

**Analisi e sviluppo di un sistema di  
raccomandazione  
per la piattaforma BeerAdvocate**  
***Davide Vettore 868855 – Metodi informatici per la gestione  
aziendale 2022/23***

***Introduzione***

L'obiettivo dell'analisi è sviluppare un recommender system: ovvero un programma che suggerisca elementi di interesse personalizzati all'utente. I dati utilizzati provengono da una comunità online di esperti di birra chiamata BeerAdvocate:

"This website is all about discovering and discussing beer. Users can search for beers and places near them, rate and review them, start or join a discussions in our forums, trade beers, make some new friends, participate in online tastings, and learn more about our in-person events."

Partendo da un dataset contenente i ratings di decine di migliaia di birre, vogliamo riuscire a proporre a ciascun utente una lista di suggerimenti basati sui voti da lui espressi in precedenza.

Per fare ciò seguiremo un approccio Collaborative Filtering: si tratta di una tecnica adottata dai recomender system la quale permette di creare suggerimenti basati sulle preferenze di utenti ritenuti simili, questo approccio non richiede alcuna conoscenza e comprensione del prodotto, il vantaggio è che i dati non necessitano di mantenimento, sono forniti direttamente dagli utenti tramite questionari espliciti (ovvero i rating). Un altro approccio utilizzato nei sistemi di raccomandazione consiste nel consigliare prodotti con caratteristiche simili a prodotti acquistati in passato, quest'ultimo viene definito content-based.

A seguito di una breve analisi esplorativa sui dati, il primo passo è definire l'algoritmo K-Nearest Neighbors che ci permetterà di identificare gli utenti "vicini", grazie ad esso riusciremo a riempire la matrice di ratings su cui baseremo poi i suggerimenti.

L'algoritmo supervisionato KNN richiede in input degli iperparametri specifici rispetto al problema che deve processare. Per ottimizzare le performance, utilizzeremo una gridsearch per verificare le performance dell'algoritmo al variare degli iperparametri, confronteremo i risultati mediante le metriche Mean Square Error e Root Mean Square Error.

Oltre all'algoritmo KNN, possiamo verificare come variano le metriche di performance utilizzando il Singular Value Decomposition. Anche in questo caso ottimizzeremo gli iperparametri del modello rispetto ai nostri dati utilizzando una gridsearch. Dopo aver ottenuto l'algoritmo ottimale rispetto al nostro problema, possiamo procedere a riempire la matrice di ratings "sparsa", così definita poiché disponiamo solo di una piccola percentuale di rating reali.

Il recommender system è ora ultimato, dopo aver rimosso i ratings reali dalla matrice, le birre che avranno un rating più alto rispetto a ciascun utente saranno quelle a lui suggerite.

L'ultima analisi che effettueremo sulla matrice incompleta sarà verificare la presenza di clusters che discriminano gli utenti in gruppi distinti, per fare ciò si utilizza l'algoritmo clustering k-means. Osserveremo poi il coefficiente di silhouette per valutare la qualità, ovvero la densità e la separazione, delle segmentazioni ottenute.


## Analisi esplorativa

Il dataset completo contiene 1.586.614 recensioni caratterizzate da 13 variabili: gli identificativi della birra (name, beerId, brewerId) e dell'utente (profileName), caratteristiche come style e AlcoholByVolume, la review testuale, il timestamp Unix e il rating (appearance, aroma, palate, taste e overall).

SCORE

95

World-Class



✓ Rate it

Beer Geek Stats | Print Shelf Talker

You: Not rated

From: Dogfish Head Craft Brewery

Style: Imperial IPA

ABV: 9%

Score: 95

Avg: 4.27 | pDev: 11.01%

Reviews: 4,114

Ratings: 16,482

Status: Active

Rated: Today at 01:37 AM

Added: Nov 19, 2001 by Todd

Wants: 899

Gots: 5,129

Reviewed by JesseJames81 from California

3.9/5 rDev -8.7%

look: 4 | smell: 4 | taste: 3.75 | feel: 4 | overall: 4

Appearance: Pours a rich, golden reddish amber color, with a thick white foam head. Good retention and leaves behind some sticky lacing.

Aroma: Sweet bready malt and pine. A subtle touch of citric fruit.

Taste: A pronounced maltiness followed by hoppy layers of citrus and pine. Some bready biscuit notes. Malt backbone and a bitterness that starts half-way through, and rides all the way to the end. Finishes fairly clean.

Overall: This is one interesting, well balanced beer.

Sep 22, 2022 Report

beer/name	beer/beerId	beer/brewerId	beer/ABV	beer/style	review/appearance	review/aroma	review/palate	review/taste	review/overall	review/time	review/profileName	review/text
90 Minute IPA	2093	10099	9.0	American Double / Imperial IPA	4	4	4	3.75	4	1264711532	JesseJames81	Appearance: Pours a rich golden reddish amber.

Figura 1: esempio di prodotto e di recensione con corrispettiva riga nel dataset

Dopo aver rimosso le osservazioni che presentano valori nulli nelle colonne di interesse, possiamo procedere conducendo delle analisi interessanti sulla totalità dei dati:

1. Osservando le correlazioni tra le diverse tipologie di rating, notiamo come il sapore della birra sia la caratteristica che più influenza la valutazione "overall", con un valore di correlazioni di Pearson del 79%. La caratteristica che conta meno è invece l'aspetto estetico, presenta una correlazione lineare positiva di appena il 50%.  
Tralasciando il rating complessivo, "taste" risulta essere molto correlata positivamente anche con "palate" e "aroma": rispettivamente 73% e 72% (figura 2).
2. Osservando il rating overall medio, al variare del volume di alcool, notiamo come le birre con ABV nella media (5-10) hanno un rating medio tra 3.5 e 4. Gli estremi, ovvero le birre con un volume alcolico molto basso o molto elevato, hanno rating medio che varia significativamente tra valori alti e

valori bassi. Questo fenomeno è sicuramente influenzato dalla numerosità inferiore delle birre agli estremi, evidenzia tuttavia come l'aumento della gradazione e il rating overall non siano linearmente correlati (figura 3).

