

PROGETTO IS - MODULO FIA

Help Seller

Presentazione di Alex Lysytsya

Indice dei contenuti



01 Introduzione

02 Descrizione dell'agente

03 I dati

04 Implementazione

05 Integrazione



01 Introduzione



PARTIAMO DALL'INIZIO, HELP SELLER È UNA PIATTAFORMA CHE SI OCCUPA DI DIGITALIZZARE IL RAPPORTO TRA LE AZIENDE ED I DISTRIBUTORI.

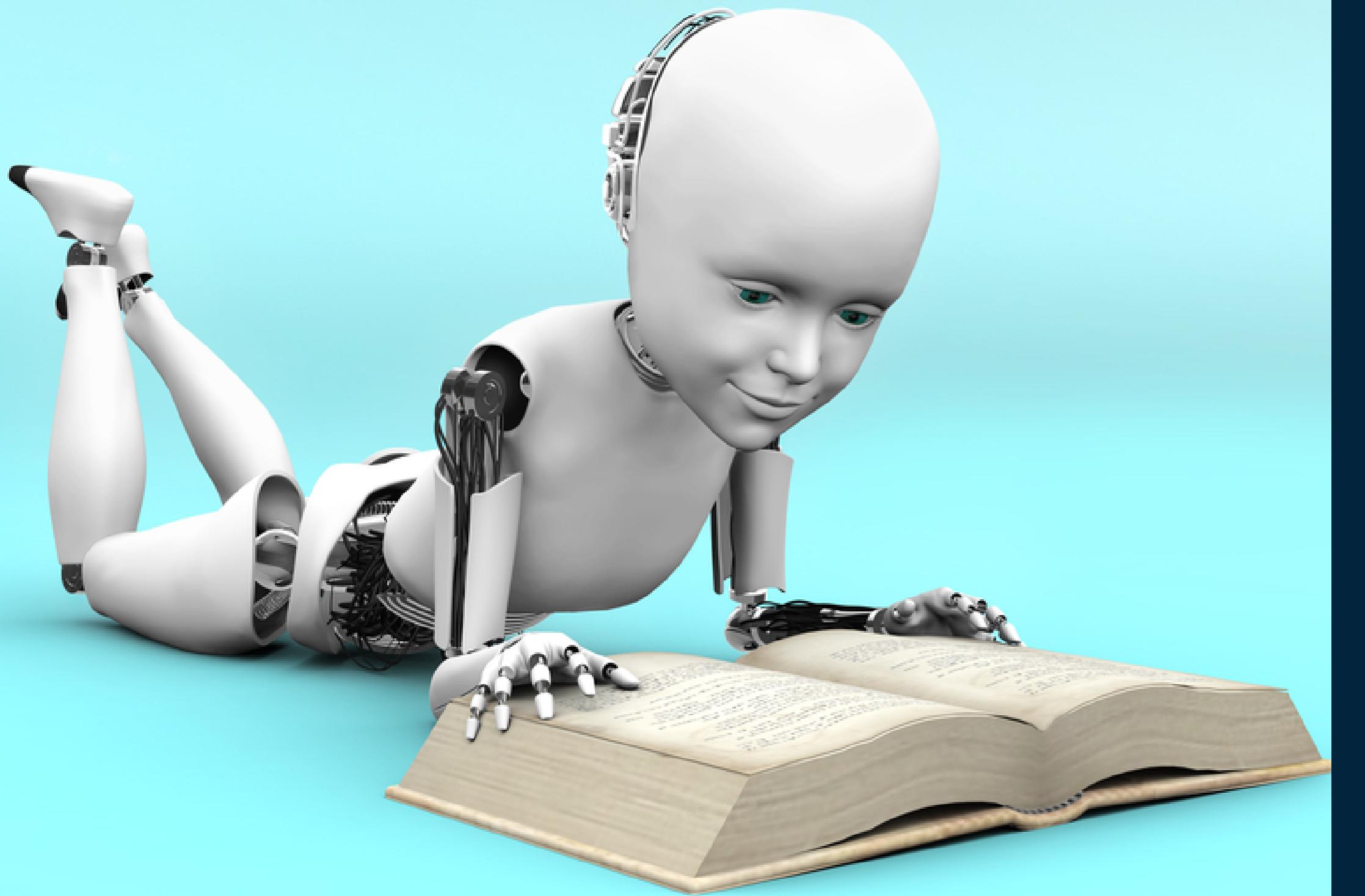
L'azienda è lo stakeholder

CHE DISPONE DI UN PRODOTTO ED HA LA NECESSITÀ DI VENDERLO SUL MERCATO

Il distributore è quella figura che compra il prodotto direttamente dall'azienda ed ha il task di distribuirla ai vari punti vendita con cui interagisce la **massa**

**STIAMO PARLANDO QUINDI
DI REALTÀ COME RISTORANTI,
NEGOZI, SUPERMERCATI E
TANTE ALTRE**





02

DESCRIZIONE DELL'AGENTE

OBIETTIVI

Sviluppare un motore di raccomandazione in grado di gestire le seguenti situazioni



Utenti appena registrati

riuscire a consigliare qualcuno senza conoscerlo



Utenti regolari

tracciare un profilo degli utenti nel tempo, determinare le loro abitudini d'acquisto per fare previsioni sempre più efficaci



Nuove aziende e quindi nuovi prodotti,

dare l'opportunità agli elementi del catalogo appena inseriti di poter compere con quelli invece già presenti

2.2 Specifica PEAS

PEAS	
Performance	La misura di performance dell'agente è direttamente proporzionale alla sua capacità nel riuscire a predire i prodotti che un utente è intenzionato a comprare, talvolta senza neanche che l'utente stesso ne sia consapevole.
Environment	L'ambiente in cui opererà l'agente è composto dall'insieme di tutti i prodotti presenti sulla piattaforma Help Seller, in congiunzione all'insieme di tutti gli utenti iscritti al sito con particolare enfasi sulla storia degli utenti, e quindi sulla loro cronologia d'acquisto. Si tratta di un ambiente: <ul style="list-style-type: none">● Dinamico fin quando la piattaforma resta online, un utente potrebbe effettuare un ordine cambiando le proprie preferenze● Sequenziale le azioni passate degli utenti influenzano le decisioni future dell'agente● Completamente Osservabile si ha accesso a tutte le informazioni relative agli utenti ed ai prodotti in ogni momento● Discreto il prezzo dei prodotti viene approssimato a due cifre decimali● Stocastico lo stato dell'ambiente cambia indipendentemente dalle azioni dell'agente● Singolo agente tecnicamente ci potrebbe essere una situazioni in cui più agenti operano in simultanea, tuttavia senza influenzarsi a vicenda.
Actuators	Gli attuatori consistono nella lista dei prodotti suggeriti sulla base del profilo dell'utente.
Sensors	I sensori dell'agente sono le view fatte al database per reperire informazioni necessarie al fine di prendere decisioni data driver

Ciascuna delle tre tipologie di modello ML studiate nel corso sono in grado di affrontare il problema delle raccomandazioni

Si potrebbe ricorrere a:

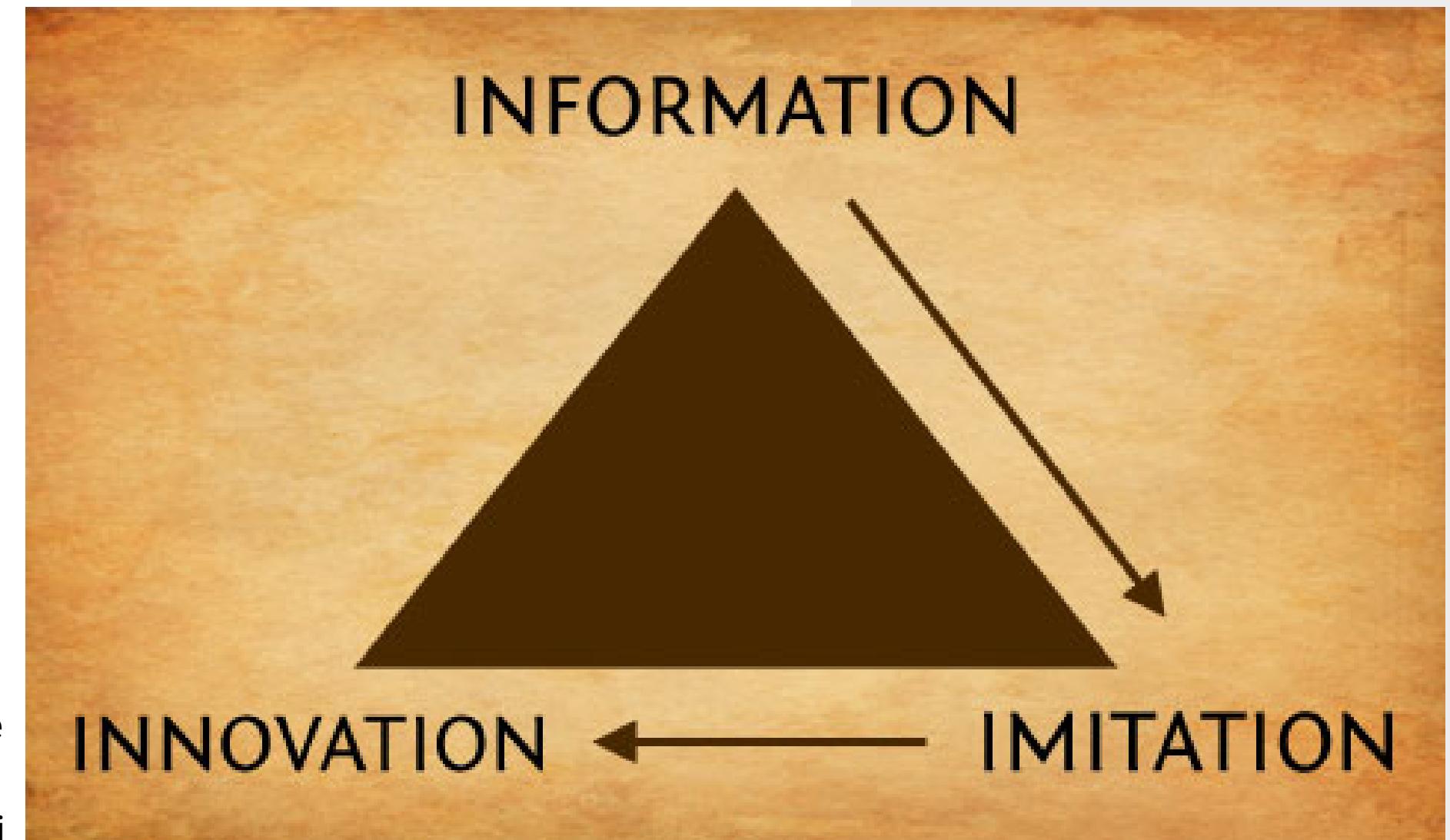
- un modello di classificazione associando una classe ad ogni utente o articolo,
- un modello di clustering Clustering raggruppando users (o items) omogenei alla ricerca di pattern nascosti
- oppure invece adottare un modello di regressione volto alla predizione di una variabile continua indice del **grado di affinità** di due elementi

prendere una decisione

- basta considerare l'enorme mole di dati trattati da una piattaforma di e-commerce come Amazon a rendersi conto l'inefficacia di un modello di classificazione
- nel database della piattaforma sono presenti tutte le informazioni necessarie alla predizione da parte del modello, ciò mi ha portato a scartare l'opzione del clustering e ad adottare invece un **modello di regressione**

I modelli di regressione costituiscono la tipologia di sistema di raccomandazione più diffusa da almeno due decenni a questa parte

l'applicazione di metodi matematici come la correlazione di Pearson e la decomposizione ai valori singolari (tanto per dirne qualcuno) al fine di ottenere una variabile continua, indice del grado di affinità tra diversi utenti o diversi item.



COLLABORATIVE FILTERING



Si parla di una classe di strumenti e meccanismi che consentono il recupero di informazioni predittive relativamente agli interessi di un insieme dato di utenti.

L'assunzione fondamentale dietro il concetto di collaborative filtering è che ogni singolo utente che ha mostrato un certo insieme di preferenze continuerà a mostrarle in futuro

PROBLEMATICA DEL COLLABORATIVE FILTERING:



Cold start

Quando un nuovo item o utente è aggiunto, esso non è associato a nessuna interazione, per tanto il modello non sarà in grado di gestirlo.



Scalability

Potenzialmente nella piattaforma ci potrebbero essere milioni di utenti e prodotti, richiedendo un'enorme potenza di calcolo per il calcolo delle raccomandazioni.



Sparsity

Il quantitativo di articoli venduti su un e-commerce è potenzialmente enorme; al tempo stesso persino gli utenti più attivi valuteranno solo una piccola porzione dell'insieme complessivo degli articoli, di conseguenza anche gli articoli più popolari avranno un quantitativo ristretto di valutazioni.

User / Item	Batman	Star Wars	Titanic
Bill	3	3	
Jane		2	4
Tom		5	

Esempio di matrice di una matrice di correlazione

Mette in relazione gli utenti con gli items

User / Item	Batman	Star Wars	Titanic
Bill	3	3	
Jane		2	4
Tom		5	

Andando a confrontare la coppia di utenti Bill e Jane, notiamo che Bill non ha visto Titanic e Jane non ha visto Batman, pertanto la relazione tra i due utenti è rafforzata soltanto da Star Wars.



