Relazione Progetto MIGA

Gabriele Carrivale & Marco Midali

Contents

1 Breve riassunto

L'obiettivo che si vuole raggiungere è implementare un sistema di raccomandazioni di ricette, utilizzando due metodologie diverse:

- Content-Based: Le raccomandazioni vengono effettuate sulla base delle caratteristiche degli items (nel nostro caso ricette) più simili.
- Collaborative Filtering: Le raccomandazioni vengono eseguite in funzione degli item/user più simili.

Una delle finalità di tale progetto è di effettuare un confronto tra le due.
Le informazioni e i dati utilizzati sono presi dal sito Food.com. In questo sito gli utenti possono interagire con un catalogo di ricette aggiungendone di nuove oppure dando una recensione a quelle pubblicate da altri utenti, la quale può essere un rate da 0 a 5 e in aggiunta anche un eventuale commento per spiegare il rate dato.
È stata effettuata un'analisi delle statistiche descrittive rilevanti per il problema in questione, raccogliendo dunque informazioni utili per estrapolare features utilizzabili nello sviluppo dei punti successivi.
Tali statistiche sono risultate utili per realizzare gli obiettivi di ciascuna delle metodologie descritte in precedenza. Differenti analisi sono state realizzate e distinti algoritmi sono stati implementati, degli esempi potrebbero essere la generazione delle matrici dei rating prodotte sfruttando i differenti algoritmi (quali kNN per entrambi e SVD per la matrix factotization per Collaborative Filtering). Inoltre, la suddivisione in classi degli utenti servendosi di differenti librerie e analizzando le metriche per compararli tra loro, oppure ancora la ricerca dei top-n items utilizzando la matrice dei rating descritta precedentemente.
Infine, mediante la libreria Pymoo è stata eseguita la formulazione e risoluzione di un problema MOO (Multi Objective Optimization) confrontando tra loro due differenti algoritmi evolutivi che sfruttano delle funzioni genetiche realizzate adhoc e quindi dunque dipendenti dal problema in questione.

2 Introduzione al problema

Una volta aver scaricato il dataset da Kaggle e aver "selezionato" le informazioni più utili ai fini di risolvere il nostro problema, è stata effettuata l'analisi delle statistiche descrittive e le effettive operazioni per la costruzione di un sistema di raccomandazioni di ricette. I due dataset che si selezionano sono i seguenti:

- RAW_interactions.csv
- RAW_recipe.csv

Il primo contiene tutte le interazioni di ciascun utente con le ricette, con relativi ratings, reviews e datetime di quanto è stata eseguita tale interazione. Il secondo contiene tutte le informazioni utili a caratterizzare una singola ricetta, abbiamo informazioni come: ingredients, tags, steps, ecc...
Non tutte le informazioni contenute nei dataframe ci saranno utili per i nostri scopi, dunque si è deciso di non prenderli in considerazione eliminandoli dal dataframe. Riportiamo le prime righe di ciascun dataframe:

- Interactions

	user_id	recipe_id	date	rating	review
0	38094	40893	2003-02-17	4	Great with a salad. Cooked on top of stove for...
1	1293707	40893	2011-12-21	5	So simple, so delicious! Great for chilly fall...
2	8937	44394	2002-12-01	4	This worked very well and is EASY. I used not...
3	126440	85009	2010-02-27	5	I made the Mexican topping and took it to bunk...
4	57222	85009	2011-10-01	5	Made the cheddar bacon topping, adding a sprin...
...	...	...	...	...	...
1132362	116593	72730	2003-12-09	0	Another approach is to start making sauce with...
1132363	583662	386618	2009-09-29	5	These were so delicious! My husband and I tru...
1132364	157126	78003	2008-06-23	5	WOW! Sometimes I don't take the time to rate ...
1132365	53932	78003	2009-01-11	4	Very good! I used regular port as well. The ...
1132366	2001868099	78003	2017-12-18	5	I am so glad I googled and found this here. Th...