3. Osserviamo i “beer style” con voto medio maggiore e quelli con voto medio minore. Le tipologie di birre mediamente più apprezzate risultano essere le American Wild Ale, quelle meno apprezzate sono invece le Low Alcohol Beer. Gli stili più apprezzati appaiono più spesso rispetto a quelli meno apprezzati, le Imperial IPA, ad esempio, hanno collettivamente una totalità di oltre 80.000 recensioni (figura 4).

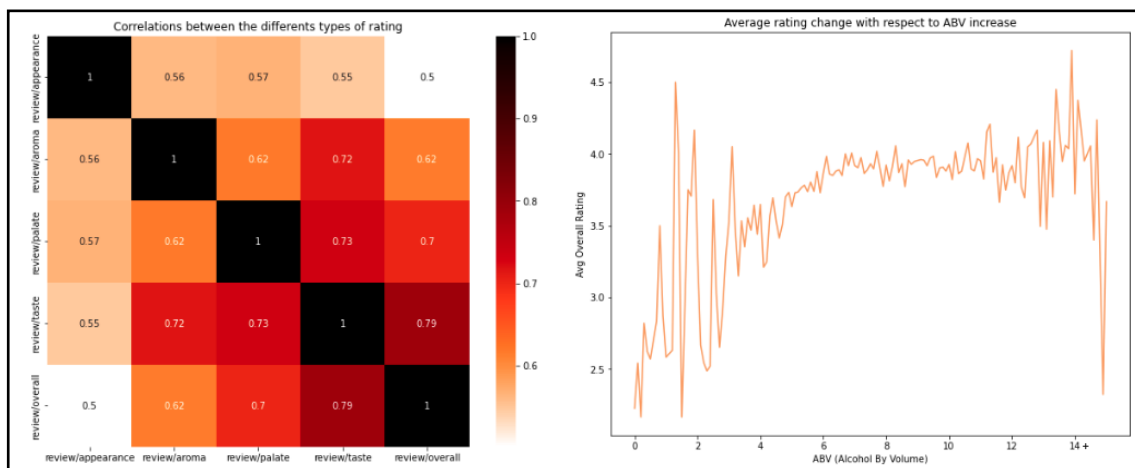


Figura 2 e 3: correlazione tra le diverse tipologie di rating e rating “overall” medio all’aumentare dell’AlcholByVolumn

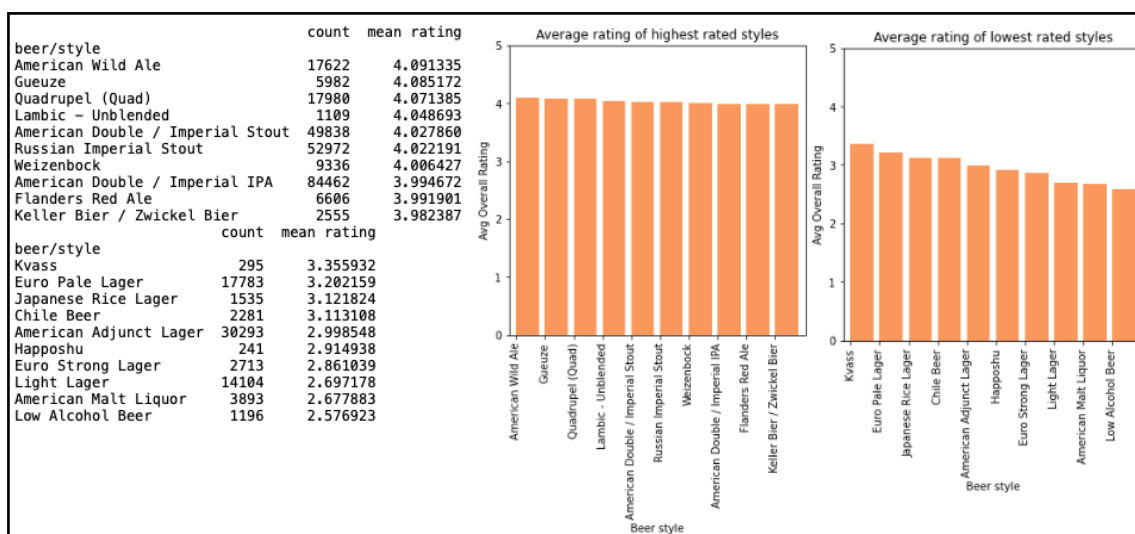


Figura 4: beer style con voto medio maggiore e con voto medio minore, con le rispettive numerosità

Osserviamo inoltre, ad esempio, l'utente che recensisce il maggior numero di birre: 'BuckeyeNation'. Incentrando l'analisi, precedentemente eseguita sui beer style, sul singolo utente, notiamo come le sue birre preferite siano le Eisbock, una tipologia di lager tedesca molto forte, con una valutazione media di 4.5 su un totale di tre prodotti recensiti. La tipologia di birra meno apprezzata risulta essere la Kvaas, la così detta "birra di pane", nonostante ne abbia recensita solo una, è seguita dalle birre a basso contenuto alcolico e dalle Light Lager (figura 5).

Possiamo inoltre verificare come variano mediamente le recensioni dell'utente nel tempo: per fare ciò bisogna inizialmente convertire il timestamp Unix in un formato data (utilizzando la libreria Python datetime), possiamo poi procedere raggruppando il rating overall rispetto ai mesi (figura 6).

Nel primo anno d'attività le recensioni subiscono ampi sbalzi, probabilmente l'utente votava pochi prodotti, come conferma possiamo vedere qualche mese senza alcuna recensione. Successivamente il rating si stabilizza tra 3.5 e 4 per una decina d'anni, solo verso gli ultimi mesi registrati iniziamo a notare una decrescita, l'utente potrebbe essere diventato più severo con le sue valutazioni.

