FROSI DAVIDE

# PROGETTO
# METODI INFORMATICI PER LA GESTIONE AZIENDALE
# a.a.2024/2025

# INDICE


## EXECUTIVE SUMMARY

## INTRODUZIONE AL PROBLEMA

### DESCRIZIONE DEI DATI

### OBIETTIVI DELL'ANALISI

### RISULTATI DELL'ANALISI ESPLORATIVA

## RISULTATI STEP BY STEP

### PROGETTO BASE

#### STEP 2

#### STEP 3

#### STEP 4

#### STEP 5

#### STEP 6

### PROGETTO INTERMEDIO

#### STEP 1

#### STEP 2

#### STEP 3

#### STEP 4

#### STEP 5

### PROGETTO AVANZATO

#### STEP 1

#### STEP 2

#### STEP 3

## CONCLUSIONE E INTERPRETAZIONE SINTETICA DEI RISULTATI

# EXECUTIVE SUMMARY

In questo progetto l'obiettivo principale è stato sviluppare diverse tipologie di Recommendation Systems (Sistemi di raccomandazione) basate sull'analisi delle review dei prodotti della categoria "*Video Games*" provenienti dal dataset Amazon Reviews:

https://huggingface.co/datasets/McAuley-Lab/Amazon-Reviews-2023

Il lavoro è stato articolato in tre sezioni:

- Progetto Base: nella prima parte è stata completata un'analisi esplorativa, comprendente sia diverse statistiche descrittive, sia analisi di correlazione su alcune delle informazioni contenute nel dataset. Successivamente è stato realizzato un sistema di raccomandazione basato su collaborative filtering, applicando l'algoritmo K-NN, con successivo confronto con la tecnica di matrix factorization, risultata essere leggermente migliore. La configurazione ottimale ottenuta ha previsto un approccio item-based con similarità coseno e K=40. Il sistema è stato valutato tramite MSE e RMSE, ed è stato esteso anche con un'analisi di clustering degli utenti mediante K-means.
- Progetto Intermedio: è stato integrato un approccio content-based, sfruttando le informazioni testuali dei prodotti: 'title' e 'description', titolo e descrizione del prodotto, processate con tecniche NLP, Natural Language Processing. Gli embedding testuali sono stati generati con due tecniche: TFIDF, tecnica basata sulla frequenza, e Transformers, tecnica neurale, successivamente confrontate fra loro. Utilizzando la similarità tra prodotti e le preferenze degli utenti, è stato sviluppato un sistema di raccomandazione alternativo, poi confrontato con quello basato su collaborative filtering del progetto base.
- Progetto Avanzato: è stata affrontata la problematica della sentiment analysis applicata alle review testuali. Dopo aver processato i testi: 'title' e 'text', titolo e corpo della recensione, con tecniche NLP, sono stati generati gli embedding, ancora una volta con le due tecniche: TFIDF e Transformers. I modelli di classificazione, Logistic Regression e Random Forest, sono stati utilizzati per predire il sentiment: negativo, neutro o positivo, utilizzando come "etichetta" il rating associato alla recensione. I risultati migliori sono stati ottenuti con la Logistic Regression e embedding TFIDF, con una accuracy dell' 87%.

Complessivamente, il progetto ha evidenziato la differenza ma anche la complementarità tra approcci: collaborative filtering e content-based, e ha mostrato la validità dell'analisi testuale nella previsione del sentiment espresso dagli utenti.

# INTRODUZIONE AL PROBLEMA

## DESCRIZIONE DEI DATI

Il dataset utilizzato per lo sviluppo del progetto è quello fornito dal McAuley Lab, relativo alle recensioni dei prodotti su Amazon. In particolare, è stata selezionata la categoria "*Video Games*", comprendente sia informazioni sulle recensioni dei prodotti (*user reviews*) sia le informazioni sui prodotti (*item metadata*). In particolare, parliamo di una categoria con più di 4 milioni di recensioni, espresse da quasi 3 milioni di utenti su più di 100 mila prodotti unici.

La parte di *user reviews* contiene informazioni essenziali quali:

- rating: valutazione assegnata dall'utente (valori da 1 a 5)
- title e text: contenuto testuale della recensione, titolo e corpo
- parent_asin: identificativo del prodotto
- user_id: identificativo dell'utente

```
 #   Column       Dtype
---  ------       -----
 0   rating       float64
 1   title        object
 2   text         object
 3   parent_asin  object
 4   user_id      object
dtypes: float64(1), object(4)
```

Le altre informazioni che invece non sono state prese in considerazione sono:

- images: le immagini postate dagli utenti dopo aver ricevuto il prodotto
- asin: identificativo del singolo prodotto scindendolo dai prodotti diversi per colore, taglia, stile etc…
- timestamp: momento in cui è avvenuta la recensione
- verified_purchase: indica se la recensione è avvenuta da un utente il cui acquisto è stato verificato o meno
- helpful_vote: numero di voti utili della recensione

In realtà, queste ultime tre voci sono state usate nell'analisi esplorativa, ma solo per fare un'analisi di correlazione rispetto al rating.

La parte di *item metadata* contiene informazioni essenziali quali:

- parent_asin: identificativo del prodotto
- title: titolo del prodotto
- description: descrizione testuale del prodotto

| # | Column | Non-Null Count | Dtype |
|---|---|---|---|
| 0 | parent_asin | 137269 non-null | object |
| 1 | title | 137269 non-null | object |
| 2 | description | 137269 non-null | object |

Le altre informazioni che invece non sono state prese in considerazione sono:

- main_category: macrocategoria del prodotto
- average_rating: media del rating mostrata sulla pagina del prodotto
- rating_number: numero di recensioni del prodotto
- features: caratteristiche del prodotto
- price: prezzo in USD
- images: immagini del prodotto
- videos: video del prodotto
- store: nome del negozio che vende il prodotto
- category: sottocategoria del prodotto
- details: dettagli del prodotto
- bought_together: prodotti consigliati da comprare insieme

Queste due fonti informative sono state utilizzate nei diversi progetti per costruire sistemi di raccomandazione e per condurre una sentiment analysis.

## OBIETTIVI DELL'ANALISI

L'analisi esplorativa iniziale aveva come obiettivo quello di comprendere la struttura e le caratteristiche fondamentali del dataset, al fine di individuare eventuali criticità o squilibri nei dati e impostare correttamente le fasi successive del progetto.

In particolare, l'analisi si è concentrata su:

- Verifica dell'assenza di valori nulli, mancanti, nelle colonne rilevanti
- Statistiche descrittive:
    - 'Head', 'tail' e 'describe' nelle colonne rilevanti per avere una prima panoramica del dataset
    - Osservazione della distribuzione dei rating per identificare eventuali sbilanciamenti
    - Analisi della numerosità di recensioni per utente e per prodotto, anche per valutare la necessità di applicare un filtraggio sui dati
- Analisi di correlazione, su tre livelli:
    - Rating vs helpful_vote, per valutare la correlazione tra il rating dato e quanto è stata trovata utile la recensione. I rating più alti ricevono più voti utili?

- Rating vs verified_purchase, per valutare la correlazione tra il rating dato e la verifica dell'acquisto. Le recensioni con acquisto verificato sono mediamente più alte?
- Rating medio nel tempo, per valutare la correlazione tra il rating medio delle recensioni e il tempo (in anni, dal 1998 al 2023). Ci sono pattern nel tempo?

Uno degli scopi di questa fase era assicurarsi che la matrice di rating risultasse sufficientemente densa per l'addestramento dei sistemi di raccomandazione, e che il dataset fosse adatto a sostenere confronti robusti tra diversi approcci (collaborative filtering e content-based).

## RISULTATI DELL'ANALISI ESPLORATIVA

Nella fase iniziale è stata effettuata un'analisi esplorativa sul dataset delle *user reviews*, focalizzandosi principalmente sulle variabili essenziali. I principali risultati ottenuti sono stati i seguenti:

```
rating          0
parent_asin     0
user_id         0
dtype: int64
```

*Controllo dei valori mancanti*: non sono stati riscontrati *missing values* nelle colonne considerate.

*STATISTICHE DESCRITTIVE:*

| | rating | parent_asin | user_id |
|---|---|---|---|
| 0 | 4.0 | B07DK1H3H5 | AGCI7FAH4GL5FI65HYLKWTMFZ2CQ |
| 1 | 5.0 | B07SRWRH5D | AGCI7FAH4GL5FI65HYLKWTMFZ2CQ |
| 2 | 5.0 | B07MFMFW34 | AGXVBIUFLFGMVLATYXHJYL4A5Q7Q |
| 3 | 5.0 | B0BCHWZX95 | AFTC6ZR5IKNRDG5JCPVNVMU3XV2Q |
| 4 | 5.0 | B00HUWA45W | AFTC6ZR5IKNRDG5JCPVNVMU3XV2Q |

| | rating | parent_asin | user_id |
|---|---|---|---|
| 4624610 | 5.0 | B0C89J78ZW | AEGJO3XG3JGBVIJW64SDJ6BIYWHQ |
| 4624611 | 5.0 | B089F1BD4W | AHTYCU6NSHJ4BY7R2YNDIDGUXS6Q |
| 4624612 | 5.0 | B0BN942894 | AF4KDWDGCJSUEB7JESQZ6ZWD2LNA |
| 4624613 | 1.0 | B09SM83KRP | AG3BHCQJCY3MPNP3UKVAENSGQM2A |
| 4624614 | 5.0 | B01GOYVNJM | AGDC4SOFDQ3UXPXMEYJ57SMM3F4A |

Con *Head e tail* controllo le prime e ultime righe del dataset.