1132367 rows × 5 columns

- Recipes

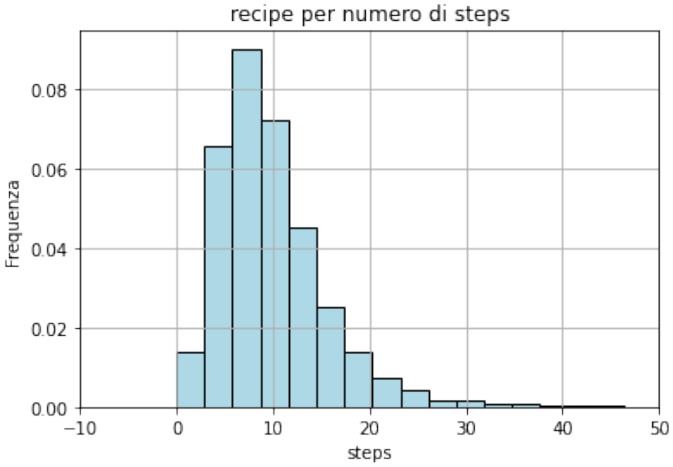
	name	id	minutes	contributor_id	submitted	tags	nutrition	n_steps	steps	description	ingredients	n_ingredients
0	arriba baked winter squash mexican style	137739	55	47892	2005-09-16	['60-minutes-or-less', 'time-to-make', 'course...	[51.5, 0.0, 13.0, 0.0, 2.0, 0.0, 4.0]	11	['make a choice and proceed with recipe', 'dep...	autumn is my favorite time of year to cook! th...	['winter squash', 'mexican seasoning', 'mixed ...	7
1	a bit different breakfast pizza	31490	30	26278	2002-06-17	['30-minutes-or-less', 'time-to-make', 'course...	[173.4, 18.0, 0.0, 17.0, 22.0, 35.0, 1.0]	9	['preheat oven to 425 degrees f', 'press dough...	this recipe calls for the crust to be prebaked...	['prepared pizza crust', 'sausage patty', 'egg...	6
2	all in the kitchen chili	112140	130	196586	2005-02-25	['time-to-make', 'course', 'preparation', 'mai...	[269.8, 22.0, 32.0, 48.0, 39.0, 27.0, 5.0]	6	['brown ground beef in large pot', 'add choppe...	this modified version of 'mom's' chili was a h...	['ground beef', 'yellow onions', 'diced tomato...	13
3	alouette potatoes	59389	45	68585	2003-04-14	['60-minutes-or-less', 'time-to-make', 'course...	[368.1, 17.0, 10.0, 2.0, 14.0, 8.0, 20.0]	11	['place potatoes in a large pot of lightly sal...	this is a super easy, great tasting, make ahea...	['spreadable cheese with garlic and herbs', 'n...	11
4	amish tomato ketchup for canning	44061	190	41706	2002-10-25	['weeknight', 'time-to-make', 'course', 'main-...	[352.9, 1.0, 337.0, 23.0, 3.0, 0.0, 28.0]	5	['mix all ingredients& boil for 2 1 / 2 hours ...	my dh's amish mother raised him on this recipe...	['tomato juice', 'apple cider vinegar', 'sugar...	8
...	...	...	...	...	...	...	...	...	...	...	...	...
231632	zydeco soup	486161	60	227978	2012-08-29	['ham', '60-minutes-or-less', 'time-to-make', ...	[415.2, 26.0, 34.0, 26.0, 44.0, 21.0, 15.0]	7	['heat oil in a 4-quart dutch oven', 'add cele...	this is a delicious soup that i originally fou...	['celery', 'onion', 'green sweet pepper', 'gar...	22
231633	zydeco spice mix	493372	5	1500678	2013-01-09	['15-minutes-or-less', 'time-to-make', 'course...	[14.8, 0.0, 2.0, 58.0, 1.0, 0.0, 1.0]	1	['mix all ingredients together thoroughly']	this spice mix will make your taste buds dance!	['paprika', 'salt', 'garlic powder', 'onion po...	13
231634	zydeco ya ya deviled eggs	308080	40	37779	2008-06-07	['60-minutes-or-less', 'time-to-make', 'course...	[59.2, 6.0, 2.0, 3.0, 6.0, 5.0, 0.0]	7	['in a bowl , combine the mashed yolks and may...	deviled eggs, cajun-style	['hard-cooked eggs', 'mayonnaise', 'dijon must...	8
231635	cookies by design cookies on a stick	298512	29	506822	2008-04-15	['30-minutes-or-less', 'time-to-make', 'course...	[188.0, 11.0, 57.0, 11.0, 7.0, 21.0, 9.0]	9	['place melted butter in a large mixing bowl a...	i've heard of the 'cookies by design' company....	['butter', 'eagle brand condensed milk', 'ligh...	10
231636	cookies by design sugar shortbread cookies	298509	20	506822	2008-04-15	['30-minutes-or-less', 'time-to-make', 'course...	[174.9, 14.0, 33.0, 4.0, 4.0, 11.0, 6.0]	5	['whip sugar and shortening in a large bowl , ...	i've heard of the 'cookies by design' company....	['granulated sugar', 'shortening', 'eggs', 'fl...	7

231637 rows × 12 columns

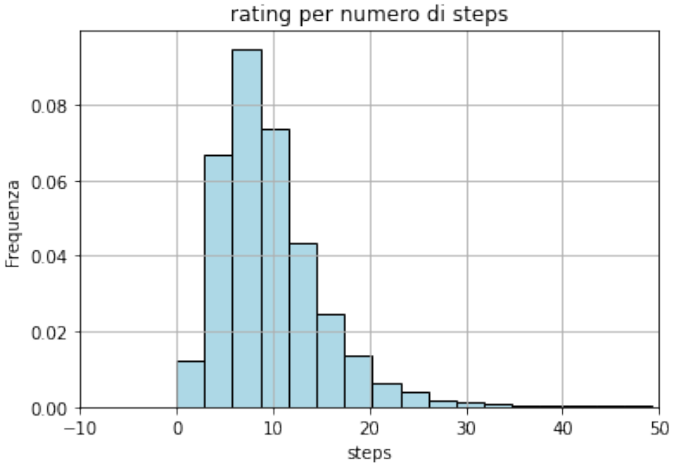
Nel dataframe "Interactions", le colonne "date" e "review", non sono necessarie ai fini dello svolgimento del progetto, dunque queste due verranno eliminate, lasciando solamente le altre 3 che sono rilevanti. Mentre nel dataframe "Recipes" sono state eliminate le colonne "name", "submitted", "steps", "description", "contributor_id", "ingredients" in quanto sono campi testuali o ridondanti (come ad esempio contributor_id che posso ricavarlo dal dataframe "Interactions").

Il grafico qui presente è un Word Cloud che è una rappresentazione grafica per valutare quanto dei termini sono più utilizzati di altri. L'importanza di ogni parola è determinata o dalla dimensione del carattere (come nel caso in figura) o dal colore. È molto utile quando si vuole fare un analisi visiva che a colpo d'occhio faccia risaltare i termini più utilizzati (come in questo caso).

