantaggio è che non può convergere al valore ottimale globale [29].

- L'ottimizzazione Bayesiana è uno dei metodi di tuning degli iperparametri automatizzati più ampiamente utilizzati per trovare il valore ottimale globale in meno passaggi, ma i suoi risultati sono sensibili ai parametri del modello surrogato e l'accuratezza dipende dalla qualità del modello di apprendimento [30]. Essa determina i livelli di valutazione futuri sulla base dei risultati precedenti.
  Utilizza due fattori chiave: un modello surrogato e una funzione di acquisizione. Il primo ha lo scopo di mappare tutti i punti attualmente osservati nella funzione obiettivo, mentre il secondo determina l'uso di punti diversi, bilanciando esplorazione e sfruttamento.
  La figura 2.8 mostra come funziona l'ottimizzazione bayesiana:

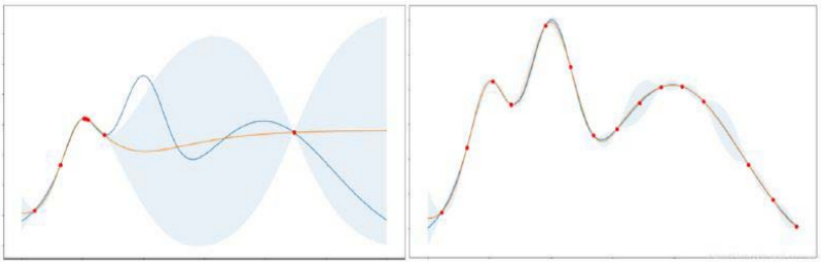

Figura 2.8: Ottimizzazione bayesiana basata sull'esplorazione (a sinistra) e basata sullo sfruttamento (a destra); l'ombra indica l'incertezza.[26]

Modelli di sostituzione comuni per l'ottimizzazione bayesiana (BO) includono il processo gaussiano (GP), la foresta casuale (RF) e il Parzen estimator (TPE).

2.4.1 Ottimizzazione degli iperparametri

Chiamata anche HPO (Hyper Parameter Optimization), ha come obiettivo trovare il valore globale ottimale x^* della funzione obiettivo $f(x)$, che può essere valutata per un qualunque $x \in X$, $x^* = \arg\min_{x \in X} f(x)$. X è uno spazio di iperparametri che può contenere variabili categoriche, discrete e continue. Le componenti dell'HPO sono:

1. *Uno stimatore* che può essere un regressore o qualsiasi classificatore con una o più funzioni obiettivo.

2. *Uno spazio di ricerca.*

3. *Un metodo di ottimizzazione* per trovare le migliori combinazioni.

4. *Una funzione* per confrontare l'efficacia delle varie configurazioni di iperparametri.

Altri autori invece hanno discusso dell'ottimizzazione degli iperparametri negli algoritmi di Machine Learning [31]. Tra i vari algoritmi c'è *Hyperband*, una famosa tecnica di ottimizzazione basata su bandit, che può essere considerata una versione migliorata della Random Search (RS). Genera piccole versioni dei dataset e assegna lo stesso budget a ciascuna combinazione di iperparametri. In ogni iterazione di Hyperband, le configurazioni di iperparametri con prestazioni scarse vengono eliminate per risparmiare tempo e risorse. Ci sono anche algoritmi metaeuristici come quelli citati in [32]:

- Algoritmo genetico (GA): rileva combinazioni di iperparametri con buone prestazioni in ogni generazione e le passa alla generazione successiva fino a identificare la combinazione migliore.

- Ottimizzazione con sciame di particelle (PSO): ogni particella comunica con le altre per rilevare e aggiornare l'attuale ottimo globale in ogni iterazione fino a rilevare l'ottimo finale.

- Algoritmi di annealing simulato classico.

- Algoritmi della colonia di formiche.

- Sono particolarmente adatti ai problemi di ottimizzazione degli iperparametri (HPO) con ampi spazi di configurazione grazie alla loro alta efficienza.

- Possono essere utilizzati in reti neurali profonde (DNN), che hanno uno spazio di configurazione ampio con più iperparametri, come i tipi di attivazione e ottimizzatore, il tasso di apprendimento, il tasso di drop-out, etc.

- Le funzioni non convesse hanno più ottimi locali, ma solo uno di questi è l'ottimo globale. La maggior parte dei problemi di ottimizzazione in ML e HPO sono problemi di ottimizzazione non convessa, quindi l'utilizzo di metodi di ottimizzazione inappropriati può portare solo a un ottimo locale invece che a uno globale.

- Esistono molti metodi tradizionali per risolvere problemi di ottimizzazione, come:

  - Discesa del gradiente: usa la direzione del gradiente negativo come direzione di ricerca per avvicinarsi all'ottimo, ma non può garantire di rilevare l'ottimo globale a meno che la funzione obiettivo non sia convessa.
  - Metodo di Newton: usa la matrice inversa della matrice Hessiana[12] per ottenere l'ottimo[13]
  - Gradiente coniugato: cerca lungo le direzioni coniugate costruite dal gradiente dei punti dati noti per rilevare l'ottimo.

Esempi di Iperparametri scelti

- In SVM, il grado della funzione kernel polinomiale deve essere ottimizzato solo quando il tipo di kernel scelto è polinomiale.

- Il numero di caratteristiche considerate in un albero decisionale dovrebbe essere compreso tra 0 e il numero totale di caratteristiche.

- Il numero di cluster in k-means non dovrebbe essere superiore al numero di punti dati.

- Le prestazioni del modello possono essere valutate utilizzando diverse metriche, come accuratezza, RMSE, F1-score e tasso di falsi allarmi.

[12]https://it.wikipedia.org/wiki/Matrice_hessiana
[13]https://www.cs.toronto.edu/ rgrosse/courses/csc421_2019/slides/lec07.pdf

Il processo dell'HPO

1. Selezionare la funzione obiettivo e le metriche di performance;
2. Selezionare gli iperparametri che richiedono l'ottimizzazione, riassumere i loro tipi e determinare la tecnica di ottimizzazione appropriata;
3. Addestrare il modello di ML utilizzando la configurazione predefinita degli iperparametri o valori comuni come modello di base;
4. Avviare il processo di ottimizzazione con un ampio spazio di ricerca come dominio di fattibilità degli iperparametri, determinato da test manuali e/o conoscenza del dominio;
5. Restringere lo spazio di ricerca in base alle regioni degli iperparametri già testati che hanno dato buoni risultati o esplorare nuovi spazi di ricerca se necessario;
6. Restituire la configurazione degli iperparametri con le migliori performance come soluzione finale.

Problemi dell'HPO

1. Molti metodi tradizionali di ottimizzazione progettati per risolvere problemi di ottimizzazione convessa o differenziabile sono spesso inadeguati per i problemi di HPO, poiché questi metodi possono restituire un ottimo locale invece di un ottimo globale.
2. Gli iperparametri dei modelli di ML includono iperparametri continui, discreti, categorici e condizionali, quindi molti metodi tradizionali di ottimizzazione numerica che mirano solo a trattare variabili numeriche o continue non sono adatti per i problemi di HPO.
3. Il tempo di valutazione della funzione è spesso ignorato in molti modelli di ottimizzazione a scatola nera (BBO), quindi molti algoritmi BBO sono spesso inadeguati per i problemi di HPO con budget di tempo e risorse limitati.

Algoritmi di apprendimento supervisionato
Gli algoritmi di apprendimento supervisionato mappano le caratteristiche di input a un obiettivo tramite l'addestramento su dati etichettati e comprendono principalmente [33]:

- Modelli lineari (regressione)
- k-nearest neighbors (KNN) (classificazione ma anche regressione)
- Macchine a vettori di supporto (SVM) (classificazione e regressione)
- Naïve Bayes (NB) (quasi solo classificazione)
- Modelli basati su alberi decisionali (classificazione e regressione)
- Alberi bagged (classificazione e regressione)

Tecniche di ottimizzazione degli iperparametri

- **Babysitting**[36]: consiste nel testare diversi valori di iperparametri basati su esperienza, intuizioni o analisi dei risultati precedenti. Il processo continua fino a una scadenza o fino a ottenere risultati soddisfacenti.

Analisi delle tecniche di ottimizzazione

- **Hyperband** bilancia le prestazioni del modello e l'uso delle risorse, risultando più efficiente rispetto a RS, soprattutto con risorse limitate.
- **GS**, **RS** e **Hyperband** presentano il limite di trattare ogni iperparametro in modo indipendente, non considerando correlazioni. Questo le rende inefficaci per algoritmi con iperparametri condizionali, come SVM, DBSCAN e regressione logistica.
- Gli algoritmi **gradient-based** supportano solo iperparametri continui e rilevano solo minimi locali (non globali) per problemi HPO non convessi, limitandosi all'ottimizzazione di iperparametri come il learning rate nei modelli DL.

2.4.2 Iperparametri nella Regressione Logistica

- Il metodo di regolarizzazione utilizzato nella penalizzazione, 'l1', 'l2', 'elasticnet' o 'none' (chiamato 'penalty' in sklearn).
- Il coefficiente 'C', che determina la forza della regolarizzazione del modello.
- Il tipo di 'solver', che rappresenta il tipo di algoritmo di ottimizzazione, può essere impostato su 'newton-cg', 'lbfgs', 'liblinear', 'sag' o 'saga'.

2.4.3 Iperparametri in KNN

Il numero di vicini più prossimi considerati, k, è l'iperparametro più cruciale [34].

- k è troppo piccolo? $\rightarrow$ il modello sarà in under-fitting.
- k è troppo grande? $\rightarrow$ il modello sarà in over-fitting e richiederà un alto tempo di calcolo.
- La funzione di pesatura utilizzata nella previsione può anche essere scelta tra 'uniform' (i punti sono pesati in modo uguale) o 'distance' (i punti sono pesati dall'inverso della loro distanza).
- La metrica di distanza e il parametro di potenza della metrica di Minkowski[14] possono anche essere ottimizzati, poiché ciò può portare a un miglioramento marginale.

2.4.4 Iperparametri in SVM

I tipi di kernel comuni in SVM includono:

- Kernel lineari
- Funzione di base radiale (RBF)
- Kernel polinomiali
- Kernel sigmoidali

Altri iperparametri da ottimizzare sono:

- γ, denotato da 'gamma' in sklearn, è l'iperparametro condizionale dell'iperparametro 'tipo di kernel' quando è impostato su polinomiale, RBF o sigmoidale.
- r, specificato da 'coef0' in sklearn, è l'iperparametro condizionale dei kernel polinomiali e sigmoidali.
- D, che rappresenta il 'grado' della funzione kernel polinomiale.
- 'epsilon', che indica l'errore di distanza della sua funzione di perdita presente nei modelli di regressione a vettori di supporto (SVR).

[14]https://mathworld.wolfram.com/MinkowskiMetric.html

2.4.5 Iperparametri in Naïve Bayes

Esistono diversi tipi di classificatori Naïve Bayes. I quattro tipi principali di modelli NB sono: Bernoulli NB, Gaussian NB, multinomial NB e complement NB. Per Gaussian NB si assume che la probabilità delle caratteristiche segua una distribuzione gaussiana. Normalmente, non ci sono iperparametri che necessitano di essere ottimizzati per gli algoritmi Naïve Bayes.

2.4.6 Iperparametri in modelli basati su alberi

- 'max_depth' è un iperparametro essenziale che controlla la complessità degli algoritmi DT, poiché un albero più profondo ha più sotto-alberi per prendere decisioni più accurate.
- La qualità delle suddivisioni può essere misurata impostando una funzione di misura, denotata da 'criterion' in sklearn.
- Il metodo di selezione delle suddivisioni, 'splitter', può anche essere impostato su 'best' per scegliere la migliore suddivisione, o 'random' per selezionare una suddivisione casuale.
- max_features: (il numero di caratteristiche considerate per generare la migliore suddivisione).
- min_samples_split/min_samples_leaf/max_leaf_nodes: (il numero minimo/massimo di punti dati per suddividere un nodo decisionale o per ottenere un nodo foglia).
- min_weight_fraction_leaf: la frazione minima ponderata dei pesi totali.

Esistono molti algoritmi di ensemble basati su alberi decisionali proposti per migliorare le prestazioni del modello combinando più alberi decisionali, inclusi modelli di foresta casuale (RF), alberi extra (ET) e potenziamento di gradiente estremo (XGBoost).
Ad esempio, in XGBoost si cerca di minimizzare la seguente funzione obiettivo [35]:

$$Obj = -\frac{1}{2}\sum_{j=1}^{t}\frac{G_j^2}{H_j+\lambda} + \gamma t \quad (22)$$

dove t è il numero di foglie in un albero decisionale, G e H sono le somme delle statistiche del gradiente di primo e secondo ordine della funzione di costo, mentre c e k sono coefficienti di penalità.

Altri parametri fondamentali da ottimizzare includono:

- `n_estimators`: numero di alberi da combinare.
- `min_child_weight`: somma minima dei pesi in un nodo figlio.
- `subsample` e `colsample_bytree`: controllano il rapporto di sotto-campionamento di istanze e caratteristiche.
- `gamma` (riduzione minima di perdita per una divisione), `alpha` (L1), `lambda` (termine di regolarizzazione L2 sui pesi) e `learning_rate`.

2.4.7 Iperparametri negli Algoritmi di apprendimento ensemble

Gli algoritmi più comuni sono:

- **Voting**: utilizza la regola di voto maggioritario per combinare stimatori singolari, migliorando l'accuratezza. Iperparametri:
  - `Hard/soft`: determina il risultato di classificazione tramite voto maggioritario o probabilità previste medie.
- **Bagging/Bootstrap aggregating**: allena multipli stimatori di base su sottoinsiemi estratti casualmente. Iperparametri:
  - `base_estimator/n_estimators`: tipo e numero di stimatori di base nell'ensemble.
  - `max_samples` e `max_features`: dimensioni di campioni e caratteristiche per sottoinsiemi diversi.
- **AdaBoost**: allena successivamente i learner di base, concentrandosi sugli errori dei learner precedenti. Iperparametri:
  - `base_estimator`: tipo di stimatore di base.
  - `n_estimators`: massimo numero di stimatori per terminare il boosting.
  - `learning_rate`: riduce il contributo di ogni classificatore.

2.4.8 Iperparametri nei modelli di deep learning

Poiché tutti i modelli di rete neurale hanno un livello di input e uno di output, la complessità dipende principalmente da:

- Numero di livelli nascosti.
- Numero di neuroni per livello.
- Tipo di funzione di perdita:
  - entropia incrociata per problemi di classificazione binaria.
  - entropia incrociata multiclasse per classificazione multiclass.
  - RMSE per problemi di regressione.
- Tipo di funzione di attivazione ('softmax', 'ReLU', 'sigmoid', 'tanh', o 'softsign').
- Tipo di ottimizzatore: discesa del gradiente stocastico (SGD), Adam, RMSprop.
- Tasso di apprendimento: determina la dimensione dei passi a ogni iterazione per convergere alla funzione obiettivo.
  - Un tasso di apprendimento elevato velocizza l'apprendimento, ma il gradiente può oscillare intorno a un minimo locale o non convergere.
  - Un tasso basso converge lentamente ma richiede più epoche.
- `Drop-out rate`: un metodo di regolarizzazione per ridurre l'overfitting, disattivando casualmente una percentuale di neuroni.
- Dimensione del mini-batch: numero di campioni processati prima dell'aggiornamento del modello.
- Numero di epoche: numero di passaggi completi attraverso il set di addestramento. Da ottimizzare per evitare l'overfitting.
- `Early stop patience`: il training si ferma quando l'accuratezza non migliora dopo un certo numero di epoche consecutive.

Algoritmi di apprendimento non supervisionato
Gli algoritmi di apprendimento non supervisionato sono usati per trovare pattern in dati non etichettati e si dividono in algoritmi di clustering e riduzione dimensionale in base agli obiettivi.

2.4.9 Iperparametri negli algoritmi di Clustering

K-means

- `n_clusters`: numero di cluster
- `init`: metodo di inizializzazione dei centroidi, può essere - 'k-means++', 'random' o array definito
- `n_init`: numero di esecuzioni con semi di centroidi diversi
- `max_iter`: numero massimo di iterazioni

EM

- `n_components`: numero di cluster o distribuzioni gaussiane
- `max_iter`: numero di iterazioni EM
- `tol`: soglia di convergenza

Clustering Gerarchico

- `linkage`: distanza tra insiemi di osservazioni, può essere - `ward`: minimizza la varianza - `complete`: massimizza la varianza - `average`: media la varianza - `single`: minima distanza tra cluster

DBSCAN

- `eps`: raggio di scansione
- `min_samples`: numero minimo di punti vicini

PCA e LDA

- `n_components`: numero di feature estratte
- `SVD solver` (PCA): estrae e separa le informazioni, può essere 'auto', 'full', 'arpack' o 'randomized'
- `shrinkage` (LDA): stima per regolarizzare la matrice di covarianza e migliorare la stabilità del modello.

