SISTEMA DI RACCOMANDAZIONE PER RECENSIONI DI AMAZON

DAMIANO PELLEGRINI

1. Sommario

Questo progetto ha avuto come obiettivo lo sviluppo di un sistema di raccomandazione utilizzando di Collaborative Filtering e Content-Based Filtering su un dataset di recensioni di prodotti Amazon. Successivamente è stata effettuata una Sentiment Analysis sulle recensioni stesse con lo scopo di predire il rating dato il testo inserito dall'utente.

La categoria del dataset soggetto ad analisi è stata scelta accuratamente in base alle performance dei vari sistemi di raccomandazione dopo aver analizzato un ampia gamma di categorie. La scelta è infine ricaduta sulla categoria Automotive.

I risultati ottenuti mostrano che il sistema di raccomandazione basato su Collaborative Filtering ha performance limitate a causa della sparseness della matrice dei rating, mentre il Content-Based Filtering mostra alta precisione con errori MSE e RMSE bassi. La sentiment analysis, condotta con entrambe le tecniche di embedding, ha prodotto risultati con errori molto bassi, indicando un'accuratezza elevata nella classificazione del sentimento.

2. Analisi preliminare

2.1. **Hardware utilizzato.**

Per eseguire gli algoritmi ho utilizzato un Macbook Pro 2021:

- Processore & GPU: Apple M1 Pro 10-core, 16-gpu-cores
- RAM: 16 GB unificata

2.2. **Scelta del dataset.**

La scelta del dataset da analizzare è ricaduta inizialmente sulla categoria "Industrial and Scientific", tuttavia dopo aver riscontato problemi con il modello K-NN per collaborative filtering causati dalla sparsità della matrice, nonostante il filtraggio per ridurla, ho deciso di trovarne un altro.

La metrica che ho utilizzato per cercare un dataset adatto è stato il numero di utenti, prodotti e review unici e la percentuale di predizioni corrette effetuate durante il calcolo della matrice di rating di un subset dei dati. La prima metrica in particolare doveva presentare un numero di utenti molto più elevato rispetto ai prodotti dato che il sistema di raccomandazione è basato sugli utenti. Inoltre era importante che queste metriche rimanessero nelle giuste proporzioni dopo il pruning e filtering del dataset.

2.3. **Visualizzazione dati.**

Andiamo ora a visualizzare alcuni dati sul dataset Automotive scelto.

