Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Final Project

Mercari Price Suggestion

Authors:
Anna Marika Biasco - 865873 - a.biasco@campus.unimib.it
Francesca Pulerà - 870005 - f.pulera@campus.unimib.it
Massimo Zarantonello - 866457 -
m.zarantonello2@campus.unimib.it
February 7, 2025

Abstract

Il progetto si concentra sulla previsione automatica dei prezzi dei prodotti online nel contesto della Mercari Price Suggestion Challenge. L'obiettivo è sviluppare un modello di apprendimento automatico che preveda con precisione i prezzi, utilizzando dati come nome, categoria, marchio, condizione e descrizione dei prodotti. Il set di dati include 1,5 milioni di righe e il lavoro comprende l'analisi e la pre-elaborazione dei dati, la creazione di funzionalità per diversi modelli e la valutazione delle prestazioni tramite metriche adeguate.

1 Introduzione

1.1 Che cos'è Mercari?

Mercari, lanciato in Giappone nel 2013, è una piattaforma di e-commerce facile ed intuitiva. Prevedere i prezzi accurati è una sfida importante, poiché aiuta i venditori a stabilire prezzi competitivi in un mercato saturo e migliora l'esperienza utente e la fluidità delle transazioni.

1.2 Composizione del Dataset

Il dataset preso in considerazione è Mercari Price Suggestion Challenge [1] ed è composto dai seguenti capi:

- **train_id** — l'identificativo di ogni inserzione.
- **name** — l'intestazione dell'inserzione. Per evitare fughe di notizie, Kaggle ha ripulito i dati rimuovendo il testo che assomiglia ai prezzi. Questi prezzi sono stati rimossi e sono mostrati come `[rm]`.
- **item_condition_id** — lo stato della merce fornito dal venditore. Va da 1 a 5, dove 1 indica lo stato non rovinato e 5 lo stato molto danneggiato.
- **category_name** — la categoria dell'inserzione.
- **brand_name** — casella di testo contenente informazioni sulla marca del prodotto.
- **shipping** — se il venditore paga le spese di spedizione, il valore è 1, altrimenti è 0.

- **item_description** — la descrizione completa dell'articolo. Per evitare fughe di notizie, Kaggle ha ripulito i dati rimuovendo il testo che assomiglia ai prezzi (ad esempio $20). Questi prezzi sono stati rimossi e sono mostrati come `[rm]`. È una delle caratteristiche più importanti.

- **price** — il prezzo di acquisto dell'articolo. Questa è la variabile target che deve essere prevista dai modelli. L'unità è USD. Questa colonna non esiste in `test.tsv` poiché è quella che verrà prevista.

wrapfig

2 Dataset

L'obiettivo è comprendere la distribuzione dei dati, identificare i valori mancanti, le caratteristiche influenti e le trasformazioni necessarie.

2.1 Statistiche Generali

- **train_id**: il dataset contiene 1.482.535 valori.

- **item_condition_id**: media di 1.91 con una deviazione standard di circa 0.90, suggerendo che la maggior parte degli articoli è in buono stato.

- **price**: media di $26,74 con una deviazione standard di $38,59. Il minimo è 0, indicando possibili articoli gratuiti o errori nei dati, mentre il massimo di $2009 suggerisce la presenza di prodotti molto costosi.

- **shipping**: il 55,3% degli articoli ha un costo di spedizione, mentre il 44,7% offre spedizione gratuita, suggerendo politiche diversificate per attrarre acquirenti.

2.2 Analisi del Target

- Prezzo medio di $26,74 con una deviazione standard di $38,59.

- Il valore massimo del prezzo è $2009, ma il 75° percentile del prezzo è $29, suggerendo una distribuzione distorta con alcuni prodotti molto costosi.

2.3 Shipping e Condition

Il dataset mostra che la maggior parte degli articoli è in **buone o ottime condizioni**, con una percentuale inferiore di articoli in stato mediocre o danneggiato.