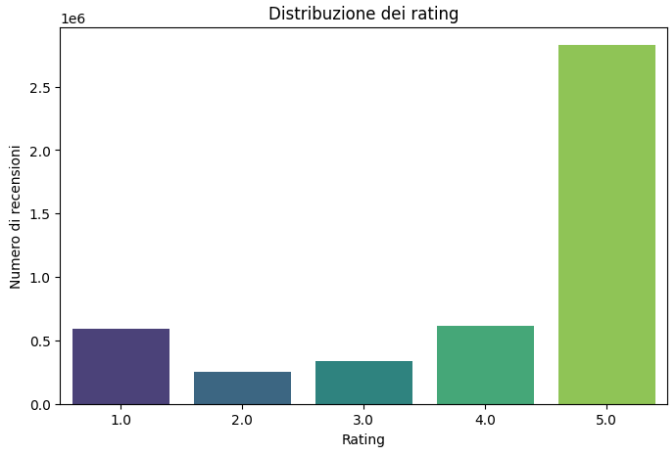
```
count    4.624615e+06
mean     4.047460e+00
std      1.430443e+00
min      1.000000e+00
25%      3.000000e+00
50%      5.000000e+00
75%      5.000000e+00
max      5.000000e+00
Name: rating, dtype: float64
```

Con *Describe*, relativo ai ratings, ottengo una panoramica sui ratings.

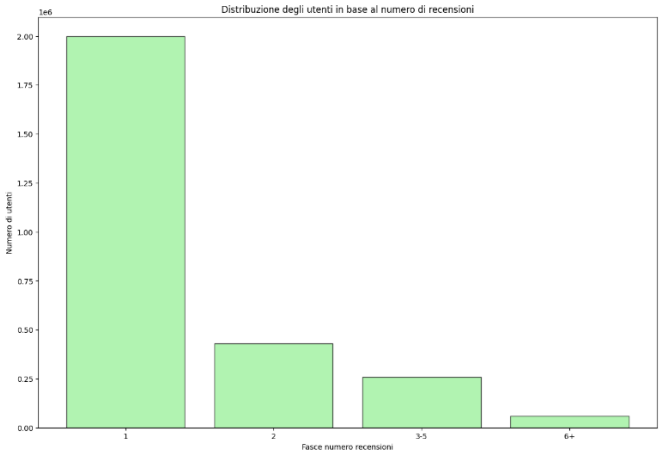
In particolare, noto che la media è intorno al 4,05, indicando una prevalenza di giudizi positivi, e che almeno il 50% delle recensioni corrispondo al punteggio massimo disponibile. Inoltre, la

deviazione standard, pari a 1,43, indica una discreta dispersione.
La distribuzione è chiaramente asimmetrica, concentrata verso punteggi positivi, il che può indicare bias positivo, soddisfazione generale del cliente o dinamiche tipiche delle recensioni online (come, per esempio, la tendenza a lasciare feedback solo quando molto soddisfatti).

| rating | |
|---|---|
| 5.0 | 2827881 |
| 4.0 | 617251 |
| 1.0 | 589519 |
| 3.0 | 340086 |
| 2.0 | 249878 |



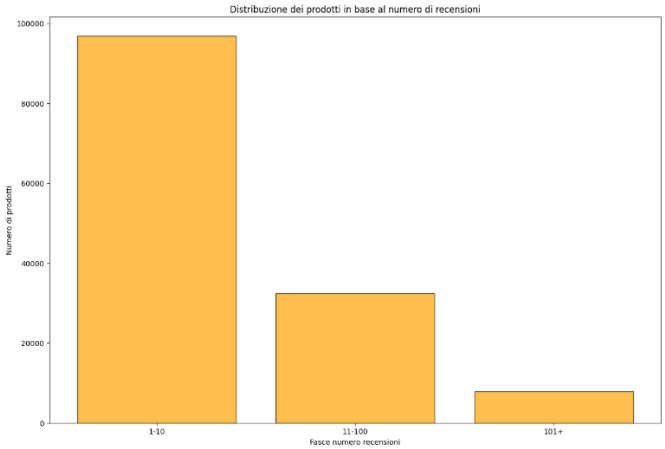

*Distribuzione dei rating*: evidenzia un forte picco sul valore 5, con frequenze decrescenti man mano che ci si avvicina al rating minimo (il rating 1 in realtà è comunque il terzo più rappresentato).




*Distribuzione degli utenti in base al numero di recensioni*: oltre 2 milioni di utenti hanno lasciato una sola recensione, e solo una piccola percentuale supera le 6 recensioni.

```
user_id
AHJRJCJMK3XVV4BSPBRAHIYEODWA    664
AGMWACNMAG74AXBF7IJ22IOZSZPA    596
AGIBXD3LM6HNDWWRTIOJHB5EKNFA    469
AEWLQYBQDYWWUWK6UHHTNWO5AHYA    425
AHEDJIDSPVYCB3GPRZKGO7YTK6XQ    346
                               ...
AHB7VCYPS6S6SXLTXFRMWTNWDVGA      1
AFEEN5MQUL2AXMIWADNK5SIJO6MA      1
AH6UTMLB6BHLXQVWJQ62VUM6K35Q      1
AF22KIFQVT2UFRECR6MOIHMYA5EQ      1
AGOH4TJVYVRHIGFDJYGR5GETCY6Q      1
Name: count, Length: 2766656, dtype: int64
```

Nell’immagine possiamo vedere una preview del conteggio delle recensioni per ogni utente, che mi ha aiutato anche nella scelta delle fasce nel grafico mostrato sopra.




*Distribuzione dei prodotti in base al numero di recensioni*: la maggioranza dei prodotti ha ricevuto meno di 10 recensioni, solo una bassa percentuale supera le 100 recensioni.

```
parent_asin
B01N3ASPNV    18105
B0BN942894    17310
B077GG9D5D    15594
B000N5Z2L4    13329
B0086VPUHI    12100
              ...
B09LR7VX11        1
B088QHR73N        1
B006TQVNBE        1
B07YR87WC7        1
B0C7W87NFX        1
Name: count, Length: 137249, dtype: int64
```

Nell’immagine possiamo vedere una preview del conteggio delle recensioni per ogni prodotto.

Per suddividere in fasce ho usato un criterio logico: la fiducia che l'utente può riporre nelle recensioni: 1-10, dati scarsi; 11-100, prodotto più consolidato; 101+, prodotto molto popolare.

Questa analisi ha permesso di comprendere la natura sbilanciata e sparsa del dataset, guidando le scelte successive di filtraggio e preprocessing, infatti, per i progetti base e intermedio, è stato applicato un filtro che considera utenti e prodotti con almeno 25 recensioni.

Questo passaggio è stato fondamentale sia per rendere i dati più significativi e quindi costruire sistemi di raccomandazione migliori, sia, in parte, per ovviare a limitazioni legate alle risorse computazionali disponibili nell’ambiente di esecuzione.

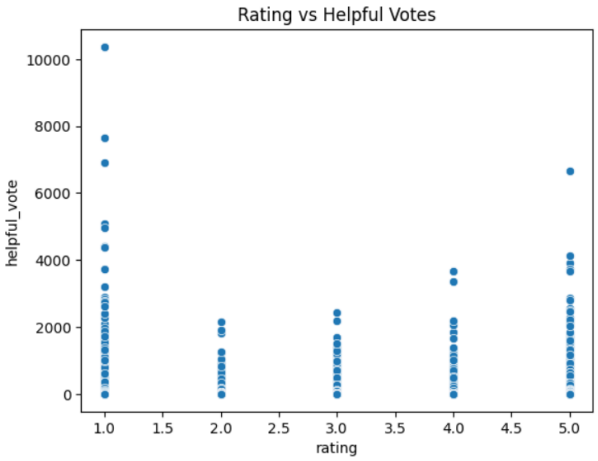
```
Numero righe prima del filtraggio: 4624615
Utenti unici: 2766656
Prodotti unici: 137249
Numero righe dopo il filtraggio: 112383
Utenti unici: 3195
Prodotti unici: 15061
```

Questo è il risultato dell'applicazione del filtro al dataset. Il subset così ottenuto conserva le caratteristiche essenziali, e anzi possiamo considerarlo più raffinato, per rispondere agli obiettivi dell'analisi.

*ANALISI DI CORRELAZIONE:*

*Correlazione rating vs helpful_vote:*

| | rating | helpful_vote |
|---|---|---|
| rating | 1.000000 | -0.031549 |
| helpful_vote | -0.031549 | 1.000000 |




il coefficiente di correlazione è -0.03, un valore molto vicino a zero, che indica una correlazione negativa debolissima, praticamente nulla. In sostanza il numero di voti utili non varia in modo significativo in funzione del rating. I prodotti con rating alti non ricevono sistematicamente più, o meno, voti utili rispetto a quelli con rating bassi.