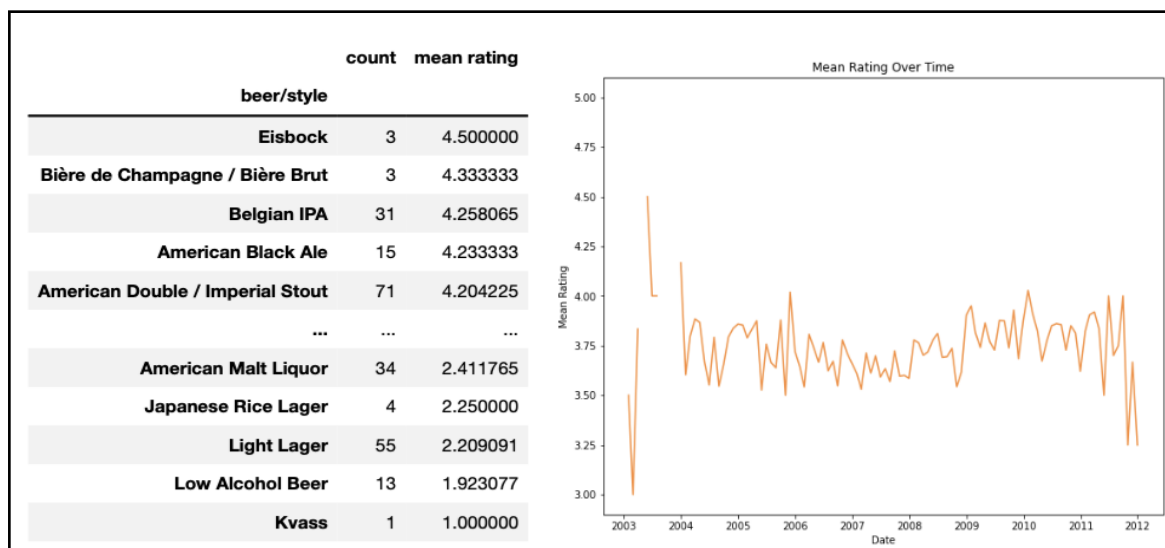


Figura 5 e 6: stili di birra più e meno apprezzati dall'utente 'BuckeyeNation'; variazione media dei rating overall riportati dall'utente al trascorrere dei mesi

Per semplicità computazionale, decidiamo di mantenere solo le colonne contenenti il nome della birra, il nome dell'utente e il rating overall, ovvero le colonne su cui si incentrerà l'analisi.

Il passo successivo è osservare le dimensioni della matrice di ratings che andrà riempita: il dataset contiene 33.387 users univoci e 56.856 items univoci, questo significa che la matrice di ratings avrà dimensione 1.898.251.272.

I rating reali, presenti all'interno del dataset, sono solo 1.561.398: la matrice è quindi vuota al 99.92%, questo fenomeno è causato dal fatto che un elevato numero di utenti vota una sola birra, inoltre un elevato numero di birre è stato votato da un solo utente.

Per ridurre i tempi computazionali, e migliorare il funzionamento dell'algoritmo KNN, decidiamo di ridurre la matrice: teniamo i 4000 utenti e le 4000 birre che appaiono più spesso all'interno del dataset. Otteniamo una matrice con dimensione 16.000.000, tra cui 1.000.360 sono rating reali (6.252%), i rating hanno mediamente valore pari a 3.62.

Per confrontare la matrice completa con la matrice ridotta osserviamo:

1. Come si distribuiscono i rating nei due casi: notiamo che, riducendo il dataset, la frequenza dei ratings non cambia significativamente (figura 7).
2. Come si distribuiscono le occorrenze degli items nei due casi: nel dataset completo solo le birre nel terzo quartile apparivano almeno 9 volte, la media di occorrenze era di appena 27. Nel dataset ridotto, la birra che appare meno volte ha 46 occorrenze, mediamente ciascun prodotto viene recensito 250 volte (figura 8).
3. Come si distribuiscono le occorrenze degli users nei due casi: nel primo caso ogni utente recensisce mediamente 46 prodotti; nonostante l'utente più attivo recensisca ben 5600 birre, solo gli utenti nel terzo quartile ne recensiscono almeno 16. A seguito della riduzione, la media aumenta del 500%, l'utente che recensisce meno prodotti appare 57 volte (figura 9).

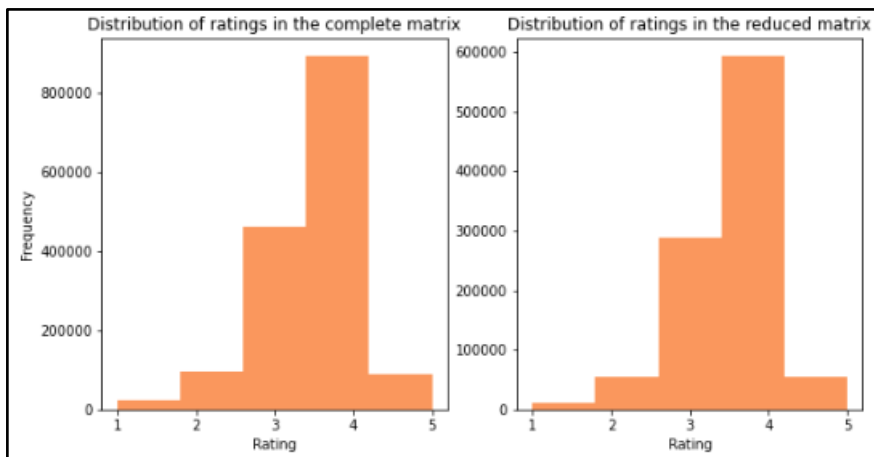


Figura 7: frequenza dei rating nella matrice completa e nella matrice ridotta

	Items occurrences in the complete matrix	Items occurrences in the reduced matrix
count	56856.000000	4000.000000
mean	27.462326	250.090000
std	118.684510	265.81839
min	1.000000	46.000000
25%	1.000000	91.000000
50%	3.000000	151.000000
75%	9.000000	292.000000
max	3206.000000	2172.000000