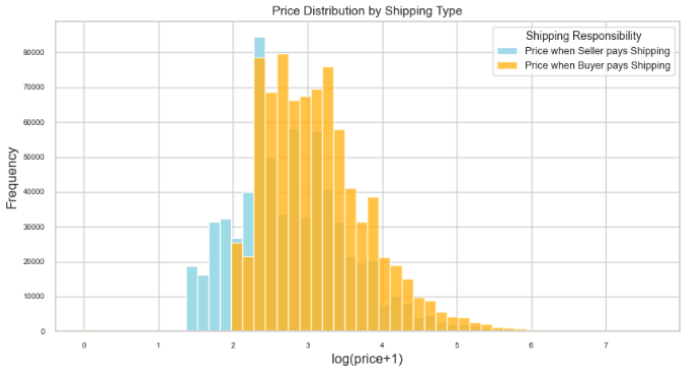

Figure 1: Questo grafico mostra la **distribuzione dei prezzi** degli articoli in base alla responsabilità della **spedizione** (se pagata dal venditore o dal compratore).

La distribuzione dei prezzi è più alta per gli articoli con spedizione gratuita (pagata dal venditore), e questo potrebbe indicare che i venditori che offrono la spedizione gratuita tendono a vendere articoli a prezzi più alti.

2.4 Categorie

La colonna **category_name** è strutturata gerarchicamente, con circa **1.287 categorie uniche**. Ogni categoria è composta da una categoria principale e due sottocategorie. Ci sono anche circa **6.327 elementi senza etichetta di categoria**, a cui è stata assegnata la categoria "**other**". Le prime cinque categorie uniche sono:

1. Women/Athletic Apparel/Pants, Tights, Leggings (60.177 occorrenze)
2. Women/Tops & Blouses/T-Shirts (46.380 occorrenze)
3. Beauty/Makeup/Face (34.335 occorrenze)
4. Beauty/Makeup/Lips (29.910 occorrenze)
5. Electronics/Video Games & Consoles/Games (26.557 occorrenze)

La maggior parte delle categorie ha una profondità di 3, ma alcune arrivano alla profondità 5. Per semplificare, si è deciso di mantenere solo la profondità 3, aggiungendo le colonne 'main_cat', 'subcat_1' e 'subcat_2'.

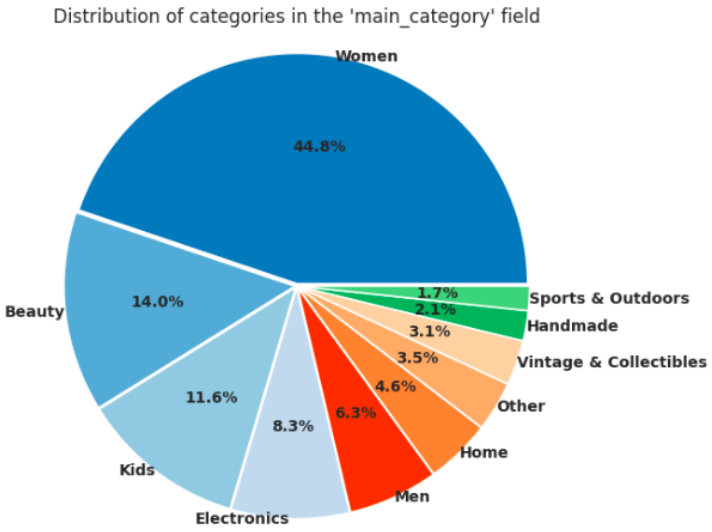

Figure 2: Grafico che mostra la distribuzione delle categorie all'interno di **main_category**.

2.5 Distribuzione dei Prezzi tra le Categorie

Il grafico 3 evidenzia una **disparità di distribuzione** tra le categorie, con alcune (come "Uomini" ed "Elettronica") che presentano una lunga coda verso prezzi più alti, mentre altre (come "Bellezza" e "Fatto a mano") sono più concentrate. La **mediana** varia significativamente tra le categorie. Sono presenti **valori anomali** in tutte le categorie, visibili nei punti esterni ai baffi del boxplot. La **varianza interna** è maggiore in categorie come "Elettronica" e "Uomini", mentre minore in "Bellezza" e "Bambini".

