UNIVERSITÀ DEGLI STUDI ROMA TRE
SCUOLA DI ECONOMIA E STUDI AZIENDALI
DIPARTIMENTO DI ECONOMIA

CORSO DI LAUREA TRIENNALE IN ECONOMIA E BIG DATA

SOLUZIONI DI PRODUCT PRICING USANDO ALGORITMI IBRIDI DI MACHINE LEARNING

Tesi di Laurea Triennale

Studente:
Daniele Banni

Docente:
Prof. Francesco Benedetto

ANNO ACCADEMICO 2023-2024

ABSTRACT

In recent decades, e-commerce platforms have gained increasing popolarity among buyers and sellers. More specifically, during the COVID-19 pandemic, researchers observed a growing trend regarding the opening of online shops and the transition from physical to digital businesses. The survey aims to discover tailored product pricing solutions directed to small businesses, empowered by Machine Learning algorithms. The major problem is connected to the instability of the prices of certain products, such as the fluctuating trends of seasonal clothing.

The study has been structured in two main parts. The first part, led by Anupama Namburu, Prabha Selvaraj, and M. Varsha, explores the performance of four algorithms (LightBoost, XGBoost, CatBoost, and X-NGBoost) following an initial preprocessing of the product sales dataset. The significance of their work lies in the development of a hybrid algorithm called X-NGBoost, enhanced by Artificial Intelligence, which proved to be the most suitable for pricing solutions.

The second part of the thesis concerns a personal study of the same dataset. During this study, three decision tree models (XGBoost, CatBoost, and Random Forest) were employed to provide optimal product pricing solutions. Crucial to this process was the optimization of these models through cross-validation algorithms, namely RandomSearch and GridSearch, which yielded good results in terms of MSE, RMSE and RMSE baseline, making the models appropriated for pricing solutions.

INDICE

INTRODUZIONE

Le piattaforme di *e-commerce* hanno sviluppato la loro popolarità tra compratori e venditori negli ultimi due decenni. Con la pandemia COVID-19 si è verificato un boom nell'apertura di negozi online, mostrando come molti venditori abbiano spostato il loro businesses verso piattaforme di *e-commerce*. Attualmente, formulare strategie di prezzo sta diventando sempre più complesso a causa del trend crescente di negozio online ed il numero di prodotti venduti in rete. Nello specifico, numerosi fattori influenzano i prezzi dei prodotti, portandoli ad essere altamente dinamici. Un esempio è la forte stagionalità dell'andamento dei prezzi dei vestiti, i quali mostrando pesanti fluttuazioni nei prezzi. Questo lavoro ha lo scopo di aiutare imprenditori e analisti di business nell'applicazione di prezzi competitivi, attraverso l'analisi delle vendite di prodotti simili su piattaforme di *e-commerce*. L'analisi consiste nell'utilizzo di algoritmi di Machine Learning con target specifico per rivenditori e imprenditori di piccola scala. L'algoritmo ibrido X-NGBoost, il quale combina l'Extreme Gradient Boost (XGBoost) con il Natural Gradient Boost (NGBoost), non fornisce solo una predizione di prezzo ma predice accuratamente la reazione dei consumatori alle sue variazioni, stimando la domanda di un dato prodotto. Il metodo proposto è messo in comparazione con i metodi di *ensemble* XGBoost, LightBoost e CatBoost, mostrando giusti suggerimenti di prezzo per migliaia di prodotti, rendendolo più profittevole e performante rispetto agli altri modelli.

SIGLE E ABBREVIAZIONI

- XGBoost = eXtreme Gradient Boosting
- LightGBM = Light Gradient-Boosting Machine
- CatBoost = Categorical Boosting
- X-NGBoost = eXtreme Natural Gradient-Boosting
- RSS = Residual Sum of Squares
- MSE = Mean Squared Error
- RMSE = Root Mean Squared Error
- GOOS = Gradient-based One-Side Sampling
- EFB = Exclusive Feature Bundling
- ML = Machine Learning

CAPITOLO 1
REVISIONE DELLA LETTERATURA

1.1 Analisi dello Studio e Scopo

Negli ultimi decenni le piattaforme di *e-commerce* hanno trasformato radicalmente lo stile di vita e il modo in cui consumatori e venditori si approcciano al mercato. Nello specifico, grazie alla comodità di poter effettuare acquisti online, senza dover obbligatoriamente uscire di casa, sta diventando sempre più complicato per i venditori offrire prodotti che possano competere con la concorrenza estera o di grandi imprese. Il problema sorge dal fatto che le piattaforme di *e-commerce* favoriscono una struttura di mercato *winner-take-all*, nella quale le imprese più grandi possono vendere prodotti a prezzi minori della concorrenza, conquistando quote di mercato nettamente maggiori al semplice quartiere o città di locazione dell'impresa. Il seguente studio effettua un'analisi su come e quali algoritmi di Machine Learning possano aiutare i piccoli commercianti a sviluppare soluzioni di *pricing* con lo scopo di ottimizzare i prezzi dei prodotti venduti online e renderli competitivi, offrendo sempre il prezzo giusto basato sullo studio dell'ambiente.

1.2 Studi Precedenti e Applicazioni Pratiche

Nello studio vengono citate, inoltre, varie pubblicazioni in materia, le quali forniscono ulteriori spunti di riflessioni per un'analisi dell'argomento più completa. Giocano un ruolo fondamentale per le finalità dello studio le analisi di mercato, marketing o studi più approfonditi sulle varie fasi del processo produttivo. Nello specifico, le pubblicazioni fanno riferimento a quattro argomenti fondamentali:

- **Studio del mercato**: in queste ricerche vengono analizzate le tendenze di mercato, i prodotti stagionali e i possibili volumi di vendite, il tutto messo in relazioni a come questi fattori possano influenzare il prezzo del bene finale;

- **Previsione del prezzo**: in queste pubblicazioni vengono approfonditi i fattori fondamentali che influenzano il prezzo ed i vari approcci di individuazione di quest'ultimo (approcci sui costi o sul valore);

- **Studio del marketing**: fondamentale per un prezzo giusto è l'analisi del valore che i consumatori attribuiscono ad un determinato bene. Nello specifico, attraverso lo studio comportamentale dei clienti è possibile effettuare campagne di marketing o di prezzo personalizzate, individuando target e segmenti di mercato;

- **Minimizzazione dei costi di produzione**: per massimizzare il profitto, gioca un ruolo centrale la minimizzazione dei costi nelle varie fasi del processo produttivo. Dunque, è fondamentale capire quali sono i fattori che influenzano la catena di approvvigionamento e come le risorse informatiche possano ottimizzarne la gestione.