Figura 8 e 9: distribuzione delle occorrenze degli items e degli users nella matrice completa e nella matrice ridotta

	Users occurrences in the complete matrix	Users occurrences in the reduced matrix
	33387.000000	4000.000000
	46.766646	250.090000
	177.726920	270.174894
	1.000000	57.000000
	1.000000	89.000000
	3.000000	150.000000
	16.000000	295.000000
	5599.000000	2759.000000

## ***Sviluppo del sistema di raccomandazione***

Come visto nell'introduzione, per poter eseguire le raccomandazioni, bisogna riempire la matrice di ratings ottenuta. A questo proposito, utilizzeremo due algoritmi diversi: K-Nearest Neighbors e Singular Value Decomposition.

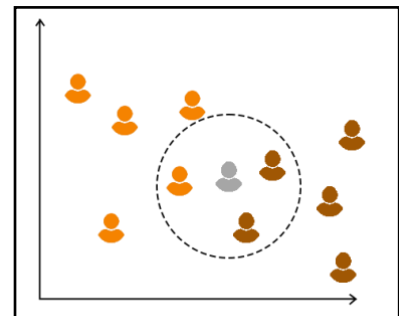
L'algoritmo K-Nearest-Neighbors associa a ciascun utente (o a ciascun prodotto) un numero k di vicini. Si tratta di un algoritmo supervisionato ampiamente utilizzato per la classificazione e la regressione, nel nostro caso verrà allenato sulla totalità dei dati e il suo obiettivo sarà di riempire la matrice. Non trattandosi di un problema di classificazione non possiamo verificare le sue performance con metriche quali accuracy, precision o recall, utilizzeremo invece le metriche di Mean Square Error e Root Mean Square Error.

L'errore quadratico medio è una misura per valutare la qualità di uno stimatore rispetto alla sua variazione e distorsione, si calcola con la seguente formula:

$$MSE = \frac{\sum_{i=1}^n (x_i - \hat{x}_i)^2}{n}$$

Dove  $x_i$  sono i valori rating reali e  $\hat{x}_i$  sono i rating previsti dal modello, più basso è il valore di MSE migliore è il modello. L'altra misura utilizzata è la radice dell'errore quadratico medio (RMSE), è tendenzialmente più interpretabile poiché fornisce un valore nella stessa unità di misura dei dati originali.

I modelli di apprendimento automatico KNN richiedono degli iperparametri in input: il più importante è il numero k di vicini che vanno considerati per calcolare la previsione. La scelta del valore di k ha un impatto sulle prestazioni del modello KNN, non esiste tuttavia una regola per determinarne il valore ottimale, dipende infatti dal contesto di analisi (figura 10).



*Figura 10: identificazioni dei "vicini" eseguita dall'algoritmo KNN in un contesto bidimensionale, il valore di k è posto pari a 3*



L'algoritmo deve inoltre utilizzare una misura di distanza che gli permetta di definire se due utenti (o prodotti) sono da considerarsi vicini o meno: vediamo la similarità del coseno e la correlazione di pearson.

La Cosine Similarity rappresenta una misura di somiglianza tra due vettori, nel nostro contesto i vettori sarebbero chiaramente le righe (o le colonne) della matrice di rating, geometricamente corrisponde al coseno dell'angolo che si viene a formare nello spazio n dimensionale tra i due vettori.

Nel caso non siano note alcune componenti del vettore, viene ad esse assegnato valore zero. Nel contesto dei rating (ovvero quando le componenti dei vettori assumono valori positivi 1-5), il valore di cosine similarity è compreso tra 0 e 1, la formula per calcolarlo è la seguente:

$$sim(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \|\vec{v}\|} = \frac{\sum_{i=1}^n u_i v_i}{\sqrt{\sum_{i=1}^n u_i^2} \sqrt{\sum_{i=1}^n v_i^2}}$$

La Pearson Correlation è data dall'indice di correlazione di pearson, ovvero il rapporto tra la covarianza e il prodotto delle deviazioni standard. L'indice assume valore da -1 (perfetta correlazione lineare negativa) a +1 (perfetta correlazione lineare positiva) dove 0 rappresenta l'incorrelazione, si calcola con la seguente formula:

$$sim(\vec{u}, \vec{v}) = \frac{\langle \vec{u} - \bar{u}, \vec{v} - \bar{v} \rangle}{\|\vec{u} - \bar{u}\| \|\vec{v} - \bar{v}\|} = \frac{\sum_{i=1}^n (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i=1}^n (u_i - \bar{u})^2} \sqrt{\sum_{i=1}^n (v_i - \bar{v})^2}}$$

Dove  $\bar{u} = |I_u|^{-1} \sum_{i \in I_u} u_i$  e  $\bar{v} = |I_v|^{-1} \sum_{i \in I_v} v_i$  rappresentano le medie dei due vettori. Qual'ora alcune componenti del vettore non siano note la formula si trasforma come segue, ove  $I_{uv}$  rappresenta l'insieme degli indici delle componenti note in entrambi i vettori:

$$sim(\vec{u}, \vec{v}) = \frac{\sum_{i \in I_{uv}} (u_i - \bar{u})(v_i - \bar{v})}{\sqrt{\sum_{i \in I_{uv}} (u_i - \bar{u})^2} \sqrt{\sum_{i \in I_{uv}} (v_i - \bar{v})^2}}$$

Esempio del calcolo delle misure di somiglianza (dati fittizi):