2.3.1. *Informazioni sul dataset.*

Il primo dato che ci interessa è sapere in che modo è fatto il dataset, quali sono i tipi e quanta memoria virtuale occupa.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19692836 entries, 0 to 19692835
Data columns (total 4 columns):
 #   Column       Dtype
---  ------       -----
 0   user_id      object
 1   parent_asin  object
 2   rating       object
 3   timestamp    object
dtypes: object(4)
memory usage: 601.0+ MB
None
```

Possiamo quindi notare come il tipo sia object (puntatore a 64-bit), ho 20 milioni di entries e occupa 601 MB di memoria virtuale.

Successivamente ho verificato che non ci fossero valori nulli.

E infine e effettuato il conteggio di quanti user e quanti item unici ci sono all'interno del dataset.

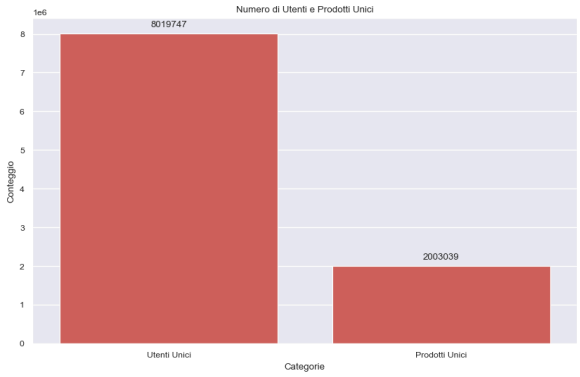

FIGURA 1. Utenti/Item unici.

2.3.2. *Analisi statistica.*

Il passo successivo è stato quindi un analisi statistica dei dati per capire distribuzioni, medie e correlazione dei campi in analisi. Questo per vedere, prima di andare a studiare il dataset con algoritmi complessi, se c'erano dei campi da sfruttare o da tenere conto durante la valutazione dei risultati ottenuti.

La prima metrica che ho voluto controllare è la distribuzione dei rating del dataset per avere un idea di quanto siano bilanciate le mie classi che dovro predire. Come possiamo notare dal grafico c'è una forte predominanza dei rating a 5 stelle.

Questo può stare a indicare diverse cose:

- Un bias positivo degli utenti della piattaforma
- Un problema di botting nella piattaforma
- Una barriera di qualità all'interno della piattaforma o un buon sistema di raccomandazione dei prodotti

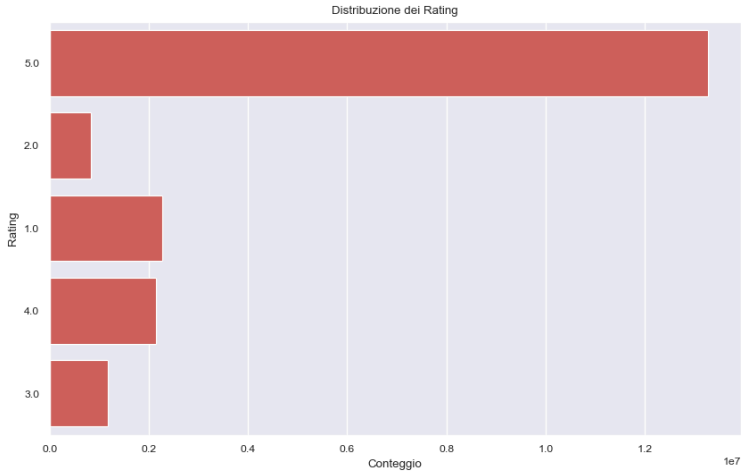

FIGURA 2. Distribuzione dei rating.

Media dei rating $\simeq 4.183$

Sono andato quindi a vedere la distribuzione di recensioni per utente in modo da capire meglio la situazione. A prima vista si può notare come ci sia un numero estremo di utenti con poche recensioni e pochi utenti con un numero considerevole che superano le 300 recensioni.

Chiaramente questi due poli del dataset vanno considerati con attenzione durante lo sviluppo del sistema di raccomandazione, sia per lo scarso contributo degli utenti con poche recensioni che il pericolo che gli utenti con tante recensioni siano bot.

Media di recensioni per utente $\simeq 2.456$

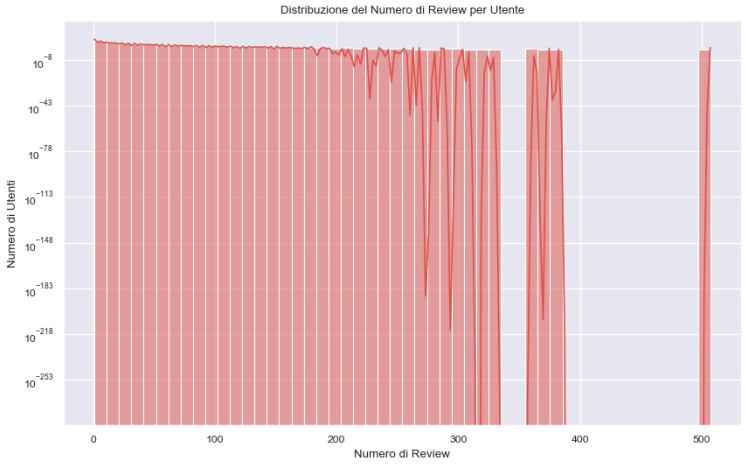