**Due diversi
approcci**

ANALIZZARE L'UTENTE

User based

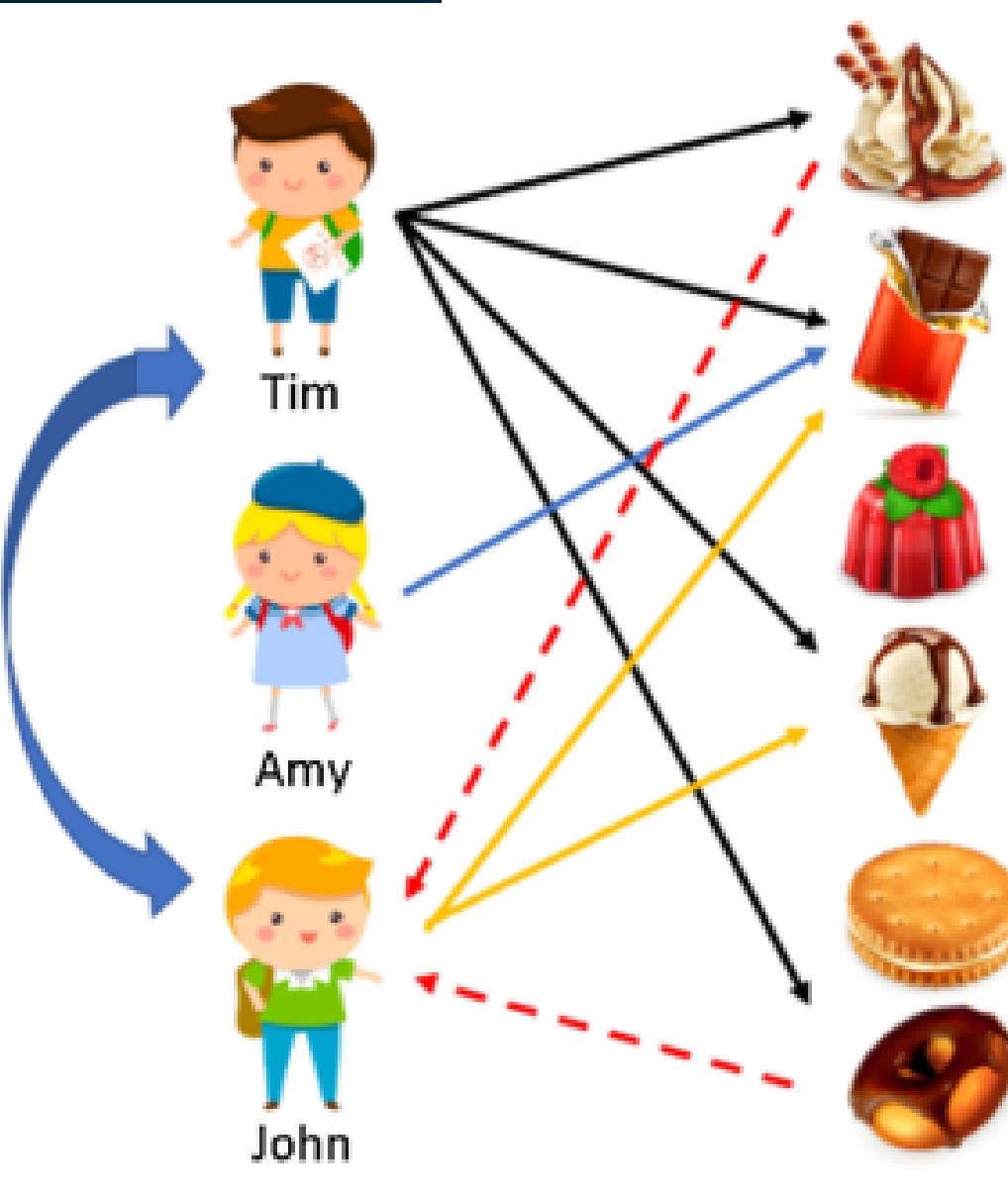
ANALIZZARE IL PRODOTTO

Item based

Ipotesi user based

nell'andare a fare una raccomandazione, vengono individuati gli utenti dagli interessi simili analizzando le loro recensioni

- Siano due users A e B tali che entrambi abbiano apprezzato gli item 1 e 2
- un sistema user based noterà la relazione tra i due utenti in quanto è presente una congiunzione nell'insieme degli item d'interesse di A e B
- ogni nuovo ordine effettuato da A potrebbe essere un buon suggerimento per B, sulla base della relazione appena individuata.

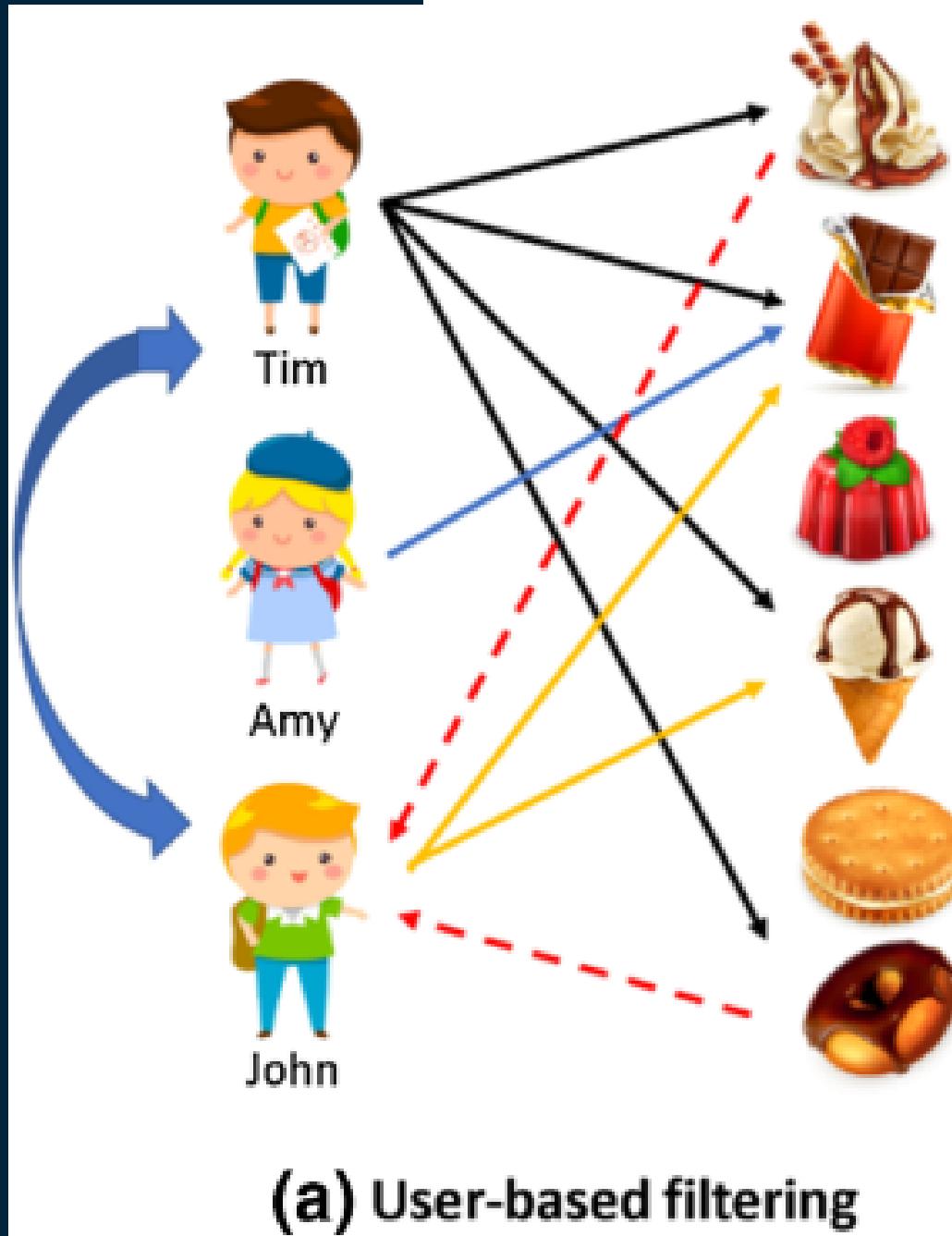


(a) User-based filtering

Ipotesi user based

Limitazioni della proposta:

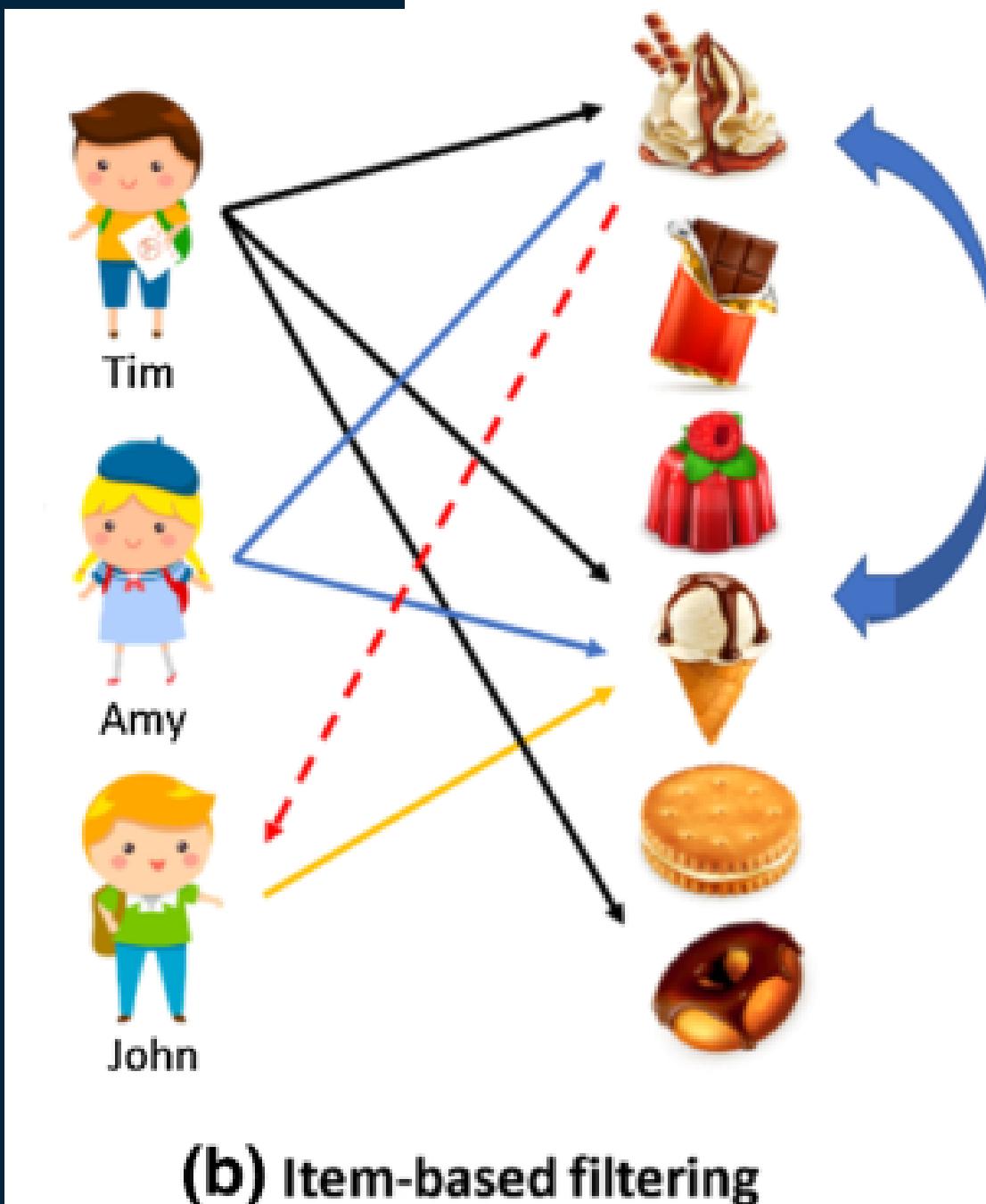
- Il sistema è poco performante se relativamente parlando ci sono molti item ma poche recensioni
- Calcolare la similarità tra tutte le coppie di utenti è costoso
- I profili degli utenti sono entità dinamiche che cambiano spesso e in breve tempo, costringendo l'intero modello del sistema ad essere ricalcolato.



Ipotesi item based

“Le persone che valutano positivamente l’item 1 come te, tendono a valutare positivamente anche l’item 2, siccome tu non hai ancora valutato l’item 2 dovresti provarlo”

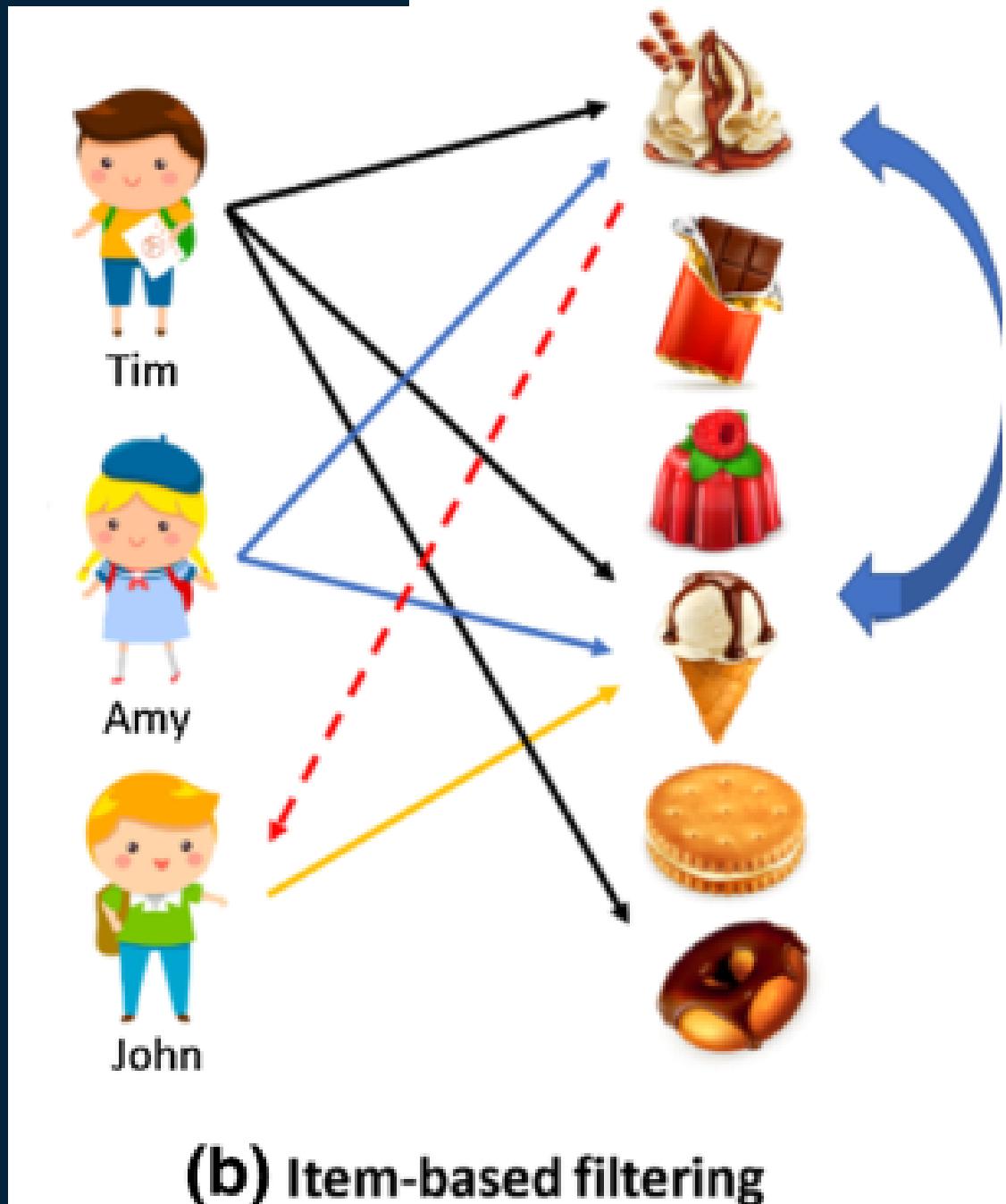
inventata da Amazon nel 1998, questa tipologia è concettualmente identica a quella User based ma l’applicazione è eseguita al contrario: anziché cercare relazioni tra utenti, vengono cercate relazioni tra gli articoli del catalogo.



Ipotesi item based

Vengono risolte le limitazioni appartenenti alla sua all'ipotesi item based

almeno nei sistemi aventi più utenti di items, ad ogni item saranno associate molteplici valutazioni, sicché il voto medio di un articolo non cambierà troppo frequentemente. Questo comporta una distribuzione più stabile nel modello minimizzando la necessità di andare a ricalcolarlo.