3 Approccio Metodologico

3.1 Preprocessing del Testo (name e item_description)

La pre-elaborazione del testo è fondamentale per rendere i dati non strutturati, come le descrizioni di prodotto, utilizzabili dai modelli di apprendimento automatico. Il processo include i seguenti passaggi:

- **Segmentazione**: il testo viene suddiviso in frasi tramite `sent_tokenize`, e successivamente in parole con `word_tokenize`.
- **Pulizia**: si rimuovono punteggiatura, numeri, caratteri speciali e stop

Figure 3: Grafico di distribuzione della **General Category**.

words, mantenendo solo le parole alfabetiche con una lunghezza maggiore di due caratteri.

- **Normalizzazione**: le parole vengono convertite in minuscolo e lemmatizzate utilizzando il lemmatizzatore WordNet.

3.2 Feature Encoding e Feature Vectorization

La trasformazione delle variabili categoriali e testuali in rappresentazioni numeriche è essenziale per permettere ai modelli di apprendimento automatico di utilizzarle efficacemente. Per evitare l'overfitting, il dataset è suddiviso in training, validation e test.

3.2.1 Vettorizzazione delle Feature Testuali

Per le feature testuali, sono stati utilizzati due approcci:

- **CountVectorizer** per il nome del prodotto (`name`), che cattura la frequenza delle parole principali, risultando in una matrice di dimensione $(1186028, 76227)$.
- **TfidfVectorizer** per la descrizione del prodotto (`item_description`), che pondera le parole in base alla loro rilevanza rispetto ai documenti, con matrice di forma $(1186028, 109127)$.

3.2.2 Word2Vec

Il secondo approccio utilizzato è **Word2Vec**, che mappa il contenuto testuale in vettori numerici densi, catturando le relazioni semantiche tra le parole e conservando il contesto. La matrice risultante ha dimensione $(1186028, 100)$.

3.3 Encoding delle Feature Categoriali

Anche per le variabili categoriali, sono stati utilizzati due approcci distinti:

- **One-Hot Encoding**: applicato a variabili come `brand_name`, `item_condition_id`, `shipping`, `main_cat`, `subcat_1`, e `subcat_2`. Ogni categoria è rappresentata come un vettore binario, e la dimensione del dataset con One-Hot Encoding è $(1186028, 191165)$.
- **Label Encoding**: applicato a variabili come `brand_name`, `main_cat`, `subcat_1`, e `subcat_2`, dove ogni categoria è assegnata a un valore numerico unico. La dimensione del dataset con Label Encoding è $(1186028, 108)$, $(148253, 108)$, $(148254, 108)$.

Con **One-Hot Encoding**, **TF-IDF** e **CountVectorizer** abbiamo creato **matrici sparse** adatte a modelli lineari come Ridge Regression. Con **Word2Vec** e **Label Encoding,** invece, abbiamo ottenuto **rappresentazioni compatte** ideali per le reti neurali.

3.4 Confronto tra Approcci

Il confronto tra One-Hot Encoding e Label Encoding evidenzia che il primo porta a un dataset più grande $(1186028, 191165)$, ma offre maggiore granularità nella rappresentazione. Al contrario, Label Encoding e Word2Vec riducono la dimensione del dataset $(1186028, 107)$ migliorando l'efficienza computazionale, ma con una possibile perdita di granularità e semantica.

3.5 Regressione

La regressione è utilizzata per prevedere valori numerici continui, come i prezzi dei prodotti, sfruttando le caratteristiche degli oggetti.

3.5.1 Ridge Regression

Ridge Regression è un metodo di regressione lineare con una penalità L2 che aiuta a gestire la multicollinearità e a prevenire l'overfitting [2]. Gestisce meglio le matrici sparse. La trasformazione logaritmica dei prezzi ha migliorato la robustezza delle previsioni. L'ottimizzazione del parametro α tramite *GridSearch* ha permesso di ridurre l'RMSE, anche se sono necessari miglioramenti per i valori estremi dei prezzi.