	Dead guy Ale	Keystone ice	moon man	winter lager	festive Ale	imperial IPA
chilidog	1.0	0.0	2.0	0.0	5.0	0.0
drunkman	5.0	2.0	0.0	2.0	2.0	0.0

Cosine Similarity (chilidog, drunkman):  $\frac{1*5+5*2}{\sqrt{1^2+5^2}\sqrt{5^2+2^2}} = 0.5463$

Pearson Correlation (chilidog, drunkman):  $I_u = \{0,2,4\}, I_v = \{0,1,3,4\}, I_{uv} = \{0,4\}$

$$\bar{u} = |I_u|^{-1} \sum_{i \in I_u} u_i = \frac{1+2+5}{3} = 2,667, \quad \bar{v} = 2,75$$

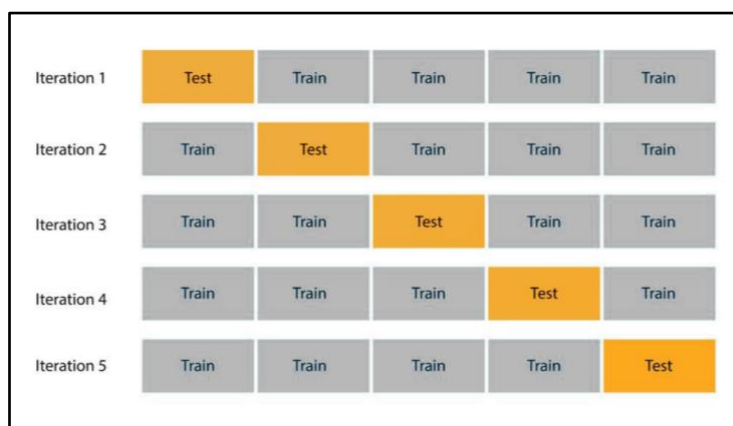
$$\frac{(1-2,667)(5-2,75)+(5-2,667)(2-2,75)}{\sqrt{(1-2,667)^2+(5-2,667)^2} \sqrt{(5-2,75)^2+(2-2,75)^2}} = -0,809$$

Il modello KNN richiede infine che venga specificato il tipo di approccio da seguire: si può allenare il modello attraverso un'ottica user-based, ovvero cercare user vicini tra loro (utilizzando quindi le righe come vettore nel calcolo delle distanze) o un'ottica item-based, ovvero cercare item vicini tra loro (utilizzando invece i vettori colonna).

Per fare in modo che gli iperparametri appena visti siano ottimizzati, rispetto al nostro contesto di analisi, si esegue una gridsearch: questa metodologia permette di considerare tutte le possibili combinazioni di iperparametri e di scegliere quella con le metriche MSE e RMSE migliori.

Per poter calcolare lo scarto quadratico medio, e la sua radice, in modo accurato bisogna suddividere i dati reali di cui disponiamo in un dataset di train e uno di test, si tratta della tecnica di model validation più comune e prende il nome di holdout method.

Per non sacrificare dati preziosi decidiamo tuttavia di utilizzare la cross validation per valutare il modello: essa consiste nel partizionare i dati a disposizione in k-folds (noi utilizziamo  $k=5$ ) mutualmente esclusivi, si stabilirà poi a turni lo spicchio utilizzato per la validazione mentre i rimanenti verranno utilizzati per il train del modello (figura 11). L'obiettivo della cross validation è ricavare metriche di MSE e RMSE non distorte.



*Figura 11: funzionamento della tecnica di model validation cross validation k-folds, con k posto uguale a 5*

La gridsearch fornisce come risultato la combinazione di iperparametri osservabile nella figura 12. Vediamo inoltre come funziona la cross validation osservando, dopo aver definito l'algoritmo KNN con gli iperparametri ottimali, le metriche calcolate utilizzando a turno ciascuno dei k-fold come validation set (figura 13).

```
Best RMSE = 0.6471
Best MSE = 0.4188
Best MSE configuration = {'k': 140, 'sim_options': {'name': 'cosine', 'user_based': True}}
```

	fold 1	fold 2	fold 3	fold 4	fold 5	mean
MSE	0.403634	0.401453	0.403568	0.402413	0.404349	0.634888
RMSE	0.635322	0.633604	0.635270	0.634360	0.635885	0.403084

*Figura 12 e 13: combinazione di iperparametri dell'algoritmo KNN ottimizzati rispetto al nostro problema applicativo e metriche di errore cross-validate rispetto ai rispettivi k-fold*

Avendo definito l'algoritmo KNNBasic (contenuto nella libreria surprise) con gli iperparametri ottimali, possiamo ora riempire la matrice applicando la funzione predict ad ogni osservazione vuota. Otteniamo la matrice completa 4000x4000 (figura 14).

item_name	Cocoa Porter Winter Warmer	Coffee Bender	Coffee Porter	Coffee Stout	Coffee Stout (Brewmaster Series)
user_name					
morebeergood	0.0	0.0	3.0	0.0	3.0
morimech	0.0	3.0	0.0	0.0	4.0
mortarit	0.0	0.0	4.0	0.0	0.0
mostpoetsdo	0.0	0.0	0.0	0.0	0.0
mothman	0.0	4.0	0.0	4.0	0.0

item_name	Cocoa Porter Winter Warmer	Coffee Bender	Coffee Porter	Coffee Stout	Coffee Stout (Brewmaster Series)
user_name					
morebeergood	2.99	3.92	3.00	3.80	3.00
morimech	2.96	3.00	3.58	3.78	4.00
mortarit	2.94	3.98	4.00	3.72	3.79
mostpoetsdo	2.89	3.89	3.62	3.76	3.83
mothman	2.95	4.00	3.59	4.00	3.83

*Figura 14: confronto tra la matrice incompleta e quella completa rispetto a 5 osservazioni casuali*

Possiamo ora procedere con il secondo algoritmo: la Singular Value Decomposition è una forma di decomposizione matriciale (matrix factorization) che permette di scomporre la matrice in una combinazione lineare di matrici di rango inferiore.