FIGURA 3. Distribuzione delle recensioni per utente. Scala logaritmica.

La stessa analisi è stata quindi fatta per i prodotti individuando i prodotti sconosciuti e quelli molto popolari, una considerazione aggiuntiva deve essere fatta anche per la distribuzione di recensioni per prodotto.

Media di recensioni per prodotto $\simeq 9.831$

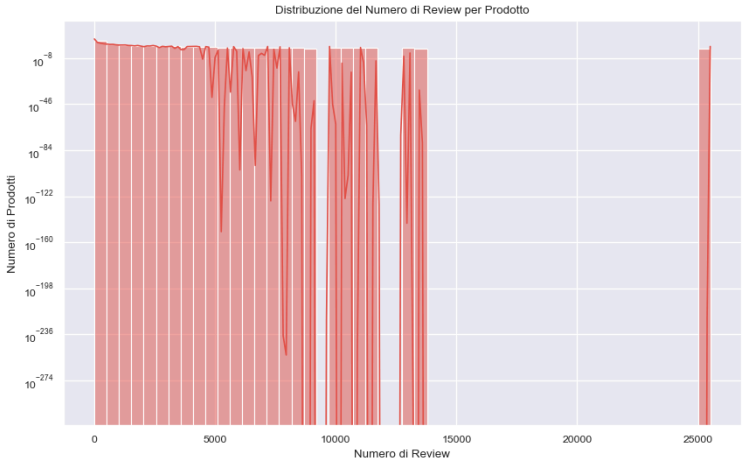

FIGURA 4. Distribuzione delle recensioni per prodotto. Scala logaritmica.

L'ultimo passo della mia analisi statistica è stato generare una matrice di correlazione tra i campi del dataset per confermare le mie aspettative, ovvero che non ci sia correlazione tra il codice del prodotto e l'id dell'utente, questo anche perchè l'utente finale non ha accesso diretto a queste informazioni.

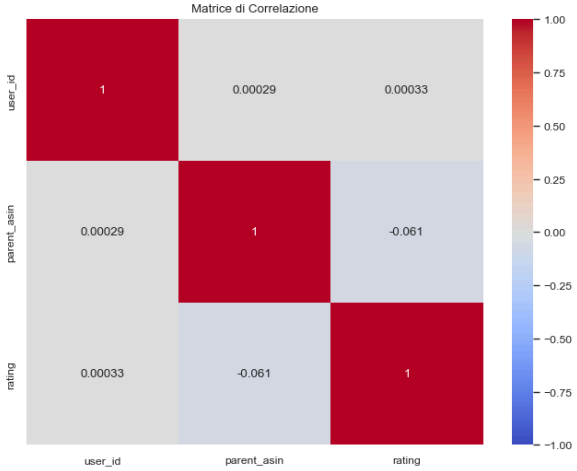

FIGURA 5. Matrice di correlazione lineare.

2.3.3. *Analisi temporale.*

L'ultima analisi che ho effettuato è stata un'analisi temporale per provare a individuare eventuali mode ricorrenti e visualizzare l'andamento della piattaforma nel tempo.

Il primo grafico che ho analizzato è stato l'andamento nel tempo del numero di recensioni con granularità mensile.

Si può subito notare come negli anni la piattaforma ha avuto un considerevole sviluppo che ha portato a un maggior numero di recensioni.

Si può inoltre notare come durante il periodo del Covid ci sono state meno recensioni, con un picco subito dopo, andando a indicare una scarsa situazione economica con successiva ripresa da parte degli utenti di Amazon.

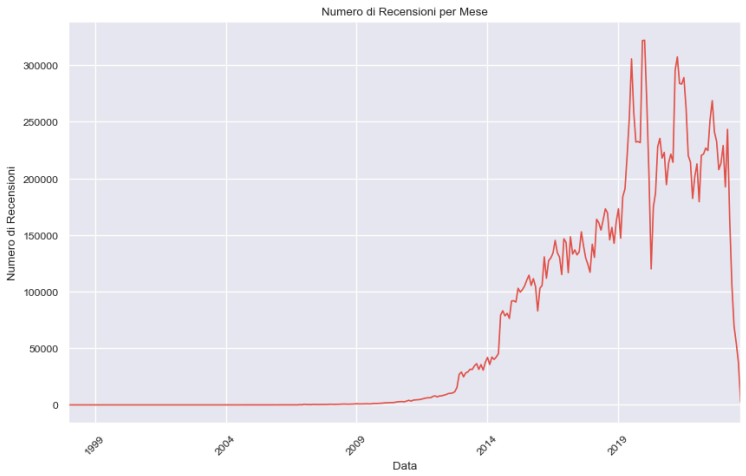

FIGURA 6. Numero di recensioni per mese.

Grafico analogo ma per la media dei rating, dove di può notare come in concomitanza dell'aumento del numero di recensioni ci sia stato anche una regolarizzazione della media dei rating.

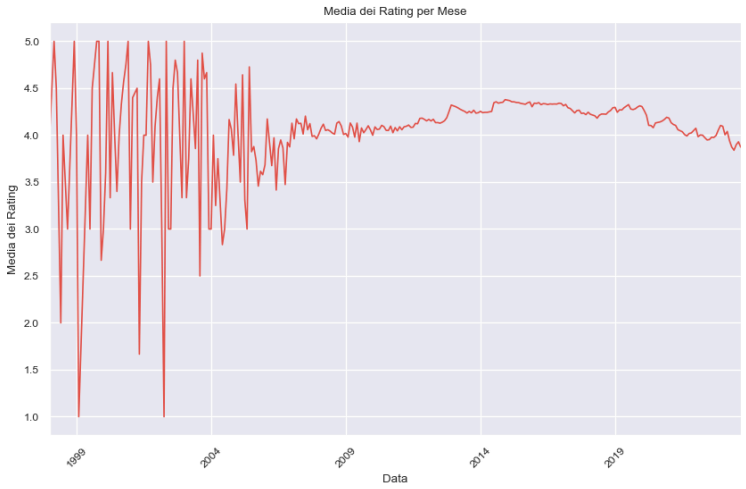

FIGURA 7. Media dei rating per mese.

Come nel caso del grafico con granularità mensile possiamo notare una crescita con uno sprofondamento e successivo picco dovuto al 2020.

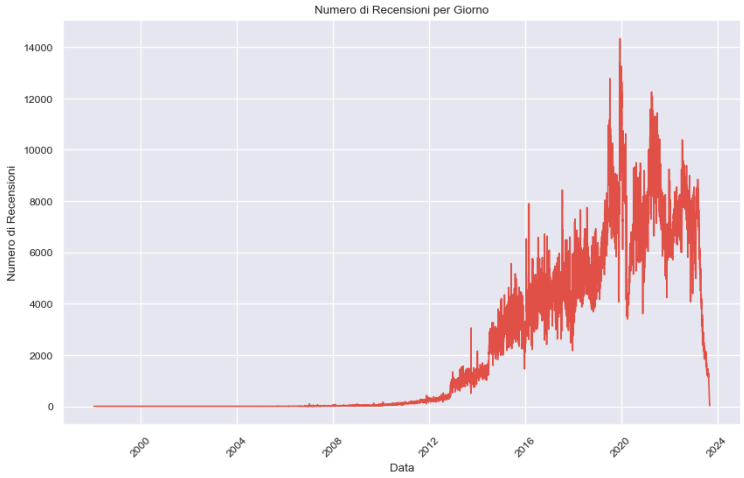

FIGURA 8. Numero di recensioni per giorno.

Grafico con granularità giornaliera. Possiamo notare lo stesso andamento che abbiamo con la granularità mensile, seppur con un maggiore rumore.

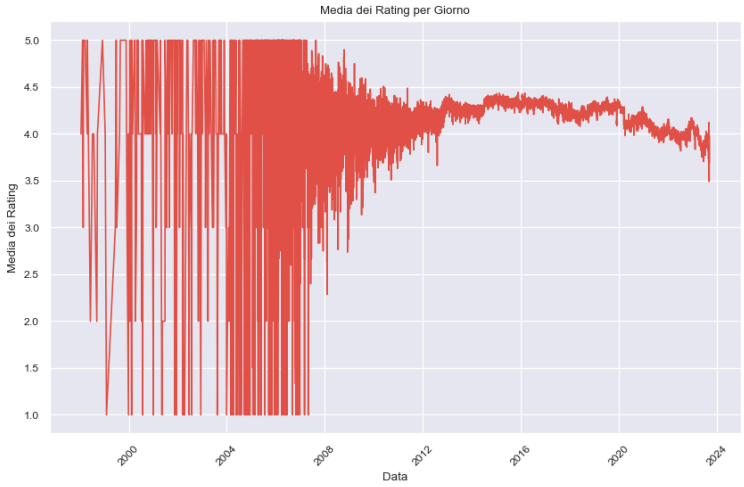

FIGURA 9. Media dei rating per giorno.

Grafico con granularità settimanale, prova ulteriore di come la moda persista con qualsiasi granularità.

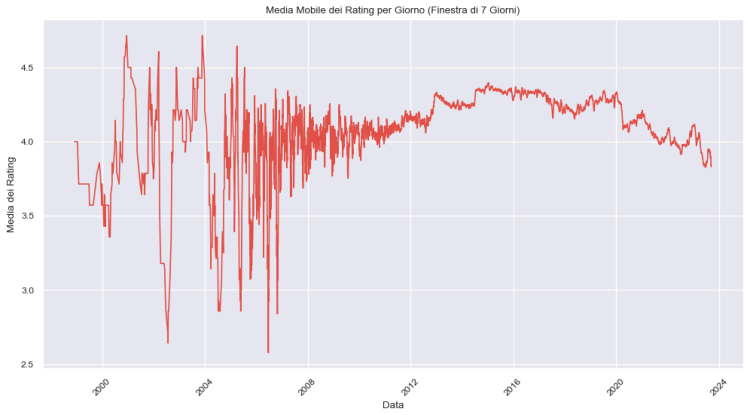