3.6 Reti Neurali

Le reti neurali sono utilizzate per la previsione dei prezzi degli articoli sfruttando le loro capacità di apprendere pattern complessi e non lineari.

3.6.1 Modello 1: Rete Densa a Tre Strati

Il primo modello è una rete feedforward semplice con tre strati densi, batch normalization e dropout del 20%. Usa l'ottimizzatore Adam e la funzione di perdita MSE. Questo modello è relativamente semplice e fornisce una solida base per il trattamento di dati non sequenziali.

Differenze rispetto ai modelli successivi: il modello 1 è il più semplice e utilizza una struttura ridotta rispetto agli altri modelli che impiegano tecniche avanzate per combattere l'overfitting.

3.6.2 Modello 2: Rete Densa con Regolarizzazione e Ottimizzatore AdamW

Il secondo modello è simile al primo, ma con l'ottimizzatore AdamW, un tasso di apprendimento più alto e regolarizzazione L2. È stato aggiunto il callback ReduceLROnPlateau per ottimizzare il tasso di apprendimento.

Differenze rispetto al modello precedente: riduce il rischio di overfitting più efficacemente rispetto al primo modello.

3.6.3 Modello 3: Rete Neurale con Maggiore Complessità

Il terzo modello è composto da quattro strati nascosti, una maggiore regolarizzazione (L2 e dropout del 30%) e un callback EarlyStopping per prevenire l'overfitting. L'ottimizzatore AdamW utilizza un tasso di apprendimento più basso.

Differenze rispetto ai modelli precedenti: aumenta la complessità con più strati e una regolarizzazione più intensa, ideale per dataset complessi.

3.6.4 Modello 4: Rete Ibrida Densa e GRU

Il quarto modello combina strati densi con GRU (Gated Recurrent Units) per apprendere sia pattern complessi che sequenziali. Include un tasso di apprendimento adattivo tramite ReduceLROnPlateau e dropout del 30%.

Differenze rispetto al modello precedente: introduce GRU, utile per catturare sequenze temporali, mentre i modelli precedenti non gestivano esplicitamente le sequenze.

3.6.5 Modello 5: Rete LSTM con Regularizzazione e Dropout

Il quinto modello utilizza LSTM (Long Short-Term Memory) per catturare informazioni a lungo termine, con tre strati LSTM e dropout del 30% e 20%. È ottimizzato per dati sequenziali.

Differenze rispetto al modello precedente: rispetto al modello 4, che usa GRU, il modello 5 sfrutta LSTM, più adatto per memorizzare informazioni temporali a lungo termine.

4 Risultati e valutazioni

4.1 Introduzione

In questo capitolo vengono presentati i risultati ottenuti dall'applicazione della Ridge Regression e delle Reti Neurali al dataset in esame. I risultati sono valutati tramite le metriche RMSE (Root Mean Squared Error) e MAE (Mean Absolute Error), calcolate sia sul set di validazione che sul test set.

4.2 Risultati Ridge Regression

La tabella 1 riassume i principali risultati ottenuti utilizzando la Ridge Regression con diverse rappresentazioni dei dati. In particolare, vengono riportati l'errore quadratico medio (RMSE) sul set di validazione, l'iperparametro α ottimale selezionato tramite Grid Search e Cross-Validation, e i valori di RMSE e MAE calcolati sul test set, che forniscono una misura dell'efficacia del modello su dati non visti. Gli Scatter Plot mostrati in figura 4 offrono un

confronto visivo tra i valori reali e quelli predetti dalla Ridge Regression, evidenziando l'accuratezza del modello per ciascuna delle due rappresentazioni.

Table 1: RMSE in validazione, α ottimale selezionato tramite Grid Search e Cross-Validation, e RMSE e MAE sul test set per valutare la performance del modello su dati non visti.