Nel contesto del collaborative filtering, l'algoritmo SVD viene utilizzato per apprendere le caratteristiche latenti degli utenti, ovvero i loro rating, e stimare le valutazioni mancanti.

Così come il K-Nearest Neighbors, anche la Singular Value Decomposition richiede degli iperparametri. Allo stesso modo utilizzeremo una gridsearch per verificare la combinazione che fornisce le metriche di Mean Square Error e di Root Mean Square Error migliori. Vediamo gli iperparametri richiesti dall'algoritmo SVD contenuto nella libreria Python surprise:

1. *n\_factors*: equivale alle dimensioni che il modello dovrà utilizzare nella scomposizione della matrice, il valore di default è 100, un valore maggiore potrebbe catturare meglio le relazioni latenti nei dati ma porta a un carico computazionale maggiore.
2. *n\_epochs*: le “epoche” corrispondono alle iterazioni complete che il modello esegue sui dati, il valore predefinito è 20 e valori più elevati rendono più lento l'apprendimento.
3. *lr\_all*: il learning rate è un iperparametro che appare in diversi modelli, si tratta di un peso che regola l'entità con cui i parametri del modello vengono aggiornati durante ciascuna interazione. Normalmente assume valore 0.005, valori più alti rendono l'addestramento più veloce, ma aumentano il rischio di instabilità
4. *reg\_all*: la regolarizzazione ha l'obiettivo di evitare l'overfitting, un valore maggiore evita che il modello si adatti eccessivamente ai dati ma rischia anche di limitare l'apprendimento.

La gridsearch fornisce come risultato la combinazione di iperparametri osservabile in figura 15. Come in precedenza, osserviamo inoltre le metriche MSE e RMSE cross-validate, ottenute applicando l'algoritmo SVD allenato con gli iperparametri ottimali (figura 16).

```
Best RMSE = 0.6258
Best MSE = 0.3916
Best MSE configuration = {'n_factors': 5, 'n_epochs': 40, 'lr_all': 0.003, 'reg_all': 0.08}
```

	fold 1	fold 2	fold 3	fold 4	fold 5	mean
MSE	0.378971	0.381560	0.378956	0.378243	0.379704	0.616024
RMSE	0.615607	0.617706	0.615594	0.615014	0.616201	0.379487

Figura 15 e 16: combinazione di iperparametri dell'algoritmo SVD ottimizzati rispetto al nostro problema applicativo e metriche di errore cross-validate rispetto ai rispettivi k-fold

Possiamo procedere riempiendo la matrice sparsa utilizzando l'algoritmo appena definito (figura 17).

item_name	Cocoa Porter	Winter Warmer	Coffee Bender	Coffee Porter	Coffee Stout	Coffee Stout (Brewmaster Series)
user_name						
morebeergood		2.83	3.74	3.00	3.52	3.00
morimech		2.98	3.00	3.51	3.68	4.00
mortarit		2.71	3.63	4.00	3.41	3.43
mostpoetsdo		3.06	3.96	3.58	3.75	3.77
mothman		2.79	4.00	3.31	4.00	3.50

Figura 17: matrice completata attraverso l'applicazione dell'algoritmo SVD con gli iperparametri ottimizzati

Volendo quindi confrontare i due algoritmi utilizzati, notiamo come, rispetto alle metriche di performance previsiva MSE e RMSE cross-validate, il modello SVD performa meglio.

Dato che l'RMSE, come visto in precedenza, ha la stessa unità di misura rispetto ai dati, possiamo dire che, nel modello migliore, la previsione del rating sbaglia mediamente di 0.616 (figura 18).

Cross validated RMSE obtained with KNN algorithm = 0.6348880782371433
Cross validated RMSE obtained with SVD algorithm = 0.6160242070225597
-----
Cross validated MSE obtained with KNN algorithm = 0.4030835230537315
Cross validated MSE obtained with SVD algorithm = 0.3794866713224371

Figura 18: confronto delle metriche cross-validate nei due modelli

Abbiamo quindi ottenuto due matrici complete, riempite rispettivamente utilizzando l'algoritmo KNN e l'algoritmo SVD. Per creare la lista di L items da raccomandare a ciascun utente, non faremo altro che osservare i rating previsti con valore maggiore.

Per evitare che vengano suggerite birre già recensite in passato, procediamo ponendo i rating reali presenti nella matrice completa (ovvero il 6% circa) pari a 0.

Confrontiamo le raccomandazioni, eseguite nei confronti di un utente casuale, estratte dalle due matrici di rating ottenute (figura 19). ‘sarahspat’ recensisce realmente 88 birre, il voto medio si assesta a 3.6, l’algoritmo KNN prevede che la birra a cui potrebbe dare una valutazione maggiore (pari a 4.45) è la “Cantillon Blåbær Lambik”. Questo risultato non ci sorprende, questa birra infatti, nonostante abbia un numero di recensioni reali nella media (123), ha un rating overall medio di quasi un punto superiore alla valutazione media di tutte le birre presenti nel dataset: 4.45 contro 3.62, ci aspettiamo quindi che appaia spesso nelle raccomandazioni. Seguono la “Geuze Cuvée J&J Blauw” e la “Veritas 004”, tutte birre mediamente molto apprezzate e popolari.

L’algoritmo SVD si trova d’accordo con le prime tre raccomandazioni fornite dal KNN, nonostante il rating medio cambi leggermente, dalla quarta posizione qualche birra viene invertita di posizione, come la “Deviation” e la “Kaggen!”, e andando avanti nella top 10 si iniziano a trovare suggerimenti differenti.

Osservando però come performano globalmente le birre suggerite, e osservando le raccomandazioni effettuate ad un secondo utente (figura 20), è evidente come la popolarità introduca un certo bias: ci aspettiamo che, almento tra le prime posizioni delle raccomandazioni, spicchino sempre le stesse birre, ovvero quelle generalmente più apprezzate.