FIGURA 10. Media mobile per giorno (Finestra 7 giorni).

2.4. **Preprocessing preliminare.**

Vista la mole molto grandi di dati era necessario eseguire un pruning preliminare comune a qualsiasi tipo di analisi effettuata, questo per due principali motivi: ridurre il tempo di esecuzione degli algoritmi, ridurre la memoria utilizzata dagli algoritmi.

Il metodo di pruning si basa sul mantenre solo gli utenti e i prodotti con un determinato numero di recensioni.

	Before	Min. review	After
Users	8019747	13	9911
Items	2003039	14	5381
Ratings	19692836	N/A	184765

TABELLA 1. Utenti/Item/Rating unici dopo il pruning.

L'ultimo preprocess effettuato in tutti e tre i casi è il casting di alcune colonne del dataset in tipi di dati meno generosi, nella fattispecie convertire il rating da `object` a `float16` e l'id del prodotto e dell'utente in `string`.

3. APPROCCIO COLLABORATIVE FILTERING

Questo approccio è basato principalmente su due algoritmi: Nearest Neighbor e Matrix Factorization. Il primo algoritmo come detto precedentemente necessita la riduzione della matrice di rating che andrà a processare, fortunatamente ci ho pensato nel preprocessing preliminare.

Successivamente ho effettuato il clustering sulla media rating dell'utente e il numero di recensioni per individuare delle fasce di utenti.

3.1. **Preprocessing.**

Non ho effettuato nessun preprocessing aggiuntivo oltre a quello preliminare.

3.2. **Ottimizzazione iperparametri.**

L'ottimizzazione degli iperparametri degli algoritmi è stata effettuata tramite l'algoritmo di GridSearch per K-NN e SVD, mentre per il K-Means ho utilizzato l'elbow method.

I risultati dell'ottimizzazzione sono i seguenti:

Algoritmo	Parametri ottimizzati	Metrica migliore
K-NN	`{'k': 10, 'sim_options': {'name': 'cosine', 'user_based': True}}`	RMSE 1.038
SVD	`{'n_factors': 100}`	RMSE 0.888
K-Means	`{'num_clusters': 5}`	WCSS $\simeq$ 48000

Tabella 2. Risultati di ottimizzazione iperparametri per ogni algoritmo.

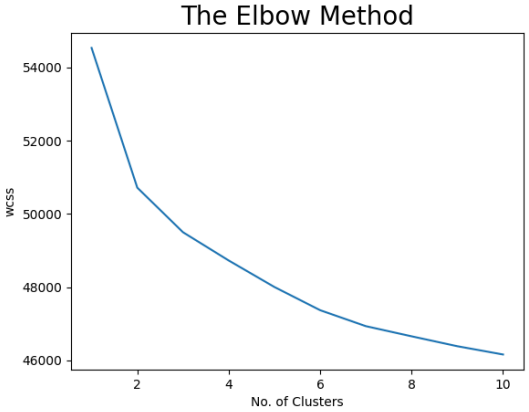

Figura 11. Grafico metriche K-Means per cluster k.

3.3. **Generazione matrice di rating.**

La parte della generazione delle matrice di rating per K-NN e SVD è molto dispendiosa in termini di memoria perciò ho limitato la matrice al numero massimo di utenti che la mia macchina riusciva a calcolare per avere un idea delle performance del modello, il numero di utenti è 1000.