In [31], c'è un confronto tra le ottimizzazioni degli iperparametri negli algoritmi più comuni (n è il numero di valori degli iperparametri e k è il numero di iperparametri):

Tabella 2.4: Metodi di Ottimizzazione degli Iperparametri (HPO)

Metodo HPO	**Punti di Forza**	**Limitazioni**	**Complessità Temporale**
GS	Semplice.	• Dispendioso in termini di tempo • Efficiente solo con IP categorici.	$O(n^k)$
RS	• Più efficiente di GS • Permette parallelizzazione.	• Non considera i risultati precedenti • Non efficiente con IP condizionali.	$O(n)$
Basato su Gradiente	• Rapida velocità di convergenza per IP continui.	• Supporta solo IP continui • Può rilevare solo ottimi locali.	$O(n^k)$
BO-GP	• Rapida velocità di convergenza per IP continui.	• Scarsa capacità di parallelizzazione • Non efficiente con IP condizionali.	$O(n^3)$
SMAC	• Efficiente con tutti i tipi di IP.	• Scarsa capacità di parallelizzazione • Non efficiente con IP condizionali.	$O(n \log n)$
BO-TPE	• Efficiente con tutti i tipi di IP • Mantiene dipendenze condizionali.	• Scarsa capacità di parallelizzazione.	$O(n \log n)$
Hyperband	• Permette parallelizzazione.	• Non efficiente con IP condizionali • Richiede sottoinsiemi rappresentativi con budget ridotti.	$O(n \log n)$
BOHB	• Efficiente con tutti i tipi di IP • Permette parallelizzazione.	• Richiede sottoinsiemi rappresentativi con budget ridotti.	$O(n \log n)$
GA	• Efficiente con tutti i tipi di IP • Non richiede buona inizializzazione.	• Scarsa capacità di parallelizzazione.	$O(n^2)$
PSO	• Efficiente con tutti i tipi di IP • Permette parallelizzazione.	• Richiede inizializzazione appropriata.	$O(n \log n)$

Successivamente, c'è una panoramica completa dei modelli ML più comuni, dei loro iperparametri, delle tecniche di ottimizzazione idonee e delle librerie Python disponibili.

Tabella 2.5: Panoramica dei Modelli di Machine Learning

Algoritmo ML	**Iperparametri Principali**	**Iperparametri Opzionali**	**Metodi HPO**	**Librerie**
Regressione Lineare	-	-	-	-
Ridge & Lasso	alpha	-	BO-GP	Skpot
Regressione Logistica	penalty, c, solver	-	BO-TPE, SMAC	Hyperopt, SMAC
KNN	n_neighbors	weights, p, algorithm	BOs, Hyperband	Skpot, Hyperopt, SMAC
SVM	C, kernel, epsilon (per SVR)	gamma, coef0, degree	BO-TPE, SMAC, BOHB	Hyperopt, SMAC, BOHB
Naive Bayes	alpha	-	BO-GP	Skpot
Albero Decisionale	criterion, max_depth, min_samples_split, min_samples_leaf, max_features	splitter, min_weight_fraction_leaf, max_leaf_nodes	GA, PSO, BO-TPE, SMAC, BOHB	TPOT, Optunity, SMAC, BOHB
Random Forest & Extra Trees	n_estimators	max_depth, criterion, min_samples_split, min_samples_leaf, max_features	GA, PSO, BO-TPE, SMAC, BOHB	TPOT, Optunity, SMAC, BOHB
XGBoost	n_estimators, max_depth, learning_rate, subsample, colsample_bytree, min_child_weight, gamma, alpha, lambda	-	GA, PSO, BO-TPE, SMAC, BOHB	TPOT, Optunity, SMAC, BOHB
Voting	estimators, voting	weights	GS	Sklearn
Bagging	base_estimator, n_estimators	max_samples, max_features	GS, BOs	sklearn, Skpot, Hyperopt, SMAC
AdaBoost	base_estimator, n_estimators, learning_rate	-	BO-TPE, SMAC	Hyperopt, SMAC
Deep Learning	numero di hidden layers, 'units' per layer, loss, optimizer, activation, learning_rate, dropout rate, epochs, batch_size, early stop patience	numero di frozen layers (se viene utilizzato il transfer learning)	PSO, BOHB	Optunity, BOHB
K-means	n_clusters	init, n_init, max_iter	BOs, Hyperband	Skpot, Hyperopt, SMAC, Hyperband
Clustering Gerarchico	n_clusters	distance_threshold, linkage	BOs, Hyperband	Skpot, Hyperopt, SMAC, Hyperband
DBSCAN	eps, min_samples	-	BO-TPE, SMAC, BOHB	Hyperopt, SMAC, BOHB
Mixture Gaussiana	n_components	covariance_type, max_iter, tol	BO-GP	Skpot
PCA	n_components	svd_solver	BOs, Hyperband	Skpot, Hyperopt, SMAC, Hyperband
LDA	n_components	solver, shrinkage	BOs, Hyperband	Skpot, Hyperopt, SMAC, Hyperband

2.4.10 Librerie Open source per HPO

Ci sono varie librerie open source per affrontare i problemi dell'ottimizzazione degli iperparametri. I più comuni sono:

1. **Sklearn:** *GridSearchCV*[15] e *RandomizedSearchCV*[16] possono essere implementati.

2. **Spearmint:** Ottimizzazione bayesiana con il processo gaussiano come modello surrogato[17].

3. **BayesOpt:** utilizza un processo gaussiano come modello surrogato per calcolare la funzione obiettivo basata sulle valutazioni passate e utilizza una funzione di acquisizione per determinare i prossimi valori[18].

4. **Hyperopt:** include RS e BO-TPE come algoritmi di ottimizzazione. Può utilizzare più modelli per modellare iperparametri gerarchici ed è parallelizzabile poiché utilizza MongoDB come database centrale per memorizzare le combinazioni di iperparametri[19].

5. **SMAC:** utilizza BO con la foresta casuale come modello surrogato e supporta variabili categoriche, continue e discrete[20].

6. **BOHB:** è una combinazione di ottimizzazione bayesiana e Hyperband. Utilizzandolo per valutare l'istanza, è possibile ottenere un compromesso tra le prestazioni del modello e il budget attuale[21].

7. **Optunity:** fornisce diverse tecniche di ottimizzazione, inclusi GS, RS, PSO e BO-TPE, e supporta tutti i tipi di iperparametri[22].

8. **Skopt:** implementa diversi modelli di ottimizzazione basata su modelli sequenziali, inclusi RS e BO-GP, mostrando buone prestazioni con spazi di ricerca piccoli e una corretta inizializzazione[23].

[15]https://tinyurl.com/52ry5b4k
[16]https://tinyurl.com/yw8y4tz7
[17]https://github.com/HIPS/Spearmint
[18]https://github.com/rmcantin/bayesopt/
[19]https://github.com/hyperopt/hyperopt
[20]https://github.com/automl/SMAC3
[21]https://github.com/goktug97/bohb-hpo
[22]https://github.com/claesenm/optunity
[23]https://github.com/scikit-optimize/scikit-optimize

9. **GpFlowOpt:** supporta l'esecuzione di BO-GP su GPU utilizzando la libreria TensorFlow ed è una buona scelta se BO viene utilizzato in modelli di deep learning con risorse GPU disponibili[24].

10. **Talos:** è progettato per l'ottimizzazione degli iperparametri con modelli Keras. È possibile implementare diverse tecniche di ottimizzazione, inclusi GS, RS e riduzione probabilistica[25].

11. **Sherpa:** supporta calcoli paralleli e ha diversi metodi di ottimizzazione, inclusi GS, RS, BO-GP (via GPyOpt), Hyperband e training basato su popolazione (PBT)[26].

12. **Osprey:** progettato per ottimizzare gli iperparametri, ha GS, RS, BO-TPE (via Hyperopt) e BO-GP (via GPyOpt) come strategie disponibili[27].

13. **FAR-HO:** è un pacchetto di ottimizzazione degli iperparametri che impiega algoritmi basati sul gradiente con TensorFlow. Contiene alcuni ottimizzatori basati sul gradiente, come i metodi di retrogradiente iper e di gradiente iper in avanti[28].

14. **Hyperband:** esiste una classe chiamata `HyperbandSearchCV` in Hyperband che può essere combinata con sklearn e utilizzata per problemi di HPO. Utilizza la cross-validation per la valutazione[29].

15. **DEAP:** contiene diversi algoritmi evolutivi come GA e PSO. Integra meccanismi di parallelizzazione come il multiprocessing e pacchetti di machine learning come Sklearn[30].

16. **TPOT:** uno strumento Python per auto-ML che utilizza la programmazione genetica per ottimizzare i pipeline di ML. Ad esempio, *TPOTClassifier* è la sua funzione principale e diversi iperparametri aggiuntivi di GA devono essere impostati per adattarsi a problemi specifici[31].

[24]https://github.com/GPflow/GPflowOpt
[25]https://github.com/autonomio/talos
[26]https://github.com/sherpa-ai/sherpa
[27]https://github.com/msmbuilder/osprey
[28]https://github.com/lucfra/FAR-HO
[29]https://github.com/thuijskens/scikit-hyperband
[30]https://github.com/DEAP/deap
[31]https://github.com/EpistasisLab/tpot

17. **Nevergrad:** libreria Python open-source che include un'ampia gamma di ottimizzatori, come fast-GA e PSO. Può essere utilizzata per ottimizzare tutti i tipi di iperparametri, inclusi iperparametri discreti, continui e categorici, scegliendo diversi ottimizzatori[32].

18. Esistono alcuni framework per automatizzare la ricerca dell'ottimizzazione degli iperparametri e della ricerca dell'architettura, come dei framework **Auto ML**[33] e **Neural Architecture Search** [34].

Infine, in letteratura [41] vengono presentati dieci principi fondamentali per implementare efficacemente tecniche di machine learning in *biologia computazionale*, anche chiamata *bioinformatica*, che si occupa di apprendere i sistemi della natura (biologici) per poterli apprendere e riprodurre così da utilizzare modelli ottimizzati già dalla natura grazie all'evoluzione [35].
Ognuno dei principi offre indicazioni pratiche per evitare errori comuni e massimizzare le prestazioni dei modelli.

1. Controllo e preparazione del dataset di input
È importante verificare la quantità di dati, normalizzare le feature, randomizzare l'ordine dei dati e rimuovere valori anomali per migliorare la qualità dell'analisi, garantendo che il modello possa apprendere da dati ben strutturati e rappresentativi del fenomeno in esame.

2. Suddivisione del dataset in tre sottoinsiemi: training, validation e test set
La suddivisione in tre insiemi evita la sovrapposizione di dati tra training e test, prevenendo punteggi di performance falsati. Il test set deve essere utilizzato solo alla fine, dopo la scelta e l'ottimizzazione degli iperparametri.

3. Inquadramento del problema biologico nella categoria di algoritmo adeguata
La corretta classificazione del problema (supervisionato o non supervisionato, classificazione o regressione) facilita la selezione dell'algoritmo più adatto.

[32]https://github.com/FacebookResearch/Nevergrad
[33]https://openml.github.io/automlbenchmark/frameworks.html
[34]https://github.com/automl/NASLib
[35]https://cbd.cmu.edu/about-us/what-is-computational-biology.html

4. Inizio con algoritmi semplici
Per evitare complessità eccessive e ridurre il rischio di errori, si consiglia di iniziare con un algoritmo semplice (ad es., k-NN o k-means). Questo permette una gestione più controllata del processo e consente di effettuare debug accurati. Algoritmi complessi dovrebbero essere introdotti solo se giustificati dalla natura specifica dei dati e del problema, quindi solo se strettamente necessari.

5. Gestione dei dataset sbilanciati
Nei dataset con classi squilibrate, strategie come il bilanciamento dei pesi delle classi e il sotto-campionamento possono migliorare la capacità del modello di riconoscere correttamente tutte le classi, anche quelle minoritarie. Queste tecniche sono essenziali per garantire un'adeguata rappresentazione delle classi nei modelli di machine learning.

6. Ottimizzazione degli iperparametri
La scelta ottimale degli iperparametri, come il numero di vicini in k-NN, può migliorare significativamente l'accuratezza del modello. L'ottimizzazione tramite tecniche come la *Grid search*(Capitolo 1) consente di trovare i parametri che offrono le migliori prestazioni utilizzando set di validazione dedicati.

7. Minimizzazione dell'overfitting
Per ridurre l'overfitting, vengono consigliate pratiche come la cross-validation e la regolarizzazione. Questi metodi permettono al modello di generalizzare efficacemente senza adattarsi eccessivamente al dataset di training, garantendo una migliore capacità predittiva su dati non osservati.

8. Valutazione delle prestazioni con il coefficiente di correlazione di Matthews (MCC)
L'MCC è una metrica affidabile, in particolare per dataset sbilanciati, poiché considera tutte le componenti della matrice di confusione. In alternativa è possibile usare la curva di Precision-Recall che rappresenta un'opzione utile per valutare le prestazioni predittive del modello in contesti biologici, offrendo una visione più accurata rispetto alla curva ROC. Ad esempio, se su un modello si ha un'accurcay del 91% e un F1-score del 95,24%, si penserebbe che il modello ha ottime prestazioni, ma non è così: se andando anche ad analizzare il valore del MCC, questo dovesse essere di 0.14, significherebbe che il modello sta funzionando in modo simile a un'ipotesi casuale. L'MCC agirebbe quindi come campanello di allarme informando che il modello statistico sta funzionando male.

9. Utilizzo di software open source
Si possono usare strumenti open source (come R, Python o Weka) che promuovono la trasparenza e la collaborazione accademica. Questi strumenti non impongono vincoli di licenza, garantendo la possibilità di riutilizzo del codice e la condivisione senza barriere istituzionali.

10. Richiesta di feedback da esperti o comunità online
Può essere utile consultare esperti o comunità Q&A online (come Cross Validated [36] e Stack Overflow [37]) come strategia utile per identificare errori, ricevere suggerimenti e migliorare l'approccio metodologico. Questo feedback anticipa eventuali critiche dei revisori e arricchisce il progetto, aumentando la qualità e la solidità della ricerca.

[36]https://stats.stackexchange.com/
[37]https://stackoverflow.com/

Capitolo 3

Esperimenti e Risultati

Il presente capitolo si propone di validare le linee guida teoriche attraverso l'applicazione su dataset reali, al fine di verificare empiricamente l'eventuale impatto positivo (o negativo) di tali pratiche sulla qualità complessiva dei modelli e delle metriche di valutazione.

3.1 Strumenti e Librerie Utilizzate

Per fare analisi dei dati sui dataset è stato utilizzato Python[1] come linguaggio di programmazione, uno dei più comuni nell'analisi dei dati. Per garantire la riproducibilità degli esperimenti indipendentemente dall'hardware disponibile, è stato scelto Google Colab[2], un servizio Jupyter Notebook ospitato che non richiede alcuna configurazione per essere utilizzato e fornisce accesso gratuito alle risorse di elaborazione, tra cui GPU e TPU. Per una scrittura più veloce e pulita del codice sono stati utilizzati alcuni strumenti di intelligenza artificiale per generare testo come ChatGPT [3] e ClaudeAI [4]

3.1.1 Specifiche Hardware

Le più importanti specifiche Hardware dell'ambiente usato (come accelerazione Hardware è stata scelta una CPU in quanto era più prestante di usare la GPU), offerto gratuitamente da Google Colab sono le seguenti:

[1]https://www.python.org/
[2]https://colab.research.google.com/
[3]https://chatgpt.com/
[4]https://claude.ai/new

Tabella 3.1: Caratteristiche Hardware CPU Google Colab

Caratteristica	**Dettaglio**
Processore	2
Vendor ID	GenuineIntel
Nome del modello	Intel(R) Xeon(R) CPU @ 2.20GHz
Frequenza CPU	2199.998 MHz
Cache size	56320 KB
CPU Cores	1
APICID	2 (per ogni processore)
Clflush Size	64
Cache Alignment	64
Address Sizes	46 bits physical, 48 bits virtual
RAM di sistema	12.7 Gb
Memoria disco	107.7 Gb

3.1.2 Librerie Usate

Le librerie e pacchetti usate per poter utilizzare i vari algoritmi per fare analisi dati sono state, per la classificazione e la regressione (in grassetto quelle comuni):

Tabella 3.2: Librerie per la Classificazione

Librerie
`tensorflow =2.11`
`scikeras`
`keras`
`seaborn`
`Optunity`
`sklearn-deap`
`colorama`
`tpot`
`ipykernel`
`jupyter`
`scikeras[tensorflow]`
`hpbandster`
`tabulate`

In più per la classificazione è stata utilizzato il framework Optuna [5].
Per utilizzare gli algoritmi Hyperband e Genetic Algorithm in entrambi gli algoritmi si è utilizzata la versione 1.0.2 di Scikitlearn, in quanto la nuova versione non supporta più i pacchetti di tali algoritmi.