Metodo	**RMSE Val.**	**α Ott.**	**RMSE Test**	**MAE Test**
One-Hot, CountVec, TF-IDF	0.70	56.90	0.51	0.38
Label Enc., Word2Vec	0.74	10.00	0.63	0.48

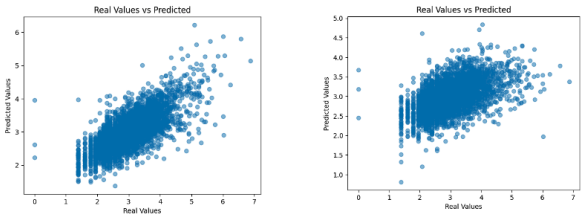

Figure 4: Lo scatter plot a sinistra mostra i risultati con One-Hot, Count Vectorizer e TF-IDF, mentre a destra quelli con Label Encoding e Word2Vec.

Nella Figura 4 l'allineamento lungo la diagonale indica la qualità della predizione, con una maggiore dispersione dai valori reali che segnala un errore maggiore.

4.3 Risultati con Reti Neurali

In questa sezione vengono presentati i risultati ottenuti con cinque diversi modelli di rete neurale, utilizzando le rappresentazioni dense generate tramite Label Encoding e Word2Vec, come descritto nella sezione precedente.

Le seguenti figure mostrano i grafici di training e validation loss (RMSE e MAE) per ciascuna delle reti neurali. Questi grafici consentono di osservare l'andamento dell'errore durante l'allenamento e la validazione, fornendo una panoramica sull'efficacia dei vari modelli.

I risultati quantitativi ottenuti sui set di test sono riassunti nella tabella 2, che riporta i valori di **RMSE** (Root Mean Squared Error) e **MAE** (Mean Absolute Error) per ciascuno dei cinque modelli

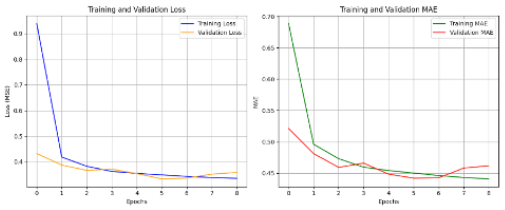

Figure 5: Training e Validation Loss (RMSE e MAE) Prima Neural Network

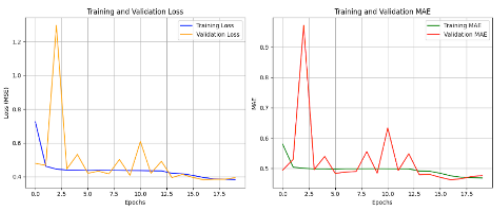

Figure 6: Training e Validation Loss (RMSE e MAE) Seconda Neural Network

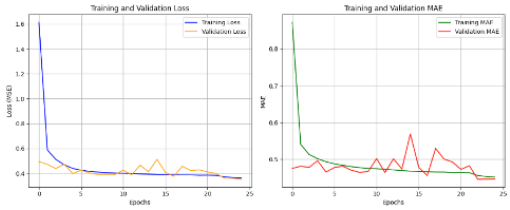

Figure 7: Training e Validation Loss (RMSE e MAE) Terza Neural Network

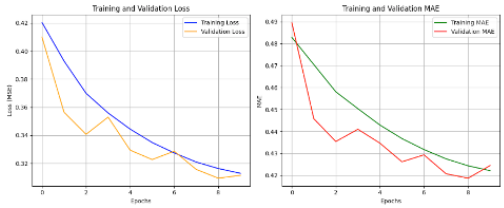

Figure 8: Training e Validation Loss (RMSE e MAE) Quarta Neural Network

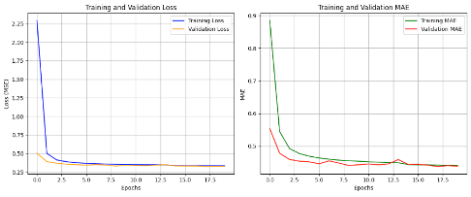

Figure 9: Training e Validation Loss (RMSE e MAE) Quinta Neural Network

Table 2: La tabella mostra i risultati delle reti neurali sui set di test, con RMSE e MAE. La Quarta NN ha ottenuto le migliori prestazioni con RMSE di 0.55 e MAE di 0.42, seguita dalla Quinta NN.