Il grafico mostra dei punti molto dispersi, senza una forma chiara o una direzione evidente, con qualche outlier, ossia recensioni con tantissimi voti utili, ma questi sono sporadici. I voti utili sembrano distribuiti su tutti i livelli di rating, confermando l'assenza di correlazione forte.

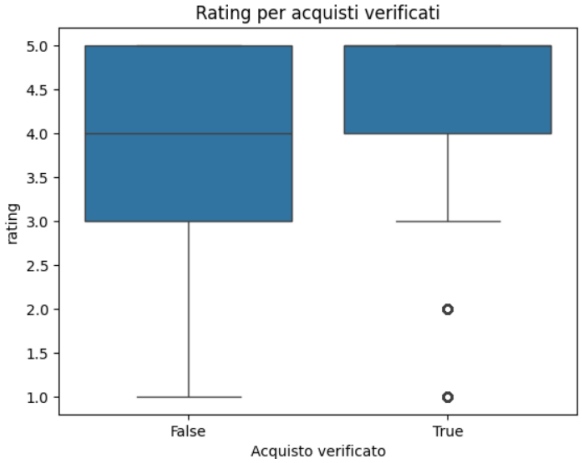
Questi dati ci suggeriscono che l'utilità percepita di una recensione non dipende dal rating assegnato, ma probabilmente da altri fattori come:

- Quanto è dettagliata o ben scritta la recensione
- Se contiene info pratiche o esperienze personali

- Quanto è recente (potrebbe influenzare la visibilità)

*Correlazione rating vs verified_purchase*:

| | rating | verified_purchase_num |
|---|---|---|
| rating | 1.000000 | 0.088801 |
| verified_purchase_num | 0.088801 | 1.000000 |



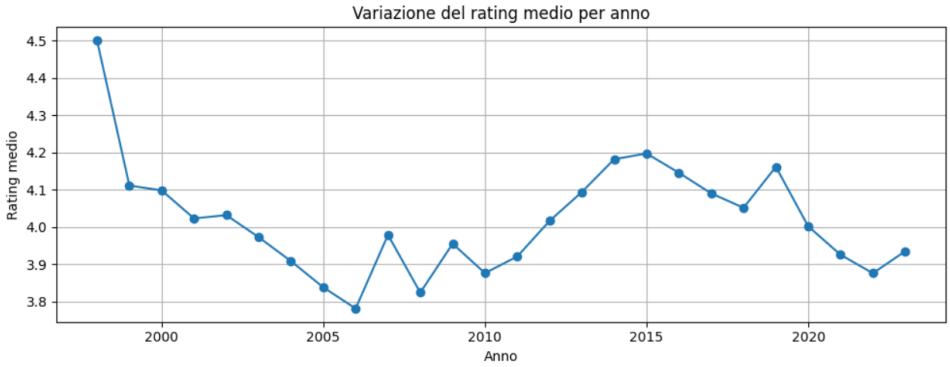

il coefficiente di correlazione è +0.088, un valore positivo ma molto debole, che indica che le recensioni con acquisto verificato tendono leggermente ad avere rating più alti, ma la relazione è quasi trascurabile.

Il boxplot mostra che le recensioni di acquisti verificati (True) hanno una mediana più alta e meno dispersione nei rating. Le recensioni non verificate (False) mostrano una maggiore variabilità, con rating più distribuiti tra 1 e 5. L'intervallo interquartile per le verificate è più stretto, segno che i giudizi sono più coerenti. Da segnalare che ci sono alcuni outlier tra le recensioni di acquisti verificati con rating basso, ma sono appunto casi isolati, trascurabili.

Questi risultati suggeriscono che gli utenti che hanno effettivamente acquistato il prodotto tendono a lasciare recensioni più positive e più coerenti, ma anche che le recensioni non verificate potrebbero includere opinioni meno affidabili o più estreme.

*Andamento del rating medio nel tempo*:

| | year | rating |
|---|---|---|
| 0 | 1998 | 4.500000 |
| 1 | 1999 | 4.111354 |
| 2 | 2000 | 4.098238 |
| 3 | 2001 | 4.023172 |
| 4 | 2002 | 4.031988 |
| 5 | 2003 | 3.972895 |
| 6 | 2004 | 3.908638 |
| 7 | 2005 | 3.838410 |
| 8 | 2006 | 3.781745 |
| 9 | 2007 | 3.979004 |
| 10 | 2008 | 3.824351 |
| 11 | 2009 | 3.955480 |
| 12 | 2010 | 3.876587 |
| 13 | 2011 | 3.921204 |
| 14 | 2012 | 4.016611 |
| 15 | 2013 | 4.094147 |
| 16 | 2014 | 4.182008 |
| 17 | 2015 | 4.197194 |
| 18 | 2016 | 4.145579 |
| 19 | 2017 | 4.090289 |
| 20 | 2018 | 4.051708 |
| 21 | 2019 | 4.161578 |
| 22 | 2020 | 4.001949 |
| 23 | 2021 | 3.925874 |
| 24 | 2022 | 3.876339 |
| 25 | 2023 | 3.935174 |




Nel 1998, il rating medio parte molto alto: 4.5, ma quel valore potrebbe riflettere un numero molto ristretto di recensioni. Dal 1999 al 2008, vediamo una discesa graduale e costante, con minimi intorno al 2005–2008, che potrebbero riflettere una maggiore severità dei consumatori o una diversa composizione dei prodotti recensiti. Tra il 2014 e il 2016, il rating medio risale sensibilmente con valori anche oltre il 4, segno di un periodo con giudizi più generosi o prodotti più soddisfacenti. Dal 2017 fino al 2023, si nota una nuova tendenza al calo: il rating medio torna sotto il 4.

La correlazione rating-tempo non è lineare, ma mostra oscillazioni cicliche. È difficile parlare di una vera correlazione statistica, ma sicuramente c'è una tendenza storica degna di nota.

Le seguenti potrebbero essere le cause di queste oscillazioni:

- Cambiamento nel comportamento degli utenti (più severi o più critici nel tempo)
- Evoluzione nel tipo di prodotti venduti (maggiore varietà porta a maggiore variabilità nella soddisfazione)
- Cambiamenti algoritmici o nei criteri di visibilità delle recensioni
- Aumento del numero totale di recensioni con conseguenti statistiche più stabili e meno influenzate da singoli outlier

# RISULTATI STEP BY STEP

In questa sezione vengono presentati, suddivisi per progetto e per step, i risultati ottenuti durante lo sviluppo e l'implementazione delle diverse componenti dei sistemi di raccomandazione e della classificazione del sentiment. Per ciascuno step vengono illustrati gli approcci utilizzati, le scelte metodologiche adottate e i principali risultati emersi, con particolare attenzione al confronto tra le diverse tecniche sperimentate.

Come detto nel capitolo precedente, per il progetto base e intermedio, è stato applicato un filtro sul dataset che considera solo gli utenti e i prodotti con almeno 25 recensioni. Le motivazioni e i risultati dell'applicazione del filtro sono stati chiariti sopra e, se necessario, verranno ripresi durante la spiegazione di alcuni risultati.

## PROGETTO BASE

L'analisi esplorativa, richiesta nello step 1 del progetto base, è già stata analizzata nel dettaglio nel capitolo precedente, quindi partiremo direttamente dallo step 2.

### STEP 2

L'obiettivo di questo step è stato individuare la configurazione ottimale per un sistema di raccomandazione basato su collaborative filtering con l'algoritmo K-Nearest Neighbors (K-NN), utilizzando la libreria Surprise.

Sono state testate diverse configurazioni variando:

- la tipologia di similarità: coseno e pearson
- la struttura: user-based o item-based
- il valore di K (numero di vicini): 10, 20 e 40, valori comuni di K

Per valutare le performance, il dataset filtrato è stato suddiviso in trainset e testset. La metrica utilizzata per confrontare le performance delle varie configurazioni è stata l'errore quadratico medio (MSE, Mean Squared Error) e la sua radice quadrata (RMSE, Root Mean Squared Error).

| | Similarity | User-Based | K | MSE | RMSE |
|---|---|---|---|---|---|
| **5** | cosine | False | 40 | 1.088697 | 1.043406 |
| **4** | cosine | False | 20 | 1.090581 | 1.044309 |
| **3** | cosine | False | 10 | 1.107331 | 1.052298 |
| **11** | pearson | False | 40 | 1.175995 | 1.084433 |
| **10** | pearson | False | 20 | 1.176120 | 1.084491 |
| **9** | pearson | False | 10 | 1.177174 | 1.084977 |
| **8** | pearson | True | 40 | 1.354978 | 1.164035 |
| **7** | pearson | True | 20 | 1.355152 | 1.164110 |
| **6** | pearson | True | 10 | 1.355311 | 1.164178 |
| **2** | cosine | True | 40 | 1.379206 | 1.174396 |
| **1** | cosine | True | 20 | 1.380975 | 1.175149 |
| **0** | cosine | True | 10 | 1.395429 | 1.181283 |

La configurazione ottimale individuata è risultata essere: item-based, similarità coseno, K = 40.