Queste analisi e gli algoritmi di Machine Learning forniscono un enorme contributo ai rivenditori a prendere decisioni di prezzo in maniera rapida e più sicura, sulla base di dati storici, della concorrenza e di predizioni scalabili, ripetibili e facilmente interpretabili da quest'ultimi.

1.3 Risultati dello Studio

I risultati ottenuti mostrano come l'impiego dell'algoritmo X-NGBoost sia appropriato per fornire soluzioni di prezzo con il minor tasso di errore, rispetto agli altri algoritmi. Inoltre, l'implementazione di tecniche di *ensemble* risulta affidabile ed efficiente con una bassa percentuale di errore, fornendo agli utenti un risultato fruibile ed un maggior tasso di soddisfazione. In aggiunta, il modello potrebbe essere esteso attraverso l'integrazione di una piattaforma di *e-commerce* che permetta di avere a disposizione risultati dinamici, rendendo lo studio più accurato.

CAPITOLO 2
BACKGROUND TEORICO

2.1 Introduzione al Machine Learning

L'apprendimento automatico, o Machine Learning, è il campo dell'informatica che studia come fornire ai computer la capacità di imparare ad eseguire un task senza essere stati esplicitamente programmati per la sua esecuzione. Evolutosi dagli studi sul riconoscimento di *pattern* e sull'apprendimento computazionale teorico nel campo dell'intelligenza artificiale, il Machine Learning esplora lo studio e la costruzione di algoritmi che permettono l'apprendimento di informazioni a partire dai dati disponibili, fornendo la capacità di predire nuove informazioni alla luce di quelle apprese. Attraverso la costruzione di un modello che impara automaticamente a predire nuovi dati a partire dalle osservazioni, questi algoritmi superano il classico paradigma delle istruzioni strettamente statiche. Il Machine Learning trova il suo impiego principale in quell'insieme di problemi di computazione in cui la progettazione e l'implementazione di algoritmi *ad-hoc* non è praticabile o è poco conveniente. Questa disciplina presenta profondi legami col campo dell'ottimizzazione matematica, il quale fornisce metodi, teorie e domini di applicazione. Molti problemi di apprendimento automatico, infatti, sono formulati come problemi di minimizzazione di una certa funzione di perdita (*loss function*) nei confronti di un determinato set di esempi (*training set*). Questa funzione esprime la discrepanza tra i valori predetti dal modello in fase di addestramento e i valori attesi per ciascuna istanza di esempio. L'obiettivo finale è dunque quello di insegnare al modello la capacità di predire correttamente i valori attesi su un set di istanze non presenti nel set di addestramento (*test set*), mediante la minimizzazione della funzione di perdita in questo insieme di istanze. Una più formale ed ampiamente citata definizione per il Machine Learning è stata formulata da Tom M. Mitchell: "Si dice che un programma apprende dall'esperienza E con riferimento a alcune classi di compiti T e con misurazione della

performance P, se le sue performance nel compito T, come misurato da P, migliorano con l'esperienza E.” Gli algoritmi di Machine Learning possono essere classificati in diverse categorie principali:

1. **Apprendimento Supervisionato**: L'apprendimento supervisionato è una tipologia di ML in cui l'algoritmo viene addestrato utilizzando un insieme di dati etichettati. Ogni esempio del set di dati di addestramento è composto da un input e un output desiderato (etichetta). L'algoritmo apprende a mappare l'input all'output, cercando di minimizzare l'errore tra le previsioni del modello e gli output effettivi;

2. **Apprendimento Non Supervisionato**: L'apprendimento non supervisionato è una tipologia di ML in cui l'algoritmo viene addestrato utilizzando un insieme di dati non etichettati. L'obiettivo è quello di identificare strutture nascoste, *pattern* o raggruppamenti nei dati;

3. **Apprendimento Semi-Supervisionato (Semi-Supervised Learning)**: L'apprendimento semi-supervisionato è una tipologia di ML che utilizza una combinazione di dati etichettati e non etichettati per l'addestramento. Questo approccio viene spesso utilizzato quando si dispone di una piccola quantità di dati etichettati e una grande quantità di dati non etichettati. L'obiettivo è migliorare l'accuratezza del modello sfruttando le informazioni presenti nei dati non etichettati;

4. **Apprendimento per Rinforzo (Reinforcement Learning)**: L'apprendimento per rinforzo è una tipologia di ML in cui un agente interagisce con un ambiente per apprendere una politica di comportamento ottimale. L'agente prende decisioni e riceve ricompense o penalità in base alle sue azioni. L'obiettivo è massimizzare la ricompensa cumulativa nel tempo.

2.2 Gli Alberi Decisionali di Regressione

Un albero decisionale è un algoritmo di apprendimento supervisionato non parametrico utilizzato sia per attività di classificazione che di regressione. Esso è costituito da una struttura gerarchica caratterizzata da:

- **Nodo radice**: Nodo iniziale dell'albero che rappresenta l'intero set di dati;
- **Nodi interni**: Punti di suddivisione basati sui valori delle variabili predittive;
- **Nodi terminali (o foglie)**: Contengono le previsioni finali determinate dal modello;
- **Rami**: Collegano i vari nodi dell'albero.

Nello studio in questione sono stati utilizzati gli alberi di regressione che effettuano una segmentazione o stratificazione iterativa dello spazio dei predittori in regioni distinte, semplificando l'analisi e l'interpretazione delle relazioni tra le variabili indipendenti e la variabile dipendente.

2.2.1 Suddivisione dello Spazio dei Predittori

La suddivisione dello spazio dei predittori è un processo chiave nella costruzione di un albero di regressione, in questo primo step procedurale tutti i predittori (X_1, X_2, …, X_p) vengono suddivisi in J regioni distinte e non sovrapposte (R_1, R_2, …,R_j).
Successivamente, per ogni osservazione che ricade in una determinata regione R_j , si esegue la previsione di un'osservazione sulla base della media di tutti i valori presenti nella stessa regione. Per costruire le regioni si determina un criterio di *split*, ovvero un criterio per selezionare la migliore suddivisione di un nodo. Il criterio di *split* è un indice statistico che garantisce il miglioramento della partizione delle variabili, scegliendo di volta in volta la migliore partizioni tra tutte le possibili, fino ad arrivare ad un criterio di arresto impostato arbitrariamente. Alla base del modello è presente un algoritmo che procede iterativamente, suddividendo il dataset in regioni sempre più piccole, cercando di migliorare la precisione delle previsioni.