3.2 Dataset Utilizzati

I dataset da analizzare scelti sono stati 3 per la classificazione e 3 per la regressione.

3.2.1 Dataset per la Classificazione

1. Blood Transfusion Center [43]
   Descrizione:
   Questo dataset è stato ottenuto dal Centro di Servizi di Trasfusione di Sangue di Hsin-Chu, Taiwan. Include informazioni su 748 donatori casualmente selezionati. Ogni record contiene le seguenti variabili:

   - **R**ecency: mesi dall'ultima donazione.
   - **F**requency: numero totale di donazioni.
   - **M**onetary: volume totale di sangue donato (in c.c.).
   - **T**ime: mesi dalla prima donazione.
   - Variabile binaria che indica se il donatore ha effettuato una donazione a marzo 2007 (1: sì, 0: no).

2. Breast Cancer Wisconsin [44]
   Descrizione:
   Questo dataset contiene caratteristiche calcolate da immagini digitalizzate di aspirati con ago sottile di masse al seno. Le caratteristiche descrivono proprietà dei nuclei cellulari e sono state selezionate tramite ricerca esaustiva. Il piano di separazione è stato calcolato usando il metodo MSM-T (Multisurface Method-Tree), descritto in:

   - K. P. Bennett, "Decision Tree Construction Via Linear Programming", 1992.
   - K. P. Bennett and O. L. Mangasarian, "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", 1992.

[5] https://github.com/optuna/optuna

3. Digits MNIST [45]
   Descrizione:
   Il database MNIST è un benchmark standard per il riconoscimento ottico di cifre scritte a mano. Le immagini sono preprocessate (segmentazione e normalizzazione) per garantire una base comune per il confronto tra algoritmi di apprendimento automatico.

3.2.2 Dataset per la Regressione

1. Life Expectancy [47]
   Descrizione:
   In questo dataset, sono stati analizzati modelli predittivi sull'aspettativa di vita utilizzando algoritmi di machine learning. Le caratteristiche includono fattori educativi, sanitari, economici e di welfare sociale. Lo studio propone miglioramenti nei modelli rispetto ai metodi tradizionali basati esclusivamente sui tassi di mortalità.

2. Obesity [42]
   Descrizione:
   Questo dataset contiene dati su abitudini alimentari e condizioni fisiche di individui di Messico, Perù e Colombia. Include 17 attributi e 2111 record, classificati in sette livelli di obesità: Peso Insufficiente, Peso Normale, Sovrappeso (I e II), Obesità (I, II e III). Il 77% dei dati è stato generato sinteticamente tramite Weka e SMOTE, mentre il 23% proviene da utenti tramite una piattaforma web.

3. Boston Housing [46]
   Descrizione:
   Questo studio utilizza dati del mercato immobiliare di Boston per stimare la disponibilità a pagare per un'aria più pulita. Un modello di prezzi edonici viene impiegato per calcolare i danni marginali dell'inquinamento, che aumentano con il livello di inquinamento e il reddito delle famiglie.

3.3 Esperimenti

I codici per eseguire gli esperimenti e i dataset sono disponibili al seguente url (Google Drive):
https://t.ly/dPZAA

L'analisi dei dati e la ricerca dei migliori iperparametri negli algoritmi è stata basata sul codice utilizzato in [31] disponibile su Github [6].

3.3.1 Procedimento

Lo scopo degli esperimenti è stato quello di valutare se le metriche migliorassero o meno applicando le linee guida (semplici e/o complesse) ad un dataset di classificazione e/o regressione.
I risultati calcolati e stampati sono stati raccolti in tabelle (successivamente convertite in grafici e poi in PDF) aventi come colonne, in ordine (per la classificazione):

1. **Algorithm**
2. **Best Params**
3. **Accuracy**
4. **Recall**
5. **Precision**
6. **F1 Score**
7. **Log Loss**
8. **Time Elapsed (s)**

E come righe, in ordine, gli algoritmi utilizzati:

1. **Random Forest**
2. **SVM**
3. **KNN**
4. **ANN**

Iterando il calcolo di questi risultati per tutte le procedure, in ordine:

1. **3_fold_cross_validation**
2. **Grid Search**
3. **Random Search**
4. **Hyperband**
5. **Bayesian Optimization**
6. **Sequential Model-Based Optimization (skopt)**

[6]https://github.com/LiYangHart/Hyperparameter-Optimization-of-Machine-Learning-Algorithms/tree/master

7. **Bayesian Optimization (scikit-optimize)**
8. **Particle Swarm Optimization**
9. **Genetic Algorithm**
10. **TPOT**

Per la regressione invece, per colonne

1. **Algorithm**
2. **Best Params**
3. **MSE**
4. **R^2**
5. **Time Elapsed (s)**

E tutto il resto identico alle tabelle per la classificazione.

3.4 Risultati non applicando le linee guida

Di seguito i risultati dall'analisi dei dataset e la costruzione del modello di regressione e classificazione per ogni dataset.
In grassetto sono segnati le metriche migliori ottenute dai vari algoritmi.
Tutti i codici per ottenere i risultati (i file Notebook Jupyter) sono disponibili su github al seguente link: https://github.com/matteotoma98/Tesi_magistrale

3.4.1 Risultati per la classificazione

Di seguito i risultati ottenuti SENZA applicare le linee guida viste in letteratura, con alcuni (per semplicità e migliore visibilità) relativi migliori iperparametri trovati per ogni procedura.
Il simbolo "-" nella tabella degli iperparametri è presente quando non è disponibile l'iperparametro stampato relativo alla procedura, questo succede perché non sempre le librerie forniscono in output la stampa di tutti i migliori iperparametri trovati).

Dataset per la classificazione
Dataset Blood Transfusion Center (Qui c'è stata solo una Standardizzazione delle feature)

Tabella 3.3: 3-Fold Cross-Validation sul dataset Blood Transfusion Center

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.758483	0.618617	0.653763	0.628966	1.035335	1.380000
SVM	0.762475	0.537127	0.569119	0.513912	0.517616	0.170000
KNN	**0.780439**	**0.624377**	**0.693996**	**0.639038**	2.155069	**0.110000**
ANN	0.764471	0.504167	0.548668	0.441236	**0.486888**	14.810000

Tabella 3.4: Grid Search sul dataset Blood Transfusion Center

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.758483	**0.624315**	0.653850	0.633114	1.314551	9.930000
SVM	0.786427	0.607972	0.719376	0.620937	0.497665	7.080000
KNN	**0.792415**	0.617852	0.726776	**0.634659**	0.594682	**0.870000**
ANN	**0.792415**	0.609075	**0.730237**	0.623336	**0.480343**	400.830000

Tabella 3.5: Random Search sul dataset Blood Transfusion Center

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.782435	0.631439	0.689179	0.644890	0.488485	12.290000
SVM	0.786427	0.607865	0.717887	0.619423	0.500247	5.920000
KNN	**0.796407**	**0.632011**	0.729887	**0.650899**	0.733426	**1.500000**
ANN	0.792415	0.606211	**0.735973**	0.620036	**0.480180**	158.220000

Tabella 3.6: Hyperband sul dataset Blood Transfusion Center

Algorithm	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**Log Loss**	**Time Elapsed (s)**
Random Forest	0.797571	0.797571	0.781160	0.783976	0.490330	18.300000
SVM	0.805668	0.805668	0.789837	0.790746	0.487650	12.000000
KNN	0.809717	0.809717	0.796208	0.783062	0.983722	**2.370000**
ANN	**0.813765**	**0.813765**	**0.799575**	**0.793472**	**0.463798**	242.820000

Tabella 3.7: Bayesian Optimization sul dataset Blood Transfusion Center

Algorithm	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**Log Loss**	**Time Elapsed (s)**
Random Forest	0.786427	**0.634170**	0.700770	0.650436	0.494689	34.640000
SVM	0.786427	0.607865	0.717887	0.619423	0.501588	21.650000
KNN	**0.796407**	0.632011	0.729887	**0.650899**	0.733426	**2.800000**
ANN	0.792415	0.603250	**0.737022**	0.615725	**0.477911**	171.480000

Tabella 3.8: Sequential Model-Based Optimization (skopt) sul dataset Blood Transfusion Center

Algorithm	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**Log Loss**	**Time Elapsed (s)**
Random Forest	0.790419	0.621910	0.697067	0.651128	0.498074	47.930000
SVM	0.786427	0.610837	0.712968	0.623947	0.503245	34.370000
KNN	0.784431	0.601196	0.708523	0.613956	0.592475	**4.030000**
ANN	**0.800399**	**0.646053**	**0.739150**	**0.666350**	**0.482264**	36.050000

Tabella 3.9: Bayesian Optimization TPE sul dataset Blood Transfusion Center

Algorithm	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**Log Loss**	**Time Elapsed (s)**
Random Forest	0.772455	**0.336111**	0.502517	**0.427368**	7.626003	35.160000
SVM	0.786427	0.276709	0.618103	0.377017	7.697946	47.000000
KNN	**0.792415**	0.285470	**0.643275**	0.394643	**7.482116**	**5.290000**
ANN	0.792415	0.201496	0.641636	0.328351	**7.482116**	548.360000

Tabella 3.10: Particle Swarm Optimization sul dataset Blood Transfusion Center

Algorithm	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**Log Loss**	**Time Elapsed (s)**
Random Forest	0.778443	0.778443	0.752038	0.752594	0.495598	15.980000
SVM	0.786427	0.786427	0.762655	0.753709	0.501094	32.760000
KNN	**0.792415**	**0.792415**	**0.770561**	**0.760712**	0.594682	**3.270000**
ANN	0.786427	0.786427	0.756289	0.756043	**0.482398**	17.590000

Tabella 3.11: Genetic Algorithm sul dataset Blood Transfusion Center

Algorithm	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**Log Loss**	**Time Elapsed (s)**
Random Forest	0.780439	**0.856287**	**0.854469**	**0.841590**	**0.337424**	5.060000
SVM	0.786427	0.822355	0.814640	0.797146	0.458900	9.550000
KNN	0.796407	0.804391	0.786957	0.779778	0.417459	**0.500000**
ANN	**0.804391**	0.810379	0.795210	0.787104	0.442071	221.480000

Tabella 3.12: TPOT sul dataset Blood Transfusion Center

Algorithm	**Accuracy**	**Recall**	**Precision**	**F1 Score**	**Log Loss**	**Time Elapsed (s)**
Random Forest	0.780439	0.780439	0.756714	0.760170	0.493051	28.360000
SVM	0.788423	0.788423	0.767700	0.755209	0.500732	13.270000
KNN	**0.796407**	**0.796407**	**0.775504**	**0.769250**	0.733426	**1.410000**
ANN	0.790419	0.790419	0.769529	0.755011	**0.470503**	73.190000

Migliori Iperparametri trovati

Tabella 3.13: Migliori iperparametri per Grid Search sul dataset Blood Transfusion Center

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N Neighbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	15	30	-	-	-	-	-	-	-
SVM	-	-	-	100	linear	-	-	-	-	-
KNN	-	-	-	-	-	5	-	-	-	-
ANN	-	-	-	-	-	-	tanh	16	50	rmsprop

Tabella 3.14: Migliori iperparametri per Random Search sul dataset Blood Transfusion Center

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	38	22	-	-	-	-	-	-	-
SVM	-	-	-	45.64	linear	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	tanh	16	50	sgd

Tabella 3.15: Migliori iperparametri per Hyperband sul dataset Blood Transfusion Center

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	entropy	13	11	-	-	-	-	-	-	-
SVM	-	-	-	4.94	poly	-	-	-	-	-
KNN	-	-	-	-	-	1	-	-	-	-
ANN	-	-	-	-	-	-	relu	64	50	rmsprop

Tabella 3.16: Migliori iperparametri per Bayesian Optimization sul dataset Blood Transfusion Center

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	entropy	14	92	-	-	-	-	-	-	-
SVM	-	-	-	45.62	linear	-	-	-	-	-
KNN	-	-	-	-	-	13	-	-	-	-
ANN	-	-	-	-	-	-	relu	16	50	rmsprop

Tabella 3.17: Migliori iperparametri per Sequential Model-Based Optimization (skopt) sul dataset Blood Transfusion Center

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	14	47	-	-	-	-	-	-	-
SVM	-	-	-	13.64	poly	-	-	-	-	-
KNN	-	-	-	-	-	2	-	-	-	-
ANN	-	-	-	-	-	-	relu	45	50	lbfgs

Tabella 3.18: Migliori iperparametri per Bayesian Optimization TPE sul dataset Blood Transfusion Center

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N Neighbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	entropy	48	31	-	-	-	-	-	-	-
SVM	-	-	-	16.88	rbf	-	-	-	-	-
KNN	-	-	-	-	-	15	-	-	-	-
ANN	-	-	-	-	-	-	tanh	48	20	rmsprop

Tabella 3.19: Migliori iperparametri per Particle Swarm Optimization sul dataset Blood Transfusion Center

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	43	42	-	-	-	-	-	-	-
SVM	-	-	-	0.64	linear	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	relu	18	50	adamax

Tabella 3.20: Migliori iperparametri per Genetic Algorithm sul dataset Blood Transfusion Center

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N Neighbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	10	58	-	-	-	-	-	-	-
SVM	-	-	-	12.02	rbf	-	-	-	-	-
KNN	-	-	-	-	-	12	-	-	-	-
ANN	-	-	-	-	-	-	relu	32	15	adam

Tabella 3.21: Migliori iperparametri per TPOT sul dataset Blood Transfusion Center

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N Neighbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	12	70	-	-	-	-	-	-	-
SVM	-	-	-	12.15	rbf	-	-	-	-	-
KNN	-	-	-	-	-	18	-	-	-	-
ANN	-	-	-	-	-	-	tanh	32	50	adam

Breast Cancer Wisconsin

Tabella 3.22: 3_fold_cross_validation sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.953159	0.947936	0.951939	0.949805	0.122428	2.220000
SVM	**0.964866**	0.958358	0.966731	0.962187	**0.091562**	0.360000
KNN	0.964855	0.956250	**0.969079**	0.961978	0.295881	**0.150000**
ANN	0.964866	**0.960436**	0.964464	**0.962368**	0.096317	5.880000

Tabella 3.23: Grid Search sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.951210	0.942222	0.953939	0.947377	0.191235	7.390000
SVM	**0.972687**	**0.967713**	**0.974108**	**0.970634**	**0.099993**	1.760000
KNN	0.964855	0.956250	0.969079	0.961978	0.295881	**0.460000**
ANN	0.962929	0.959925	0.960945	0.960421	0.121945	375.570000

Tabella 3.24: Random Search sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.955108	0.950540	0.953462	0.951935	0.202927	10.900000
SVM	**0.968788**	**0.963551**	**0.969743**	**0.966404**	**0.094221**	3.800000
KNN	0.960956	0.953135	0.963515	0.957864	0.616376	**0.750000**
ANN	0.962883	0.954692	0.966388	0.959921	0.120760	154.360000

Tabella 3.25: Hyperband sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	**1.000000**	**1.000000**	**1.000000**	**1.000000**	0.133409	17.050000
SVM	1.000000	1.000000	1.000000	1.000000	0.043708	2.580000
KNN	1.000000	1.000000	1.000000	1.000000	**0.000000**	**1.420000**
ANN	0.982456	0.982456	0.983292	0.982550	0.178981	227.640000

Tabella 3.26: Bayesian Optimization sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.957069	0.952098	0.956010	0.953949	0.188816	32.930000
SVM	**0.968788**	**0.963551**	**0.969743**	**0.966404**	**0.091701**	15.320000
KNN	0.960945	0.950010	0.967520	0.957465	0.184484	**1.740000**
ANN	0.951175	0.945332	0.950435	0.947653	0.135625	140.190000