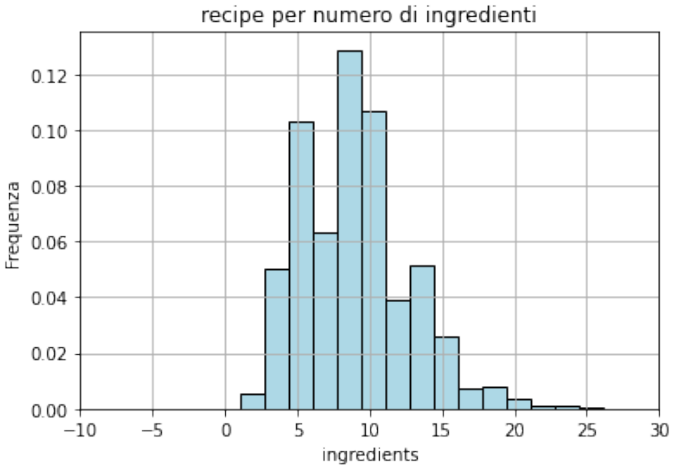
Questo word cloud è relativo al campo "tags" del dataframe ricette riportato qui sopra, per valutare appunto quali sono i tags più utilizzati per coloro che pubblicano le ricette. Si nota che i tags più utilizzati sono "Make course", "less time", "main ingredients", mentre invece tags come "pot slow", "carb healthy" sono invece molto meno usati.



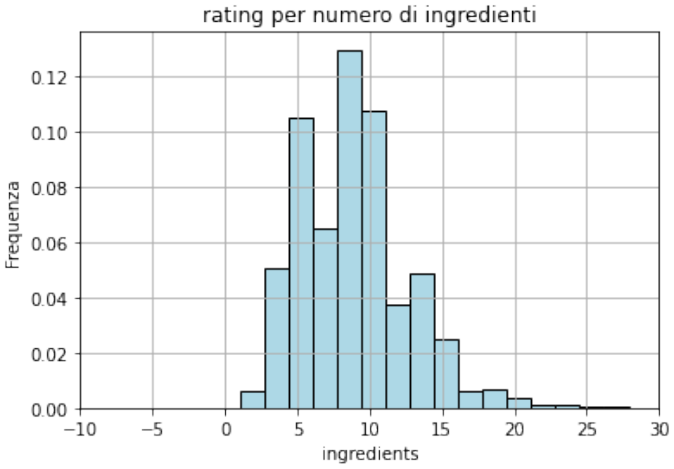
Il grafico soprastante è un istogramma che è una rappresentazione approssimativa della distribuzione dei dati numerici. Questo istogramma esamina la frequenza delle ricette in base al numero di steps. Questo ci dice che c'è una maggiore frequenza di ricette con né troppi pochi, né troppi steps, infatti, come possiamo notare le ricette più gettonate sono quelle da circa 5 a poco meno di 10 steps. Questo istogramma ha un andamento "a campana" tipico di una normale, infatti parte crescendo finché non arriva al picco più alto e poi diminuisce.



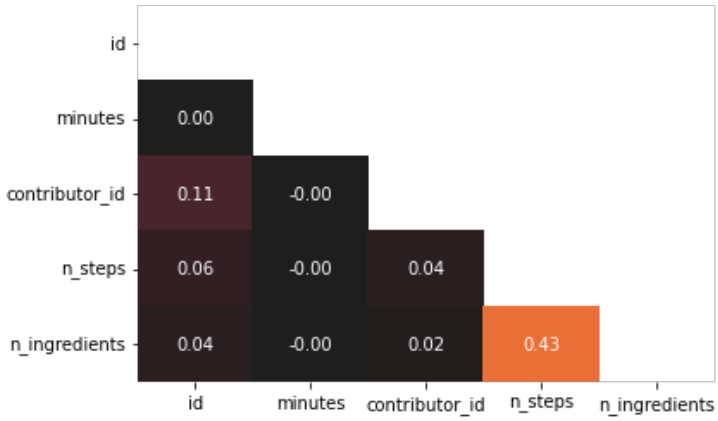
In quest'altro istogramma invece si sta valutando la frequenza dei rating in base al numero di steps, come possiamo notare comparando i due grafici i 2 grafici sono proporzionali.



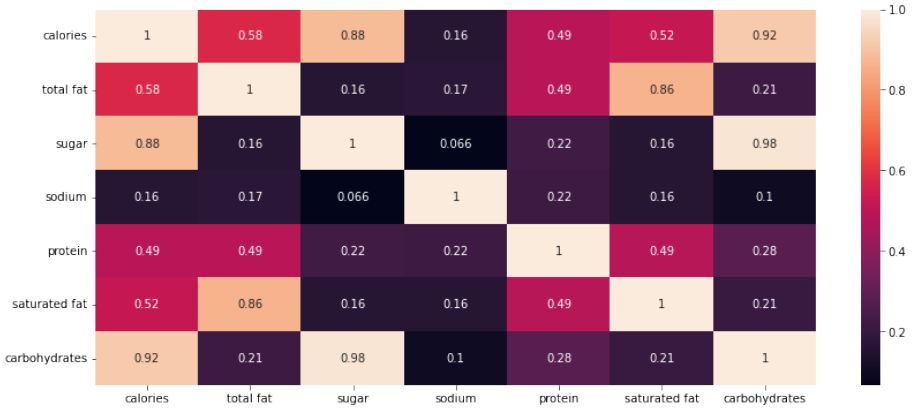
Simile a sopra si possono anche esaminare gli stessi istogrammi rispetto però al numero di ingredienti e non più rispetto al numero di steps, la prima cosa che notiamo e che rispetto a prima si ha un più un andamento sinusoidale (ovvero che cambia di intensità man mano che si cresce con il numero di ingredienti) rispetto all'andamento normale di quello precedente. Anche qui si riscontra un picco poco prima di 10 ingredienti.