Ogni divisione è effettuata in base al criterio di *split*, andando a minimizzare la somma dei quadrati dei residui (*RSS: Residual Sum of Squares*):

$$RSS = \sum_{j=1}^{j} \sum_{i \in \mathbb{R}} \left(yi - \hat{y}_{R_J} \right)^2 \qquad (2.\ 1)$$

$\hat{y}_{R_J} = \frac{1}{n} \sum_{i \in \mathbb{R}} y_i$ è la media dei valori osservati nella regione $j - esima$ (R_j), j è il numero totale di regioni, mentre y_i è il valore osservato della variabile dipendente.
La minimizzazione dell'RSS garantisce che ogni divisione produca sottogruppi di dati più omogenei rispetto al valore della variabile dipendente. Tuttavia, il calcolo di tutte le possibili suddivisioni nelle regioni risulta dispendioso a livello computazionale, perciò si adottano approcci come la suddivisione binaria ricorsiva.

2.2.2 Suddivisione Binaria Ricorsiva

L'algoritmo di suddivisione binaria ricorsiva è un metodo '*Top-Down*' e '*Greedy*' utilizzato per costruire alberi di regressione. Questo metodo procede in maniera ricorsiva, scegliendo la miglior suddivisione possibile in ogni fase, partendo dall'insieme generale delle osservazioni fino a determinare insiemi più piccoli (*Top-Down*). Tutte le divisioni vengono eseguite valutando la miglior partizione in ogni passo iterativo, senza considerare le implicazioni future (*Greedy*). Questo approccio implica i seguenti passi:

a. Scelta del Predittore e del Punto di Suddivisione:

Per ogni variabile predittiva $\boldsymbol{X_p}$ e per ogni possibile valore di suddivisione $\boldsymbol{S}$, si calcola la somma dei quadrati dei residui (RSS), risultante dalla divisione dell'insieme dei dati in due sottogruppi:

$$RSS_{(j,s)} = \sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1(j,s)})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2(j,s)})^2 \qquad (2.\ 2)$$

dove $R_1(j,s)$ e $R_2(j,s)$ sono le regioni definite dalla suddivisione del predittore $\boldsymbol{X_j}$ al valore di soglia $\boldsymbol{S}$:

$$R_1(j,s) = \{X|X_j < s\} \ e \ R_2(j,s) = \{X|X_j \geq s\} \qquad (2.3)$$

b. Scelta del Miglior Split:

L'algoritmo sceglie la miglior coppia di semipiani (j^*, s^*) che va a minimizzare la (2.2), ovvero l'RSS per le due regioni:

$$(j^*, s^*) = \min_{j,s} RSS_{(j,s)} \qquad (2.4)$$

c. Suddivisione del Dataset:

Dopo aver determinato la coppia di semipiani, l'algoritmo procede con la partizione dello spazio dei predittori nelle regioni:

$$R_1(j^*, s^*) \ e \ R_2(j^*, s^*) \qquad (2.5)$$

d. Ricorsione:

Il processo si ripete in modo ricorsivo secondo i passaggi descritti (1-3) per ciascuna delle due regioni finché non viene soddisfatto un criterio di arresto preimpostato arbitrariamente.

2.2.3 I Criteri di Arresto

I criteri di arresto sono fondamentali per evitare che l'albero di regressione diventi eccessivamente complesso e vada in *overfitting* (sovradattamento ai dati di addestramento). Alcuni dei criteri di arresto più utilizzati, includono:

- **Profondità massima dell'albero**: Si impone un limite alla profondità dell'albero, cioè un numero massimo di nodi che possono essere attraversati dalla radice a una foglia;

- **Numero minimo di osservazioni per nodo**: Ogni nodo deve contenere un numero minimo di osservazioni per poter essere suddiviso ulteriormente;

- **Riduzione minima dell'RSS**: La suddivisione di un nodo deve produrre una riduzione dell'RSS che superi una certa soglia predefinita.

Questi criteri, seppur abbiano la finalità di evitare l'*overfitting*, tendono a generare alberi non ottimali per eseguire previsioni accurate. Tuttavia, esistono tecniche ausiliarie per l'individuazione di modelli migliori, come ad esempio la potatura dell'albero (*pruning*).

2.2.4 La Potatura degli Alberi

La potatura degli alberi è una tecnica essenziale per ottimizzare le prestazioni dei modelli di alberi decisionali. La tecnica "*Cost Complexity Pruning*", nota anche come "*Weakest Link Pruning*" (potatura del ramo più debole), è ampiamente utilizzata per bilanciare la complessità dell'albero con la sua elevata tendenza all'*overfitting*. Questa tecnica consiste nell'analisi di molteplici sottoalberi indicizzati da un parametro di Tuning (α) non negativo. Per ogni valore di α esiste un sotto-albero $T \subset T_0$ che minimizza la seguente formula:

$$R_\alpha(T) = \sum_{m=1}^{|T|} \sum_{\text{i}:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|\text{T}| \qquad (2.\ 6)$$

|T| = Numero di nodi terminali (foglie) dell'albero T.

R_m: Rettangolo (il sottoinsieme dello spazio dei predittori) corrispondente al nodo terminale.

$\hat{y}_{R_m}$: Risposta prevista associata a R_m, cioè la media delle osservazioni di addestramento in R_m.
α: Parametro di Tuning che controlla il *trade-off* tra la complessità del sotto-albero e il suo adattamento ai dati di addestramento.

- Quando α = 0, il sotto-albero T sarà uguale a T_0, ossia all'albero non potato. In questo caso la (2.6) misura solo l'errore di addestramento;
- All'aumentare di α, si aggiunge una penalità per gli alberi con molti nodi terminali, quindi la quantità (2.6) tenderà ad essere minimizzata per sotto-alberi più piccoli.

Lo scopo di questa tecnica consiste nel minimizzare la funzione che tiene conto sia dell'errore di previsione (2.1) che della complessità dell'albero. La potatura viene svolta in modo iterativo, eliminando i rami meno significativi nell'ottimizzazione dell'errore di previsione, calcolato attraverso l'espressione (2.6).
Per individuare il valore ottimale del parametro α vengono utilizzate tecniche di validazione incrociata come la *K-fold Cross-Validation*.