KNN recommender:	
item_name	sarahspat
Cantillon Blåbær Lambik	4.45
Geuze Cuvée J&J (Joost En Jessie) Blauw (Blue)	4.41
Veritas 004	4.38
Deviation – Bottleworks 9th Anniversary	4.38
Kaggen! Stormaktsporter	4.37
Portsmouth Kate The Great	4.37
Pliny The Younger	4.36
Heady Topper	4.35
Trappist Westvleteren 12	4.35
Reality Czeck	4.34
SVD recommender:	
item_name	sarahspat
Cantillon Blåbær Lambik	4.41
Geuze Cuvée J&J (Joost En Jessie) Blauw (Blue)	4.41
Veritas 004	4.36
Kaggen! Stormaktsporter	4.34
Deviation – Bottleworks 9th Anniversary	4.33
Citra DIPA	4.31
Heady Topper	4.31
Pliny The Younger	4.29
Live Oak HefeWeizen	4.29
Reality Czeck	4.28

*Figura 19: confronto tra le raccomandazioni eseguite mediante l'utilizzo dell'algoritmo KNN e mediante l'utilizzo dell'algoritmo SVD, rispetto all'utente 'sarahspat'*

KNN recommender:	
item_name	AgentMunky
Cantillon Blåbær Lambik	4.46
Geuze Cuvée J&J (Joost En Jessie) Blauw (Blue)	4.41
Veritas 004	4.39
Pliny The Younger	4.39
Live Oak HefeWeizen	4.38
Deviation - Bottleworks 9th Anniversary	4.38
Kaggen! Stormaktsporter	4.38
Founders CBS Imperial Stout	4.36
Heady Topper	4.36
Citra DIPA	4.36
SVD recommender:	
item_name	AgentMunky
Geuze Cuvée J&J (Joost En Jessie) Blauw (Blue)	4.45
Cantillon Blåbær Lambik	4.43
Veritas 004	4.39
Deviation - Bottleworks 9th Anniversary	4.37
Kaggen! Stormaktsporter	4.36
Citra DIPA	4.35
Heady Topper	4.34
Pliny The Younger	4.33
Live Oak HefeWeizen	4.32
Maple Bacon Coffee Porter	4.31

*Figura 20: confronto tra le raccomandazioni eseguite mediante l'utilizzo dell'algoritmo KNN e mediante l'utilizzo dell'algoritmo SVD, rispetto all'utente 'AgentMunky'.*

*Vengono evidenziati i prodotti che non apparivano nella top 10 dell'utente visto in precedenza*



## ***Clusterizzazione degli utenti***

La clusterizzazione è una tecnica che permette di classificare utenti simili, in termini di simili caratteristiche. Nel contesto dei ratings, gli utenti simili sono idealmente coloro che danno le stesse valutazioni agli stessi prodotti.

Gli algoritmi di clusterizzazione sono definiti non-supervisionati, questo poiché i cluster prodotti non sono chiaramente etichettati, talvolta possiamo dedurre le motivazioni latenti alla divisione effettuata osservando gli elementi che appartengono a ciascun cluster.

L'obiettivo della clusterizzazione è massimizzare la similarità interna ai cluster e minimizzare le similarità tra cluster differenti.

L'algoritmo K-means permette di partizionare gli utenti in k cluster, dove k è un iperparametro che va ottimizzato. Ogni cluster viene rappresentato da un centroide, essendo i rating variabili continue, gli utenti vengono assegnati al cluster il cui centroide è più vicino a loro, l'algoritmo K-means esegue quindi una suddivisione sferica nello spazio delle osservazioni.

La metrica di distanza geometrica utilizzata dall'algoritmo K-means (implementato nel pacchetto Python scikit-learn) è la distanza euclidea:

$$D_{uv} = \sqrt{\sum_{k=1}^n (x_{ku} - x_{kv})^2}$$

L'algoritmo k-means utilizza la distanza euclidea per calcolare la distanza tra i punti dati e i centroidi dei cluster. I centroidi rappresentano i punti centrali di ciascun cluster. Durante l'esecuzione dell'algoritmo, i punti dati vengono assegnati al cluster il cui centroide è più vicino in base alla distanza euclidea. In seguito, i centroidi vengono aggiornati calcolando la media dei punti dati assegnati a ciascun cluster. Questo processo viene ripetuto fino a quando i centroidi convergono, o si raggiunge un numero limite di iterazioni, i cluster risultanti rappresentano idealmente gruppi di utenti con gusti simili.

Come accennato in precedenza, il numero  $k$  di cluster va ottimizzato in base alla natura del problema: si può scegliere a priori (qual'ora avessimo già un'idea sulla segmentazione), si può impostare convenzionalmente  $k = \sqrt{n/2}$  (per dataset contenenti poche osservazioni), altrimenti si possono osservare metriche di omogenità interna al cluster come la Within Cluster Sum of Squares. Questa metrica andrebbe formalmente minimizzata, ma vedremo in seguito come, nella pratica, la soluzione è trovare un compromesso. Altra metrica che viene talvolta utilizzata per la valutazione dell'algoritmo è la BCSS (between cluster sum of squares).