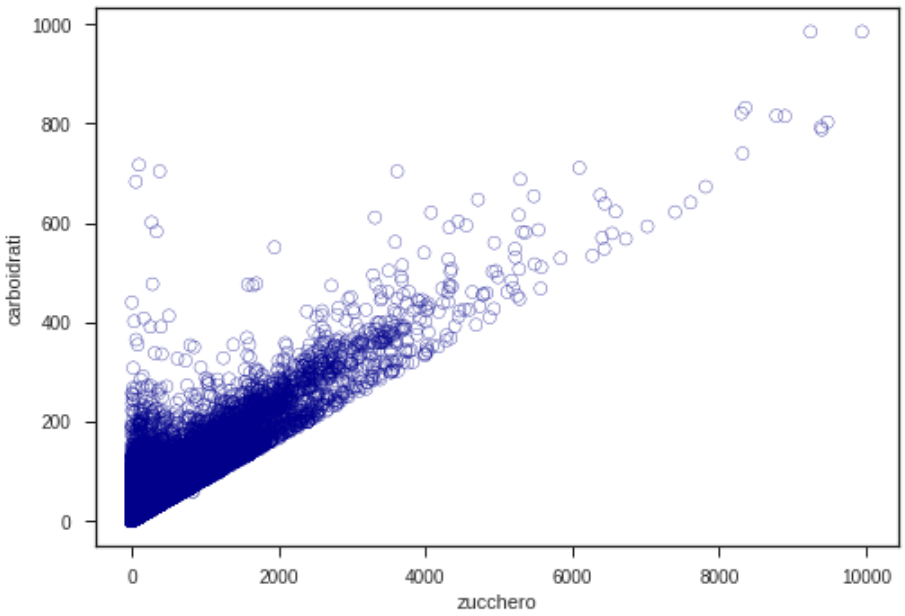
Come fatto prima per il numero di steps anche qua plottando e confrontando i due istogrammi recipe per numero di ingredienti e rating per numero di ingredienti si riscontra che sono anch'essi proporzionali, questo si nota dal fatto che la frequenza rimane pressoché la stessa per lo stesso numero di ingredienti.



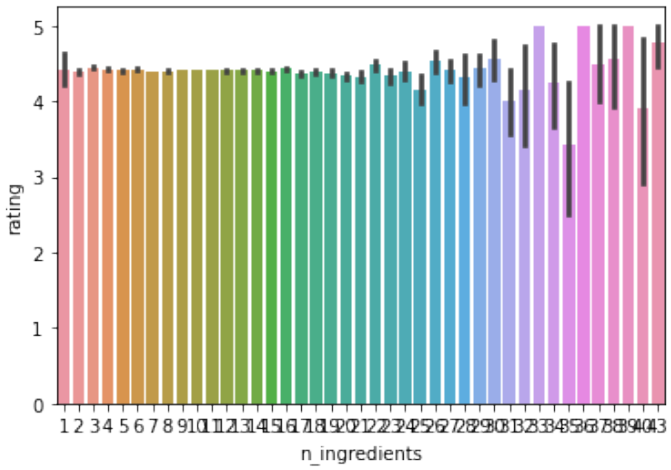
Ciò che si sta analizzando è una heatmap, la quale è una rappresentazione grafica dei dati dove i singoli valori contenuti in una matrice sono rappresentati da colori e da un numero, la quale evidenzia le correlazioni tra (in questo caso) delle colonne del dataframe. Se il numero è tra 0 (non incluso) e 0.5 si ha una correlazione positiva/negativa debole, mentre da 0.5 a 1 si ha una correlazione positiva/negativa forte, se il numero è positivo allora la correlazione sarà positiva e viceversa. Mentre se si ha 0 come numero indica che le due colonne che si intersecano nello 0 non sono correlate. Da questo notiamo che il n_steps è positivamente correlato ad n_ingredients, stranamente non c'è una correlazione positiva tra i minuti ed n_steps e n_ingredients.