2.2.5 K-fold Cross-Validation

La procedura di validazione incrociata è una tecnica di ricampionamento (*resampling method*) dei dati, in particolare la tecnica *K-fold Cross-Validation* suddivide omogeneamente il dataset in K sottoinsiemi (*fold*). L'albero viene addestrato K volte, ogniuna su $K-1$ sottoinsiemi per l'addestramento, mentre uno viene utilizzato per la validazione. Questo processo viene ripetuto finché il modello non utilizza tutti i K-fold esistenti come *validation set*.
Il metodo stima un singolo e robusto errore quadratico medio di test (MSE_{test}), calcolato con la media di tutti gli MSE_{test} prodotti per ogni K-fold, come segue:

$$MSE_{(K-fold)} = \frac{1}{K}\sum_{i=1}^{K} MSE_i \qquad (2.7)$$

Nella tecnica della potatura dell'albero, la *K-fold Cross-Validation* viene utilizzata per determinare l'errore quadratico medio di test minore, in funzione del parametro di Tuning (α), individuando così il numero ottimale di nodi terminali di un sotto-albero (T). Quest'ultimo sarà l'albero "potato" con un buon equilibrio tra tendenza all'*overfitting* ed abilità di generalizzazione su nuovi dati.

CAPITOLO 3
METODOLOGIA

3.1 Descrizione del Dataset

Il dataset utilizzato contiene diverse caratteristiche dei prodotti venduti su piattaforme di *e-commerce*. È suddiviso in due parti:

- **Training set**: Contiene 2453 osservazioni e viene utilizzato per addestrare il modello. I dettagli delle caratteristiche del *training set* sono illustrati nella Tabella 1.

Tabella 1: Tabella del training set

S.no.	Product	Product_Brand	Item_Category	subcategory_1	subcategory_2	Item_Rating	Date	Sellng_Price
0	P-2610	B-659	Bags wallets belts	Bags	Handbags	4.3	2/312017	291.0
1	P-2453	B-3078	Clothing	Women's clothing	Western wear	3.1	7/1/2015	897.0
2	P-6802	B-1810	Home décor festive needs	Show pieces	ethnic	3.5	1/12/2019	792.0
3	P-4452	B-3078	Beauty and personal care	Eye care	h2opluseyecare	4.0	12/12/2014	837..0
4	P-8454	B-3078	Clothing	Men's clothing	Tshirts	4.3	12/1212013	470.0

Fonte: Product pricing solutions using hybrid machine learning algorithm

- **Test set**: Contiene 1052 osservazioni e viene utilizzato per valutare le prestazioni del modello addestrato. I dettagli delle caratteristiche del *test set* sono illustrati nella Tabella 2.

Tabella 2: Tabella del test set

S. no.	Product	Product_Brand	Item_Category	Subcategory_1	Subcategory_2	Item_Rating	Date
0	P-11284	B-2984	Computers	Network components	Routers	4.3	1/12/2018
1	P-6580	B-1732	Jewellery	Bangles Bracelets armlets	Bracelets	3.0	20/12/2012
2	P-5843	B-3078	Clothing	Women's clothing	Western wear	1.5	1/12/2014
3	P-5334	B-1421	Jewellery	Necklaces chains	Necklaces	3.9	1/12/2019
4	P-5586	B-3078	Clothing	Women's clothing	Western wear	1.4	1/12/2017

Fonte: Product pricing solutions using hybrid machine learning algorithm

Il dataset presenta otto variabili:

- **Product**: Nome del prodotto;
- **Product_brand**: Logo del prodotto;
- **Item_Category**: La più ampia categoria di appartenenza del prodotto;
- **Subcategory_1**: La sottocategoria a cui il prodotto appartiene (un livello più profondo);
- **Subcategory_2**: La specifica categoria di appartenenza del prodotto (due livelli più profondo);
- **Item_Rating**: La valutazione rilasciata dai compratori del prodotto;
- **Date**: Data di vendita del prodotto;
- **Selling_Price**: Prezzo di vendita del prodotto.

3.2 Preprocessing dei Dati

La fase di *preprocessing* dei dati è iniziata con una prima analisi esplorativa. Inserendo la variabile 'Selling Price' come variabile target, è possibile notare come essa abbia una distribuzione sbilanciata a sinistra, Figura 1. Normalizzata in seguito attraverso una trasformazione logaritmica, Figura 2.
Successivamente sono state aggiunte nuove colonne per la gestione delle features categoriche e temporali.

- La variabile temporale 'Date' è stata eliminata, aggiungendo al suo posto sette nuove colonne ('Month', 'Day', 'Day of Year', 'Week', 'Quarter', 'Is_month_start', 'Is_month_end') che ne migliorano la gestione e la manipolazione.

- Le variabili categoriche 'Item_Category', 'subcategory_1' e 'subcategory_2' sono state gestite attraverso la creazione di tre nuove colonne ('Unique_Item_category_per_product_brand', 'Unique_Subcategory_1_product_brand' e 'Unique_Subcategory_2_product_brand') le quali forniscono un valore unico

per ciascuna categoria e sottocategoria dei prodotti, migliorandone l'analisi.

3.3 Panoramica dei Modelli Utilizzati

Lo studio utilizza i modelli LightBoost, XGBoost, CatBoost e X-NGBoost per proporre soluzioni di pricing su piattaforme di *e-commerce*. Queste tecniche di *ensemble learning* forniscono soluzioni sistematiche per unire le capacità predittive di più modelli, poiché, affidarsi ad un unico risultato di un modello di Machine Learning potrebbe non essere sufficiente.

3.3.1 LightBoost

Il LightGBM è una struttura *open-source* di *Gradient Boosting*, conosciuto per la sua efficienza con dataset molto grandi. Esso è stato sviluppato dal team di Microsoft con a capo Guolin Ke ed introdotto nel 2017 in un documento intitolato "LightGBM: Un albero decisionale con Gradient Boosting altamente efficiente". Il modello è stato sviluppato per migliorare la velocità di addestramento e le performance predittive negli algoritmi di *Gradient Boosting*. Le innovazioni chiavi: *Gradient-based One-Side Sampling* (GOSS) ed *Exclusive Feature Bundling* (EFB) mirano ad istanze di gradienti più grandi per performare meglio nella selezione automatica delle *features*, migliorando il processo di *boosting*. Questo algoritmo accelera l'addestramento mantenendo o addirittura migliorando l'accuratezza predittiva, rendendolo ideale per gestire dati tabulari estesi per operazioni di regressione e classificazione.
Il LightGBM implementa una collezione di alberi decisionali per formare un modello che fornisca forti predizioni. L'enfatizzazione sull'efficienza è evidente grazie al suo approccio di crescita fogliare e l'uso di algoritmi basati su istogrammi. La costruzione degli alberi avviene attraverso la selezione dei nodi con la riduzione di *loss* maggiore, permettendo di avere alberi più profondi e migliorarne l'accuratezza, essendo tuttavia suscettibile a potenziali *overfitting* in caso di datasets piccoli.