La WCSS si ottiene con la seguente formula:

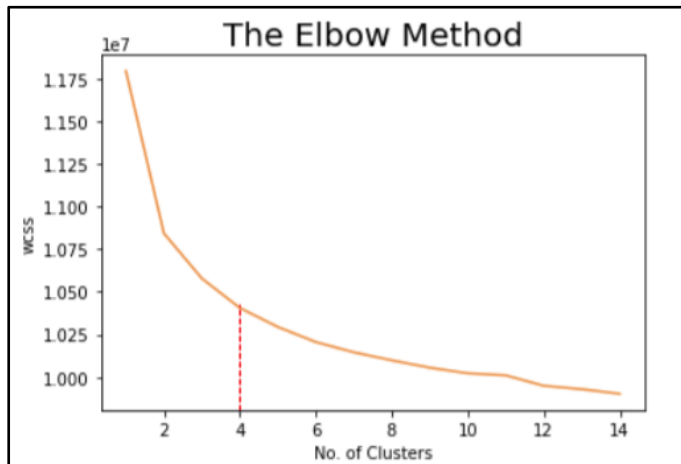
$$\sum_{i=1}^k \sum_{x \in S_i} \|x - \mu_i\|^2$$

Dove  $k$  è il numero di cluster,  $S_i$  è il cluster  $i$ -esimo e  $\mu_i$  è la media del cluster, precedentemente definita come centroide. Durante la convergenza dell'algoritmo k-means, a ciascuna iterazione i centroidi sono calcolati come segue:

$$\mu_i = \frac{1}{|S_i|} \sum_{x \in S_i} x$$

Per scegliere il numero ottimale di cluster rappresentiamo quindi graficamente il variare della WCSS al variare del numero di cluster, da 1 a 15, applichiamo poi una comune euristica, definita Elbow method, utilizzata per l'ottimizzazione matematica. In pratica si sceglie il punto in cui il cambiamento di WCSS inizia a stabilizzarsi, in quel punto i rendimenti decrescenti non valgono più il costo aggiuntivo, ovvero l'aumento del numero di cluster.

Applicando quanto detto fino ad ora alla matrice di dati incompleta (per non introdurre distorsioni dovute alle previsioni), decidiamo che il numero ottimale di cluster con cui applicare l'algoritmo k-means è 4 (figura 21).



*Figura 21: rappresentazione grafica del variare della WCSS all'aumentare del numero di cluster*

Procediamo inizializzando l'algoritmo K-means sulla matrice di rating incompleta: oltre all'iperparametro fondamentale  $k$ , impostiamo il numero massimo di iterazioni (parametro `max_iter`) a 300 e il numero di inizializzazioni (`n_init`) a 10, questo significa che l'algoritmo verrà eseguito 10 volte partendo da centroidi differenti.

I quattro cluster ottenuti hanno numerosità omogenea (figura 22). Non avendo una suddivisione reale con cui confrontare i risultati, possiamo valutare il modello utilizzando il coefficiente di Silhouette: questa metrica viene calcolata rispetto a ciascuna osservazione e la media complessiva viene interpretata.

Cluster 1:	770 utenti
Cluster 2:	1197 utenti
Cluster 3:	1002 utenti
Cluster 4:	1031 utenti

*Figura 22: numerosità dei cluster ottenuti tramite l'algoritmo K-means con  $k=4$*

Il coefficiente di Silhouette rispetto a una singola osservazione si calcola come segue:

$$s = \frac{b - a}{\max(a, b)}$$

Dove  $a$  equivale alla distanza media tra l'osservazione e tutte le osservazioni appartenenti allo stesso cluster mentre  $b$  equivale alla distanza media tra l'osservazione e tutte le osservazioni appartenenti al cluster più vicino.

Il coefficiente di Silhouette calcolato sulla totalità dei dati assume valore da -1 (clusterizzazione errata) a 1 (cluster densi e separati), dove 0 significa che i cluster sono sovrapposti.

Calcolando il coefficiente rispetto ai cluster precedentemente definiti, otteniamo un valore pari a -0.030, le segmentazioni ottenute con l'algoritmo K-means presentano quindi sovrapposizioni. Provare ad aumentare o diminuire il numero di cluster  $k$  non migliora il valore del coefficiente.

Per quanto il risultato della segmentazione degli utenti appaia deludente, il fatto che i cluster siano sovrapposti non è una sorpresa, bisogna ricordare che la matrice è al 94% composta da valori nulli, questo rende i cluster, e in primis le distanze euclidee, complesse da definire.

## Conclusioni

Grazie all'utilizzo degli algoritmi K-Nearest Neighbors e Singular Value Decomposition, abbiamo raggiunto l'obiettivo di analisi: siamo riusciti a riempire la matrice sparsa di rating.

Come abbiamo osservato, basare i suggerimenti solo sui valori predetti, introduce una distorsione causata dalla popolarità che mette in secondo piano il concetto di collaborative filtering. Per valutare, e quindi ottimizzare, un sistema di raccomandazione, andrebbero tenute in considerazione altre metriche oltre all'accuratezza, tra cui:

1. *Coverage*: ovvero quanti prodotti vengono suggeriti, vogliamo massimizzarla per quanto possibile in modo da diversificare i suggerimenti.
2. *Novelty*: si riferisce alla capacità del sistema di raccomandare items nuovi e inaspettati, non banalmente i best-seller.
3. *Serendipity*: similamente alla novelty, bisogna riuscire a suggerire prodotti inaspettati che possano suscitare l'interesse dell'utente.
4. *Diversity*: similamente alla coverage, bisogna riuscire a presentare una varietà di prodotti che coprano più categorie.
5. *Robustess*: il modello deve riuscire a superare problemi quali il cold-start (assenza di rating precedenti negli utenti nuovi) o la presenza di rating fake, garantendo una qualità consistente delle raccomandazioni.
6. *Scalability*: il sistema deve riuscire a gestire un carico computazionale sempre crescente.

Nella parte inerente al clustering abbiamo constatato che il contesto di analisi non si presta a una segmentazione efficace. Idealmente una matrice con più valori reali migliorerebbe i risultati, alternativamente si potrebbe provare a eseguire l'algoritmo K-means utilizzando una misura di distanza diversa (come la cosine distance).

Altra possibilità sarebbe quella di valutare altri algoritmi di segmentazione più complessi che permettano di gestire meglio grandi quantità di dati mancanti: ad esempio il clustering gerarchico.