I risultati sulle predizioni per 1000 utenti sono i seguenti:

Algoritmo	Nº Predizioni impossibili	% predizione impossibili su effettuate
K-NN	5808	15.43%
SVD	1207	3.21%

Tabella 3. Risultati preliminari sulle performance dei modelli.

3.4. **Clustering degli utenti.**

Sfortunatamente l'algoritmo di clustering presenta lo stesso problema del K-NN, inoltre presenta una difficoltà nel visualizzare i risultati data la struttura del data-

set e richiederebbe algoritmi come t-SNE o PCA per la riduzione di dimensionalità per la visualizzazione.

Quello che sono andato a fare nella pratica è stato quindi un clustering basato sulle medie e il numero di recensioni cosi da poter individuare eventuali fasce di recensori, dai recensori sporadici a quelli accaniti.

Di seguito riporto i risultati di tale analisi.

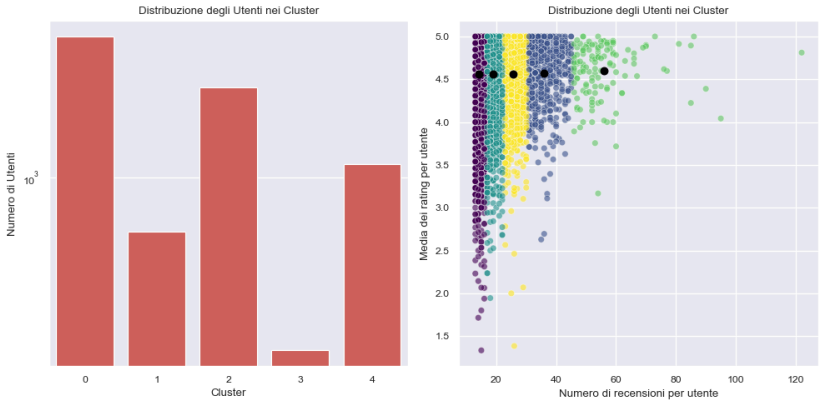

FIGURA 12. Grafico K-Means su medie recensioni utenti e numero recensioni.

Possiamo notare come la media di recensioni salga con l'aumentare del numero di recensioni indicando che esiste una fascia di utenti che solitamente non recensisce prodotti e l'unica volta che l'ha fatto non si è trovata particolarmente bene con il suo acquisto.

3.5. **Raccomandazioni utente.**

Passiamo ora alla generazione della lista di raccomandazione per un utente campione, entrambi gli algoritmi hanno generato la lista per lo stesso utente.

Di seguito per ogni algoritmo le top 10 raccomandazioni nella forma `<parent_asin>: <rating_predetto>`

	K-NN	Matrix factorization (SVD)
1.	B000IX99RO: 5	9539751047: 5
2.	B004D5SUIC: 5	B07BGHR4PN: 5
3.	B0009OR8V6: 5	B07CJL99N2: 5
4.	B07BFGGSG2: 5	B07CBMDT6K: 5
5.	B0B8PDNXG4: 5	B07C7KHB9W: 5
6.	B0046DSTJG: 5	B07C6Y83CK: 5
7.	B07BCHDTQ7: 5	B07C5S18M3: 5
8.	B0009TCRB2: 5	B07C4W858Y: 5
9.	B000VUASMY: 5	B07C2VZGJY: 5
10.	B000C9LOHW: 5	B07C2VH2MV: 5

Come sospettavo data la maggioranza di recensioni a 5 stelle anche le raccomandazioni presentano tutte predizioni a 5 stelle, a maggior ragione se prendo le top k.

3.6. **Confronto metriche.**

I risultati dei modelli sono tutto fuorché sorprendenti, il K-NN performa molto peggio rispetto a SVD, e questo è dovuto soprattuto alla matrice di rating molto sparsa e la mancanza di dati con correlazioni significative da dare in pasto al modello.

Di seguito la tabella con le metriche per ogni modello

Algoritmo	MSE	RMSE
K-NN	1.076573	1.037580
SVD	0.787765	0.887561

Tabella 4. Metriche MSE e RMSE per algoritmo.

4. Approccio Content-Based Filtering