Il metodo basato sugli istogrammi permette di discretizzare le *features* continue all'interno dell'istogramma, riducendo la complessità computazionale e l'uso di memoria, rendendolo particolarmente vantaggioso per gestire datasets di larga scala.

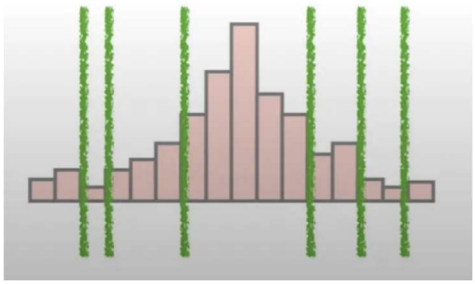
Tutte queste tecniche vengono combinate a capacità computazionali distribuite e parallele, fornendo addestramenti rapidi, con bassi consumi di memoria, alta scalabilità e rendendo l'algoritmo applicabile per diversi compiti di Machine Learning.

3.3.2 XGBoost

XGBoost, acronimo *di eXtreme Gradient Boosting*, è un'implementazione avanzata dell'algoritmo di *Gradient Boosting* progettata per essere altamente efficiente, flessibile e portabile. Grazie alle sue caratteristiche, XGBoost è diventato estremamente popolare nel campo della Data Science, in particolare per l'analisi di dati strutturati e tabellari. Questo algoritmo non solo offre prestazioni eccezionali ma è anche ottimizzato per lavorare con dataset di grandi dimensioni, risultando spesso vincitore in numerose competizioni di Machine Learning.
L'algoritmo costruisce un modello additivo dove ogni nuovo albero correggere gli errori commessi dai modelli precedenti, applicando ulteriori ottimizzazioni rispetto all'algoritmo originario di *Gradient Boosting,* migliorando l'efficienza computazionale e la capacità predittiva. Le caratteristiche e ottimizzazioni principali di tale algoritmo sono:

- **Algoritmo Greedy Approssimato**: l'XGBoost utilizza un algoritmo *greedy* per la costruzione degli alberi, esso si basa sulla divisione dei nodi in base ai valori di *gain*, senza considerare come tali divisioni influenzeranno le future. Nello specifico, l'algoritmo divide i valori delle *features* in quartili ed attraverso l'addestramento in parallelo e l'algoritmo di *sketching*, tecnica utilizzata per approssimare la distribuzione dei dati, vengono pesati i quartili delle *features* (Figura 1). I quartili non avranno necessariamente lo stesso numero di campioni poiché sono pesati:

Figura 1: Quartili pesati nella costruzione dell'albero decisionale



Fonte: Medium. "XGBoost — How does this work"

- **Gestione dei valori mancanti**: quando l'algoritmo incontra i valori mancanti durante il processo di costruzione degli alberi ed effettua una suddivisione basata sul loro impatto per il risultato predittivo. Questo approccio permette al modello di sfruttare le informazioni inerenti alla mancanza di dati, migliorando le prestazioni predittive rispetto ad algoritmi che imputano i dati mancanti solo con valori fissi;

- **Controllo accesso alle cache**: il XGBoost implementa varie tecniche per controllare l'accesso alle cache:

    - **Blocchi di Compressione**: I dati vengono suddivisi in blocchi e compressi, riducendo il consumo di memoria e migliorando la velocità di accesso durante l'addestramento;
    - **Accesso a Colonne per Split**: Durante la costruzione dell'albero, XGBoost accede ai dati in un formato colonna-centrico (*column-major order*), migliorando l'efficienza del calcolo degli *split*;
    - **Cache dei Gradienti**: XGBoost mantiene in cache i gradienti e gli hessiani (secondi derivati) delle funzioni obiettivo, riducendo il numero di calcoli necessari durante la costruzione degli alberi;
    - **Pre-fetching**: Tecniche di *pre-fetching* vengono utilizzate
      e per caricare anticipatamente i dati necessari, minimizzando i tempi di attesa e aumentando l'efficienza complessiva.

$$O^{(t)} = \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + p_i f_i(x_i) + \frac{1}{2} q_i f_t^2(x_i)] + \Omega(f_t) \qquad (3.1)$$

$i = i - esimo$ campione.

t: Rappresenta l'iterazione.

$y_i^{(t-1)}$: è il valore predetto dall'iterazione $(t-1)$.

p_i *e* q_i: Sono la derivata prima e seconda.

$\Omega(f_t)$: Rappresenta il termine di regolarizzazione.

3.3.3 CatBoost

Il CatBoost (*Categorical Boosting*) è una struttura *open-source* di *Gradient Boosting* ad alte performance, sviluppato da Yandex. Esso è stato progettato per risolvere un vasto largo di problemi di Machine Learning (regressione, classificazione, ranking) con particolare enfasi per la gestione efficiente di *features* categoriche, rendendolo popolare per l'analisi di datasets reali. Il CatBoost spicca per la sua velocità, accuratezza e semplicità di utilizzo nell'affrontare dati strutturati.

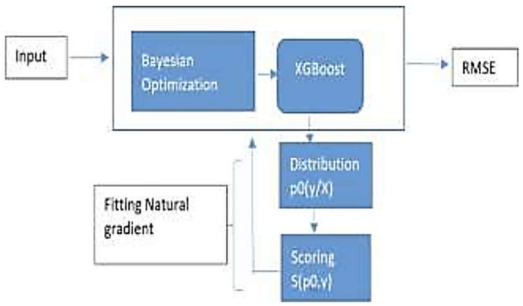
Questo algoritmo utilizza un metodo chiamato "*ordered boosting*" per processare i dati categorici senza trasformazioni o codifiche, avendo come risultato un addestramento più veloce e un modello più performante, nel quale vengono incorporate tecniche di regolarizzazione per evitare l'*overfitting*. Fondamentale, per il funzionamento dell'algoritmo è la sua grande efficienza nel ranking delle *features*, rendendo la selezione di quest'ultime e la scelta dei modelli più semplice.