Questa configurazione ha prodotto il valore di RMSE più basso, risultando quindi la più accurata nella predizione dei rating nel testset.

Nella scelta dei K per la configurazione ottimale sono stati testati anche K superiori a 40 ma, come pronosticabile, K troppo grandi rischiano di peggiorare le performance perché vanno ad includere anche quei vicini non rilevanti.

## STEP 3

Dopo aver identificato la configurazione ottimale dell'algoritmo K-NN è stata costruita la matrice di rating completa utilizzando tale modello per predire i valori mancanti.

L'obiettivo dello step è riempire la matrice utente-prodotto con tutti i rating predetti per le coppie utente–item non presenti nei dati originali, così da poter generare raccomandazioni personalizzate anche per prodotti mai valutati in precedenza da un dato utente.

È stato utilizzato l'oggetto KNNBasic di Surprise, addestrato sul trainset completo, includendo tutti i dati filtrati. Per generare le predizioni su tutte le coppie possibili non osservate nel trainset, è stato costruito un testset artificiale (tutte le coppie utente–item mancanti).

```
# Prendo tutti gli user / item mancanti nel training set
trainset_users = set([trainset.to_raw_uid(u) for u in trainset.all_users()])
trainset_items = set([trainset.to_raw_iid(i) for i in trainset.all_items()])

# Ora genero tutte le possibili coppie user-item
coppie_possibili = []
for uid in trainset_users:
    for iid in trainset_items:
        if not trainset.knows_user(trainset.to_inner_uid(uid)) or not trainset.knows_item(trainset.to_inner_iid(iid)):
            continue
        if not trainset.ur[trainset.to_inner_uid(uid)] or iid in [trainset.to_raw_iid(i[0]) for i in trainset.ur[trainset.to_inner_uid(uid)]]:
            continue
        coppie_possibili.append((uid, iid))
print(f"Numero di rating da predire: {len(coppie_possibili)}")
```

Il numero totale di rating da predire è risultato molto elevato (oltre 45 milioni), rendendo l'operazione computazionalmente pesante. Per questo motivo è stata successivamente applicata una selezione di un campione casuale di utenti nelle fasi successive (step 5).

Anche se il risultato del filling non è stato esportato interamente in un nuovo dataframe (per motivi di efficienza), il modello ha generato internamente tutte le predizioni, rendendo possibile la costruzione di raccomandazioni personalizzate.

Nella seguente immagine possiamo vederne un esempio: predizioni fatte per un utente su dieci prodotti che non aveva mai valutato.

```
Numero di rating da predire: 45188592
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B00HGAWFJG', 4.2695370720530335)
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B07BWH4GX3', 5)
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B01F9B019Y', 4.2695370720530335)
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B000GPW2QO', 5)
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B003Q9RG9K', 5)
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B07QLZZSM8', 4.2695370720530335)
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B06XHMPPLH', 5)
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B0BGBHSYNW', 4.2695370720530335)
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B00HWMP0OU', 5)
('AE35KI75TRXMYJFK6AQBJBZEFLVQ', 'B073P8X6Z1', 5)
```

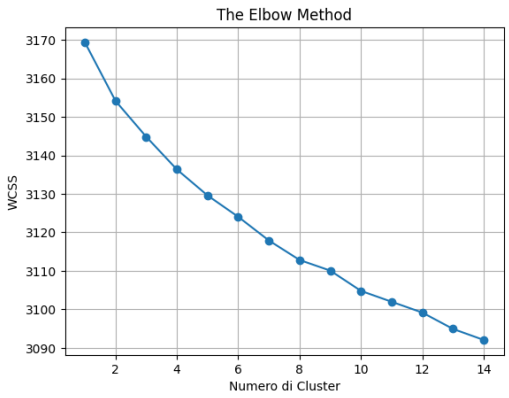
## STEP 4

Lo scopo di questo step è stato quello di segmentare gli utenti sulla base delle loro preferenze espresse tramite i rating, in modo da identificare gruppi omogenei di comportamento. Questo tipo di analisi può risultare utile per creare raccomandazioni differenziate per cluster o analizzare il comportamento degli utenti in maniera più strutturata.

È stata costruita una matrice utente–prodotto a partire dai rating reali, non predetti, dove ogni riga rappresenta un utente e ogni colonna un prodotto. I valori mancanti sono stati sostituiti con 0, assumendo che l'assenza di un rating implichi mancanza di interazione, si tratta di una scelta pratica e necessaria. I dati sono stati normalizzati per riga, centrando ogni utente rispetto alla propria media di rating.

È stato applicato l'algoritmo di clustering K-Means, utilizzando la similarità coseno come distanza.

Il numero di cluster k è stato determinato tramite Elbow Method, che ha suggerito una curvatura ottimale intorno a k = 8, anche se i valori del silhouette score risultavano molto bassi (massimo 0.0035), indicando una bassa separabilità tra i cluster.




```
K=2 → silhouette score: 0.0029
K=3 → silhouette score: 0.0031
K=4 → silhouette score: 0.0035
K=5 → silhouette score: 0.0028
K=6 → silhouette score: 0.0027
K=7 → silhouette score: 0.0027
K=8 → silhouette score: 0.0030
K=9 → silhouette score: 0.0026
K=10 → silhouette score: 0.0023
K=11 → silhouette score: 0.0026
K=12 → silhouette score: 0.0027
K=13 → silhouette score: 0.0026
K=14 → silhouette score: 0.0026
```

Il clustering è stato portato avanti e ha permesso di assegnare ogni utente a un cluster.

| parent_asin | user_id | cluster |
|---|---|---|
| 0 | AE25Y2LOSEKTPUJFDPWYJNYCQ7EQ | 4 |
| 1 | AE25ZDXYBK3LHKCZ7XUODANPME4A | 3 |
| 2 | AE27WQWQY6ZIJPFSWXMBDVHKXVVA | 4 |
| 3 | AE2A5TMJ6YE6ZNWUAFTC6P5XAHXA | 4 |
| 4 | AE2AZ2MNROPF33U6SS53VI22OXJA | 0 |

Il clustering ha sofferto della sparsità e omogeneità dei dati (molti utenti con pattern di rating simili e pochi rating diversi da 5), tuttavia, l'esperimento ha dimostrato la difficoltà insita nella segmentazione basata unicamente su rating; sarebbe stato infatti necessario integrare, per esempio, le caratteristiche degli utenti, considerando età, provenienza, sesso etc… o contenuti testuali come fatto poi nel progetto intermedio.

## STEP 5

In questo step, è stato implementato un sistema che, per ogni utente, seleziona i Top-N prodotti con i rating predetti più alti, ovvero quelli ritenuti più adatti ai suoi gusti.

È stato costruito un testset contenente tutte le coppie utente–item non ancora valutate, e su queste è stata effettuata la predizione dei rating con la configurazione ottimale. Per motivi computazionali, è stato selezionato un campione casuale di 100 utenti tra quelli presenti nel trainset.

Quindi, per ogni utente del campione sono stati considerati solo gli item non ancora recensiti e tra questi, sono stati selezionati i Top 5 con i rating predetti più alti, considerando che, a parità di rating, diamo peso al numero di recensioni totali del prodotto (priorità a prodotti più popolari).

Nella seguente immagine possiamo vedere l'head, i primi 10 utenti e i loro Top 5 prodotti raccomandati:

| | user_id | item_1 | rating_1 | reviews_1 | item_2 | rating_2 | reviews_2 | item_3 | rating_3 | reviews_3 | item_4 | rating_4 | reviews_4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | AFVYDI4KWVNBEKHFQDPLAMGCD67A | B07YBXFDYN | 5.00 | 296 | B07YBWT3PK | 5.00 | 211 | B01N3ASPNV | 5.00 | 200 | B087NNPYP3 | 5.00 | 175 |
| 1 | AG6V7FRKPA6ILOYTABMYMRLWFRAQ | B00K0NV5J2 | 5.00 | 139 | B004LLHFAW | 5.00 | 137 | B001E8WQKY | 5.00 | 115 | B072V478NR | 5.00 | 107 |
| 2 | AGGRZR7RQQXUDJIVEWFEBD2CNTVQ | B00KIWEMIG | 5.00 | 168 | B0088TN73M | 5.00 | 139 | B004LLHFAW | 5.00 | 137 | B07YBXFF99 | 5.00 | 133 |
| 3 | AEFJNOB2BIMMXHVX6GH3XUA6DONA | B09MGJXGBF | 5.00 | 42 | B07D3JSZ8F | 5.00 | 42 | B081243BT6 | 5.00 | 37 | B08P1NS2X1 | 5.00 | 35 |
| 4 | AEYYKAJ6GVAFJZEN6BMQPQAS6BHA | B009VUHVPM | 5.00 | 132 | B07624RBWB | 5.00 | 118 | B0062UOBW0 | 5.00 | 93 | B07PZ8NZSZ | 5.00 | 85 |
| 5 | AEGW3QTWMAFAB634BXMKKRZ5XPYQ | B009VUHVPM | 4.27 | 132 | B01GY35GIM | 4.27 | 110 | B087SHFL9B | 4.27 | 109 | B087NN2K41 | 4.27 | 100 |
| 6 | AFRULJ47WEAMKTW4S7YO6WAVFDFA | B07CPZ3PJG | 5.00 | 28 | B0007TFLLC | 5.00 | 27 | B07GRP33YM | 5.00 | 27 | B0C6L9RG7G | 5.00 | 25 |
| 7 | AHPJHWUFX7DFIVS5B3XNEK7JLSAQ | B001EYUPP6 | 5.00 | 37 | B001D7JEJM | 5.00 | 36 | B003N18O5Q | 5.00 | 35 | B00KTORA0K | 5.00 | 35 |
| 8 | AE6C3RB4MNC36ONGCU3TMSLMER6Q | B07YBXFDYN | 5.00 | 296 | B0086VPUHI | 5.00 | 265 | B07YBWT3PK | 5.00 | 211 | B00BGA9WK2 | 5.00 | 203 |
| 9 | AG32ISGSRQE4ZRTMDTPEHZ4GDVEA | B004RMK5QG | 5.00 | 147 | B008CZN458 | 5.00 | 141 | B001EYUPHO | 5.00 | 139 | B00K0NV5J2 | 5.00 | 139 |