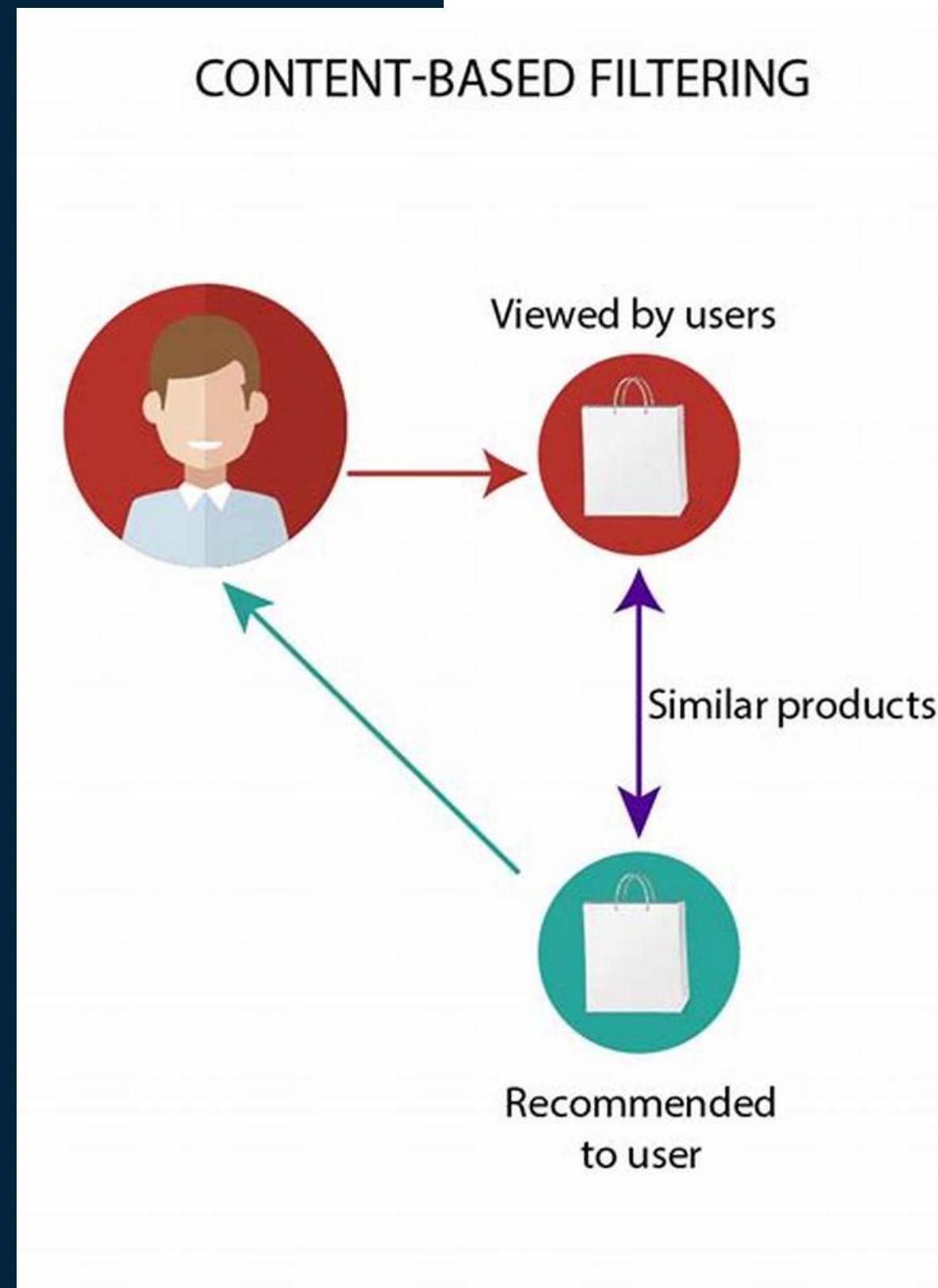


ipotesi numero tre:

CONTENT-BASED FILTERING

Non giudicare un libro dai suoi dati ma dal contenuto!

I content-based recommenders trattano i suggerimenti come problemi di classificazione specifici al singolo user. Ad ogni utente è associato un profilo dove vengono descritte le sue preferenze e avversioni relative alle caratteristiche degli items. In poche parole, il modello cercherà di suggerire all'utente articoli dalle caratteristiche simili a quelle degli articoli precedentemente acquistati dall'utente in questione.



CONTENT-BASED FILTERING



03

I Dati

tutto ciò che è necessario al modello per poter fare un suggerimento è contenuto all'interno della table “recensione”

	idrecensione	testo	voto	data	idProdotto	idDistributore
▶	1	adoro il piccante	5	NULL	9	1
	2	adoro il piccante	5	NULL	8	1
	3	adoro il piccante	5	NULL	10	1
	4	buona la cola	5	NULL	1	2
	5	buona la cola	5	NULL	2	2
	6	buona la cola	5	NULL	3	2
	7	ottimo surgelato	5	NULL	4	3
	8	ottimo surgelato	5	NULL	5	3
	9	ottimo surgelato	5	NULL	6	3
	10	anche a me piace il piccante	5	NULL	9	4
	11	sono un patito della cola	5	NULL	1	5
	12	molto comodi da cucinare	5	NULL	4	6
	13	pessimo sapore	1	NULL	3	4
	14	troppo piccante	1	NULL	10	5
	NUL	NUL	NUL	NUL	NUL	NUL

Sono sufficienti le informazioni che ci consentono di andare a riempire una matrice di correlazione

Ci bastano quindi gli attributi: "voto", "idProdotto" e "idDistributore"

debugger view della matrice ottenuta dalla table della slide precedente
tramite la funzione pivot_table di python della libreria pandas

```
X = ratings.pivot_table(values='voto', index='idProdotto', columns='idDistributore', fill_value=0)
```

	1	2	3	4	5	6
1	0	5	0	0	5	0
2	0	5	0	0	0	0
3	0	5	0	1	0	0
4	0	0	5	0	0	5
5	0	0	5	0	0	0
6	0	0	5	0	0	0
8	5	0	0	0	0	0
9	5	0	0	5	0	0
10	5	0	0	0	1	0

04

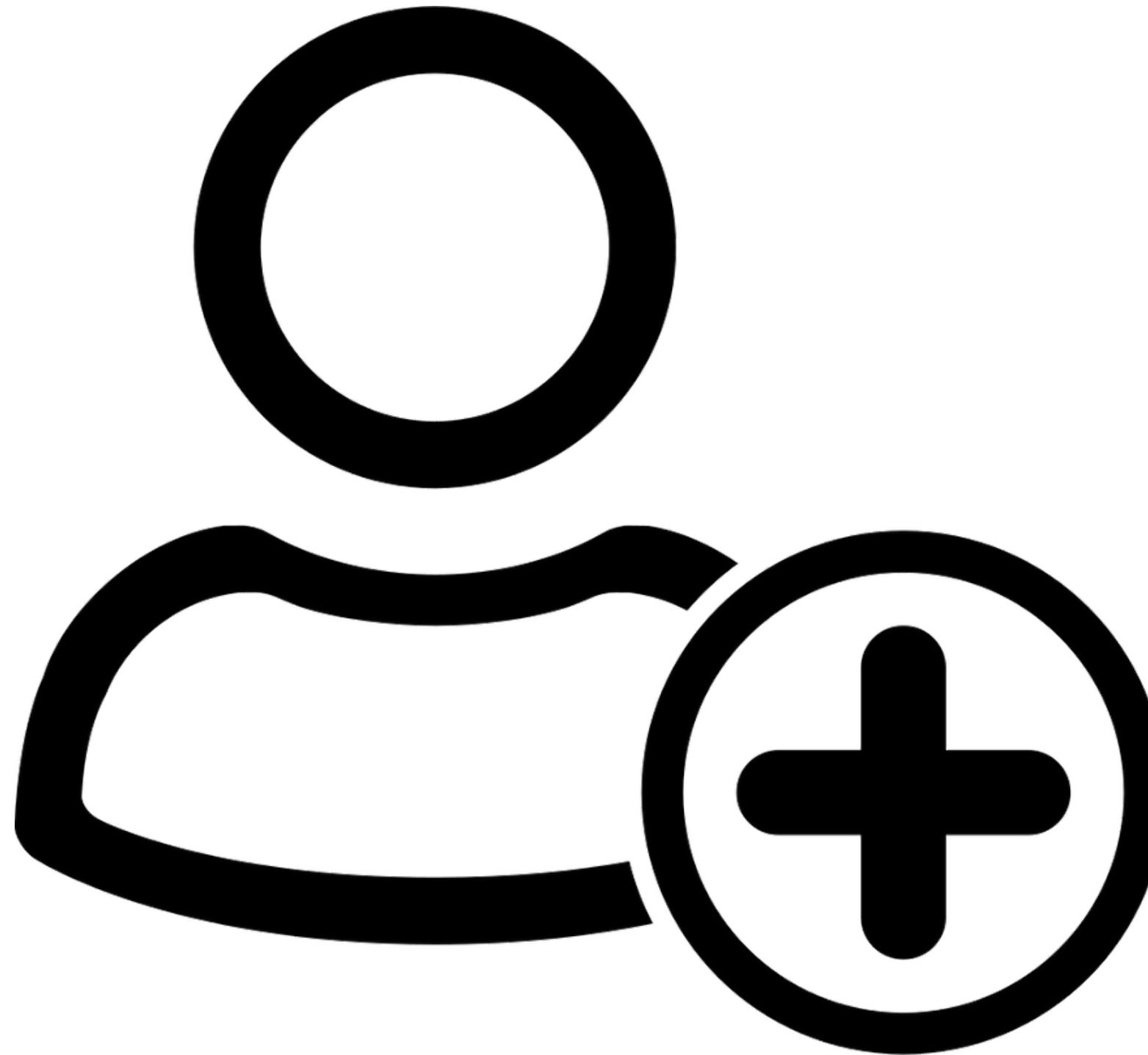
Implementazione

```
    if (parseInt(header1.css('padding-top'), 10) > header1_initialDistance) {
        header1.css('padding-top', '' + $window.scrollTop() + 'px')
    }
} else {
    header1.css('padding-top', '' + header1_initialPadding + 'px')
}

if ($window.scrollTop() > header2_initialDistance) {
    if (parseInt(header2.css('padding-top'), 10) > header2_initialDistance) {
        header2.css('padding-top', '' + $window.scrollTop() + 'px')
    }
}
```