3.3.4 X-NGBoost

Lo scopo principale dello studio è lo sviluppo di un modello innovativo basato sull'intelligenza artificiale che ci aiuti nella predizione delle strategie di prezzo. Il modello proposto consolida l'algoritmo XGBoost ed il modello NG-Boost, avendo come risultato una metodologia innovativa rinominata X-NGBoost. La struttura effettua un *preprocessing* dei dati utilizzando il XGBoost, il quale è stato modificato con un algoritmo di predizione probabilista naturale. L'addestramento preliminare dei dati avviene con il modello XGBoost.

I suoi iperparametri sono scelti attraverso prove ed errori, ed ottimizzati in seguito tramite l'ottimizzazione Bayesiana. Il seguente modello di predizione ottimizzata riesce a elevare significativamente l'accuratezza, migliorando così le performance del sistema di predizione dei prezzi.

Figura 2: Funzionamento schematico dell'algoritmo X-NGBoost



Fonte: Product pricing solutions using hybrid machine learning algorithm

3.4 Comparazione dei Modelli

I modelli di *ensemble* LightGBM, XGBoost e CatBoost forniscono appropriate soluzione per le strategie di prezzo. Il loro vantaggio proviene dalla stima dell'importanza delle *features* tramite predizione. Nello specifico, il punteggio delle *features* viene utilizzato nella fase di *boost* degli alberi decisionali, all'interno dei modelli, per migliorare il risultato delle predizioni.

Dai risultati della Tabella 3, è possibile concludere che, per il seguente data set, l'algoritmo X-NGBoost fornisce le soluzioni di prezzo più appropriate, deducibile dagli $RMSE_{training}$ e $RMSE_{test}$ minori.

Tabella 3: Comparazione dei risultati dei modelli

Model	RMSE training	RMSE testing
XGBoost	6.62	7.48
LightGBM	6.62	7.44
CatBoost	5.91	6.96
X-NGBoost	4.23	5.34

Fonte: Product pricing solutions using hybrid machine learning algorithm

Tabella 4: Comparazione trai modelli

Function	XGBoost	LightGBM	CatBoost
Splits	Non usa nessuna tecnica di campioni pesati, rendendo il suo processo di divisione più lento del GOSS e MVS	Fornisce il GOSS, il quale utilizza istanze con gradienti più grandi e campioni casuali con piccoli gradienti per selezionare la segmentazione	Forsnisce una nuova tecnica chiamata Minimal Variance Sampling (MVS), dove il campionamento pesatoa si verifica al livello dell'albero rispetto che al livello di split
Valori mancanti	I valori mancanti vengono allocati al lato che riduce la loss per ogni divisione	I valori mancanti vengono allocati al lato che riduce la loss per ogni divisione	Ha le modalità 'Min' e 'Max' per processare i valori mancanti
Crescita fogliare	Esso divide per la specificata max_depth e, in seguito, inizia a potare l'albero (backwards) attraverso l'eliminazione delle divisioni che non hanno un gain positivo,poiché divisioni senza riduzione della loss protrebbe essere seguite da divisioni con perdita di loss	Utilizza la miglior prima crescita dell'albero, poiché seleziona le foglie che minimizzano la crescita della loss, permettendo tuttavia la crescita di alberi sbilanciati con overfitting in caso di dataset piccoli	Costruisce un albero bilanciato ad ogni livello dell'albero, la coppia di segmentazione dell'albero che produce la loss minore è selezionata ed usata per tutti i nodi di quel livello
Velocità d'addestramento	Più lento di Catboost e LightGBM	Più veloce di CatBoost e XGBoost	Più veloce del XGBoost
Gestione delle features categoriche	Non ha un metodo al suo interno per la classificazione delle features, l'utente deve codificarle	Il metodo ordina le categorie secondo l'obiettivo di training. Categorical_features: specifica le features da considerare durante l'addestramento del modello	Combina il one-hot encoding e l'advanced media encoding. One_hot_max_size: esegue un encoding attivo per tutte le funzione con differenti valori multipli.
Parametri per controllare l'overfitting	Learning rate - Maximum depth - Minimum child weight	Learning rate - Manimum depth - Number of leavers - Minimum data in leaf	Learning rate - Depth - L2 leaf reg

Fonte: Product pricing solutions using hybrid machine learning algorithm

CAPITOLO 4
IMPLEMENTAZIONE

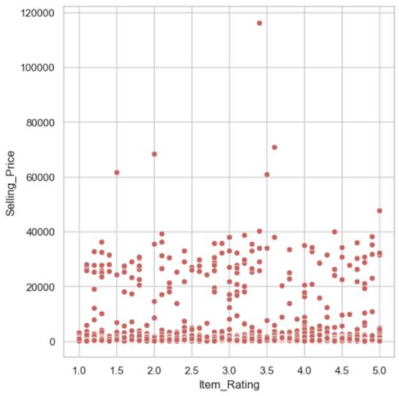
4.1 Ambiente di Sviluppo (Python, librerie utilizzate)

Python si rivela uno strumento potente e versatile per l'analisi dei dati e la costruzione di modelli di Machine Learning. In questo progetto vengono utilizzate diverse librerie di Python che ne facilitano l'intero processo, dalla manipolazione alla modellazione dei dati, fino alla valutazione delle prestazioni dei modelli. Pandas viene utilizzato per la gestione e l'elaborazione dei dati, fornendo strutture dati efficienti, come i DataFrame. NumPy supporta operazioni matematiche su array multidimensionali, mentre Matplotlib e Seaborn permettono di creare visualizzazioni grafiche per esplorare i dati e interpretare i risultati. Infine, la libreria Scikit-learn fornisce una vasta gamma di strumenti per l'analisi dei dati nel Machine Learning, dal preprocessing dei dati all'implementazione dei modelli, la loro ottimizzazione e la rappresentazione grafica dei risultati.