Tabella 3.27: Sequential Model-Based Optimization (skopt) sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.945350	0.916667	0.931184	0.921693	0.195295	69.190000
SVM	**0.960991**	**0.937500**	0.957966	**0.947453**	**0.162898**	11.920000
KNN	0.945304	0.859375	**0.994444**	0.921140	0.752590	**1.680000**
ANN	0.960945	0.937500	0.958059	0.947410	0.952150	40.440000

Tabella 3.28: Bayesian Optimization TPE sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.931682	0.890625	0.936975	0.913994	0.257292	72.400000
SVM	**0.960991**	**0.937500**	0.958586	**0.947563**	0.170785	8.310000
KNN	0.945304	0.859375	**0.994444**	0.921140	0.752590	**3.610000**
ANN	0.902339	0.791667	0.928619	0.856510	**0.153530**	78.890000

Tabella 3.29: Particle Swarm Optimization sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.949249	0.949249	0.949194	0.949082	0.248510	32.920000
SVM	**0.968788**	**0.968788**	**0.968936**	**0.968649**	**0.090571**	4.460000
KNN	0.960956	0.960956	0.961294	0.960722	0.616376	**1.380000**
ANN	0.959007	0.959007	0.959283	0.959041	0.263088	25.090000

Tabella 3.30: Genetic Algorithm sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.960938	0.982422	0.982528	0.982374	0.087784	7.000000
SVM	0.968750	0.974609	0.975270	0.974449	0.081104	0.720000
KNN	0.960938	0.984375	0.984756	0.984305	0.045675	**0.370000**
ANN	**0.974609**	**0.990234**	**0.990385**	**0.990208**	**0.035977**	363.460000

Tabella 3.31: TPOT sul dataset Breast Cancer

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.959007	0.959007	0.959040	0.958894	0.136144	63.580000
SVM	**0.970703**	**0.970703**	**0.971088**	**0.970547**	**0.088797**	3.210000
KNN	0.964855	0.964855	0.965534	0.964595	0.295881	**0.660000**
ANN	0.966804	0.966804	0.967078	0.966734	0.120356	53.700000

Migliori Iperparametri trovati

Tabella 3.32: Migliori iperparametri per Grid Search sul dataset Breast Cancer

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N Neighbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	15	30	-	-	-	-	-	-	-
SVM	-	-	-	100	linear	-	-	-	-	-
KNN	-	-	-	-	-	5	-	-	-	-
ANN	-	-	-	-	-	-	tanh	16	50	rmsprop

Tabella 3.33: Migliori iperparametri per Random Search sul dataset Breast Cancer

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	38	22	-	-	-	-	-	-	-
SVM	-	-	-	45.64	linear	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	tanh	16	50	sgd

Tabella 3.34: Migliori iperparametri per Hyperband sul dataset Breast Cancer

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	entropy	13	11	-	-	-	-	-	-	-
SVM	-	-	-	4.94	poly	-	-	-	-	-
KNN	-	-	-	-	-	1	-	-	-	-
ANN	-	-	-	-	-	-	relu	64	50	rmsprop

Tabella 3.35: Migliori iperparametri per Bayesian Optimization sul dataset Breast Cancer

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	entropy	14	92	-	-	-	-	-	-	-
SVM	-	-	-	45.62	linear	-	-	-	-	-
KNN	-	-	-	-	-	13	-	-	-	-
ANN	-	-	-	-	-	-	relu	16	50	rmsprop

Tabella 3.36: Migliori iperparametri per Sequential Model-Based Optimization (skopt) sul dataset Breast Cancer

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	14	47	-	-	-	-	-	-	-
SVM	-	-	-	13.64	poly	-	-	-	-	-
KNN	-	-	-	-	-	2	-	-	-	-
ANN	-	-	-	-	-	-	relu	45	50	lbfgs

Tabella 3.37: Migliori iperparametri per Bayesian Optimization TPE sul dataset Breast Cancer

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	39	52	13	-	-	-	-	-	-
SVM	-	-	-	11.063	poly	-	-	-	-	-
KNN	-	-	-	-	-	2	-	-	-	-
ANN	-	-	-	-	-	-	relu	48	20	adam

Tabella 3.38: Migliori iperparametri per Particle Swarm Optimization sul dataset Breast Cancer

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	43	42	-	-	-	-	-	-	-
SVM	-	-	-	0.64	linear	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	relu	18	50	adamax

Tabella 3.39: Migliori iperparametri per Genetic Algorithm sul dataset Breast Cancer

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	entropy	25	36	-	-	-	-	-	-	-
SVM	-	-	-	12.34	rbf	-	-	-	-	-
KNN	-	-	-	-	-	4	-	-	-	-
ANN	-	-	-	-	-	-	relu	20	100	adam

Tabella 3.40: Migliori iperparametri per TPOT sul dataset Breast Cancer

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	18	48	-	-	-	-	-	-	-
SVM	-	-	-	2.35	sigmoid	-	-	-	-	-
KNN	-	-	-	-	-	7	-	-	-	-
ANN	-	-	-	-	-	-	sigmoid	32	30	nadam

Digits MNIST

Tabella 3.41: 3-fold Cross Validation sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.944352	0.944223	0.945732	0.944265	0.427887	2.620000
SVM	0.969950	0.969786	0.970958	0.969916	0.161959	1.760000
KNN	0.962716	0.962494	0.963858	0.962621	0.466567	**0.230000**
ANN	**0.989427**	**0.989337**	**0.989639**	**0.989393**	**0.050931**	14.060000

Tabella 3.42: Grid Search sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.936004	0.935917	0.937570	0.935648	0.466019	9.910000
SVM	**0.973845**	**0.973780**	**0.974703**	**0.973846**	**0.159104**	12.830000
KNN	0.968280	0.968129	0.969136	0.968234	0.586421	**1.250000**
ANN	0.951586	0.951580	0.952496	0.951526	0.166835	639.160000

Tabella 3.43: Random Search sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.929883	0.929883	0.931561	0.929670	0.544428	31.330000
SVM	0.973845	0.973845	0.974740	0.973896	0.159714	20.120000
KNN	0.968280	0.968280	0.969148	0.968315	0.586421	**2.660000**
ANN	**0.991096**	**0.991096**	**0.991245**	**0.991085**	**0.035876**	283.920000

Tabella 3.44: Hyperband sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.983333	0.983333	0.983437	0.983318	0.332234	43.530000
SVM	0.986111	0.986111	0.986196	0.986075	**0.088163**	24.060000
KNN	**0.988889**	**0.988889**	**0.988954**	**0.988850**	0.127179	**2.170000**
ANN	0.980556	0.980556	0.980787	0.980590	0.109046	297.790000

Tabella 3.45: Bayesian Optimization sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.931553	0.931507	0.933245	0.931364	0.417181	46.860000
SVM	0.974402	0.974336	0.975269	0.974416	0.159219	40.030000
KNN	0.963829	0.963550	0.965475	0.963799	0.500377	**2.650000**
ANN	**0.994435**	**0.994377**	**0.994586**	**0.994407**	**0.033928**	249.790000

Tabella 3.46: Sequential Model-Based Optimization (skopt) sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.918197	0.916559	0.915269	0.913100	0.429773	135.720000
SVM	**0.973845**	**0.973780**	**0.974703**	**0.973846**	**0.159030**	72.200000
KNN	0.953812	0.953444	0.956136	0.953421	0.227953	**2.920000**
ANN	0.927101	0.926870	0.933268	0.927499	0.232204	166.550000

Tabella 3.47: Bayesian Optimization TPE sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.903728	0.906473	0.908028	0.902424	0.422909	119.200000
SVM	**0.973845**	**0.973780**	**0.974703**	**0.973846**	**0.160284**	69.100000
KNN	0.953812	0.953444	0.956136	0.953421	0.227953	**2.920000**
ANN	0.928978	0.928901	0.933289	0.928935	0.229861	230.170000

Tabella 3.48: Particle Swarm Optimization sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.931553	0.931553	0.933634	0.931492	0.498710	52.540000
SVM	**0.973845**	**0.973845**	**0.974740**	**0.973896**	**0.158170**	41.960000
KNN	0.968280	0.968280	0.969148	0.968315	0.586421	**3.170000**
ANN	0.929883	0.929883	0.933255	0.930162	0.532839	61.470000

Tabella 3.49: Genetic Algorithm sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.941013	**1.000000**	**1.000000**	**1.000000**	0.089524	27.100000
SVM	0.973845	1.000000	1.000000	1.000000	0.023431	14.500000
KNN	0.968280	0.993322	0.993356	0.993309	0.012752	**1.800000**
ANN	**1.000000**	1.000000	1.000000	1.000000	**0.000024**	480.060000

Tabella 3.50: TPOT sul dataset Digits

Algorithm	Accuracy	Recall	Precision	F1 Score	Log Loss	Time Elapsed (s)
Random Forest	0.927101	0.927101	0.928871	0.927035	0.342839	54.240000
SVM	**0.973845**	**0.973845**	**0.974740**	**0.973896**	**0.159606**	22.790000
KNN	0.968280	0.968280	0.969148	0.968315	0.586421	**3.210000**
ANN	0.952699	0.952699	0.953959	0.952668	0.260716	88.690000

Migliori Iperparametri trovati

Tabella 3.51: Migliori iperparametri per Grid Search sul dataset Digits

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	15	30	-	-	-	-	-	-	-
SVM	-	-	-	10	rbf	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	relu	32	20	sgd

Tabella 3.52: Migliori iperparametri per Random Search sul dataset Digits

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	32	88	-	-	-	-	-	-	-
SVM	-	-	-	17.11	rbf	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	relu	64	20	rmsprop

Tabella 3.53: Migliori iperparametri per Hyperband sul dataset Digits

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	entropy	39	38	-	-	-	-	-	-	-
SVM	-	-	-	2.24	poly	-	-	-	-	-
KNN	-	-	-	-	-	7	-	-	-	-
ANN	-	-	-	-	-	-	relu	46	45	sgd

Tabella 3.54: Migliori iperparametri per Bayesian Optimization sul dataset Digits

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	31	64	-	-	-	-	-	-	-
SVM	-	-	-	7.73	rbf	-	-	-	-	-
KNN	-	-	-	-	-	4	-	-	-	-
ANN	-	-	-	-	-	-	relu	32	20	rmsprop

Tabella 3.55: Migliori iperparametri per Sequential Model-Based Optimization (skopt) sul dataset Digits

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	31	75	-	-	-	-	-	-	-
SVM	-	-	-	25.30	rbf	-	-	-	-	-
KNN	-	-	-	-	-	12	-	-	-	-
ANN	-	-	-	-	-	-	relu	16	50	sgd

Tabella 3.56: Migliori iperparametri per Bayesian Optimization TPE sul dataset Digits

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	42.0	75.0	-	-	-	-	-	-	-
SVM	-	-	-	21.39	rbf	-	-	-	-	-
KNN	-	-	-	-	-	12.0	-	-	-	-
ANN	-	-	-	-	-	-	relu	64.0	20.0	adam

Tabella 3.57: Migliori iperparametri per Particle Swarm Optimization sul dataset Digits

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	entropy	27	50	-	-	-	-	-	-	-
SVM	-	-	-	15.40	linear	-	-	-	-	-
KNN	-	-	-	-	-	5	-	-	-	-
ANN	-	-	-	-	-	-	sigmoid	32	30	adam

Tabella 3.58: Migliori iperparametri per Genetic Algorithm sul dataset Digits

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	35	45	-	-	-	-	-	-	-
SVM	-	-	-	12.35	sigmoid	-	-	-	-	-
KNN	-	-	-	-	-	7	-	-	-	-
ANN	-	-	-	-	-	-	tanh	64	25	adam

Tabella 3.59: Migliori iperparametri per TPOT sul dataset Digits

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	20	100	-	-	-	-	-	-	-
SVM	-	-	-	30.25	rbf	-	-	-	-	-
KNN	-	-	-	-	-	4	-	-	-	-
ANN	-	-	-	-	-	-	relu	32	50	sgd

3.4.2 Risultati per la regressione

Dataset per la Regressione
Di seguito i risultati ottenuti SENZA applicare le linee guida viste in letteratura.
Life Expectancy

Tabella 3.60: 3-Fold Cross-Validation sul dataset Life Expectancy

Algorithm	MSE	R^2	Time Elapsed (s)
Random Forest	**4.212946**	**0.952961**	25.75
SVM	15.179601	0.829707	2.11
KNN	11.728416	0.868422	**0.11**
ANN	12.667394	0.843278	78.42

Tabella 3.61: Grid Search sul dataset Life Expectancy

Algorithm	MSE	R^2	Time Elapsed (s)
Random Forest	**4.392258**	**0.993335**	17.59
SVM	7.267193	0.944014	67.44
KNN	11.30248	0.95215	**0.29**
ANN	10.834532	0.87619	2294.43

Tabella 3.62: Random Search sul dataset Life Expectancy

Algorithm	MSE	R^2	Time Elapsed (s)
Random Forest	**4.307486**	**0.990603**	85.41
SVM	7.994456	0.933478	23.36
KNN	11.495462	0.939574	**0.58**
ANN	10.023566	0.850261	397.16

Tabella 3.63: Hyperband sul dataset Life Expectancy

Algorithm	MSE	R^2	Time Elapsed (s)
Random Forest	**3.880105**	**0.993164**	408.27
SVM	14.814833	0.847687	126.94
KNN	10.809046	0.969299	**15.56**
ANN	8.259958	0.930822	376.0

Tabella 3.64: Bayesian Optimization sul dataset Life Expectancy

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**3.982204**	**0.994413**	73.68
SVM	7.952088	0.934374	35.75
KNN	11.30248	0.95215	**1.59**
ANN	9.308879	0.923604	476.6

Tabella 3.65: Sequential Model-Based Optimization (skopt) sul dataset Life Expectancy

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**4.024313**	**0.994797**	42.72
SVM	8.09356	0.931229	20.79
KNN	11.881784	0.968989	**1.3**
ANN	3051.408357	0.863182	109.88

Tabella 3.66: Bayesian Optimization TPE sul dataset Life Expectancy

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**4.095693**	**0.993966**	30.75
SVM	7.545369	0.938334	53.83
KNN	11.30248	0.95215	**1.6**
ANN	12.780012	0.88027	524.56

Tabella 3.67: Particle Swarm Optimization sul dataset Life Expectancy

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**4.163161**	**0.99159**	98.1
SVM	7.944236	0.934496	63.97
KNN	11.495462	0.939574	**1.43**
ANN	11.94535	0.891484	974.0

Tabella 3.68: Genetic Algorithm sul dataset Life Expectancy sul dataset Digits

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**4.439632**	**0.991399**	33.03
SVM	8.363296	0.931973	15.96
KNN	12.058614	0.929111	**0.94**
ANN	9.067398	0.891323	1035.61

Tabella 3.69: TPOT sul dataset Life Expectancy

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.845652**	**0.990512**	72.45
SVM	5.82396	0.934659	34.27
KNN	4.26501	0.95215	**1.26**
ANN	7.047518	0.922305	137.15

Migliori Iperparametri trovati

Tabella 3.70: Migliori iperparametri per Grid Search sul dataset Life Expectancy

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	-	30	30	-	-	-	-	-	-	-
SVM	-	-	-	100	rbf	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	relu	16	50	adam

Tabella 3.71: Migliori iperparametri per Random Search sul dataset Life Expectancy

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman_mse	41	32	-	-	-	-	-	-	-
SVM	-	-	-	49.68	poly	-	-	-	-	-
KNN	-	-	-	-	-	4	-	-	-	-
ANN	-	-	-	-	-	-	relu	16	50	rmsprop

Tabella 3.72: Migliori iperparametri per Hyperband sul dataset Life Expectancy

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman_mse	20	- 98 -	-	-	-	-	-	-	
SVM	-	-	-	0.95	rbf	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	tanh	32	100	adam

Tabella 3.73: Migliori iperparametri per Bayesian Optimization sul dataset Life Expectancy

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman_mse	18	75	-	-	-	-	-	-	-
SVM	-	-	-	50	rbf	-	-	-	-	-
KNN	-	-	-	-	-	5	-	-	-	-
ANN	-	-	-	-	-	-	tanh	16	50	rmsprop

Tabella 3.74: Migliori iperparametri per Sequential Model-Based Optimization (skopt) sul dataset Life Expectancy