L'approccio su Content-Based richiede un'attenzione maggiore sui passi di preprocessing e un dataset più ricco però vedremo come in realtà è abbastanza banale.

Il dataset che utilizziamo è sempre quello Automotive, tuttavia questa volta lo integriamo con i metadati dei prodotti.

4.1. **Preprocessing.**

Questa volta abbiamo due parti del dataset da preprocessare: quello dei rating e quello dei metadati, partiamo da quello dei rating. Il preprocess rimane lo stesso dell'approccio precedente, quindi filtro per numero minimo di recensioni per utenti e prodotti e conversioni di tipi di dato.

Per quanto rigurda il dataset dei metadati invece abbiamo qualche passo in più da effettuare, di seguito un breve lista:

1. Ridurre campi quali `'images'` e `'videos'` da array a booleani col prefisso `'has_'` che indicano la presenza o meno di video o immagini.
2. Rimuovere tutti i campi inutilizzati dal modello.
3. Conversione del campo `'description'` da `array` a `string`.
4. Rimuovere tutte le entry con valori nulli nei campi `'title'` e `'description'`.
5. Merge del dataset dei metadati con quello dei ratings.

Effettuati tutti questi passaggi possiamo passare al preprocessing atto a migliorare quello che sarà l'embedding dei nostri campi di testo.

Andiamo prima a unire i campi `'title'` e `'description'` in un unico `'corpus_text'` sul quale verrà effettuato uno stemming e uno stop-word removal in modo da migliorare quella che sarà la generazione di feature dell'embedding tramite BoW.

A questo proposito ho scelto di utilizzare lo stemming e non il lemmatizing per una semplice ragione, voglio avere il minor numero di feature possibile per evitare di andare in overfitting del modello con embedding BoW. Come nota aggiuntiva lo stemming è anche computazionalmente meno dispendioso.

4.2. **Embedding.**

L'embedding è quella tecnica che permette di ridurre un testo di lunghezza indefinita in un vettore di lunghezza definita in modo che possa essere dato in input a vari modelli di machine learning.

Nel mio caso ho utilizzato due tecniche di embedding una basata sulla frequenza e una basata sui transformers (reti neurali) per poi dare i risultati delle tecniche in pasto a un modello K-NN.

4.2.1. *Bag of Words.*

Il modello utilizzato per l'embedding tramite Bag of Words è il `CountVectorizer` con tokenizer personalizzato e set di stop-word inglese.

4.2.2. *Transformers.*