4.2 Preprocessing dei Dati

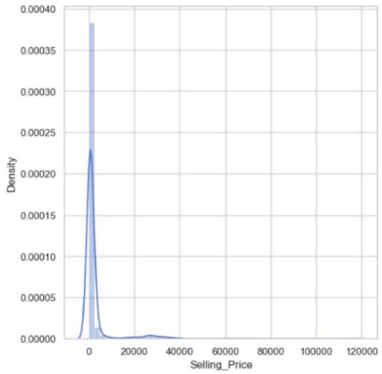
Nella fase di preprocessing, si preparano i dati per facilitarne la gestibilità in vista delle fasi successive di analisi. Innanzitutto, si importano i dataset di training e test scaricati da Kaggle. In seguito, si effettua una piccola analisi esplorativa del dataset per le variabili 'Selling_Price' ed 'Item_Rating' in relazione col prezzo di vendita (Figura 3), notando una distribuzione sbilanciata a sinistra per 'Selling_Price'. (Figura 4)

Figura 3: Scatterplot delle variabili 'Selling_Price' e 'Item_Rating'



Fonte: Analisi personale del dataset

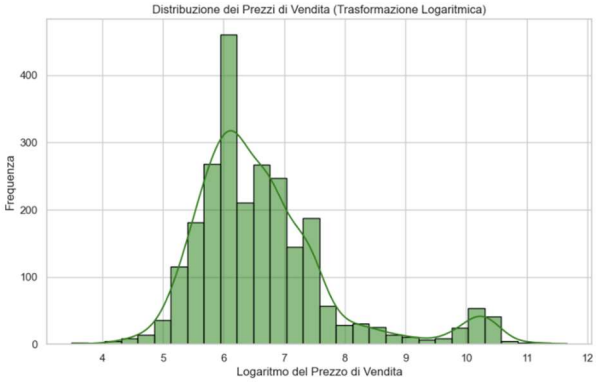
Figura 4: Distribuzione della variabile 'Selling_Price'



Fonte: Analisi personale del dataset

Per semplificare l'analisi, la variabile 'Selling_Price' è stata trasformata in scala logaritmica per normalizzarne la distribuzione. (Figura 5)

Figura 5: Distribuzione di 'Selling_Price' in scala logaritmica



Fonte: Analisi personale del dataset

Successivamente, sono state trasformate le variabili ‘Date’, ‘Item_Category’, ‘Subcategory_1’ e ‘Subcategory_2’, seguendo l’esempio dello studio precedente.

- **Date**: è stata trasformata aggiungendo le variabili (‘Month’, ‘Day’, ‘Day of Year’, ‘Week’, ‘Quarter’, ‘Is_month_start’, ‘Is_month_end’);

- **Item_Category, Subcategory_1, Subcategory_2**: sono state trasformate tramite codifica *one-shot*, aggiungendo le variabili (‘Unique_Item_category_per_product_brand’, ‘Unique_Subcategory_1_product_brand’ e ‘Unique_Subcategory_2_product_brand’).

4.3 Implementazione dei Modelli

Come primo passo per l’implementazione dei modelli, sono state definite le *features* per entrambi i dataset e la variabile target:

- **Features**: 'Item_Rating', 'Month', 'Day', 'DayofYear', 'Is_month_start', 'Is_month_end','Unique_Item_category_per_product_brand', 'Unique_Subcategory_1_product_brand', 'Unique_Subcategory_2_product_brand';

- **Variabile target**: ‘Selling_Price’.

4.3.1 Random Forest Regressor

Per la definizione dell’algoritmo Random Forest sono state utilizzare le seguenti metriche:

- **Max_depth**: indica la profondità massima degli alberi nella foresta, nello specifico il valore ‘20’ rappresenta che la lunghezza del percorso più lungo dalla radice a una foglia è di venti nodi;
- **Max_features**: indica il numero massimo di caratteristiche da considerare durante la ricerca della miglior divisione, ‘sqrt’ significa che per ogni divisione il modello

considererà un numero massimo di caratteristiche pari alla radice quadrata del numero totale di caratteristiche;

- **Min_samples_leaf**: indica il numero minimo di campioni che un nodo foglia deve contenere, nel nostro caso '3';
- **Min_samples_split**: indica il numero minimo di campioni richiesti per dividere un nodo interno, in questo caso '2';
- **N_estimators**: indica il numero di alberi nella foresta, nel nostro caso verranno costruiti '278' alberi.

4.3.2 XGBoost Regressor

Per la definizione dell'algoritmo XGBoost sono state utilizzare le seguenti metriche:

- **Colsample_bytree**: indica la frazione di caratteristiche da considerare per ogni albero, con un valore di '0.6' il modello considera il 60% delle caratteristiche per costruire ogni albero;
- **Learning_rate**: Il passo di apprendimento utilizzato per aggiornare i pesi di ogni albero. Valori più bassi rendono l'addestramento più lento ma possono portare a una migliore generalizzazione, in questo caso avremo un valore di '0.01' in cui ogni albero contribuisce per l'1% alle modifiche dei pesi;
- **Max_depth**: in questo caso avremo una profondità di '7';
- **Subsample**: indica la frazione di campioni utilizzati per costruire ogni albero, con un valore di '0.8' ogni albero sarà addestrato su un campione causale contenente l'80% dei dati di training;
- **N_estimators**: avremo un modello che costruirà '300' alberi.

4.3.3 CatBoost Regressor

Per la definizione dell'algoritmo CatBoost sono state utilizzare le seguenti metriche:

- **Colsample_bytree**: con il valore '0.8', il modello prenderà in considerazione l'80% delle caratteristiche per costruire l'albero;
- **Learning_rate**: avremo un valore di '0.1';

- **Max_depth**: in questo caso avremo una profondità di ‘6’;
- **Subsample**: con un valore di ‘0.6’, il modello verrà addestrato su un campione causale con il 60% dei campioni;
- **N_estimators**: avremo un modello che costruirà ‘300’ alberi.

4.4 Ottimizzazione

Nel contesto dell'Intelligenza Artificiale e del Machine Learning, un iperparametro si riferisce a un parametro o a un'impostazione di configurazione che determina la struttura generale e il comportamento di un modello di ML prima dell'inizio del processo di addestramento. Gli iperparametri sono cruciali nell'influenzare le prestazioni e l'efficacia di un modello, controllandone la capacità di apprendere, generalizzare ed evitare l'adattamento eccessivo o insufficiente ai dati di addestramento.

4.4.1 GridSearchCV

La ricerca dei migliori iperparametri e l’ottimizzazione dei modelli XGBoost e CatBoost, è stato svolta attraverso l’algoritmo di *Cross-Validation* GridSearchCV. Sono state effettuate due prove con cinque e dieci *folds* per la *Cross-Validation*, ottenendo risultati leggermente migliori con 10 *folds* (gli iperparametri sopra inseriti fanno riferimento a 10 *folds*).

4.4.1 RandomizedSearchCV

La ricerca dei migliori iperparametri e l’ottimizzazione del modello Random Forest, è stato svolta attraverso l’algoritmo di *Cross-Validation* RandomizedSearchCV. Sono state effettuate due prove con cinque e dieci *folds* per la *Cross-Validation*, ottenendo risultati leggermente migliori con 10 *folds* (gli iperparametri sopra inseriti fanno riferimento a 10 *folds*).