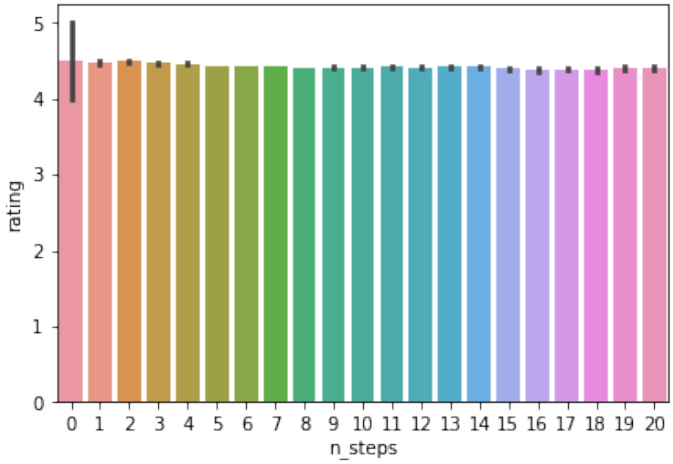


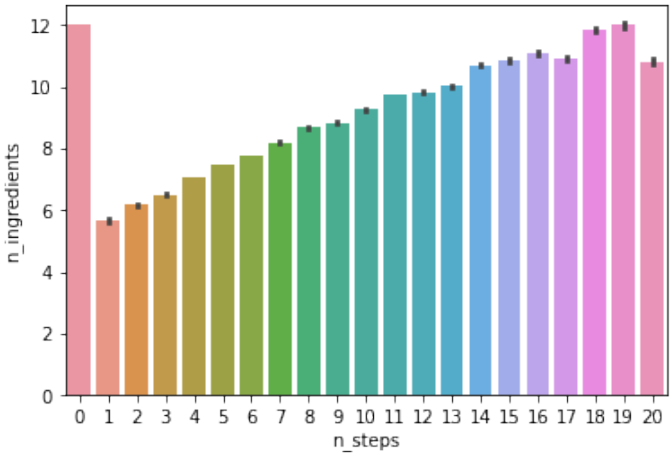
In questa heatmap dei nutrienti possiamo notare molteplici correlazioni tutte positive. In particolare, per esempio, "calories" è in una forte correlazione positiva con "sugar" e "Carbohydrates", in correlazione moderata con "total grass", "protein", "saturated fat", mentre c'è una correlazione debole con "sodium". La più alta correlazione che possiamo notare è quella tra "sugar" e "carbohydrates" che è di 0.98. Analizziamo più nel dettaglio la seguente correlazione:



Il seguente grafico (scatterplot o grafico a dispersione) è un tipo di grafico in cui due variabili di un set di dati sono riportate su uno spazio cartesiano.
I dati sono visualizzati su questo scatterplot tramite una collezione di punti ciascuno con una posizione sull'asse orizzontale determinato dallo zucchero e sull'asse verticale determinato dai carboidrati.
tramite l'analisi dell'andamento delle seguenti due variabili: sugar e carbohydrates si vede che al crescere dello zucchero cresce anche il valore legato ai carboidrati e viceversa. Dal grafico si evince una correlazione positiva forte, come già ci saremmo aspettati appunto dal valore visto sopra nella heatmap (0.98)


rating
n_ingredients


rating
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
n_steps

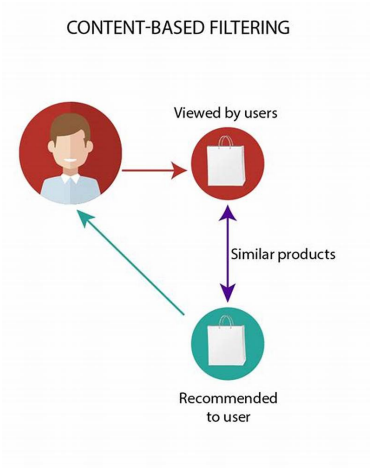


Un barplot è un grafico che è utilizzato per fare delle indagini statistiche.
In questi barplot si vuole fare 3 confronti rating per numero di ingredienti, rating per numero di step e numero di ingredienti per numero di steps. Come si può notare nei primi 2 barplot (che guardano la media di rating in base al numero di ingredienti/steps) sono abbastanza bilanciati, tengono (più o meno) entrambi una media tra il 4 e il 5. L'ultimo barplot invece serve per far vedere la relazione che c'è tra numero di steps e numero di ingredienti, come si può vedere (e come avevamo visto precedentemente dalla heatmap) il numero di ingredienti cresce al crescere del numero di steps (e chiaramente vale anche il contrario).

3 Recommender System

Anzitutto per effettuare l'analisi e costruzione di un Recommender System sfruttando le due diverse metodolige discusse in precedenza, è stato necessario andare a eseguire un partizionamento del dataset delle interazioni andando a selezionare esclusivamente gli utenti e le ricette con più interazioni (nel nostro caso, sono stati selezionati gli utenti con più di 20 interazioni e le ricette con più di 10 interazioni)

3.1 Content-Based



Come discusso precedentemente, la metodologia Content-Based permette di effettuare raccomandazioni basate sul contenuto, ovvero sulla descrizione del prodotto che si sta raccomandando. In questo approccio, le raccomandazioni sono generate utilizzando informazioni sul contenuto degli elementi, come ad esempio le caratteristiche, i generi o le parole chiave. Il sistema analizza le preferenze dell'utente e confronta i contenuti degli elementi consigliati con quelli già graditi dall'utente. Questo metodo è molto utile per raccomandare elementi simili a quelli che l'utente ha già gradito in passato.

3.1.1 Encode Categorical Value

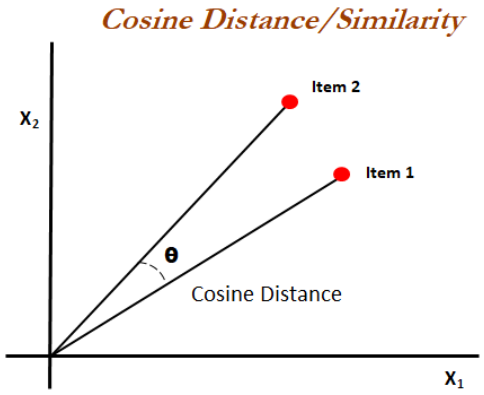
Anzitutto, per poter applicare dei metodi Content-Based per la raccomandazione delle ricette è stato necessario eseguire un encoding degli elementi contenuti all'interno della colonna "tags", i quali indicano appunto le etichette associate a ciascuna ricetta. Come visto nell'analisi statistica, anzitutto si indicano dei valori ripetuti e con quale frequenza, una volta aver trovato una certa corrispondenza, si procede con l'estrapolamento dei tags più utilizzati e per ciascuno di essi viene creata una singola colonna del dataframe contenente il nome del tag estratto. Questo ha permesso di avere per ogni ricetta un vettore composto esclusivamente da 0 e 1, in modo tale da poterlo usare per confrontare

e calcolare la cosine similarity (una tecnica euristica per la misurazione della similitudine tra due vettori effettuata calcolando il coseno tra di loro).
Ricordiamo che i valori di questo vettore assumono il seguente significato:

- 1 se quel tag è stato assegnato a quella ricetta
- 0 altrimenti

3.1.2 Cosine Distance come Metrica di distanza

La metrica utilizzata per determinare la distanza tra le ricette è appunto la Cosine Similarity, la quale prende in input due vettori (in questo caso i soli 0 e 1 associati alle colonne appena create) e produce in output la loro distanza. Banalmente, minore sarà questa distanza tra due ricette, più queste saranno simili.