L'algoritmo utilizzato per l'embedding tramite reti neurali è il `SentenceTransformer` con modello `sentence-transformers/average_word_embeddings_komninos`.

4.3. **Ottimizzazioni.** Data l'eccessiva quantità di tempo per l'addestramento dei modelli ho optato per una libreria di parallelizzazione per addestrare i modelli e andare a ridurre i tempi di esecuzione dell'algoritmo da 3:52h a "sole" 2:50h.

4.4. **Confronto metriche.**

Una strategia che i venditori potrebbero adottare supponendo diverse cose sul sistema di raccomandazione di Amazon è quella di usare parole molto generiche o che richiedono poco preprocessing, in modo che, se Amazon basasse i rating sulla velocità di embedding, loro otterrebbero un punteggio più alto. Un'altra strategia è utilizzare parole di prodotti forti tra le quali: "Apple", "iPhone", "Nvidia", etc.

Il K-NN non è un buon algoritmo soprattuto se come metrica uso cosine similarity dato che usa il prodotto scalare tra vettori che in presenza di molti valori nulli dà risultati troppo simili. Questa cosa la possiamo notare anche dal fatto che l'embedding con BoW presenta un errore inferiore di quello con le reti neurali, che generano un embedding con diversi campi uguali a 0, andando a rendere inefficaci la maggior parte delle feature dell'embedding stesso.

Algoritmo	MSE	RMSE
K-NN	1.0766	1.0376
SVD	0.7878	0.8876
K-NN + BoW	0.0783	0.2798
K-NN + Transformers	0.0797	0.2822

Tabella 5. Metriche MSE e RMSE per algoritmo.

5. Sentiment Analysis

L'ultimo step del progetto è stato fare analisi del sentimento sulle recensioni. Gli scopi per quest'analisi sono molteplici e vanno dall'individuare eventuali bot a capire il vero pensiero del recensore a seconda di quello che scrivono. Quest'ultima capacità di comprensione del testo nasconde la vera utilità dell'analisi del sentimento.

5.1. **Preprocessing.**

Come ultimo dataset andremo a utilizzare quello delle recensioni che comprende anche i campi di testo e non solo il rating e gli id di prodotto e utente. Il preprocess fortunatamente rimane come lo stesso dell'approccio con collaborative filtering, con l'aggiunta del filtro per acquisti verificati, i quali erano precedentemente filtrati dal subset predo da HuggingFace.

A differenza del secondo approccio i campi su cui effettuare preprocessing testuale sono i campi `'title'` e `'text'` ai quali effettueremo lo stesso preprocess.