CAPITOLO 5
ANALISI DEI RISULTATI

5.1 Metriche di Valutazione

La valutazione di un modello di Machine Learning è fondamentale per determinare se il nostro modello riuscirà a predire correttamente la destinazione o il valore di dati nuovi e futuri. Poiché le istanze future hanno valori di destinazione ignoti, è necessario verificare il parametro di accuratezza del modello su dati dei quali si conosce già la risposta e utilizzare questa valutazione come *proxy* per la precisione predittiva sui dati futuri. Nello specifico per la valutazione dei nostri modelli sono state utilizzate tre statistiche principali: MSE, RMSE e la RMSE baseline.

5.1.1 Mean Squared Error (MSE)

Il *Mean Squared Error* (MSE) è una metrica statistica utilizzata per misurare la qualità di un modello di regressione. Il MSE rappresenta la media degli errori al quadrato tra i valori effettivi e quelli previsti dal modello. (5.1)

$$MSE = \frac{1}{n}\sum_{I=1}^{n}(y_i - \hat{y}_i)^2 \qquad (5.1)$$

5.1.2 Root Mean Squared Error (RMSE)

La *Root Mean Squared Error* (RMSE) è una metrica che misura la differenza tra i valori previsti dal modello e i valori effettivi. Viene calcolata attraverso la radice quadrata della media degli errori quadratici. (5.2)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \qquad (5.\ 2)$$

5.1.3 RMSE baseline

La *Root Mean Squared Error* (RMSE) *baseline* è una misura di errore utilizzata come punto di riferimento per valutare le prestazioni di un modello di regressione. La RMSE baseline viene calcolata utilizzando un modello molto semplice, di solito un modello che predice sempre la media del valore target nel set di training. Questa metrica fornisce un supporto a capire come il modello addestrato si comporta rispetto a una previsione semplice.

$$\bar{y} = \frac{1}{n}\sum_{i=0}^{n} y_i \qquad (5.\ 3)$$

$$\hat{y}_i = \bar{y}\ per\ tutte\ le\ i \qquad (5.\ 4)$$

$$MSE_{baseline} = \frac{1}{m}\sum_{j=1}^{m}(y_j - \bar{y})^2 \qquad (5.\ 5)$$

$$RMSE_{baseline} = \sqrt{MSE_{baseline}} \qquad (5.\ 6)$$

5.2 Confronto e Discussioni sui Risultati tra i Modelli

L'algoritmo XGBoost ha mostrato le performance migliori in termini di MSE e RMSE rispetto a CatBoost e Random Forest, sia con 5-folds che 10-folds di *Cross-Validation*. Tuttavia, l'algoritmo Random Forest presenta la deviazione standard per MSE e RMSE minore, suggerendo maggior stabilità.

Tutti i modelli hanno performance nettamente migliori rispetto al RMSE baseline, dimostrandosi efficaci nelle previsioni rispetto alla semplice media dei target.

Tabella 3: Risultati delle metriche

5 Folds	XGBoost	CatBoost	RandomForest
MSE	0.47790	0.48455	0.49032
SD MSE	0.05922	0.05778	0.05040
RMSE	0.69001	0.69486	0.69932
SD RMSE	0.04227	0.04143	0.03577
RMSE base	1.16416		

10 Folds	XGBoost	CatBoost	RandomForest
MSE	0.47558	0.48538	0.49090
SD MSE	0.05669	0.05619	0.05085
RMSE	0.68840	0.69554	0.69971
SD RMSE	0.04097	0.03999	0.03607
RMSE base	1.16416		

Fonte: Analisi personale del dataset

BIBLIOGRAFIA

1. *Prathamesh Sonawane, "Medium: XGBoost — How does this work", Articolo di ricerca, 2023. https://medium.com/@prathameshsonawane/xgboost-how-does-this-work-e1cae7c5b6cb*

2. *Wei Wang, Medium:" Bayesian Optimization Concept Explained in Layman Terms", Articolo di ricerca, 2020. https://towardsdatascience.com/bayesian-optimization-concept-explained-in-layman-terms-1d2bcdeaf12f*

3. *M. Mohtasim Hossain, Medium:" Mastering LightGBM: An In-Depth Guide to Efficient Gradient Boosting", Articolo di ricerca, 2024. https://medium.com/@mohtasim.hossain2000/mastering-lightgbm-an-in-depth-guide-to-efficient-gradient-boosting-8bfeff15ee17*

4. *Bnova, "Algoritmi di Machine Learning: come funzionano e quali sono", Articolo di ricerca, 2023. https://www.bnova.it/intelligenza-artificiale/algoritmi-machine-learning/*

5. *Omar Burzio "Machine Learning per la qualità nell'industria 4.0", Tesi di Laurea Magistrale, POLITECNICO DI TORINO, 2022. https://webthesis.biblio.polito.it/23576/1/tesi.pdf*

6. *shilpamahq2o, "Regression using CatBoost.", Articolo di ricerca, 2024. https://www.geeksforgeeks.org/regression-using-catboost/*

7. *Md Sumon Gazi, Md Rokibul hasan, Nisha Gurung and Anik Mitra, "Ethical Considerations in AI-driven Dynamic Pricing in the USA: Balancing Profit Maximization with Consumer Fairness and Transparency", Articolo di ricerca, 2024. https://al-kindipublisher.com/index.php/jefas/article/view/7099/5950*

8. *AppMaster, "Iperparametro", **Articolo di ricerca, 2023.** **https://appmaster.io/it/glossary/iperparametro***

9. *Emiliano Simonelli, "ANALISI DEL MERCATO AIRBNB DI ROMA: PREVISIONI DI PREZZO ATTRAVERSO MODELLI DI APPRENDIMENTO AUTOMATICO", Tesi triennale, 2024.*

10. *Robert Frank, Philip J. Cook, "The Winner-Take-All Society: Why the Few at the Top Get So Much More Than the Rest of Us", 1995.*

11. *Jonatasv, Medium: "Metrics Evaluation: MSE, RMSE, MAE and MAPE", Articolo di ricerca, 2024. https://medium.com/@jonatasv/metrics-evaluation-mse-rmse-mae-and-mape-317cab85a26b*

12. *HODSON, Timothy O.,"Root mean square error (RMSE) or mean absolute error (MAE): When to use them or not", Geoscientific Model Development Discussions, v. 2022, p. 1–10, 2022.*
13. *Tia Plagata, Medium:" Interpreting the Root Mean Squared Error of a Linear Regression Model", Articolo di ricerca, 2020. https://tiaplagata.medium.com/interpreting-the-root-mean-squared-error-of-a-linear-regression-model-5166e6b10db8*