Nella maggior parte dei casi, i rating predetti risultavano molto alti, confermando lo sbilanciamento positivo osservato nei dati originali. Solo una piccola parte degli utenti mostrava predizioni significativamente inferiori a 5. Le raccomandazioni sono risultate piuttosto omogenee, segno che l'approccio collaborative filtering, in presenza di dati molto positivi e ripetitivi, tende a suggerire i prodotti più popolari e già largamente apprezzati.

## STEP 6

In questo step è stato applicato un approccio alternativo, basato su matrix factorization, con l'obiettivo di confrontarne le performance con il modello K-NN sviluppato negli step precedenti.

È stato utilizzato l'algoritmo SVD, dalla libreria Surprise, che permette di fattorizzare la matrice utente–item in uno spazio latente di dimensioni ridotte.

Il modello è stato allenato sullo stesso trainset e valutato sullo stesso testset usati per il K-NN, al fine di garantire un confronto equo. Sono state utilizzate le metriche MSE e RMSE per valutare le performance predittive.

```
CONFRONTO FINALE
KNN → RMSE: 1.0434 | MSE: 1.0887
SVD → RMSE: 0.9448 | MSE: 0.8926
```

Come si può osservare, il modello SVD ha ottenuto risultati migliori in entrambe le metriche: MSE e RMSE. Questo è coerente con la natura dell'algoritmo, che riesce a cogliere pattern latenti tra utenti e prodotti, superando alcune delle limitazioni dell'approccio K-NN, che si basa su similarità diretta.

La Matrix Factorization è particolarmente efficace in contesti con grandi volumi di dati e pattern impliciti tra utenti e prodotti, tuttavia, a differenza del K-NN, i modelli a fattorizzazione non offrono interpretabilità diretta in termini di similarità tra utenti o prodotti. Il confronto ha evidenziato un miglioramento netto nelle performance, confermando la validità dell'approccio SVD in contesti reali.

# PROGETTO INTERMEDIO

## STEP 1

Lo scopo di questo step è stato quello di preparare i dati testuali contenuti nel metadata degli item, in particolare 'title' e 'description', al fine di poterli utilizzare per costruire un sistema di raccomandazione content-based.

Dal dataset contenente le informazioni testuali sui prodotti, sono stati selezionati:

- parent_asin: identificativo univoco del prodotto
- title: titolo del prodotto
- description: descrizione testuale

I campi 'title' e 'description' sono stati concatenati in un'unica colonna chiamata 'text', allo scopo di creare un'unica rappresentazione testuale del prodotto. Questa fase ha permesso di semplificare il successivo processamento linguistico.

È stata implementata una funzione di pulizia del testo basata su tecniche di Natural Language Processing. In particolare, sono stati applicati i seguenti passaggi:

- Conversione in minuscolo
- Rimozione di punteggiatura e caratteri speciali
- Rimozione dei numeri
- Tokenizzazione del testo
- Rimozione delle stopwords
- Stemmer (riduzione delle parole alla loro radice)
- Eliminazione di spazi multipli

Nella seguente immagine possiamo vedere il risultato della pulizia e processing dei testi:

```
7         extremer soft touch top shell front hous facep...
11        turbo super stunt squad nintendo ds product de...
15        warhamm dawn war game year pc manufactur game ...
18                                                  eeekit
                                  ...
118595    dc superman fun pack lego dimens mysteri power...
119799    die light stay human playstat twenti year ago ...
123440    kiwi design ultrasoft clipon headphon compat q...
128144                 chronicl riddick escap butcher bay pc
133855        thrustmast ferrari nrl wheel stand lite bundl
Name: text_clean, Length: 15061, dtype: object>
```

## STEP 2

Una volta puliti i campi testuali (title + description) degli item, è stato necessario trasformare il testo in una rappresentazione numerica, ovvero un embedding, che potesse essere utilizzata per calcolare la similarità tra prodotti. A tal fine, sono state utilizzate due tecniche:

1. TFIDF (Term Frequency - Inverse Document Frequency), tecnica basata sulla frequenza delle parole.
2. Transformers, tecnica neurale basata su modelli pre-addestrati.

Embedding con TFIDF:

È stato utilizzato lo strumento TfidfVectorizer della libreria scikit-learn per trasformare il testo in una matrice, in cui ogni riga rappresenta un prodotto e ogni colonna un termine. Il valore rappresenta l'importanza relativa del termine nel prodotto rispetto al corpus.

Sono stati scelti i seguenti parametri, ottimizzati in base alle dimensioni del dataset:

- max_features = 5000 → si considerano solo le 5000 parole più informative
- min_df = 5 → una parola deve comparire in almeno 5 prodotti
- max_df = 0.95 → esclude parole troppo frequenti (quelle presenti nel 95% dei prodotti)

TFIDF shape: (15061, 5000)

| parent_asin | aa | aaa | ab | abandon | abduct | abe | abil | abl | aboard | absolut | ... | zelda | zenimax | zero | zip | zipper | zombi | zone | zoo | zoom | zumba |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B00Z9TLVK0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 |
| B07H93H878 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 |
| B00BJH85SW | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.052455 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 |
| B001EYUX4Y | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.045893 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.124815 | 0.0 |
| B01N9HGQTM | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 |

5 rows × 5000 columns

La matrice risultante (15.061: prodotti unici, 5000: termini), presenta molti zeri (matrice sparsa), pronosticabile e naturale in questo tipo di rappresentazioni.

Embedding con Transformers:

È stato utilizzato un modello pre-addestrato di tipo BERT (encoder-only). Nello specifico è stato impiegato all-MiniLM-L6-v2 dalla libreria sentence-transformers. Ogni riga di testo è stata passata al modello per ottenere un vettore denso di 384 dimensioni, che rappresenta semanticamente il contenuto del prodotto. Il modello è stato applicato su tutti i prodotti con testo pulito.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 374 | 375 | 376 | 377 | 378 | 379 | 380 | 381 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| parent_asin | | | | | | | | | | | | | | | | | | | |
| B00Z9TLVK0 | -0.049201 | 0.016514 | -0.016202 | -0.117681 | -0.071885 | 0.118582 | 0.147196 | 0.012877 | 0.053295 | 0.122205 | ... | 0.072063 | 0.038335 | 0.023418 | 0.076885 | 0.003659 | -0.067073 | 0.094093 | 0.005379 |
| B07H93H878 | -0.007167 | -0.041943 | 0.034650 | 0.031046 | 0.044565 | -0.019741 | 0.066344 | 0.021511 | -0.054020 | 0.063921 | ... | 0.019827 | 0.042098 | 0.012906 | -0.046862 | 0.049012 | 0.041983 | 0.087601 | 0.076375 |
| B00BJH85SW | -0.103978 | -0.047863 | -0.011609 | -0.042338 | -0.052997 | 0.059457 | 0.002432 | 0.038821 | -0.096055 | -0.021072 | ... | 0.027572 | 0.097387 | -0.018617 | -0.028803 | 0.002629 | 0.001190 | 0.010532 | -0.031860 |
| B001EYUX4Y | 0.009404 | -0.022230 | -0.023390 | -0.045710 | -0.080369 | 0.065583 | 0.012872 | -0.008100 | -0.064673 | 0.032947 | ... | -0.020327 | 0.021483 | -0.002586 | 0.101969 | 0.039033 | 0.008975 | 0.146602 | -0.108721 |
| B01N9HGQTM | -0.023801 | 0.040475 | 0.062559 | 0.006697 | 0.043779 | -0.113267 | 0.165839 | 0.091189 | 0.037370 | -0.062194 | ... | 0.067761 | 0.045697 | -0.004716 | 0.018272 | -0.077087 | 0.023157 | 0.148370 | 0.003346 |

5 rows × 384 columns

La matrice risultante è densa e permette di calcolare similarità semantiche tra prodotti in modo molto più sofisticato rispetto a TFIDF. Ogni valore rappresenta quanto fortemente, positivamente o negativamente, una determinata caratteristica semantica è presente nel testo del prodotto.