Primo obiettivo: utenti appena registrati

Quello che si vuole risolvere è il problema del **cold start** definito in precedenza



l'idea è che se una cosa piace ad n persone, è probabile che possa piacere anche alla persona n+1

Pertanto verranno individuati e suggeriti ai new users quelli che risultano essere i prodotti più popolari tra gli altri utenti della piattaforma già esistenti

semplice script volto all'ordinamento dei prodotti in base alla loro popolarità (numero di recensioni)
seguito da un grafico a barre fornito dalla libreria matplotlib

```
oFia Two Layer > Two Layer.py
Two Layer.py

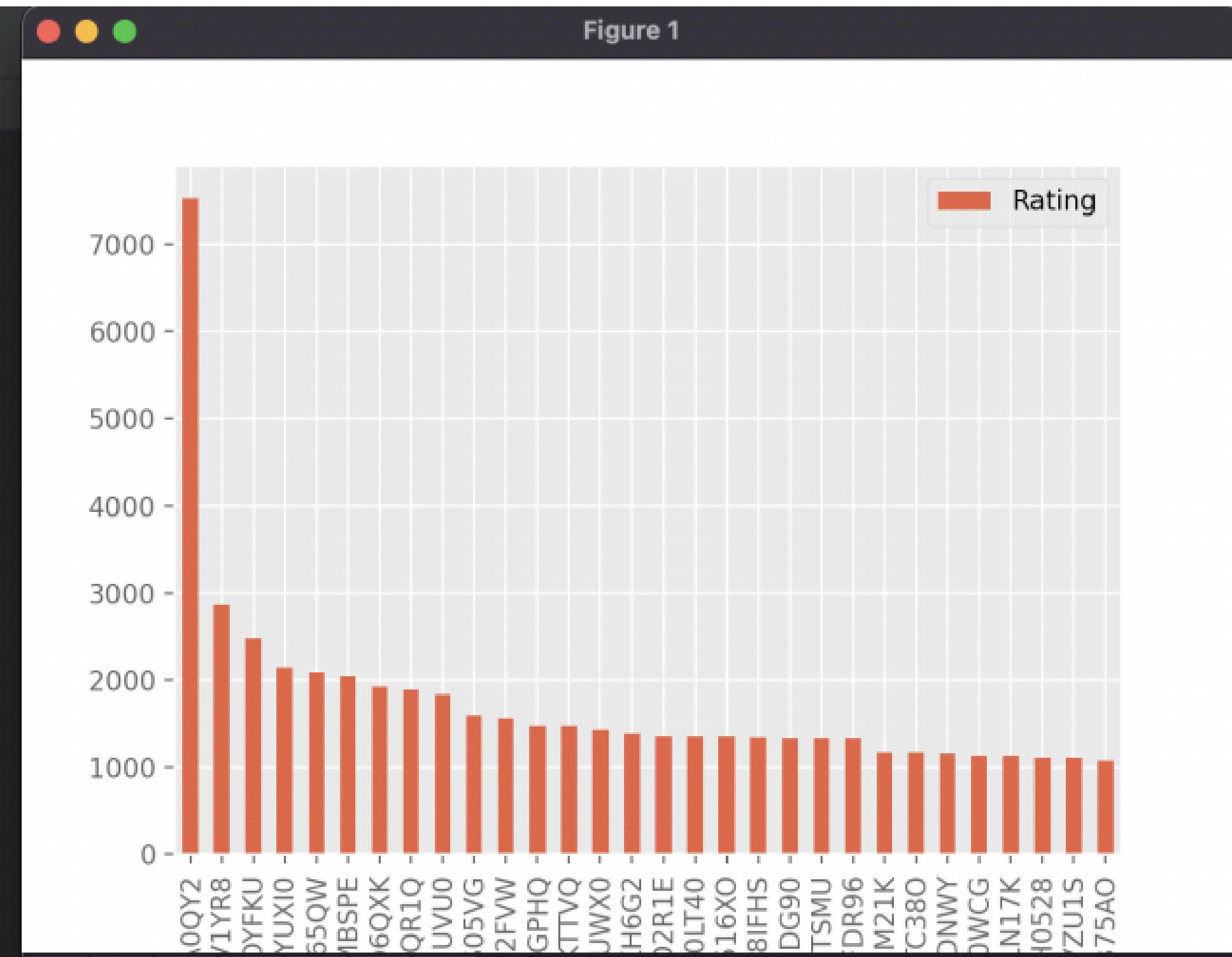
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

plt.style.use("ggplot")

import sklearn
from sklearn.decomposition import TruncatedSVD

ratings = pd.read_csv('ds.csv')
ratings = ratings.dropna()
# print(ratings.shape)

popular_products = pd.DataFrame(ratings.groupby('ProductId')['Rating'].count())
most_popular = popular_products.sort_values('Rating', ascending=False)
# print(most_popular.head(30))
most_popular.head(30).plot(kind='bar')
plt.show()
```



Secondo obiettivo:

UTENTI REGOLARI

Il modello deve
essere in grado di

01

Tracciare un profilo
dell'user

02

Individuare
affinità

03

Suggerire un
prodotto

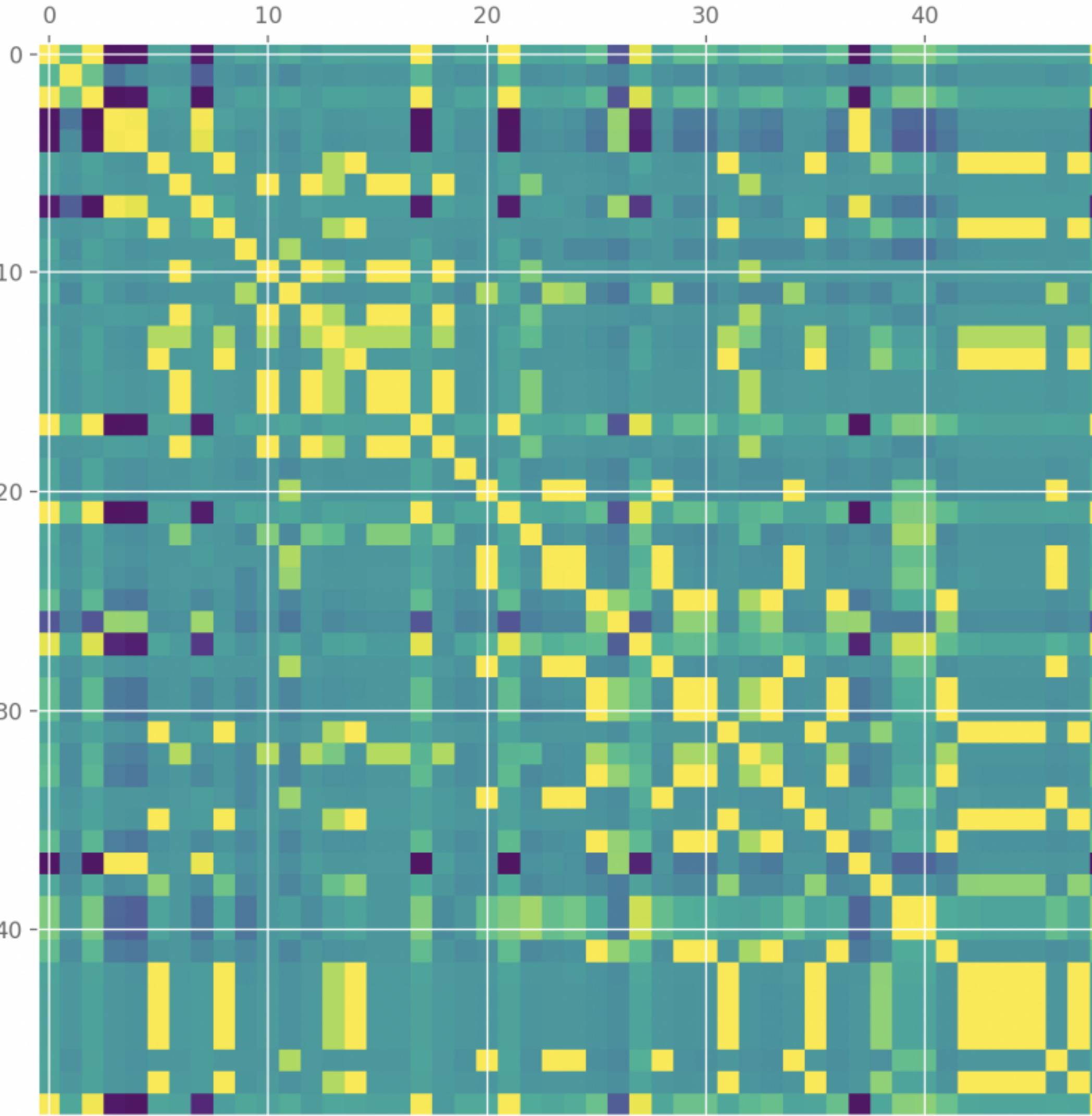
01. Tracciare un profilo Utente

Verrà mappata una matrice avente per indici gli utenti e per colonne i prodotti. Il valore di tale matrice sarà il rating che l'utente ha dato al prodotto in seguito ad un acquisto.

02. Trovare affinità

È possibile determinare il grado di affinità tra users & items applicando il metodo della decomposizione ai valori singolari, il tutto è già implementato nel package `sklearn.decomposition` di Python.

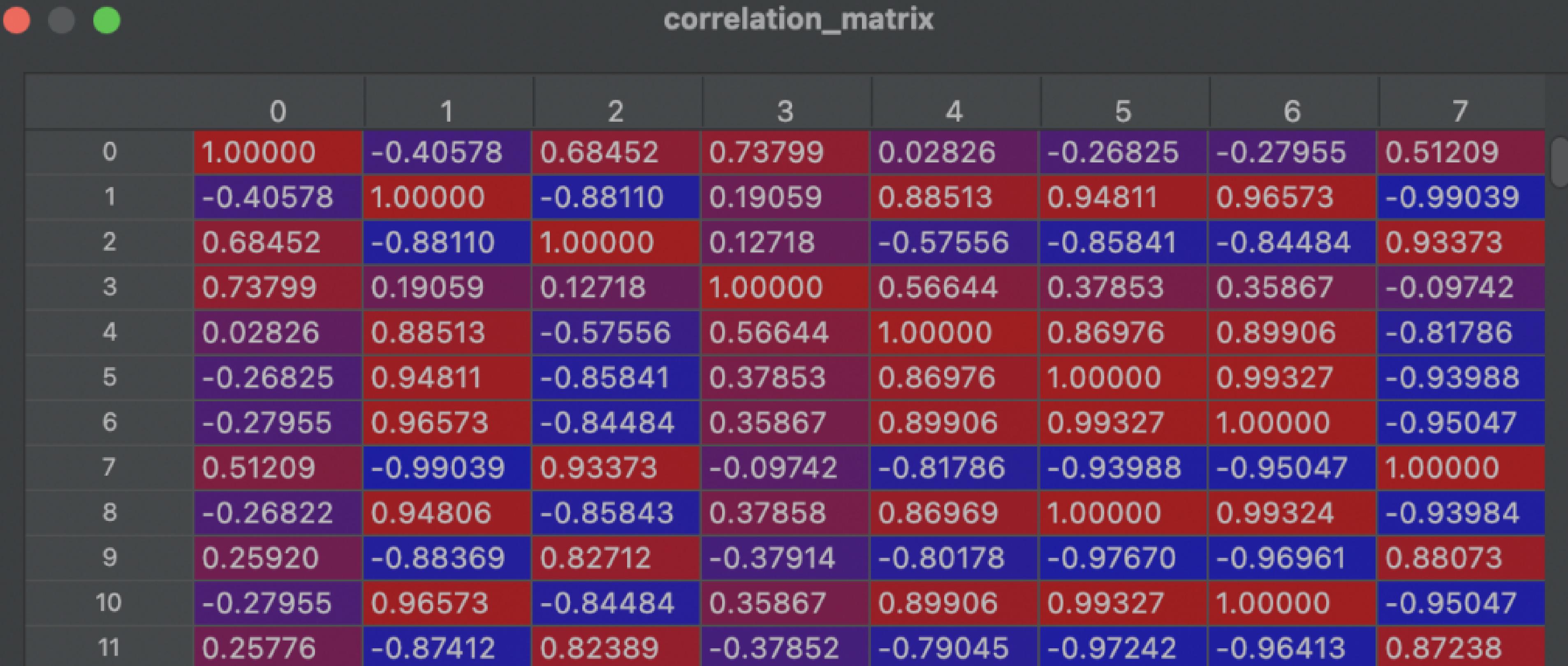
```
SVD = TruncatedSVD(n_components=10) #Decomposizione ai valori singolari
decomposed_matrix = SVD.fit_transform(X)
# print(decomposed_matrix)
correlation_matrix = np.corrcoef(decomposed_matrix)
plt.matshow(correlation_matrix)
```



Di fianco è presentata la rappresentazione grafica di quello che si ottiene applicando il metodo della scomposizione ai valori singolari ad una matrice di correlazione.

notare la diagonale che intercorre nel mezzo, descrive il fattore di correlazione tra l'elemento e se stesso, per tanto assume valore massimo. Più il valore contenuto nella cella si avvicina ad 1, maggiore sarà l'affinità.

Debugger view della matrice, anche qui è possibile notare la diagonale di affinità tra elementi.