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman_mse	27	100	-	-	-	-	-	-	-
SVM	-	-	-	42.52	rbf	-	-	-	-	-
KNN	-	-	-	-	-	2	-	-	-	-
ANN	-	-	-	-	-	-	tanh	-	-	rmsprop

Tabella 3.75: Migliori iperparametri per Bayesian Optimization TPE sul dataset Life Expectancy

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman_mse	19	67	-	-	-	-	-	-	-
SVM	-	-	-	56.1	rbf	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	relu	48	50	adam

Tabella 3.76: Migliori iperparametri per Particle Swarm Optimization sul dataset Life Expectancy

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	-	20	50	-	-	-	-	-	-	-
SVM	-	-	-	5.32	rbf	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	relu	32	50	adam

Tabella 3.77: Migliori iperparametri per Genetic Algorithm sul dataset Life Expectancy

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	gini	25	100	-	-	-	-	-	-	-
SVM	-	-	-	2.56	linear	-	-	-	-	-
KNN	-	-	-	-	-	5	-	-	-	-
ANN	-	-	-	-	-	-	tanh	32	100	adam

Tabella 3.78: Migliori iperparametri per TPOT sul dataset Life Expectancy

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	squared_error	93	69	-	-	-	-	-	-	-
SVM	-	-	-	42.21	poly	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	logistic	auto	200	lbfgs

Obesity

Tabella 3.79: 3-Fold Cross-Validation sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.400459**	**0.898819**	5.31
SVM	3.138676	0.17997	1.39
KNN	1.135786	0.703269	**0.2**
ANN	1.817694	0.436194	88.74

Tabella 3.80: Grid Search sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.391839**	**0.987898**	3.6
SVM	1.946485	0.514228	14.13
KNN	0.991488	0.930377	**0.18**
ANN	1.142892	0.755335	1811.11

Tabella 3.81: Random Search sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.470239**	**0.955741**	28.5
SVM	1.963755	0.506758	10.45
KNN	0.991488	0.930377	**0.59**
ANN	0.966399	0.829063	414.86

Tabella 3.82: Hyperband sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.374629**	**0.976352**	149.44
SVM	3.074266	0.229334	77.68
KNN	1.015013	0.879892	**10.78**
ANN	1.96925	0.50163	298.17

Tabella 3.83: Bayesian Optimization sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.391096**	**0.98846**	44.47
SVM	1.963599	0.504652	28.29
KNN	0.991488	0.930377	**0.82**
ANN	1.069398	0.768698	338.5

Tabella 3.84: Sequential Model-Based Optimization (skopt) sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.37492**	**0.989077**	25.83
SVM	1.959369	0.506514	13.82
KNN	0.991488	0.930377	**0.79**
ANN	3.017177	0.762038	132.27

Tabella 3.85: Bayesian Optimization TPE sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.527393**	**0.941208**	14.18
SVM	1.949986	0.513478	17.33
KNN	0.991488	0.930377	**1.73**
ANN	1.620988	0.686322	414.1

Tabella 3.86: Particle Swarm Optimization sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.393669**	**0.967696**	14.88
SVM	1.986564	0.498604	520.4
KNN	0.991488	0.930377	**0.73**
ANN	1.66027	0.658063	386.66

Tabella 3.87: Genetic Algorithm sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.465344**	**0.961209**	18.26
SVM	1.988604	0.506503	10.66
KNN	1.125685	0.83579	**0.58**
ANN	1.03486	0.835935	434.33

Tabella 3.88: TPOT sul dataset Obesity

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**0.107448**	**0.971988**	65.7
SVM	1.890643	0.5071	10.63
KNN	0.267056	0.930377	**1.49**
ANN	1.142584	0.690249	151.12

Migliori Iperparametri trovati

Tabella 3.89: Migliori iperparametri per Grid Search sul dataset Obesity

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	-	20	30	-	-	-	-	-	-	-
SVM	-	-	-	100	rbf	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	tanh	16	50	rmsprop

Tabella 3.90: Migliori iperparametri per Random Search sul dataset Obesity

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	poisson	37	27	-	-	-	-	-	-	-
SVM	-	-	-	33.38	rbf	-	-	-	-	-
KNN	-	-	-	-	-	4	-	-	-	-
ANN	-	-	-	-	-	-	tanh	16	50	adam

Tabella 3.91: Migliori iperparametri per Hyperband sul dataset Obesity

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman	35	68	-	-	-	-	-	-	-
SVM	-	-	-	0.95	rbf	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	tanh	64	100	adam

Tabella 3.92: Migliori iperparametri per Bayesian Optimization sul dataset Obesity

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	squared	42	100	-	-	-	-	-	-	-
SVM	-	-	-	34.81	rbf	-	-	-	-	-
KNN	-	-	-	-	-	4	-	-	-	-
ANN	-	-	-	-	-	-	tanh	64	50	rmsprop

Tabella 3.93: Migliori iperparametri per Sequential Model-Based Optimization (skopt) sul dataset Obesity

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman	46	100	-	-	-	-	-	-	-
SVM	-	-	-	42.52	rbf	-	-	-	-	-
KNN	-	-	-	-	-	2	-	-	-	-
ANN	-	-	-	-	-	-	tanh	-	-	adam

Tabella 3.94: Migliori iperparametri per Bayesian Optimization TPE sul dataset Obesity

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman	32	59	-	-	-	-	-	-	-
SVM	-	-	-	-26.82	rbf	-	-	-	-	-
KNN	-	-	-	-	-	4	-	-	-	-
ANN	-	-	-	-	-	-	tanh	16	30	adam

Tabella 3.95: Migliori iperparametri per Particle Swarm Optimization sul dataset Obesity

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	-	25	29	-	-	-	-	-	-	-
SVM	-	-	-	47.41	2.50	-	-	-	-	-
KNN	-	-	-	-	-	3	-	-	-	-
ANN	-	-	-	-	-	-	0.60	0.95	46	0.32

Tabella 3.96: Migliori iperparametri per Genetic Algorithm sul dataset Obesity

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	squared	37	65	-	-	-	-	-	-	-
SVM	-	-	-	43.60	rbf	-	-	-	-	-
KNN	-	-	-	-	-	5	-	-	-	-
ANN	-	-	-	-	-	-	tanh	64	20	rmsprop

Tabella 3.97: Migliori iperparametri per TPOT sul dataset Obesity

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	squared	66	141	-	-	-	-	-	-	-
SVM	-	-	-	40.63	rbf	-	-	-	-	-
KNN	-	-	-	-	-	2	-	-	-	-
ANN	-	-	-	-	-	-	tanh	auto	200	adam

Boston Housing

Tabella 3.98: 3-Fold Cross-Validation sul dataset Boston Housing

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**11.834**	**0.855**	3.52
SVM	66.493	0.208	0.09
KNN	49.421	0.411	**0.03**
ANN	45.396	0.508	60.38

Tabella 3.99: Grid Search sul dataset Boston Housing

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**12.084**	**0.976**	2.37
SVM	45.283	0.5	1.44
KNN	47.677	0.602	**0.07**
ANN	33.056	0.594	916.31

Tabella 3.100: Random Search sul dataset Boston Housing

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**12.207**	**0.981**	9.83
SVM	50.07	0.439	1.09
KNN	47.549	0.563	**0.18**
ANN	32.677	0.667	190.79

Tabella 3.101: Hyperband sul dataset Boston Housing

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**10.472**	**0.98**	102.11
SVM	76.349	0.156	4.62
KNN	45.661	0.752	**0.99**
ANN	34.743	0.659	103.85

Tabella 3.102: Bayesian Optimization sul dataset Boston Housing

Algorithm	**MSE**	**R²**	**Time Elapsed (s)**
Random Forest	**14.243**	**0.908**	36.77
SVM	49.918	0.439	22.66
KNN	46.868	0.583	**2.12**
ANN	36.003	0.576	201.08

Tabella 3.103: Sequential Model-Based Optimization (skopt) sul dataset Boston Housing

Algorithm	**MSE**	**R²**	**Time Elapsed (s)**
Random Forest	**11.672**	**0.982**	18.53
SVM	49.832	0.439	6.93
KNN	46.868	0.583	**1.11**
ANN	105.264	0.517	98.19

Tabella 3.104: Bayesian Optimization TPE sul dataset Boston Housing

Algorithm	**MSE**	**R²**	**Time Elapsed (s)**
Random Forest	**12.477**	**0.964**	10.38
SVM	45.455	0.507	1.35
KNN	47.549	0.563	**0.84**
ANN	42.377	0.593	257.15

Tabella 3.105: Particle Swarm Optimization sul dataset Boston Housing

Algorithm	**MSE**	**R²**	**Time Elapsed (s)**
Random Forest	**13.552**	**0.951**	21.18
SVM	50.596	0.43	2.42
KNN	47.677	0.602	**0.77**
ANN	35.97	0.632	518.12

Tabella 3.106: Genetic Algorithm sul dataset Boston Housing

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**14.232**	**0.946**	11.87
SVM	57.692	0.39	1.33
KNN	47.89	0.82	**0.12**
ANN	30.314	0.747	426.14

Tabella 3.107: TPOT sul dataset Boston Housing

Algorithm	**MSE**	**R^2**	**Time Elapsed (s)**
Random Forest	**3.415**	**0.959**	28.14
SVM	47.514	0.433	1.5
KNN	34.819	0.584	**0.69**
ANN	26.007	0.641	55.55

Migliori Iperparametri trovati

Tabella 3.108: Migliori iperparametri per Grid Search sul dataset Boston Housing

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	-	30	10	-	-	-	-	-	-	-
SVM	-	-	-	100	poly	-	-	-	-	-
KNN	-	-	-	-	-	5	-	-	-	-
ANN	-	-	-	-	-	-	relu	32	20	adam

Tabella 3.109: Migliori iperparametri per Random Search sul dataset Boston Housing

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	squared	38	86	-	-	-	-	-	-	-
SVM	-	-	-	28.82	poly	-	-	-	-	-
KNN	-	-	-	-	-	13	-	-	-	-
ANN	-	-	-	-	-	-	relu	32	50	rmsprop

Tabella 3.110: Migliori iperparametri per Hyperband sul dataset Boston Housing

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman_mse	49	77	-	-	-	-	-	-	-
SVM	-	-	-	0.95	rbf	-	-	-	-	-
KNN	-	-	-	-	-	10	-	-	-	-
ANN	-	-	-	-	-	-	relu	16	100	adam

Tabella 3.111: Migliori iperparametri per Bayesian Optimization sul dataset Boston Housing

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman	33	100	-	-	-	-	-	-	-
SVM	-	-	-	32.95	poly	-	-	-	-	-
KNN	-	-	-	-	-	13	-	-	-	-
ANN	-	-	-	-	-	-	relu	32	50	adam

Tabella 3.112: Migliori iperparametri per Sequential Model-Based Optimization (skopt) sul dataset Boston Housing

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman	18	100	-	-	-	-	-	-	-
SVM	-	-	-	44.48	poly	-	-	-	-	-
KNN	-	-	-	-	-	13	-	-	-	-
ANN	-	-	-	-	-	-	tanh	-	-	rmsprop

Tabella 3.113: Migliori iperparametri per Bayesian Optimization TPE sul dataset Boston Housing

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	friedman	28	43	-	-	-	-	-	-	-
SVM	-	-	-	42.37	-	-	-	-	-	-
KNN	-	-	-	-	-	6	-	-	-	-
ANN	-	-	-	-	-	-	-	64	30	-

Tabella 3.114: Migliori iperparametri per Particle Swarm Optimization sul dataset Boston Housing

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	-	41	86	-	-	-	-	-	-	-
SVM	-	-	-	29.17	0.41	-	-	-	-	-
KNN	-	-	-	-	-	15	-	-	-	-
ANN	-	-	-	-	-	-	-	0.64	49	1.95

Tabella 3.115: Migliori iperparametri per Genetic Algorithm sul dataset Boston Housing

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	squared	29	32	-	-	-	-	-	-	-
SVM	-	-	-	43.13	poly	-	-	-	-	-
KNN	-	-	-	-	-	5	-	-	-	-
ANN	-	-	-	-	-	-	tanh	32	20	rmsprop

Tabella 3.116: Migliori iperparametri per TPOT sul dataset Boston Housing

Algorithm	Criterion	Max Depth	N Estimators	C	Kernel	N hbors	Activation	Batch Size	Epochs	Optimizer
Random Forest	squared	39	172	-	-	-	-	-	-	-
SVM	-	-	-	48.46	poly	-	-	-	-	-
KNN	-	-	-	-	-	13	-	-	-	-
ANN	-	-	-	-	-	-	relu	auto	200	lbfgs

3.4.3 Risultati applicando delle linee guida alla classificazione

In grassetto i risultati migliori tra prima e dopo aver applicato il quality check. È stato omesso il tempo trascorso per chiarezza visuale delle tabelle, e inoltre non è fondamentale viste le altre varie metriche da valutare.
La tabella compara i risultati applicando la linea guida **rimozione degli outlier** con i risultati applicando l'aggiuntiva linea guida alla rimozione degli outliers, cioè la **selezione delle feature più significative**.
Dataset Blood Transfusion Center
I nomi delle procedure di ottimizzazione sono stati abbreviati (per chiarire la formattazione della tabella) come segue:

- 3-fold cross-validation → 3-fold
- Grid Search → GS
- Random Search → RS
- Hyperband → HB
- Bayesian Optimization → BO
- Sequential Model-Based Optimization (skopt) → SMBO
- Bayesian Optimization TPE → BO-TPE
- Particle Swarm Optimization → PSO

- Genetic Algorithm → GA
- TPOT

Quality Check applicato: Algoritmo *SMOTE* per bilanciare le classi nel training set

Tabella 3.117: Confronto delle metriche prima e dopo il quality check

Procedura	Algoritmo	Accuracy		Recall		Precision		F1		Log Loss	
		Prima	Dopo	Prima	Dopo	Prima	Dopo	Prima	Dopo	Prima	Dopo
3-fold	RF	0.758	**0.802**	0.619	**0.802**	0.654	**0.811**	0.629	**0.800**	1.035	**0.632**
	SVM	0.762	0.697	0.537	**0.697**	0.569	**0.702**	0.514	**0.695**	0.518	0.577
	KNN	0.780	0.742	0.624	**0.742**	0.694	**0.747**	0.639	**0.741**	2.155	**1.589**
	ANN	0.764	0.706	0.504	**0.706**	0.549	**0.706**	0.441	**0.705**	0.487	0.584
GS	RF	0.758	**0.803**	0.624	**0.803**	0.654	**0.815**	0.633	**0.801**	1.315	**0.959**
	SVM	0.786	0.718	0.608	**0.718**	0.719	**0.721**	0.621	**0.717**	0.498	0.552
	KNN	0.792	0.763	0.618	**0.763**	0.727	**0.767**	0.635	**0.762**	0.595	3.168
	ANN	0.792	0.719	0.609	**0.719**	0.730	0.721	0.623	**0.719**	0.480	0.575
RS	RF	0.782	**0.783**	0.631	**0.783**	0.689	**0.788**	0.645	**0.782**	0.488	0.522
	SVM	0.786	0.718	0.608	**0.718**	0.718	**0.720**	0.619	**0.717**	0.500	0.557
	KNN	0.785	0.760	0.617	**0.760**	0.727	**0.768**	0.634	**0.758**	0.572	2.945
	ANN	0.775	0.725	0.589	**0.725**	0.632	**0.730**	0.615	**0.723**	0.487	0.563
HB	RF	0.792	**0.801**	0.632	**0.801**	0.680	**0.784**	0.655	**0.798**	1.222	**0.592**
	SVM	0.782	0.725	0.574	**0.725**	0.632	**0.730**	0.591	**0.726**	0.477	0.531
	KNN	0.796	0.762	0.615	**0.762**	0.726	**0.770**	0.630	**0.762**	0.579	2.945
	ANN	0.786	0.730	0.568	**0.730**	0.615	**0.724**	0.573	**0.730**	0.466	0.576
BO	RF	0.791	**0.806**	0.626	**0.806**	0.674	**0.793**	0.630	**0.804**	1.079	**0.616**
	SVM	0.778	0.731	0.598	**0.731**	0.634	**0.739**	0.593	**0.730**	0.487	0.561
	KNN	0.794	0.767	0.626	**0.767**	0.732	**0.779**	0.636	**0.762**	0.589	2.895
	ANN	0.788	0.729	0.575	**0.729**	0.620	**0.721**	0.576	**0.726**	0.464	0.564
SMBO	RF	0.792	**0.804**	0.631	**0.804**	0.684	**0.790**	0.648	**0.803**	1.201	**0.609**
	SVM	0.783	0.724	0.585	**0.724**	0.638	**0.738**	0.596	**0.723**	0.482	0.543
	KNN	0.795	0.764	0.616	**0.764**	0.731	**0.769**	0.633	**0.763**	0.591	2.952
	ANN	0.785	0.728	0.570	**0.728**	0.612	**0.722**	0.578	**0.728**	0.469	0.570
BO-TPE	RF	0.795	**0.808**	0.638	**0.808**	0.690	**0.795**	0.655	**0.807**	1.118	**0.598**
	SVM	0.780	0.725	0.591	**0.725**	0.641	**0.736**	0.598	**0.724**	0.490	0.554
	KNN	0.791	0.765	0.621	**0.765**	0.732	**0.773**	0.635	**0.763**	0.584	2.928
	ANN	0.785	0.728	0.573	**0.728**	0.618	**0.720**	0.577	**0.726**	0.469	0.563
PSO	RF	0.790	**0.806**	0.635	**0.806**	0.683	**0.789**	0.652	**0.804**	1.189	**0.615**
	SVM	0.782	0.728	0.586	**0.728**	0.636	**0.738**	0.596	**0.726**	0.491	0.554
	KNN	0.792	0.766	0.621	**0.766**	0.730	**0.774**	0.632	**0.765**	0.590	3.115
	ANN	0.784	0.728	0.569	**0.728**	0.616	**0.722**	0.578	**0.730**	0.466	0.570
GA	RF	0.791	**0.805**	0.629	**0.805**	0.676	**0.792**	0.642	**0.801**	1.129	**0.610**
	SVM	0.781	0.723	0.590	**0.723**	0.634	**0.735**	0.595	**0.722**	0.493	0.555
	KNN	0.793	0.767	0.622	**0.767**	0.734	**0.772**	0.637	**0.763**	0.592	3.004
	ANN	0.786	0.731	0.574	**0.731**	0.619	**0.721**	0.577	**0.728**	0.468	0.566
TPOT	RF	0.793	**0.805**	0.632	**0.805**	0.681	**0.793**	0.645	**0.803**	1.150	**0.604**
	SVM	0.784	0.725	0.590	**0.725**	0.639	**0.737**	0.600	**0.725**	0.495	0.558
	KNN	0.795	0.768	0.623	**0.768**	0.735	**0.775**	0.636	**0.764**	0.585	2.978
	ANN	0.787	0.730	0.576	**0.730**	0.622	**0.723**	0.579	**0.729**	0.470	0.574