## STEP 3

In questo step è stato costruito un sistema di raccomandazione content-based che suggerisce prodotti simili in base alle loro caratteristiche testuali, sfruttando gli embeddings ottenuti con le due tecniche precedenti: TFIDF e Transformers.

L'idea è quella di calcolare la similarità tra i prodotti e, per ogni utente, raccomandare item simili a quelli che ha già valutato positivamente.

1. Costruzione della matrice di similarità tra prodotti:

Per entrambi gli embeddings, TFIDF e transformers, è stata calcolata una matrice di similarità coseno tra tutti i prodotti. Questo ha permesso di ottenere, per ogni prodotto, i suoi "vicini più simili".

2. Raccomandazioni per ogni utente:

Per ciascun utente (da un campione casuale di 100 utenti, identico a quello del progetto base), sono stati selezionati gli item già recensiti con valutazioni elevate e per ogni item valutato, sono stati individuati i k vicini più simili (K = 5), pesando la similarità con il rating dato dall'utente. È stata costruita una "classifica" degli item, escludendo quelli già valutati, con rating predetti stimati tramite media pesata tra similarità e valutazione.

3. Top-N raccomandazioni

Per ogni utente, sono stati selezionati i Top 5 item con i rating predetti più alti, e, come nel progetto base, a parità di rating si preferisce prendere prodotti con il maggior numero di recensioni. È stato anche applicato un filtro per assicurarsi che i valori predetti rimanessero nel range [1, 5], di fatto normalizzandosi.

## STEP 4

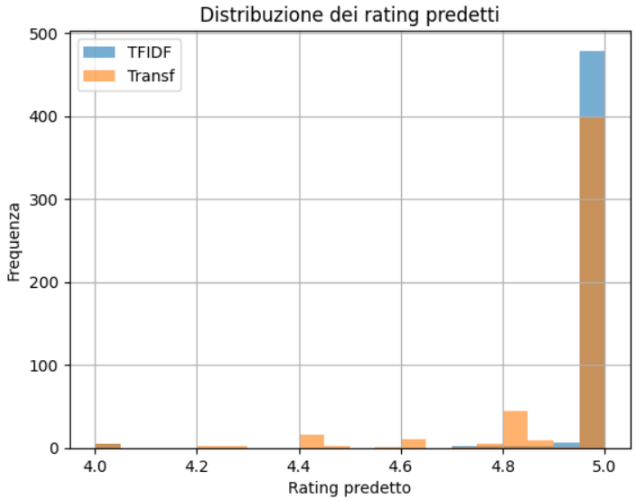
Nell'ambito del sistema di raccomandazione content-based, sono stati confrontati due approcci di embedding testuale: TFIDF, una tecnica basata sulla frequenza, e Transformers, una tecnica neurale.

Differenze concettuali:

- TFIDF (Term Frequency-Inverse Document Frequency) valuta l'importanza relativa di ciascuna parola nei testi, producendo una rappresentazione sparsa e focalizzata su termini specifici.
- Transformers, al contrario, produce vettori densi che catturano il significato complessivo del testo, sfruttando un contesto semantico più profondo.

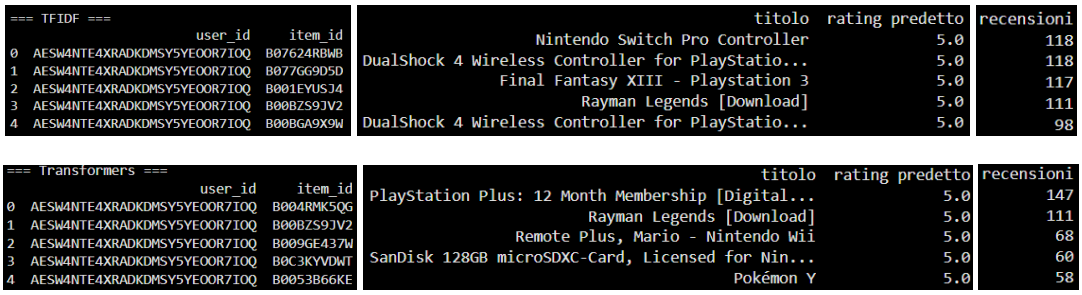
Per valutare il grado di convergenza tra le due tecniche, è stato calcolato l'overlap medio dei Top 5 item raccomandati per ogni utente nel campione (ossia quanti item mediamente coincidevano nelle raccomandazioni proposte usando le due tecniche), risultato essere pari a 0.15, un valore basso, che conferma che le due tecniche generano raccomandazioni diverse.

È stata poi analizzata la distribuzione dei rating predetti, dove entrambi i metodi mostrano una tendenza generale a predire rating elevati, compresi tra 4 e 5, coerentemente con la distribuzione sbilanciata del dataset.




Tuttavia, mentre TFIDF ha mostrato predizioni più concentrate verso il massimo, 5, la tecnica neurale Transformers ha prodotto valori leggermente più variabili, suggerendo una maggiore sensibilità alle differenze semantiche tra i prodotti.

Qui nell'immagine i risultati di un utente distinti, in ordine, per TFIDF e Transformers:



=== TFIDF ===

| | user_id | item_id | titolo | rating predetto | recensioni |
|---|---|---|---|---|---|
| 0 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B07624RBWB | Nintendo Switch Pro Controller | 5.0 | 118 |
| 1 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B077GG9D5D | DualShock 4 Wireless Controller for PlayStatio... | 5.0 | 118 |
| 2 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B001EYUSJ4 | Final Fantasy XIII - Playstation 3 | 5.0 | 117 |
| 3 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B00BZS9JV2 | Rayman Legends [Download] | 5.0 | 111 |
| 4 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B00BGA9X9W | DualShock 4 Wireless Controller for PlayStatio... | 5.0 | 98 |

=== Transformers ===

| | user_id | item_id | titolo | rating predetto | recensioni |
|---|---|---|---|---|---|
| 0 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B004RMK5QG | PlayStation Plus: 12 Month Membership [Digital... | 5.0 | 147 |
| 1 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B00BZS9JV2 | Rayman Legends [Download] | 5.0 | 111 |
| 2 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B009GE437W | Remote Plus, Mario - Nintendo Wii | 5.0 | 68 |
| 3 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B0C3KYVDWT | SanDisk 128GB microSDXC-Card, Licensed for Nin... | 5.0 | 60 |
| 4 | AESW4NTE4XRADKDMSY5YEOOR7IOQ | B0053B66KE | Pokémon Y | 5.0 | 58 |

Già da questo singolo esempio possiamo notare come nel caso della tecnica TFIDF ci siano due parole, "Playstation" e "Controller", ripetute ben 3 volte nelle Top 5 raccomandazioni, indice del fatto che il sistema ha suggerito item con termini ricorrenti e simili, letteralmente, ai preferiti dall'utente.

Al contrario possiamo vedere che nel caso della tecnica Transformers l'utente ha ricevuto delle raccomandazioni ben diversificate, riuscendo a cogliere similitudini semantiche più profonde, che vanno oltre la semplice condivisione di parole esplicite.

Possiamo concludere quindi che l'approccio basato su Transformers sembra fornire raccomandazioni più raffinate e personalizzate, distinguendo meglio tra prodotti anche all'interno di categorie simili.

## STEP 5

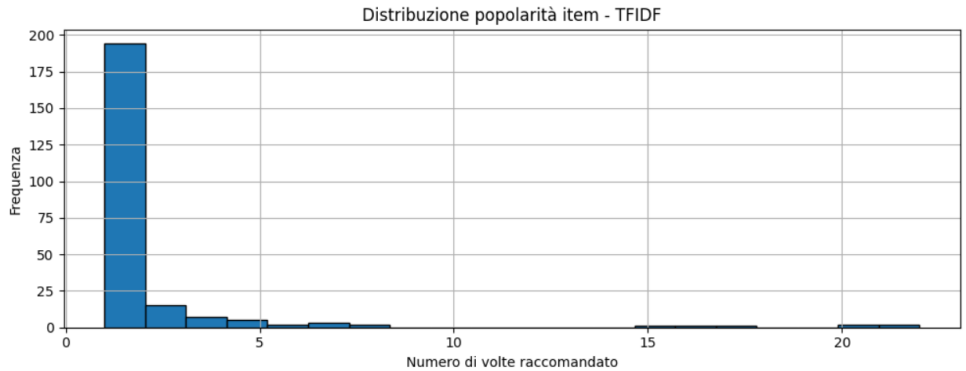
In questo step il confronto è stato effettuato tra il sistema content-based (CB) e il sistema collaborative filtering (CF) sviluppato nel progetto base.