	0	1	2	3	4	5	6	7
0	1.00000	-0.40578	0.68452	0.73799	0.02826	-0.26825	-0.27955	0.51209
1	-0.40578	1.00000	-0.88110	0.19059	0.88513	0.94811	0.96573	-0.99039
2	0.68452	-0.88110	1.00000	0.12718	-0.57556	-0.85841	-0.84484	0.93373
3	0.73799	0.19059	0.12718	1.00000	0.56644	0.37853	0.35867	-0.09742
4	0.02826	0.88513	-0.57556	0.56644	1.00000	0.86976	0.89906	-0.81786
5	-0.26825	0.94811	-0.85841	0.37853	0.86976	1.00000	0.99327	-0.93988
6	-0.27955	0.96573	-0.84484	0.35867	0.89906	0.99327	1.00000	-0.95047
7	0.51209	-0.99039	0.93373	-0.09742	-0.81786	-0.93988	-0.95047	1.00000
8	-0.26822	0.94806	-0.85843	0.37858	0.86969	1.00000	0.99324	-0.93984
9	0.25920	-0.88369	0.82712	-0.37914	-0.80178	-0.97670	-0.96961	0.88073
10	-0.27955	0.96573	-0.84484	0.35867	0.89906	0.99327	1.00000	-0.95047
11	0.25776	-0.87412	0.82389	-0.37852	-0.79045	-0.97242	-0.96413	0.87238

03. Determinare i prodotti da suggerire

Non ci resta che estrapolare dalla matrice i prodotti dall'affinità più alta e inviarli al frontend sicché possano essere visualizzati dall'utente che sta navigando

```
i = X.index[8] #id del prodotto con indice n
product_names = list(X.index)
product_ID = product_names.index(i)
correlation_product_ID = correlation_matrix[product_ID]
Recommend = list(X.index[correlation_product_ID > 0.5])
# Rimuovo l'iesimo item
Recommend.remove(i)
print("Il prodotto {} ha una relazione con: ".format(i))
print(Recommend)
print(correlation_product_ID)
```

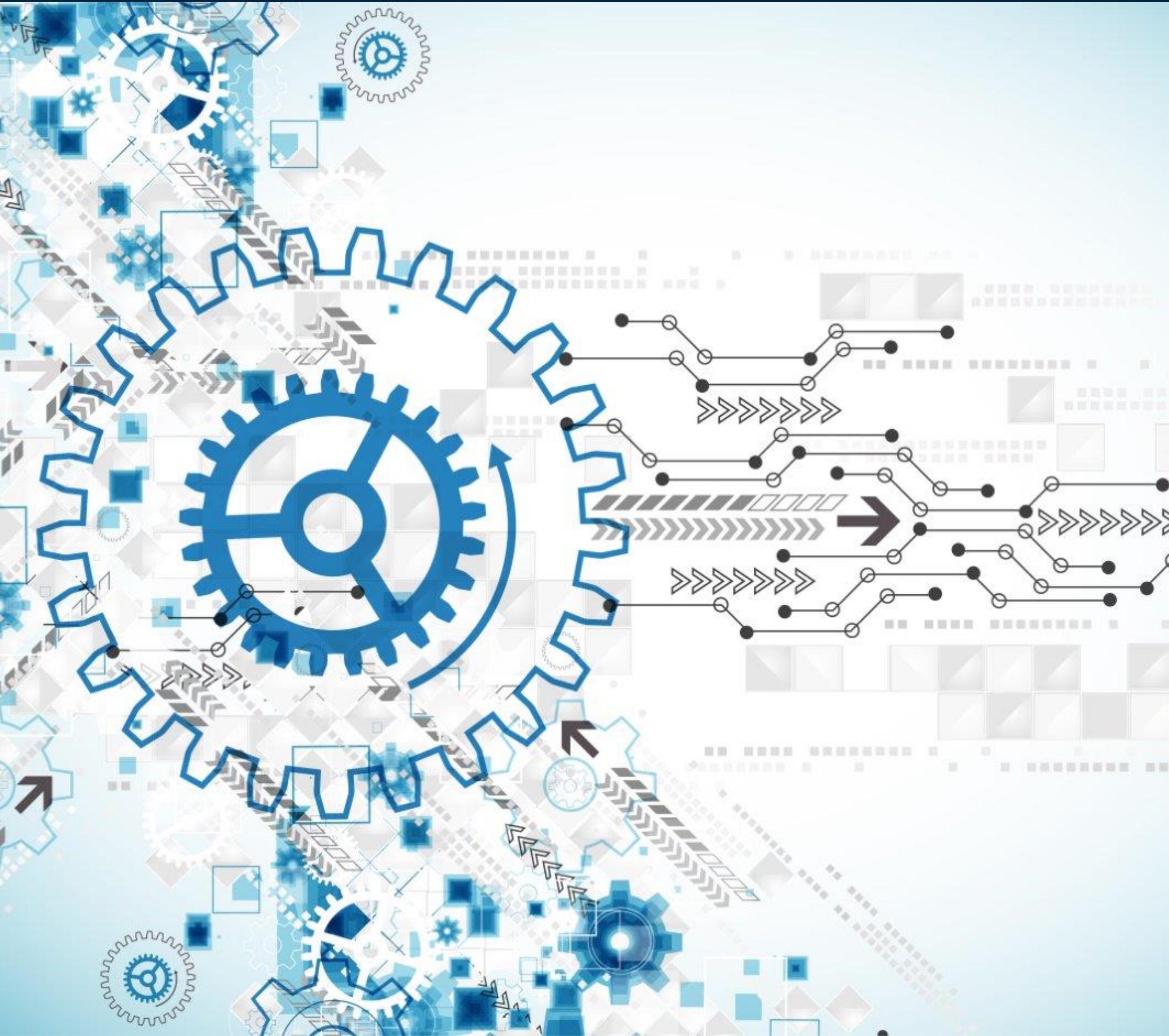
Ipotizzando che l'utente abbia effettuato l'acquisto dell'item con index 8, viene ricercato un riscontro tra tale item e gli altri sulla base delle recensioni di utenti che hanno comprato lo stesso prodotto.

Output dello snippet di codice della slide precedente

```
Run: Two Layer HS ×
C:\Users\xlits\PycharmProjects\ML1\venv\Scripts\python.exe "C:/Users/xlits/PycharmProjects/ML1/moduloFIA/test/Two Layer HS.py"
Il prodotto 10 ha una relazione con:
[8, 9]
[ 0.26813664 -0.10761533 -0.12972179 -0.3349523 -0.11604304 -0.11604304
 0.9911012  0.74640947  1.      ]
Process finished with exit code 0
```

05

Integrazione con la piattaforma



La prima parte dell'integrazione consiste nella lettura tramite query delle istanze delle recensioni presenti all'interno del DB

```
y:  
connection = mysql.connector.connect(host='localhost',  
                                      database='helpseller',  
                                      user='root',  
                                      password='root')  
  
sql_select_Query = "select * from recensione"  
cursor = connection.cursor()  
cursor.execute(sql_select_Query)  
# get all records  
records = cursor.fetchall()  
print("Total number of rows in table: ", cursor.rowcount)  
  
    print("\nPrinting each row")  
    for row in records:  
        print("Id = ", row[0], )  
        print("testo = ", row[1])  
        print("voto = ", row[2])  
        print("idProdotto = ", row[4])  
        print("idDistributore = ", row[5], "\n")  
  
    except mysql.connector.Error as e:  
        print("Error reading data from MySQL table", e)  
    finally:  
        if connection.is_connected():  
            connection.close()  
            cursor.close()  
            print("MySQL connection is closed")
```

script in Python di connessione al database, recupera le istanze dalla table recensioni...

```
# Create the csv file
with open('bigData.csv', 'w', newline='') as f_handle:
    writer = csv.writer(f_handle)
    # Add the header/column names
    header = ['id', 'testo', 'voto', 'data', 'idProdotto', 'idDistributore']
    writer.writerow(header)
    # Iterate over `data` and write to the csv file
    for row in records:
        writer.writerow(row)
```

...e le memorizza in un file CSV

```
id,testo,voto,data,idProdotto,idDistributore
1,adoro il piccante,5,,9,1
2,adoro il piccante,5,,8,1
3,adoro il piccante,5,,10,1
4,buona la cola,5,,1,2
5,buona la cola,5,,2,2
6,buona la cola,5,,3,2
7,ottimo surgelato,5,,4,3
8,ottimo surgelato,5,,5,3
9,ottimo surgelato,5,,6,3
10,anche a me piace il piccante,5,,9,4
11,sono un patito della cola,5,,1,5
12,molto comodi da cucinare ,5,,4,6
13,pessimo sapore,1,,3,4
14,troppo piccante,1,,10,5
```

File CSV ottenuto dalla table 'recensioni'
mostrata qualche slide fa