Dataset Breast Cancer Wisconsin
Quality Check applicato: Algoritmo *SMOTE* per bilanciare le classi nel training set

Tabella 3.118: Confronto delle metriche prima e dopo il quality check

Procedura	Algoritmo	Accuracy		Recall		Precision		F1		Log Loss	
		Prima	Dopo	Prima	Dopo	Prima	Dopo	Prima	Dopo	Prima	Dopo
3-fold	RF	0.944	**0.962**	0.944	**0.962**	0.946	**0.962**	0.944	**0.962**	0.428	**0.111**
	SVM	0.970	**0.972**	0.970	**0.972**	0.971	**0.973**	0.970	**0.972**	0.162	**0.083**
	KNN	0.963	**0.974**	0.962	**0.974**	0.964	**0.975**	0.963	**0.974**	0.467	**0.207**
	ANN	0.989	0.984	0.989	0.984	0.990	0.984	0.989	0.984	0.051	0.060
GS	RF	0.936	**0.960**	0.936	**0.960**	0.938	**0.961**	0.936	**0.960**	0.466	**0.179**
	SVM	0.974	**0.978**	0.974	**0.978**	0.975	**0.979**	0.974	**0.978**	0.159	**0.084**
	KNN	0.968	**0.978**	0.968	**0.978**	0.969	**0.978**	0.968	**0.978**	0.586	**0.262**
	ANN	0.952	**0.980**	0.952	**0.980**	0.952	**0.980**	0.952	**0.980**	0.167	**0.081**
RS	RF	0.930	**0.964**	0.930	**0.964**	0.932	**0.964**	0.930	**0.964**	0.544	**0.130**
	SVM	0.974	**0.978**	0.974	**0.978**	0.975	**0.978**	0.974	**0.978**	0.160	**0.068**
	KNN	0.968	**0.978**	0.968	**0.978**	0.969	**0.978**	0.968	**0.978**	0.586	**0.262**
	ANN	0.991	0.978	0.991	0.978	0.991	0.978	0.991	0.978	0.036	0.065
HB	RF	0.983	0.959	0.983	0.959	0.983	0.959	0.983	0.959	0.332	**0.098**
	SVM	0.986	**0.988**	0.986	**0.988**	0.986	**0.988**	0.986	**0.988**	0.088	**0.078**
	KNN	0.989	0.965	0.989	0.965	0.989	0.965	0.989	0.965	0.127	0.301
	ANN	0.981	0.971	0.981	0.971	0.981	0.971	0.981	0.971	0.109	**0.051**
BO	RF	0.932	**0.966**	0.932	**0.966**	0.933	**0.966**	0.931	**0.966**	0.417	**0.145**
	SVM	0.974	**0.978**	0.974	**0.978**	0.975	**0.978**	0.974	**0.978**	0.159	**0.068**
	KNN	0.964	**0.974**	0.964	**0.974**	0.965	**0.975**	0.964	**0.974**	0.500	**0.207**
	ANN	0.994	0.980	0.994	0.980	0.995	0.980	0.994	0.980	0.034	0.081
SMBO	RF	0.918	**0.956**	0.917	**0.968**	0.915	**0.953**	0.913	**0.960**	0.430	**0.175**
	SVM	0.974	0.974	0.974	0.952	0.975	**0.996**	0.974	0.973	0.159	**0.108**
	KNN	0.954	**0.970**	0.953	0.944	0.956	**0.996**	0.953	**0.969**	0.228	0.319
	ANN	0.927	**0.970**	0.927	**0.976**	0.933	**0.965**	0.927	**0.970**	0.232	0.377
BO-TPE	RF	0.904	**0.954**	0.906	**0.960**	0.908	**0.956**	0.902	**0.958**	0.423	**0.182**
	SVM	0.974	**0.978**	0.974	**0.984**	0.975	0.973	0.974	**0.978**	0.160	**0.083**
	KNN	0.954	**0.970**	0.953	0.944	0.956	**0.996**	0.953	**0.969**	0.228	0.319
	ANN	0.929	**0.968**	0.929	**0.964**	0.933	**0.980**	0.929	**0.974**	0.230	**0.089**
PSO	RF	0.932	**0.958**	0.932	**0.958**	0.934	**0.959**	0.931	**0.958**	0.499	**0.175**
	SVM	0.974	**0.980**	0.974	**0.980**	0.975	**0.980**	0.974	**0.980**	0.158	**0.070**
	KNN	0.968	**0.978**	0.968	**0.978**	0.969	**0.978**	0.968	**0.978**	0.586	**0.262**
	ANN	0.930	**0.982**	0.930	**0.982**	0.933	**0.982**	0.930	**0.982**	0.533	**0.060**
GA	RF	0.941	**0.958**	1.000	0.990	1.000	0.990	1.000	0.990	0.090	**0.054**
	SVM	0.974	**0.980**	1.000	0.994	1.000	0.994	1.000	0.994	0.023	**0.018**
	KNN	0.968	**0.974**	0.993	0.982	0.993	0.982	0.993	0.982	0.013	0.045
	ANN	1.000	0.990	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.002
TPOT	RF	0.927	**0.960**	0.927	**0.960**	0.929	**0.960**	0.927	**0.960**	0.343	**0.179**
	SVM	0.974	**0.978**	0.974	**0.978**	0.975	**0.979**	0.974	**0.978**	0.160	**0.084**
	KNN	0.968	**0.978**	0.968	**0.978**	0.969	**0.978**	0.968	**0.978**	0.586	**0.262**
	ANN	0.953	**0.986**	0.953	**0.986**	0.954	**0.986**	0.953	**0.986**	0.261	**0.072**

Dataset Digits MNIST
La matrice di correlazione mostra come le features siano quasi completamente scorrelate (essendo di colore azzurro chiaro), come visibile in figura 3.1

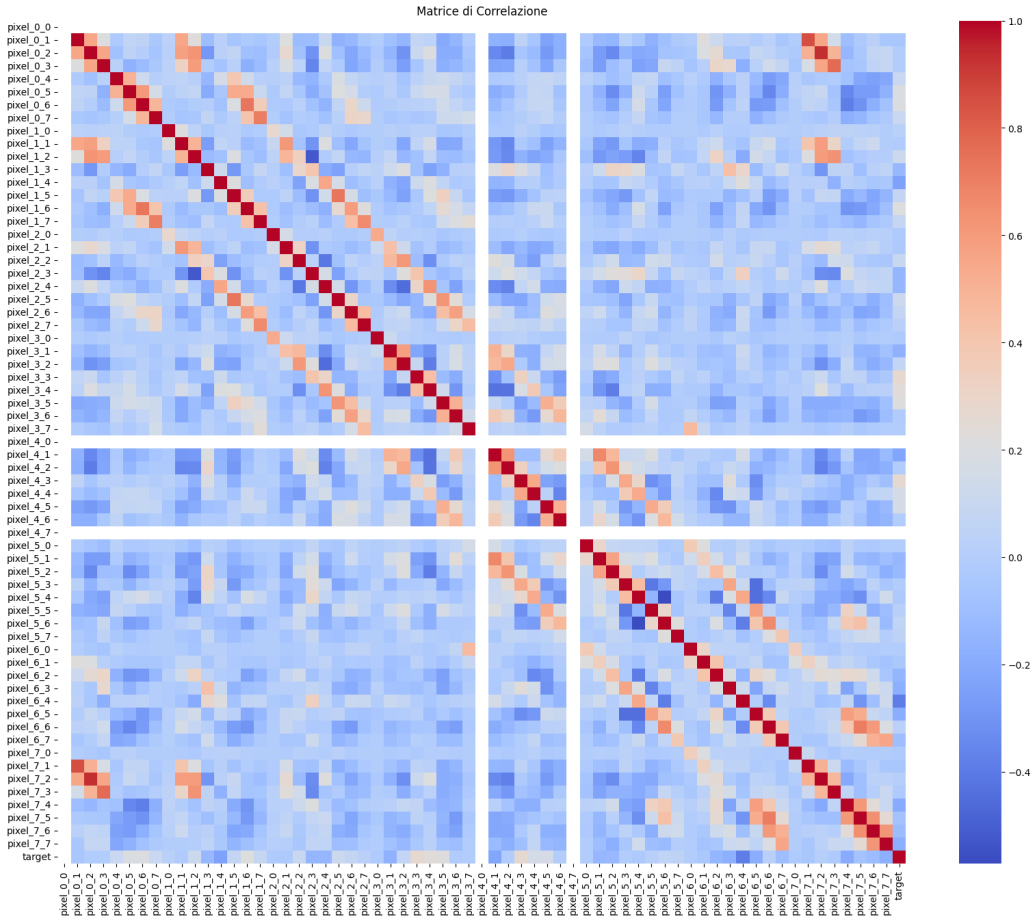

Figura 3.1: Correlazione tra feature nel dataset Digits

Anche qui, nonostante il dataset fosse quasi completamente bilanciato, dopo aver trovato la classe più sbilanciata, si è applicato l'algoritmo *SMOTE* per bilanciarla.

Quality Check applicato: **Algoritmo *SMOTE* per bilanciare le classi nel training set**

Tabella 3.119: Confronto delle metriche prima e dopo il quality check

Procedura	Algoritmo	Accuracy		Recall		Precision		F1		Log Loss	
		Prima	Dopo	Prima	Dopo	Prima	Dopo	Prima	Dopo	Prima	Dopo
3-fold	RF	0.944	**0.948**	0.944	**0.948**	0.946	**0.950**	0.944	**0.948**	0.428	**0.412**
	SVM	0.970	0.970	0.970	0.970	0.971	0.971	0.970	0.970	0.162	**0.157**
	KNN	0.963	**0.967**	0.962	**0.967**	0.964	**0.967**	0.963	**0.967**	0.467	**0.416**
	ANN	0.989	**0.990**	0.989	**0.990**	0.990	0.990	0.989	**0.990**	0.051	**0.047**
GS	RF	0.936	0.936	0.936	0.936	0.938	0.937	0.936	0.936	0.466	**0.458**
	SVM	0.974	**0.975**	0.974	**0.975**	0.975	**0.976**	0.974	**0.975**	0.159	**0.154**
	KNN	0.968	**0.972**	0.968	**0.972**	0.969	**0.972**	0.968	**0.972**	0.586	**0.516**
	ANN	0.952	**0.960**	0.952	**0.960**	0.952	**0.960**	0.952	**0.959**	0.167	**0.159**
RS	RF	0.930	0.930	0.930	0.930	0.932	0.932	0.930	0.929	0.544	**0.525**
	SVM	0.974	**0.976**	0.974	**0.976**	0.975	**0.977**	0.974	**0.976**	0.160	**0.153**
	KNN	0.968	0.967	0.968	0.967	0.969	0.967	0.968	0.967	0.586	**0.416**
	ANN	0.991	**0.995**	0.991	**0.995**	0.991	**0.995**	0.991	**0.995**	0.036	**0.028**
HB	RF	0.983	0.973	0.983	0.973	0.983	0.973	0.983	0.973	0.332	**0.255**
	SVM	0.986	0.978	0.986	0.978	0.986	0.979	0.986	0.978	0.088	0.108
	KNN	0.989	0.986	0.989	0.986	0.989	0.987	0.989	0.986	0.127	**0.046**
	ANN	0.981	0.975	0.981	0.975	0.981	0.976	0.981	0.975	0.109	**0.089**
BO	RF	0.932	**0.933**	0.932	**0.933**	0.933	**0.936**	0.931	**0.933**	0.417	0.450
	SVM	0.974	**0.976**	0.974	**0.976**	0.975	**0.977**	0.974	**0.976**	0.159	**0.152**
	KNN	0.964	0.963	0.964	0.963	0.965	0.964	0.964	0.963	0.500	**0.318**
	ANN	0.994	0.987	0.994	0.987	0.995	0.988	0.994	0.987	0.034	0.039
SMBO	RF	0.918	**0.932**	0.917	**0.938**	0.915	**0.938**	0.913	**0.938**	0.430	**0.399**
	SVM	0.974	**0.975**	0.974	**0.975**	0.975	**0.976**	0.974	**0.975**	0.159	**0.156**
	KNN	0.954	0.952	0.953	0.952	0.956	0.954	0.953	0.952	0.228	**0.208**
	ANN	0.927	0.924	0.927	0.924	0.933	0.927	0.927	0.924	0.232	0.241
BO-TPE	RF	0.904	**0.923**	0.906	**0.922**	0.908	**0.930**	0.902	**0.921**	0.423	0.462
	SVM	0.974	**0.975**	0.974	**0.975**	0.975	**0.976**	0.974	**0.975**	0.160	**0.154**
	KNN	0.954	0.952	0.953	0.952	0.956	0.955	0.953	0.952	0.228	**0.195**
	ANN	0.929	**0.987**	0.929	**0.989**	0.933	**0.989**	0.929	**0.986**	0.230	**0.041**
PSO	RF	0.932	**0.937**	0.932	**0.937**	0.934	**0.939**	0.931	**0.937**	0.499	**0.447**
	SVM	0.974	**0.975**	0.974	**0.975**	0.975	**0.976**	0.974	**0.975**	0.158	**0.154**
	KNN	0.968	**0.969**	0.968	**0.969**	0.969	**0.970**	0.968	**0.969**	0.586	**0.468**
	ANN	0.930	0.925	0.930	0.925	0.933	0.930	0.930	0.926	0.533	0.541
GA	RF	0.941	0.940	1.000	0.998	1.000	0.998	1.000	0.998	0.090	0.252
	SVM	0.974	**0.976**	1.000	1.000	1.000	1.000	1.000	1.000	0.023	0.024
	KNN	0.968	0.967	0.993	0.991	0.993	0.991	0.993	0.991	0.013	**0.008**
	ANN	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.001
TPOT	RF	0.927	**0.931**	0.927	**0.931**	0.929	**0.933**	0.927	**0.931**	0.343	0.742
	SVM	0.974	**0.975**	0.974	**0.975**	0.975	**0.976**	0.974	**0.975**	0.160	**0.155**
	KNN	0.968	**0.972**	0.968	**0.972**	0.969	**0.972**	0.968	**0.972**	0.586	**0.516**
	ANN	0.953	**0.956**	0.953	**0.956**	0.954	**0.958**	0.953	**0.956**	0.261	**0.152**