3.1.3 Algoritmo kNN

Anzitutto si procede andando a generare quella che è definita la **matrice dei rating non fillata**, ovvero la matrice che ha come righe gli "users_id" degli utenti e come colonne i "recipe_id" delle ricette. Tale matrice è realizzata mediante la funzione **pivot_table** di **pandas**. $pivot\_table(df, values =' rating', index =' user\_id', columns =' recipe\_id', fill\_value = 0)$, la seguente funzione ha come primo parametro il dataframe, nel secondo parametro si indica il valore del singolo elemento (ovvero cosa ci sarà nella posizione [i,j] della matrice), il terzo parametro indica che elementi ci saranno nelle righe della matrice (in questo caso user_id), nel penultimo ci sarà il riferimento per le colonne (che in questo caso sarà recipe_id), mentre nell'ultimo parametro indica il valore da inserire nel caso in cui manca il rating da parte di un utente per una ricetta. Al momento è non fillata, dunque vi saranno presenti esclusivamente i rating "reali" definiti. Ricordiamo che i rating vanno da 1 a 6 e dunque i valori mancanti sono stati sostituiti con 0.

recipe_id	519	536	607	749	822	1005	1356	1985	2072	2496	...	367414	373346	376786	384800	385071	388397	400137	428004	428116	429576
user_id																					
1535	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3288	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4291	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4439	0	6	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4470	1	0	0	0	0	0	6	0	0	0	...	0	0	0	0	0	0	0	0	6	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2324285	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2549237	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	5	0
1800054678	0	0	0	0	0	0	6	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1802849661	0	0	0	0	0	0	0	0	0	6	...	0	0	0	0	0	0	0	0	0	0
2000431901	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

988 rows × 1696 columns

Adesso passiamo alla generazione della **matrice dei rating fillata**
Per prima cosa si considera per ciascun utente la lista delle ricette al quale lui ha dato un rating (ricordiamo però che in questa fase si andranno a considerare tutte le coppie di utenti e ricette identificate da una singola cella della matrice, che non hanno ancora un rating assegnato, dunque, quelli che hanno nella suddetta cella il valore 0). Successivamente si andrà a specificare il numero delle k-ricette che dovranno essere considerate per poter effettuare le nostre predizioni. Nel nostro caso sono state selezionate tutte le ricette con cui l'utente ha interagito. Successivamente, si è eseguita la fit del modello utilizzando il set di ricette descritto precedentemente.
L'applicazione del modello ci restituisce i top_items con le relative distanze dei k items più simili alla ricetta presa in analisi. Dunque si andrà a predirre il rating della coppia (user, recipe) andando a calcolare la media delle ricette più simili a quella presa in analisi e tra quelle recensite dall'utente. Infine il valore 0 della cella viene sostituito con il rating predetto. Facendo questo per ogni coppia (user, recipe), si ottiene la seguente matrice dei rating fillata:

	519	536	607	749	822	1005	1356	1985	2072	2496	...	367414	373346	376786	384800	385071	388397	400137	428004	428116	429576
user_id																					
1535	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
3288	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
4291	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
4439	5	6	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
4470	1	5	5	5	5	5	6	5	5	5	...	5	5	5	5	5	5	5	5	6	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2324285	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
2549237	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
1800054678	5	5	5	5	5	5	6	5	5	5	...	5	5	5	5	5	5	5	5	5	5
1802849661	5	5	5	5	5	5	5	5	5	6	...	5	5	5	5	5	5	5	5	5	5
2000431901	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5

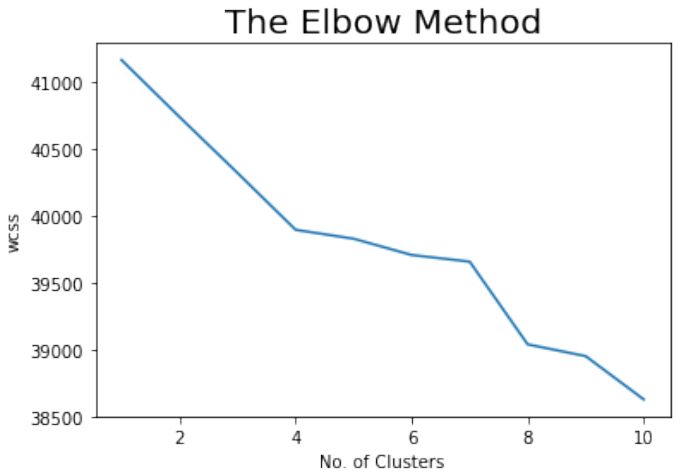
988 rows × 1696 columns

Quando si calcolano gli errori si avrà:

$$\text{Best MSE} = 0.6037245851230516$$
$$\text{Best RMSE} = 0.776997159018649$$

3.1.4 K-Means

Utilizzando la libreria di Scikit-Learn relativa allo sviluppo dell'algoritmo K-Means.