Loss Function	**Prima NN**	**Seconda NN**	**Terza NN**	**Quarta NN**	**Quinta NN**
RMSE Test	0.60	0.61	0.58	**0.55**	0.57
MAE Test	0.46	0.47	0.45	**0.42**	0.44

5 Discussione

In questo capitolo vengono analizzate le prestazioni e le capacità di generalizzazione dei modelli testati in questo studio, esaminandone i risultati elencati nella sezione precedente.

5.1 One-Hot Ridge Regression

Il valore di RMSE pari a 0.51 ottenuto sul test set (Tabella 1) indica una buona capacità del modello di predire i prezzi logaritmici, pur evidenziando una certa variabilità non completamente catturata.

L'ottimizzazione del parametro α ha contribuito a migliorare sia la stabilità che la precisione del modello rispetto alla configurazione iniziale, riducendo l'errore di generalizzazione, come riportato nella Tabella 1.

Un'analisi più approfondita dei risultati mostra che il modello tende leggermente a sottostimare i prezzi più alti e a sovrastimare quelli più bassi. Ciò suggerisce che gli estremi della distribuzione non siano ancora modellati in maniera ottimale, come si può osservare nel grafico 9.

5.2 Label Encoder Ridge Regression

L'RMSE più elevato sul validation set (0.7388, Tabella 1) rispetto a quello ottenuto con One-Hot Encoding (RMSE = 0.5030) suggerisce una minore capacità di generalizzazione del modello. Questo fenomeno potrebbe essere attribuito alla natura della codifica tramite Label Encoding, che introduce una relazione ordinale tra le categorie. A differenza dell'One-Hot Encoding, che mantiene una rappresentazione sparsa, il Label Encoding può indurre il modello a interpretare erroneamente le categorie come valori numerici con un ordine intrinseco, influenzando negativamente l'apprendimento dei parametri.

Il miglior valore di α selezionato è 10.0 (Tabella 1), il quale ha permesso di ottimizzare le prestazioni del modello. Tuttavia, la dispersione osservata nei risultati indica che anche in questo caso il modello fatica a catturare con precisione la struttura logaritmica dei prezzi in alcune fasce.

5.3 Reti Neurali

La Tabella 2 riporta i risultati delle reti neurali in termini di RMSE e MAE sui set di test. Le principali osservazioni sono:

1. **Miglior modello**: la quarta rete neurale, con RMSE di 0.55 e MAE di 0.42, è la più precisa.
2. **Secondo miglior modello**: la quinta rete neurale, con RMSE di 0.57 e MAE di 0.44, è un'ottima alternativa.
3. **Modelli meno performanti**: la seconda rete neurale ha le prestazioni peggiori, con RMSE di 0.61 e MAE di 0.47.
4. **Differenze minime**: le prestazioni dei modelli sono simili, con margini di miglioramento.

La quarta rete neurale risulta la migliore, sfruttando layer GRU per una maggiore capacità di generalizzazione.

I grafici 5-9 mostrano un buon comportamento di generalizzazione, senza segni di overfitting, con il miglioramento progressivo dell'errore medio assoluto (MAE) durante l'addestramento.

6 Conclusioni

Abbiamo analizzato la previsione dei prezzi nella *Mercari Price Suggestion Challenge*, confrontando **Ridge Regression** e reti neurali. Il modello Densa-GRU ha ottenuto il miglior RMSE (0.55) e MAE (0.42), ma con costi computazionali molto elevati.

Miglioramenti futuri includono embedding avanzati (BERT), modelli ensemble e ottimizzazione degli iperparametri. Le reti neurali si sono rivelate promettenti, ma con sfide di scalabilità e generalizzazione.

References

[1] E. Sam, "Mercari price suggestion challenge," 2017, accessed: 2025-02-07. [Online]. Available: https://www.kaggle.com/datasets/elizabethsam/mercari-price-suggestion-challenge

[2] D. W. Marquardt and R. D. Snee, "Ridge regression in practice," pp. 3–20, 1975. [Online]. Available: https://doi.org/10.1080/00031305.1975.10479105