3.4.4 Risultati applicando delle linee guida alla regressione

Dataset per la regressione
Dataset Life Expectancy
Quality Check applicato: **Rimozione degli outliers**

Tabella 3.120: Confronto delle metriche prima e dopo il quality check

Procedura	Algoritmo	MSE		R^2		Tempo (s)	
		Prima	Dopo	Prima	Dopo	Prima	Dopo
3-fold	RF	4.213	**4.079**	0.953	0.949	25.75	36.09
	SVM	15.180	**14.864**	0.830	0.816	2.11	**0.97**
	KNN	11.728	12.640	0.868	0.844	0.11	**0.06**
	ANN	12.667	13.803	0.843	0.843	78.42	**72.36**
GS	RF	4.392	**4.134**	0.993	0.993	17.59	**15.38**
	SVM	7.267	7.599	0.944	0.937	67.44	**37.20**
	KNN	11.302	12.309	0.952	0.944	0.29	**0.23**
	ANN	10.835	11.097	0.876	0.869	2294.43	2375.13
RS	RF	4.307	4.539	0.991	0.985	85.41	**74.99**
	SVM	7.994	8.357	0.933	0.922	23.36	**20.12**
	KNN	11.495	12.339	0.940	0.931	0.58	**0.37**
	ANN	10.024	10.625	0.850	**0.874**	397.16	433.64
HB	RF	3.880	**3.780**	0.993	0.993	408.27	**392.76**
	SVM	14.815	**13.909**	0.848	0.840	126.94	**120.34**
	KNN	10.809	**9.525**	0.969	0.951	15.56	**13.13**
	ANN	8.260	**7.440**	0.931	0.928	376.00	390.52
BO	RF	3.982	**3.915**	0.994	0.994	73.68	**65.79**
	SVM	7.952	8.071	0.934	0.927	35.75	47.00
	KNN	11.302	12.309	0.952	0.944	1.59	**1.05**
	ANN	9.309	11.314	0.924	0.877	476.60	**386.52**
SMBO	RF	4.024	**3.932**	0.995	0.994	42.72	**40.95**
	SVM	8.094	8.226	0.931	0.924	20.79	24.00
	KNN	11.882	12.712	0.969	0.964	1.30	**0.94**
	ANN	3051.408	3058.616	0.863	**0.866**	109.88	**106.24**
BO-TPE	RF	4.096	4.255	0.994	0.987	30.75	**24.24**
	SVM	7.545	7.730	0.938	0.935	53.83	**39.57**
	KNN	11.302	12.309	0.952	0.944	1.60	2.66
	ANN	12.780	13.127	0.880	0.865	524.56	**460.84**
PSO	RF	4.163	4.193	0.992	0.989	98.10	123.08
	SVM	7.944	8.125	0.934	0.925	63.97	93.39
	KNN	11.495	12.339	0.940	0.931	1.43	**1.42**
	ANN	11.945	12.460	0.891	0.874	974.00	1003.05
GA	RF	4.440	**4.169**	0.991	0.990	33.03	**13.85**
	SVM	8.363	**7.295**	0.932	0.932	15.96	28.08
	KNN	12.059	**10.652**	0.929	**0.968**	0.94	**0.72**
	ANN	9.067	**8.800**	0.891	**0.933**	1035.61	**994.02**
TPOT	RF	0.846	**0.440**	0.991	**0.995**	72.45	**72.32**
	SVM	5.824	6.224	0.935	0.923	34.27	**25.45**
	KNN	4.265	4.522	0.952	0.944	1.26	2.34
	ANN	7.048	10.941	0.922	0.863	137.15	201.15

Dataset Obesity
Quality Check applicato: **Rimozione degli outliers**
La rimozione degli outliers dal dataset è stata fatta provandola su 3 modelli:
Isolation Forest[7]:
Modello configurato con un livello di contaminazione (`contamination`) pari a 0.1 e un seme casuale (`random_state`) impostato a 42 per garantire la riproducibilità.

Elliptic Envelope[8]:
Modello configurato con:

- un livello di contaminazione (`contamination`) pari a 0.1,
- una frazione di supporto (`support_fraction`) impostata a 0.95,
- un seme casuale (`random_state`) impostato a 42.

Local Outlier Factor (LOF)[9]: Modello configurato con:

- un livello di contaminazione (`contamination`) pari a 0.1,
- la modalità `novelty` disabilitata (`novelty` = `False`).

[7]https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/
[8]https://medium.com/@ssssppppp/outliers-detection-using-elliptic-envelope-in-python-673a39e3b315
[9]https://medium.com/@pramodch/understanding-lof-local-outlier-factor-for-implementation-1f6d4ff13ab9

Tabella 3.121: Confronto delle metriche prima e dopo il quality check

Procedura	**Algoritmo**	**MSE**		**R^2**		**Tempo (s)**	
		Prima	**Dopo**	**Prima**	**Dopo**	**Prima**	**Dopo**
3-fold	RF	0.400	**0.383**	0.899	0.899	5.31	**2.36**
	SVM	3.139	**2.827**	0.180	**0.246**	1.39	**0.48**
	KNN	1.136	**0.927**	0.703	**0.752**	0.20	**0.05**
	ANN	1.818	**1.802**	0.436	**0.492**	88.74	**72.43**
GS	RF	0.392	**0.382**	0.988	0.988	3.60	**3.59**
	SVM	1.946	**1.907**	0.514	**0.523**	14.13	**11.24**
	KNN	0.991	**0.879**	0.930	0.905	0.18	**0.13**
	ANN	1.143	**1.086**	0.755	0.751	1811.11	**1755.19**
RS	RF	0.470	**0.467**	0.956	**0.975**	28.50	**24.21**
	SVM	1.964	**1.945**	0.507	**0.509**	10.45	**8.98**
	KNN	0.991	1.030	0.930	0.840	0.59	**0.36**
	ANN	0.966	1.053	0.829	0.791	414.86	**375.64**
HB	RF	0.375	**0.337**	0.976	**0.977**	149.44	160.95
	SVM	3.074	**2.703**	0.229	**0.329**	77.68	**62.40**
	KNN	1.015	**0.807**	0.880	**0.942**	10.78	**10.01**
	ANN	1.969	**1.925**	0.502	0.499	298.17	**224.68**
BO	RF	0.391	0.391	0.988	0.988	44.47	**43.94**
	SVM	1.964	**1.919**	0.505	**0.508**	28.29	29.05
	KNN	0.991	**0.879**	0.930	0.905	0.82	1.20
	ANN	1.069	**0.906**	0.769	**0.811**	338.50	344.93
SMBO	RF	0.375	0.376	0.989	**0.990**	25.83	**19.98**
	SVM	1.959	**1.941**	0.507	**0.513**	13.82	**12.71**
	KNN	0.991	**0.934**	0.930	**0.941**	0.79	0.85
	ANN	3.017	3.227	0.762	0.762	132.27	**110.95**
BO-TPE	RF	0.527	**0.431**	0.941	**0.976**	14.18	15.97
	SVM	1.950	**1.947**	0.513	0.509	17.33	**13.42**
	KNN	0.991	**0.902**	0.930	0.886	1.73	**1.47**
	ANN	1.621	**1.420**	0.686	**0.705**	414.10	465.03
PSO	RF	0.394	0.463	0.968	0.958	14.88	16.08
	SVM	1.987	**1.964**	0.499	**0.502**	520.40	**427.29**
	KNN	0.991	**0.879**	0.930	0.905	0.73	**0.37**
	ANN	1.660	**1.585**	0.658	**0.689**	386.66	**328.29**
GA	RF	0.465	**0.395**	0.961	**0.964**	18.26	**11.01**
	SVM	1.989	**1.856**	0.507	**0.526**	10.66	**7.04**
	KNN	1.126	**0.872**	0.836	**0.913**	0.58	0.70
	ANN	1.035	**0.847**	0.836	**0.887**	434.33	893.10
TPOT	RF	0.107	**0.062**	0.972	**0.984**	65.70	**55.87**
	SVM	1.891	**1.829**	0.507	**0.513**	10.63	13.12
	KNN	0.267	0.356	0.930	0.905	1.49	**1.20**
	ANN	1.143	**1.133**	0.690	0.685	151.12	**105.78**

Dataset Boston Housing
Quality Check applicato: **Rimozione degli outliers**

Tabella 3.122: Confronto delle metriche prima e dopo il quality check

Procedura	**Algoritmo**	**MSE**		**R^2**		**Tempo (s)**	
		Prima	**Dopo**	**Prima**	**Dopo**	**Prima**	**Dopo**
3-fold	RF	11.834	**9.744**	0.855	0.844	3.52	**1.90**
	SVM	66.493	**47.506**	0.208	**0.263**	0.09	0.10
	KNN	49.421	**33.234**	0.411	**0.488**	0.03	0.03
	ANN	45.396	**38.360**	0.508	0.481	60.38	**28.64**
GS	RF	12.084	**9.596**	0.976	**0.977**	2.37	**2.22**
	SVM	45.283	**32.306**	0.500	**0.528**	1.44	**0.73**
	KNN	47.677	**31.558**	0.602	**0.859**	0.07	**0.05**
	ANN	33.056	**24.171**	0.594	**0.700**	916.31	1031.28
RS	RF	12.207	**9.140**	0.981	0.964	9.83	**7.12**
	SVM	50.070	**34.169**	0.439	**0.501**	1.09	**0.67**
	KNN	47.549	**31.558**	0.563	**0.859**	0.18	**0.15**
	ANN	32.677	**26.279**	0.667	0.616	190.79	**183.41**
HB	RF	10.472	11.217	0.980	0.977	102.11	**97.74**
	SVM	76.349	**54.595**	0.156	**0.220**	4.62	**3.87**
	KNN	45.661	**38.904**	0.752	0.746	0.99	1.00
	ANN	34.743	**33.122**	0.659	0.447	103.85	**92.01**
BO	RF	14.243	**9.257**	0.908	**0.969**	36.77	**35.61**
	SVM	49.918	**34.175**	0.439	**0.496**	22.66	**22.03**
	KNN	46.868	**31.693**	0.583	**0.799**	2.12	**1.22**
	ANN	36.003	**22.064**	0.576	**0.627**	201.08	**188.92**
SMBO	RF	11.672	**9.349**	0.982	0.950	18.53	**15.21**
	SVM	49.832	**34.695**	0.439	**0.494**	6.93	8.18
	KNN	46.868	**31.558**	0.583	**0.859**	1.11	**0.63**
	ANN	105.264	192.223	0.517	0.289	98.19	123.77
BO-TPE	RF	12.477	**9.894**	0.964	0.961	10.38	**8.80**
	SVM	45.455	**33.478**	0.507	**0.511**	1.35	**1.34**
	KNN	47.549	**31.558**	0.563	**0.859**	0.84	**0.63**
	ANN	42.377	46.093	0.593	0.366	257.15	**247.43**
PSO	RF	13.552	**9.395**	0.951	**0.966**	21.18	23.36
	SVM	50.596	**34.845**	0.430	**0.492**	2.42	**2.25**
	KNN	47.677	**31.558**	0.602	**0.859**	0.77	1.00
	ANN	35.970	**30.863**	0.632	0.623	518.12	594.90
GA	RF	14.232	**10.524**	0.946	**0.953**	11.87	**5.78**
	SVM	57.692	**41.033**	0.390	**0.439**	1.33	**0.51**
	KNN	47.890	**35.492**	0.820	**0.861**	0.12	**0.07**
	ANN	30.314	**27.145**	0.747	0.674	426.14	495.71
TPOT	RF	3.415	**2.122**	0.959	**0.967**	28.14	**26.97**
	SVM	47.514	**32.097**	0.433	**0.499**	1.50	**1.09**
	KNN	34.819	**13.138**	0.584	**0.795**	0.69	0.94
	ANN	26.007	**16.402**	0.641	**0.699**	55.55	59.71

3.4.5 Applicazione di altri quality check al dataset Boston Housing

Quality Check applicato: **Selezione delle feature più significative**

Tabella 3.123: Confronto delle metriche prima e dopo il quality check

Procedura	**Algoritmo**	**MSE**		**R^2**		**Tempo (s)**	
		Prima	**Dopo**	**Prima**	**Dopo**	**Prima**	**Dopo**
3-fold	RF	9.744	17.040	0.844	0.804	1.90	4.23
	SVM	47.506	**39.203**	0.263	**0.558**	0.10	0.22
	KNN	33.234	**20.060**	0.488	**0.763**	0.03	0.08
	ANN	38.360	**19.401**	0.481	**0.783**	28.64	47.78
GS	RF	9.596	15.764	0.977	0.976	2.22	**1.82**
	SVM	32.306	**17.207**	0.528	**0.848**	0.73	4.25
	KNN	31.558	**19.738**	0.859	**0.898**	0.05	0.08
	ANN	24.171	**17.651**	0.700	**0.817**	1031.28	1189.26
RS	RF	9.140	16.395	0.964	0.927	7.12	9.08
	SVM	34.169	**16.831**	0.501	**0.843**	0.67	4.06
	KNN	31.558	**20.060**	0.859	**0.863**	0.15	**0.10**
	ANN	26.279	**17.452**	0.616	**0.824**	183.41	230.93
HB	RF	11.217	14.028	0.977	0.971	97.74	**93.48**
	SVM	54.595	**36.530**	0.220	**0.625**	3.87	5.67
	KNN	38.904	**20.277**	0.746	**0.863**	1.00	1.06
	ANN	33.122	**21.417**	0.447	**0.777**	92.01	**78.61**
BO	RF	9.257	14.478	0.969	0.947	35.61	**34.55**
	SVM	34.175	**16.907**	0.496	**0.841**	22.03	22.25
	KNN	31.693	**19.738**	0.799	**0.898**	1.22	**0.91**
	ANN	22.064	**17.624**	0.627	**0.808**	188.92	223.66
SMBO	RF	9.349	15.564	0.950	0.923	15.21	17.98
	SVM	34.695	**16.784**	0.494	**0.842**	8.18	10.46
	KNN	31.558	**20.703**	0.859	0.837	0.63	0.65
	ANN	192.223	402.762	0.289	**0.754**	123.77	**117.10**
BO-TPE	RF	9.894	15.710	0.961	**0.971**	8.80	**8.36**
	SVM	33.478	**16.950**	0.511	**0.847**	1.34	8.29
	KNN	31.558	**19.738**	0.859	**0.898**	0.63	0.67
	ANN	46.093	**18.689**	0.366	**0.821**	247.43	295.47
PSO	RF	9.395	15.025	0.966	0.944	23.36	50.79
	SVM	34.845	**17.476**	0.492	**0.840**	2.25	6.26
	KNN	31.558	**20.060**	0.859	**0.863**	1.00	1.08
	ANN	30.863	**18.750**	0.623	**0.819**	594.90	647.17
GA	RF	10.524	15.084	0.953	0.953	5.78	14.36
	SVM	41.033	**17.174**	0.439	**0.843**	0.51	2.59
	KNN	35.492	**20.055**	0.861	**0.863**	0.07	0.11
	ANN	27.145	**17.334**	0.674	**0.820**	495.71	649.73
TPOT	RF	2.122	**0.732**	0.967	**0.992**	26.97	**24.58**
	SVM	32.097	**14.030**	0.499	**0.842**	1.09	2.83
	KNN	13.138	**9.057**	0.795	**0.898**	0.94	**0.74**
	ANN	16.402	20.027	0.699	**0.793**	59.71	86.51

Quality Check applicato: **Log Transformation del target**
La trasformazione logaritmica è un metodo utilizzato per ridurre l'asimmetria e avvicinare la distribuzione dei dati alla normalità. Applicando una trasformazione logaritmica alla variabile target, si possono rendere i dati più simmetrici, stabilizzare la varianza e migliorare le prestazioni dei nostri modelli di regressione [50].