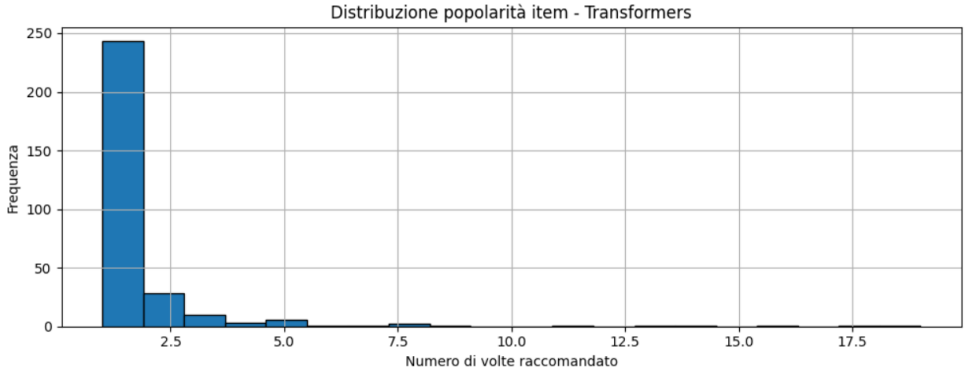
Anche in questo caso è stato calcolato l'overlap medio tra raccomandazioni, risultato essere pari a:

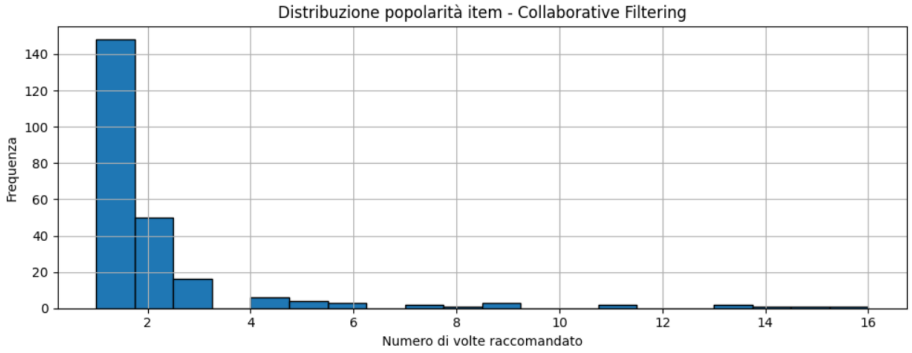
- 0.01 nel caso di confronto tra il sistema di raccomandazione content-based con embeddings generati con tecnica TFIDF e il sistema di raccomandazione collaborative filtering
- 0,02 nel caso di confronto tra il sistema di raccomandazione content-based con embeddings generati con tecnica Transformers e il sistema di raccomandazione collaborative filtering

Valori che confermano come i due approcci, CF e CB, forniscano raccomandazioni completamente diverse.

Nei grafici che seguono possiamo notare come nel sistema CB, distinto nelle due tecniche di generazione degli embeddings, ci sia maggiore diversità di prodotti raccomandati, infatti, quasi la metà di tutte le raccomandazioni includono prodotti distinti dagli altri (raccomandati una sola volta), rispetto al sistema di raccomandazione CF dove gli stessi prodotti sono consigliati molte volte agli utenti.






Il sistema collaborative filtering, nel nostro caso item-based, tende a suggerire prodotti simili a quelli già valutati positivamente dall'utente, sulla base dei comportamenti comuni di altri utenti, di conseguenza, è spesso influenzato dalla popolarità degli item e dalla densità dei dati, mentre il sistema content-based punta invece a suggerire prodotti simili a quelli già apprezzati dall'utente, anche se meno noti, sfruttando, nel nostro caso, informazioni testuali. In altre parole, il CF si è basato su comportamenti collettivi, il CB sul contenuto oggettivo.

Sappiamo inoltre che, nella pratica, i due sistemi possono coesistere grazie all'adozione di approcci ibridi, capaci di combinare i punti di forza di entrambe le strategie. In particolare, il sistema content-based risulterebbe prezioso in tutti quei casi di cold-start, ovvero quando sulla piattaforma arrivano nuovi utenti oppure viene lanciato un nuovo prodotto. Da un lato, infatti, il nuovo utente non ha ancora espresso preferenze; dall'altro, l'item non ha ricevuto recensioni o valutazioni, rendendo difficile l'utilizzo del sistema di collaborative filtering in modo autonomo.

# PROGETTO AVANZATO

## STEP 1

Per il progetto avanzato, l'analisi si è concentrata sui testi delle recensioni scritte dagli utenti, utilizzando i campi 'title' e 'text' contenuti nel dataset delle *user reviews*. L'obiettivo di questo step è stato il preprocessing dei dati testuali, per renderli idonei a un successivo embedding e classificazione del sentiment.

È stata creata una nuova colonna, 'review_text', ottenuta dalla concatenazione di title e text. Successivamente, è stato eseguito un processo di pulizia linguistica, che ha incluso:

- Conversione del testo in minuscolo;
- Rimozione di punteggiatura, simboli e numeri;
- Eliminazione delle stopwords;
- Tokenizzazione e lemmatizzazione dei termini;
- Rimozione di spazi multipli o indesiderati.

Il testo risultante è stato salvato nella nuova colonna ‘review_text_clean’, che rappresenta il contenuto testuale “pulito”, normalizzato, su cui si baseranno le successive fasi di embedding e classificazione del sentiment.

Questa attività ha consentito di ridurre la variabilità linguistica e di standardizzare il formato delle review, rendendo più efficace l’analisi semantica nei passaggi successivi.

Nella seguente immagine possiamo vedere il risultato della pulizia e processing dei testi:

```
0    pretty sexual fav im playing p interesting uni...
1    good bit slow nostalgic fun bit slow hope dont...
2    order kid really enjoyed playing pc game order...
3    great alt pro controller work great use batter...
4    solid product would recommend anyone looking a...
```

## STEP 2

Una volta completata la fase di pulizia testuale, per ragioni computazionali e di rappresentazione ho considerato solo un campione di tutte le review, pari a 40.000.

Ho poi proceduto con la trasformazione del testo in rappresentazioni numeriche (embedding), utilizzando due approcci distinti:

TFIDF:

La prima tecnica utilizzata è il classico approccio basato sulla frequenza, tramite il modello TFIDF, implementato con Scikit-Learn.
I parametri adottati sono stati:

- max_features = 5000: massimo numero di termini considerati;
- min_df = 15: esclusi i termini presenti in meno di 15 recensioni
- max_df = 0.95: esclude parole troppo frequenti (quelle presenti nel 95% dei prodotti)

Il risultato è una matrice sparsa con 40.000 righe (recensioni del campione) e 5.000 colonne (termini).

TFIDF shape: (40000, 5000)

| parent_asin | aa | aaa | ab | ability | able | absolute | absolutely | absurd | abuse | ac | ... | yr | zelda | zero | zip | zipper | zombie | zone | zoo | zoom | zumba |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B00LV8IF8O | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| B003EYV6ZC | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| B01N4NTNO2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| B00LV8IF8O | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| B00R6IMBYW | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

5 rows × 5000 columns

Transformers:

La seconda tecnica ha utilizzato un approccio neurale basato su modelli pre-addestrati, in particolare un modello Sentence-BERT tramite la libreria SentenceTransformers. Il modello scelto, sentence-transformers/all-MiniLM-L6-v2, ha prodotto embeddings densi da 384 dimensioni per ciascun testo.

Il risultato è una matrice densa con 40.000 righe (recensioni del campione) e 384 colonne (dimensioni).

Transformers shape: (40000, 384)

| parent_asin | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 374 | 375 | 376 | 377 | 378 | 379 | 380 | 381 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B00LV8IF8O | -0.102825 | 0.007101 | -0.042542 | -0.085328 | 0.001586 | 0.042518 | 0.049935 | -0.029095 | -0.130061 | 0.126028 | ... | 0.044087 | 0.061112 | 0.008239 | 0.002317 | 0.027677 | -0.005778 | 0.017276 | -0.066726 |
| B003EYV6ZC | -0.046288 | -0.003909 | 0.040954 | -0.070055 | -0.072025 | 0.026521 | 0.109352 | -0.029236 | 0.018429 | 0.040137 | ... | 0.174877 | 0.099864 | -0.008840 | 0.088912 | -0.052635 | 0.091710 | 0.043290 | 0.044920 |
| B01N4NTNO2 | -0.067015 | 0.102699 | 0.058834 | -0.036538 | -0.047689 | -0.066083 | 0.056369 | -0.028590 | -0.063117 | -0.022633 | ... | 0.018947 | -0.048282 | -0.042199 | 0.045464 | 0.021686 | 0.029550 | -0.025983 | 0.109893 |
| B00LV8IF8O | -0.006637 | 0.037771 | -0.002374 | -0.034469 | -0.053515 | -0.007128 | 0.086956 | 0.019114 | -0.029804 | 0.027519 | ... | 0.122656 | 0.032554 | -0.000325 | 0.073568 | 0.096665 | 0.031350 | 0.088017 | 0.003671 |
| B00R6IMBYW | -0.077074 | -0.084953 | 0.043919 | -0.061731 | -0.006320 | -0.042999 | 0.115134 | 0.032760 | -0.058699 | 0.054921 | ... | 0.006092 | 0.083705 | -0.063400 | -0.025767 | 0.103932 | 0.025437 | 0.073414 | -0.044573 |

5 rows × 384 columns

Entrambe le rappresentazioni, TFIDF e Transformer, sono state successivamente utilizzate per l'addestramento di modelli di classificazione del sentiment nello step successivo.