L'ultimo passo del preprocessing è convertire i rating in sentiment, sapendo che sono 5 rating unici, secondo la seguente tabella:

Rating	Sentiment
1	negative
2	negative
3	neutral
4	positive
5	positive

Tabella 6. Mapping rating-sentiment.

5.2. **Embedding.**

Le tecniche di embedding utilizzare rimangono invariate rispetto al secondo approccio con l'unica differenza nel numero massimo di feature impostato per il `CountVectorizer`.

Da illimitato sono passato a 2000 feature massime data la memoria limitata della mia macchina.

5.3. **Modello ML.**

Il modello utilizzato per la predizione è il K-NN ma, come specificato precedentemente, potrebbe non essere la scelta migliore se ho tra le mani una matrice di embedding troppo sparsa.

5.4. **Confronto metriche.**

Come nel caso del Content-based Filtering il modello allenato con embedding tramite BoW ha dimostrato risultati migliore di quello con embedding tramite Transformers.

Di seguito la tabella con le metriche osservate:

Embedding	MSE	RMSE
BoW	0.2456	0.4956
Transformers	0.262	0.5118

Tabella 7. Metriche MSE e RMSE per embedding.

6. Conclusioni

In conclusione, il progetto ha mostrato che:

1. La sparseness della matrice dei rating è una sfida significativa per i modelli di Collaborative Filtering, ma può essere mitigata con un adeguato filtraggio.
2. I modelli di Content-Based Filtering, sia basati su Bag-of-Words che su Transformers, riescono a fornire risultati altamente precisi.
3. L'analisi del sentimento come nel caso del content-based filtering dimostra un'accuratezza elevata, seppur deviata dallo sbilanciamento presente all'interno delle classi di rating.

Questi risultati suggeriscono che, mentre il Collaborative Filtering ha limitazioni nei contesti di dati sparsi, le tecniche basate sul contenuto e l'analisi del sentiment possono fornire raccomandazioni e insight utili con alta precisione.

References

Hou, Y. *et al.* (2024) «Bridging Language and Items for Retrieval and Recommendation», *arXiv preprint arXiv:2403.03952* [Preprint]

Hug, N. (2020) «Surprise: A Python library for recommender systems», *Journal of Open Source Software*, 5(52), p. 2174–2175. Disponibile su: https://doi.org/10.21105/joss.02174

Řehůrek, R. e Sojka, P. (2010) «Software Framework for Topic Modelling with Large Corpora», in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, pp. 45–50

Metodi Informatici per la Gestione Aziendale, Università degli Studi di Milano-Bicocca
Email address: d.pellegrini10@campus.unimib.it