Tabella 3.124: Confronto delle metriche prima e dopo il quality check

Procedura	**Algoritmo**	**MSE**		**R^2**		**Tempo (s)**	
		Prima	**Dopo**	**Prima**	**Dopo**	**Prima**	**Dopo**
3-fold	RF	17.040	**0.026**	0.804	**0.819**	4.230	**3.840**
	SVM	39.203	**0.033**	0.558	**0.781**	0.220	**0.120**
	KNN	20.060	**0.034**	0.763	**0.766**	0.080	0.110
	ANN	19.401	**0.069**	0.783	0.578	47.780	**33.820**
GS	RF	15.764	**0.026**	0.977	0.968	1.820	**1.620**
	SVM	17.207	**0.033**	0.528	**0.849**	4.250	12.600
	KNN	19.738	**0.034**	0.898	0.847	0.080	**0.040**
	ANN	17.651	**0.035**	0.700	**0.807**	1189.260	**928.840**
RS	RF	16.395	**0.026**	0.927	**0.965**	9.080	**8.030**
	SVM	16.831	**0.040**	0.843	**0.883**	4.060	**2.670**
	KNN	20.060	**0.036**	0.863	0.853	0.100	**0.090**
	ANN	17.452	**0.033**	0.824	**0.835**	230.930	**177.200**
HB	RF	14.028	**0.023**	0.977	0.964	93.480	98.960
	SVM	36.530	**0.036**	0.625	**0.848**	5.670	6.810
	KNN	20.277	**0.035**	0.863	0.853	1.060	**0.910**
	ANN	21.417	**0.035**	0.777	0.733	78.610	**66.220**
BO	RF	14.478	**0.028**	0.947	0.941	34.550	**8.600**
	SVM	16.907	**0.043**	0.841	**0.852**	22.250	**3.300**
	KNN	19.738	**0.034**	0.898	0.847	0.910	**0.570**
	ANN	17.624	**0.030**	0.808	**0.821**	223.660	**178.350**
SMBO	RF	15.564	**0.026**	0.923	**0.979**	17.980	**17.180**
	SVM	16.784	**0.045**	0.842	0.774	10.460	**7.120**
	KNN	20.703	**0.035**	0.837	**0.839**	0.650	**0.460**
	ANN	402.762	**0.286**	0.754	0.710	117.100	**107.310**
BO-TPE	RF	15.710	**0.028**	0.961	0.941	8.360	8.600
	SVM	16.950	**0.043**	0.847	**0.852**	8.290	**3.300**
	KNN	19.738	**0.034**	0.898	0.847	0.670	**0.570**
	ANN	18.689	**0.030**	0.821	0.821	295.470	**178.350**
PSO	RF	15.025	**0.026**	0.944	**0.975**	50.790	**17.100**
	SVM	17.476	**0.034**	0.840	**0.868**	6.260	**4.550**
	KNN	20.060	**0.035**	0.863	0.839	1.080	**0.970**
	ANN	18.750	**0.066**	0.819	0.749	647.170	**573.150**
GA	RF	15.084	**0.026**	0.953	0.953	14.360	21.640
	SVM	17.174	**0.043**	0.843	0.769	2.590	2.690
	KNN	20.055	**0.034**	0.863	0.847	0.110	**0.070**
	ANN	17.334	**0.030**	0.820	**0.847**	649.730	**531.530**
TPOT	RF	0.732	**0.005**	0.992	0.966	24.580	25.940
	SVM	14.030	**0.019**	0.842	**0.877**	2.830	**1.630**
	KNN	9.057	**0.023**	0.898	0.847	0.740	**0.590**
	ANN	20.027	**0.029**	0.793	0.776	86.510	**54.770**

Capitolo 4

Conclusioni

Un'ampia ricerca in letteratura ha portato alla scoperta di diverse linee guida/quality check per costruire un buon modello di regressione (alcune linee guida sono applicabili anche alla classificazione). Applicando delle semplici linee guida come la **rimozione degli outlier** e/o il **bilanciamento delle classi**, c'è stato un miglioramento significativo per quasi tutte le metriche in quasi tutti i dataset di classificazione e di regressione.

Ad esempio, nella tabella 3.119, nonostante la scarsa correlazione tra le feature della figura 3.1, provando comunque a bilanciare le feature c'è stato un piccolo miglioramento nelle metriche: ad esempio, l'accuracy con la procedura di Grid Search è passata nelle Random Forest da *0.936* a *0.936*, in SVM da *0.974* a *0.975*, in KNN da *0.968* a *0.972* e in ANN da *0.952* a *0.960*.
Applicare quindi anche un solo semplice quality check ha migliorato il modello.

I risultati per il Dataset di regressione Boston Housing non sono invece migliorati molto da come si evince in tabella 3.122.
Infatti, nonostante i sia stata una riduzione dell'MSE in quasi tutte le procedure e un aumento dell'R^2, quando si è applicata la rete neurale l'R^2 è diminuito nella procedure 3-fold, Random Search, Hyperband, SMBO, BO-TPE, PSO e GA.
Questo è sicuramente giustificato dal fatto che rimuovendo gli outlier ci può essere una perdita di informazione e la rete neurale può fare più fatica.

Riguardo il Tempo trascorso dalle varie procedure per venire eseguite, non c'è da dire molto se non che, (banalmente) in alcuni casi, il tempo trascorso è diminuito di poco in alcuni casi dopo la rimozione degli outliers, perché ci sono meno dati da processare.

Riguardo la rete neurale, il tempo trascorso è sempre più alto di qualunque algoritmo per ogni procedura vista la complessità dell'algoritmo: infatti la rete neurale ha 3 layer e inoltre può avere molti iperparametri da ottimizzare (ad esempio, numero di neuroni, learning rate, batch size), che possono portare a una maggiore esplorazione dello spazio degli iperparametri e questo processo aumenta il tempo complessivo.

Riguardo il dataset Breast Cancer del Wisconsin, nella tabella 3.118 si nota che per quasi tutte le procedure, bilanciando le classi, c'è stato un miglioramento per tutte le metriche, l'unica procedura che non ha notato miglioramenti (se non nell'accuracy) è stato l'algoritmo genetico, è possibile che tale algoritmo performi meglio in dataset con classi non perfettamente bilanciate, almeno in dataset biologici.

Questo si nota anche nel dataset di Mnist nella tabella 3.119, che come si è visto ha già tutte le feature quasi completamente bilanciate.
Gli algoritmi genetici, infatti, sono generalmente meno sensibili al problema dello sbilanciamento delle classi rispetto ad altri approcci, come le reti neurali o le random forest, perché si concentrano sull'evoluzione e sull'ottimizzazione globale delle soluzioni, piuttosto che sulla separazione precisa tra le classi.

Alcuni studi come [49] e suggeriscono che, sebbene tecniche come SMOTE possano migliorare le prestazioni su dataset sbilanciati, gli algoritmi genetici potrebbero non mostrare un significativo miglioramento con il bilanciamento delle classi. Infatti, alcuni metodi evolutivi, come il *GenSample*, mirano a migliorare l'oversampling della classe minoritaria ma con approcci specifici.

Per ultima cosa si è visto in tabella 3.123 che, nel dataset di regressione di Boston Housing, applicando altre linee guida quali la **selezione delle feature**, l'MSE e l'R^2 sono migliorati: rispettivamente, uno è diminuito e l'altro è aumentato, il che significa che il modello è in grado di spiegare meglio una grande parte della varianza presente nei dati.

Quasi tutte le procedure hanno avuto le metriche migliorate rispetto a prima, tranne la Random Forest, che essendo già ben prestante prima di selezionare le feature non ha avuto risultati significativamente migliori.
Stessa cosa per la tabella 3.124: quasi tutte le metriche sono migliorate in quasi tutte le procedure.

È stato quindi dimostrato con esperimenti pratici su 6 dataset in totale (3 di regressione e 3 di classificazione), che applicare le linee guide teoriche trovate in letteratura ha migliorato le metriche dei vari algoritmi e procedure utilizzate, soprattutto anche per merito della ricerca e dell'ottimizzazione degli iperparametri.

In conclusione, è stato dimostrato che è quindi possibile, nel campo del Machine Learning, non basarsi solo sull'intuito e l'esperienza ma anche appoggiandosi a delle regole, lista di quality check da verificare e linee guida da utilizzare per gli utenti esperti e non, per costruire un modello di Machine Learning che sia il più possibile affidabile e performante.

Bibliografia

[1] C. Szegedy et al. Intriguing properties of neural networks. *Research Gate*, pp. 6, December 2013.

[2] B. Whitfield. Random forest: A complete guide for machine learning. *Built_in*, March 2024.

[3] P. Piacenti. Come funziona un artificial neural network (spiegato in maniera semplice). *Medium*, October 2019.

[4] W.W.Wang et al. knn e ann. *Microsoft*, August 2024.

[5] A. Barbieri. Svm, support vector machines: guida ai modelli dell'ia. *Multinazionali Tech*, August 2023.

[6] Z. Bobbitt. What is sturges rule? (definition and example). *Statology*, January 2021.

[7] A. D'Agostino. Cosa è la cross-validazione nel machine learning. *Diario Di Un Analista*, August 2022.

[8] A. D'Agostino. Cosa è la grid search e come applicarla in python con sklearn. *Diario Di Un Analista*, April 2023.

[9] M. H. Hassan. Tuning model hyperparameters with random search. *Medium*, November 2023.

[10] L. Li et al. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(16-558):3, June 2018.

[11] W. Wang. Bayesian optimization concept explained in layman terms. *Medium*, March 2020.

[12] M. Naila. Conception et développement des méthodes de prédiction de la durée de séjour hospitalier centrées sur des techniques de machine learning. *Research Gate*, pp. 87, January 2022.

[13] S. Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *Arxiv*, pp. 3–4, May 2023.

[14] A. Tam. A gentle introduction to particle swarm optimization. *Machine Learning Mastery*, October 2021.

[15] S. Katoch et al. A review on genetic algorithm: past, present, and future. *Springer*, 80(8091–8126):4, February 2021.

[16] R. Agrawal. Know the best evaluation metrics for your regression model ! *Analytics Vidhya*, September 2024.

[17] S. Kumar. Metrics to evaluate your classification model to take the right decisions. *Analytics Vidhya*, June 2024.

[18] P. Antoniadis. Machine learning: What is ablation study? *Baeldung*, March 2024.

[19] A. Zheng. Evaluating machine learning models a beginner's guide to key concepts and pitfalls. *O'Reilly Media*, pp. 12–56, September 2015.

[20] A. Sharma. Is n = 30 really enough? a popular inductive fallacy among data analysts. *Medium*, November 2020.

[21] J. Siebert et al. Construction of a quality model for machine learning systems. *SpringerLink*, June 2021.

[22] F. Hutter et al. *Automated Machine Learning*. Springer Cham, May 2019.

[23] S. Studer et al. Towards crisp-ml(q): A machine learning process model with quality assurance methodology. *Mdpi - Machine Learning and Knowledge Extraction*, pp. 2–16, April 2021.

[24] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, NY, August 2006.

[25] L. Sumbati. Ensemble learning in machine learning: Bagging, boosting and stacking. *Medium*, March 2024.

[26] M. R. Hossain and D. Timmer. *Machine Learning Model Optimization with Hyper Parameter Tuning Approach*, volume 21. Global Journal, August 2021.

[27] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012.

[28] E. Hazan et al. Hyperparameter optimization: A spectral approach. *Research Gate*, pp. 2, June 2017.

[29] S. Andradóttir. *A Review of Random Search Methods*. Springer, New York, NY, September 2014.

[30] S. N. M.A. Amirabadi, M.H. Kahaei. *Novel suboptimal approaches for hyperparameter tuning of deep neural network [under the shelf of optical communication]*. Elsevier, April 2020.

[31] A. S. Li Yang. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Science Direct*, pp. 1–19, July 2020.

[32] S. Sun et al. A survey of optimization methods from a machine learning perspective, 2019.

[33] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd international conference on Machine learning - ICML*, 2006:2, June 2006.

[34] W. Zuo et al. On kernel difference-weighted k-nearest neighbor classification. *Pattern Analysis and Applications*, 8:247–257, 2008.

[35] L. Yang et al. Tree-based intelligent intrusion detection system in internet of vehicles. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 4, 2019.

[36] F. Abbas et al. Hyperparameter optimization for landslide susceptibility mapping: A comparison between baseline, bayesian and metaheuristic hyperparameter optimization techniques for machine learning algorithms. *Research Gate*, pp. 10, June 2023.

[37] P. Mehta et al. A high-bias, low-variance introduction to machine learning for physicists. *Physics Reports*, 810:1–124, 2019. A high-bias, low-variance introduction to Machine Learning for physicists.

[38] J. Chakraborty et al. Bias in machine learning software: Why? how? what to do? *Research Gate*, pp. 1–3, May 2021.

[39] J.-Y. Kim and S.-B. Cho. An information theoretic approach to reducing algorithmic bias for machine learning. *Neurocomputing*, 500:26–38, 2022.

[40] T. Fahse et al. *Managing Bias in Machine Learning Projects*, pp. 94–109. Research Gate, October 2021.

[41] D. Chicco. Ten quick tips for machine learning in computational biology. *Springer Nature Link*, pp. 1–15, December 2017.

[42] F. M. Palechor and A. D. la Hoz Manotas. Estimation of Obesity Levels Based On Eating Habits and Physical Condition . UCI Machine Learning Repository, 2019.

[43] I. Yeh et al. Blood Transfusion Service Center. UCI Machine Learning Repository, 2008.

[44] W. Street et al. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1993.

[45] L. Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[46] D. Harrison and D. L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5(1):81–102, 1978.

[47] V. Bali et al. Life expectancy: Prediction & analysis using ml, September 2021.

[48] A. Navlani. Sensitivity analysis in python, March 2022.

[49] V. K. Karia et al. Gensample: A genetic algorithm for oversampling in imbalanced datasets. *ArXiv*, 2019.

[50] S. Echessa. Log transforming target variables and enhancing tree ensembles, July 2024.

Ringraziamenti

Prima di tutto, vorrei ringraziare la mia relatrice e correlatore, per avermi insegnato in maniera egregia e avermi fatto appassionare alle loro materie.

Vorrei ringraziare soprattutto i miei genitori per aver sempre creduto in me, anche ogni singola volta in cui io non lo facevo, e avermi spronato a dare sempre il meglio di me.

Ringrazio i miei nonni per avermi sempre sostenuto e voluto bene da lontano, sin da quando sono piccolo, che mi hanno aiutato a crescere ed essere sempre positivo anche verso le difficoltà.

Ringrazio mia sorella Federica per avermi sempre sostenuto in questi anni anche se a molta distanza e sentendoci poco, ti sento sempre con me e dalla mia parte.

Ringrazio mio fratello Luca per avermi accompagnato in ogni giornata, brutta e bella (soprattutto le belle), per la compagnia nei buonissimi pasti, in tutti i pomeriggi e sere al cinema per staccare dallo studio e per tutto lo sport fatto insieme.

Ringrazio la mia ragazza Alessia, che nonostante la conosco da meno di un anno mi ha supportato (e sopportato) in una maniera smisurata, spontanea e dolce, spero che tutti i miei sforzi mi porteranno a una vita futura in cui ci sei tu.

Ringrazio la mia coinquilina nonché migliore amica Francesca, per le infinite chiacchere, gossip, discorsi profondi fatti in solo un anno e mezzo, per il suo infinito affetto nonostante le varie discussioni e litigate, anche se spesso per sciocchezze.

Ringrazio Fabio, per la sua infinita gentilezza, intelligenza, simpatia e maturità,

mi ha sempre fatto riflettere, ridere e ragionare su qualsiasi cosa, sia le cose stupide che le cose serie.

Ringrazio Aurora, per avermi sostenuto e aver creduto in me anche se a distanza e per essere sempre stata aperta al dialogo e al confronto, soprattutto sulle recensioni dei film/serie visti in comune.

Ringrazio Daniele, per avermi sempre spronato a dare il meglio, anche essendo severo, e per i discorsi profondi e impegnativi che abbiamo sempre avuto.

Ringrazio Riccardo ed Elisabetta, per aver creduto in me e avermi capito sempre da ogni punto di vista, per tutta la compagnia, la gentilezza, la simpatia e l'educazione che sono in pochi ad avere.

Ringrazio Maria e Sophia per avermi fatto passare delle bellissime serate a casa loro tra il trash, i giochi di società e il cibo buono.

Ringrazio Ruben e Davide per condividere le nostre passioni nerd e il mangiare insieme.

Ringrazio me stesso, per non aver mai smesso di lottare anche quando ero stanco e stressato, di sperare in un futuro migliore e di rialzarmi ogni volta che avevo problemi nella vita.

Un ultimo ringraziamento va alle aziende OpenAI e Anthropic, che mi hanno permesso di ridurre il tempo necessario per studiare e comprendere i concetti, facendomi scoprire e appassionare ancora di più a questo ambito.