## STEP 3

L'obiettivo di questo step è stato quello di classificare il sentiment espresso nelle recensioni degli utenti. Il sentiment è stato definito sulla base del rating associato a ciascuna recensione:

- Rating 1-2 → Sentiment negativo
- Rating 3 → Sentiment neutro
- Rating 4-5 → Sentiment positivo

La colonna 'review_text_clean', trasformata in rappresentazioni numeriche tramite i due metodi di embeddings, qui viene utilizzata come input per gli algoritmi di classificazione.

In particolare, sono stati testati due modelli:

- Logistic Regression
- Random Forest

Entrambi i modelli sono stati addestrati e valutati separatamente per ciascun embedding.

Le performance sono state misurate tramite:

- Accuracy
- Precision
- Recall
- F1-score
- Support (numero di istanze per ciascuna classe)

TFIDF - Logistic Regression

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| neg | 0.79 | 0.73 | 0.76 | 1436 |
| neu | 0.53 | 0.14 | 0.22 | 586 |
| pos | 0.89 | 0.97 | 0.93 | 5978 |
| accuracy | | | 0.87 | 8000 |
| macro avg | 0.74 | 0.61 | 0.63 | 8000 |
| weighted avg | 0.85 | 0.87 | 0.85 | 8000 |

TFIDF - Random Forest

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| neg | 0.81 | 0.62 | 0.70 | 1436 |
| neu | 0.82 | 0.09 | 0.17 | 586 |
| pos | 0.86 | 0.98 | 0.92 | 5978 |
| accuracy | | | 0.85 | 8000 |
| macro avg | 0.83 | 0.57 | 0.60 | 8000 |
| weighted avg | 0.85 | 0.85 | 0.82 | 8000 |

Transformers - Logistic Regression

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| neg | 0.72 | 0.70 | 0.71 | 1436 |
| neu | 0.66 | 0.11 | 0.20 | 586 |
| pos | 0.88 | 0.96 | 0.92 | 5978 |
| accuracy | | | 0.85 | 8000 |
| macro avg | 0.75 | 0.59 | 0.61 | 8000 |
| weighted avg | 0.83 | 0.85 | 0.83 | 8000 |

Transformers - Random Forest

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| neg | 0.76 | 0.41 | 0.53 | 1436 |
| neu | 0.90 | 0.02 | 0.03 | 586 |
| pos | 0.81 | 0.98 | 0.89 | 5978 |
| accuracy | | | 0.81 | 8000 |
| macro avg | 0.82 | 0.47 | 0.48 | 8000 |
| weighted avg | 0.81 | 0.81 | 0.76 | 8000 |

Dall'analisi dei classificatori emerge una tendenza piuttosto chiara:

- Classe positiva (rating 4–5):

classe fortemente dominante nel dataset (quasi il 75% delle review), e ciò si riflette nelle performance dei modelli, che ottengono valori elevati di precision e recall su questa classe, indipendentemente dalla tecnica di embedding o dal classificatore utilizzato. Ad esempio, con TFIDF e Logistic Regression si ottiene un F1-score di 0.93, mentre con Transformers si mantiene stabile su 0.92.

- Classe neutra (rating 3) risulta invece la più problematica:

in tutti i casi, il recall è molto basso, tra 0.02 e 0.14, e il F1-score oscilla tra 0.03 e 0.22, a indicare che il modello fatica a distinguere le recensioni "medie".

Questo comportamento è probabilmente dovuto alla scarsa rappresentatività della classe nel dataset (circa il 7% del totale), che porta i modelli a penalizzare la predizione di questa categoria per ottimizzare la performance complessiva.

- Classe negativa (rating 1–2):

mostra performance più stabili rispetto alla neutra, ma comunque inferiori rispetto alla classe positiva. I valori di F1-score si aggirano tra 0.70 e 0.76 nei migliori casi (TFIDF e Logistic Regression), ma crollano a 0.53 con Transformers e Random Forest, segno che la sensibilità dei modelli su questa classe è comunque limitata.

Andando poi a confrontare gli embedding:

- TFIDF sembra garantire una maggiore stabilità e precisione sulle classi minori, soprattutto se abbinato a Logistic Regression.
- Transformers, pur ottenendo ottimi risultati sulla classe positiva, mostra fluttuazioni più marcate sulle classi minori, in particolare con Random Forest, dove il recall della classe neutra si azzera praticamente.

Confrontando invece i modelli di classificazione:

- Logistic Regression si dimostra più bilanciato e coerente nei risultati, in particolare sulle classi minoritarie.
- Random Forest ottiene buone performance su precision e recall delle classi maggiori, ma può essere altamente instabile sulle minoranze, come evidenziato con Transformers.

L'elevata accuracy (~85%) riportata da quasi tutti i modelli è fortemente influenzata dall'alta incidenza della classe positiva. In presenza di un dataset sbilanciato, come in questo caso, le metriche più adatte per valutare le performance dei modelli sono la macro average (media) e la weighted average (media pesata) del F1-score, poiché forniscono una visione complessiva più bilanciata rispetto alla semplice accuracy.

Nel complesso il miglior compromesso si ottiene con TFIDF e Logistic Regression, grazie al buon bilanciamento tra classi, soprattutto nelle performance sulla categoria negativa, mentre possiamo dire che gli embeddings con tecnica Transformer, pur sofisticati, non hanno mostrato chiari vantaggi nel contesto di classificazione supervisionata su questo tipo di testo, probabilmente anche per via della scarsità di esempi "neutri" da apprendere.

# CONCLUSIONE E INTERPRETAZIONE SINTETICA DEI RISULTATI

Il progetto ha affrontato il problema della raccomandazione di prodotti nella categoria *Video Games* di Amazon attraverso due approcci principali, collaborative filtering e content-based, a cui si è affiancata un'analisi del sentiment sulle recensioni testuali degli utenti.

Nel progetto base è stato sviluppato un sistema di raccomandazione basato su collaborative filtering, ottenendo le migliori performance predittive nella configurazione item-based con similarità coseno e K=40 (RMSE = 1.04). La successiva segmentazione degli utenti tramite K-Means ha mostrato limiti interpretativi, principalmente legati alla natura sparsa della matrice e alla bassa qualità dei cluster generati. È stato quindi realizzato un sistema di raccomandazione personalizzato per ogni utente, che successivamente è stato confrontato con un approccio alternativo basato su Matrix Factorization, che ha mostrato performance simili, anche leggermente migliori in termini di accuratezza.

Nel progetto intermedio, è stato introdotto un approccio content-based basato sull'analisi testuale dei prodotti, combinando titolo e descrizione dei prodotti. Gli embedding sono stati generati tramite due tecniche ben differenti: TFIDF e Transformers. Le raccomandazioni sono state costruite a partire dalla similarità tra item, considerando le preferenze espresse dagli utenti. Il confronto tra le due tecniche ha mostrato un basso livello di overlap (= 0.15), evidenziando differenze nel tipo di raccomandazioni prodotte. Il confronto con il sistema collaborative filtering ha invece mostrato un overlap quasi nullo, a conferma della natura complementare tra i due approcci. Inoltre, l'analisi della popolarità ha evidenziato una maggiore tendenza del sistema collaborative filtering a raccomandare ripetutamente gli stessi prodotti a più utenti, mostrando una minore diversificazione rispetto al content-based.

Nel progetto avanzato si è affrontato il problema della classificazione del sentiment delle recensioni, partendo dal testo delle recensioni, titolo e corpo. Le recensioni sono state etichettate come positive, neutre o negative sulla base del rating. I modelli di classificazione addestrati su embedding TFIDF e Transformers, abbinati a classificatori come Logistic Regression e Random Forest, hanno mostrato buone prestazioni complessive, raggiungendo livelli di accuracy superiori all'80%. Tuttavia, la classe neutra è stata riconosciuta con maggiore difficoltà, a causa dello sbilanciamento intrinseco del dataset. Altre metriche, come F1-score, sia macro che weighted average, confermano la bontà dei modelli nei confronti delle classi principali, fornendo perciò una valutazione sicuramente più rappresentativa, bilanciata e attendibile.

In conclusione, l'integrazione di metodi collaborative filtering, content-based e di sentiment analysis ha permesso di esplorare a fondo il problema della raccomandazione, mostrando come approcci differenti possano offrire prospettive diverse ma anche complementari. Ogni approccio ha evidenziato punti di forza e limiti specifici, e l'analisi combinata ha consentito una comprensione più profonda dei dati, delle preferenze degli utenti e del contenuto informativo disponibile.

Il lavoro ha messo in luce l'importanza della quantità ma soprattutto qualità dei dati, della rappresentazione testuale e della scelta delle metriche di valutazione nell'ottenere risultati affidabili e utili in contesti reali di raccomandazione.