Con l'elbow method abbiamo ricavato il numero di cluster che serviranno poi per creare il modello tramite la classe KMeans che ha come parametri init (che avrà come valore 'kmeans++' che seleziona i cluster iniziali, per il kmeans in un modo intelligente per far si che converga più velocemente), n_clusters (il numero di clusters ovvero il numero di centroidi da generare. Nel nostro caso, come visto sopra con l'elbow method, questo valore sarà 8) e n_init (Numero di volte in cui l'algoritmo kmeans verrà eseguito con differenti centroidi). Mentre, la funzione fit_predict applicata al modello costruito prima permette di predirre le classi di appartenenza di ogni user e ritorna le relative labels. Dopodiché si calcola la Silhouette che è una metrica con valori tra 0 e 1 che serve per stimare quanto il modello creato sia buono. Più è vicino ad 1 meglio è.

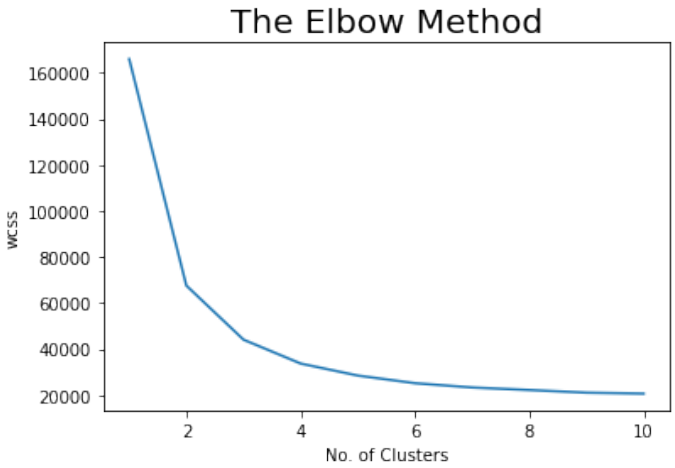
3.1.5 Varianti K-Means UserxUser RecipexRecipe

Si eseguono gli stessi passi effettuati prima, però, anziché fillare le celle vuote della matrice con la media generale dei rating calcoliamo la media user per user, come risultato avremo:

recipe_id	519	536	607	749	822	1005	1356	1985	2072	2496	...	367414	373346	376786	384800	385071	388397	400137	428004	428116
user_id																				
1535	5.717391	5.717391	5.717391	5.717391	5.717391	5.717391	5.717391	5.717391	5.717391	5.717391	...	5.717391	5.717391	5.717391	5.717391	5.717391	5.717391	5.717391	5.717391	5.717391
3288	5.206897	5.206897	5.206897	5.206897	5.206897	5.206897	5.206897	5.206897	5.206897	5.206897	...	5.206897	5.206897	5.206897	5.206897	5.206897	5.206897	5.206897	5.206897	5.206897
4291	5.740741	5.740741	5.740741	5.740741	5.740741	5.740741	5.740741	5.740741	5.740741	5.740741	...	5.740741	5.740741	5.740741	5.740741	5.740741	5.740741	5.740741	5.740741	5.740741
4439	5.735294	5.735294	5.735294	5.735294	5.735294	5.735294	5.735294	5.735294	5.735294	5.735294	...	5.735294	5.735294	5.735294	5.735294	5.735294	5.735294	5.735294	5.735294	5.735294
4470	5.896552	5.896552	5.896552	5.896552	5.896552	5.896552	5.896552	5.896552	5.896552	5.896552	...	5.896552	5.896552	5.896552	5.896552	5.896552	5.896552	5.896552	5.896552	6.000000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2324285	5.478261	5.478261	5.478261	5.478261	5.478261	5.478261	5.478261	5.478261	5.478261	5.478261	...	5.478261	5.478261	5.478261	5.478261	5.478261	5.478261	5.478261	5.478261	5.478261
2549237	5.052632	5.052632	5.052632	5.052632	5.052632	5.052632	5.052632	5.052632	5.052632	5.052632	...	5.052632	5.052632	5.052632	5.052632	5.052632	5.052632	5.052632	5.052632	5.000000
1800054678	5.871795	5.871795	5.871795	5.871795	5.871795	5.871795	5.871795	5.871795	5.871795	5.871795	...	5.871795	5.871795	5.871795	5.871795	5.871795	5.871795	5.871795	5.871795	5.871795
1802849661	5.666667	5.666667	5.666667	5.666667	5.666667	5.666667	5.666667	5.666667	5.666667	5.666667	...	5.666667	5.666667	5.666667	5.666667	5.666667	5.666667	5.666667	5.666667	5.666667
2000431901	5.111111	5.111111	5.111111	5.111111	5.111111	5.111111	5.111111	5.111111	5.111111	5.111111	...	5.111111	5.111111	5.111111	5.111111	5.111111	5.111111	5.111111	5.111111	5.111111

988 rows × 1696 columns

tramite l'elbow method troviamo un numero di cluster pari a 3:

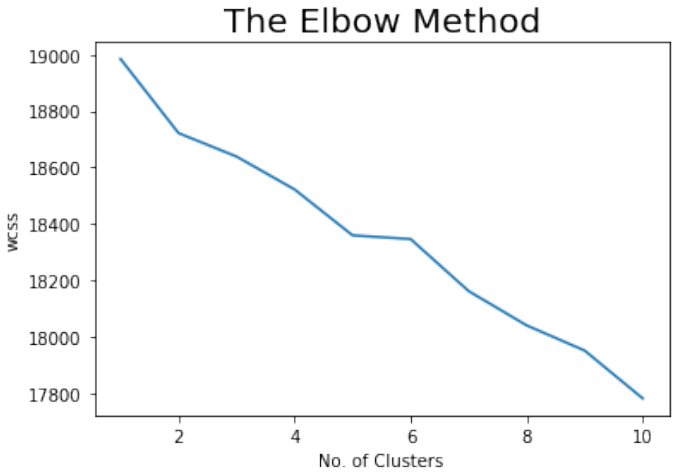


Da come si nota, la curva, a differenza della prima risulta più smooth.
Eseguendo gli altri passi eseguiti prima (passando questa matrice di rating), si ha alla fine come silhoutte 0.43 circa, che è molto maggiore (e quindi migliore) rispetto a quella precendete.
Si eseguono gli stessi passi effettuati prima, però, anziché fillare le celle vuote della matrice con la media generale dei rating e con la media user per user calcoliamo la media recipe per recipe, come risultato avremo:

recipe_id	519	536	607	749	822	1005	1356	1985	2072	2496	...	367414	373346	376786	384800	385071	388397	400137	428004	428116
user_id																				
1535	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	5.625
3288	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	5.625
4291	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	5.625
4439	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	5.625
4470	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	6.000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2324285	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	5.625
2549237	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	5.000
1800054678	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	5.625
1802849661	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	5.625
2000431901	5.916667	5.0	5.625	5.846154	5.6	4.933333	5.736842	5.875	5.352941	5.916667	...	5.0	6.0	5.842105	5.636364	5.857143	5.75	5.8	5.555556	5.625

988 rows × 1696 columns

tramite l'elbow method troviamo un numero di cluster pari a 9:



La curva in questo caso è molto più simile alla prima. Eseguendo gli altri passi eseguiti prima (passando questa matrice di rating), si ha alla fine come silhoutte 0.33 circa, che è maggiore (e quindi migliore) rispetto alla prima, ma risulta peggiore della precedente.

3.1.6 Top-n Items

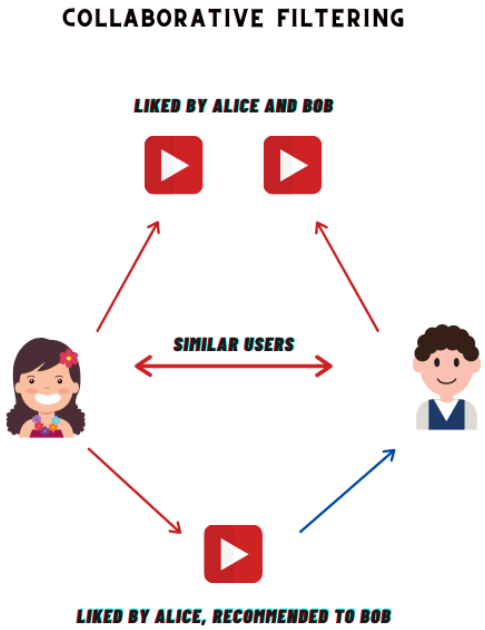
In questo punto si è andato a considerare le migliori n (numero arbitrario) ricette da raccomandare all'utente. Naturalmente si è considerata la matrice dei rating fillata calcolata precedentemente. Banalmente, si è andato a considerare le ricette con un rating più alto, tenendo conto che l'utente non debba aver già recensito una di queste ricette. Quello che si è ottenuto è la seguente matrice:

	0	1	2	3	4	5	6	7	8	9
user_id										
1535	100408	100417	95606	95823	95416	95311	95808	95222	95711	96514
3288	99476	99430	99272	99156	99103	99086	98869	101101	101954	98867
4291	99156	99272	99103	99430	99476	98869	99570	99086	519	99709
4439	100526	100513	100480	100474	100417	95416	100408	100005	99870	95311
4470	98447	102909	102734	102677	102617	98415	76453	102612	103060	102351
...	...	...	...	...	...	...	...	...	...	...
2324285	99272	100005	99870	99843	99709	99570	99476	99430	95808	94211
2549237	94528	99570	100480	100474	100417	100408	92136	100005	99870	99843
1800054678	101041	101816	73754	101803	101751	101657	101391	101104	101101	77554
1802849661	99156	98724	99272	98447	98636	98867	98844	99103	99086	98869
2000431901	94031	99870	99843	99709	99570	92022	99430	99272	99156	92072

988 rows × 10 columns

In questo caso si è scelto un numero $n = 10$

3.2 Collaborative Filtering



Secondo questo approccio, invece, le raccomandazioni sono generate utilizzando informazioni sul comportamento degli utenti, come ad esempio le valutazioni, i "Mi piace" o le recensioni. Il sistema analizza le preferenze degli utenti e confronta le loro valutazioni o recensioni con quelle degli altri utenti. Questo metodo è utile per raccomandare elementi che sono stati graditi da altri utenti con preferenze simili alle proprie. Il Collaborative Filtering può essere implementato sia con un approccio basato sulle similitudini degli utenti sia con un approccio basato sulle similitudini degli elementi.

3.2.1 Algoritmo kNN

Inizialmente, si è effettuata una trasformazione del dataset contente le interazioni tra gli utenti e le ricette con i relativi ratings, in un datframe di surprise con l'apposita funzione **Dataset.load_from_df** con il relativo **reader** il quale imposta il range di valori dei rating, nel nostro caso nel range $[1, 6]$.
Successivamente, si è realizzato uno splitting del dataset principale in due distinti, sequendo il seguente criterio:

- Training set, corrisponde all'80%
- Testing set, corrisponde al 20%

Una volta aver eseguito il fitting (usando il training set) e il testing (usando il testing set) dell'algoritmo, sono state calcolate le due metriche di errore **MSE** e **RMSE**:

```
MSE: 0.6538
RMSE: 0.8086
```

L'MSE (indice che fornisce una misura della precisione dello stimatore, in quanto calcola la media dei quadrati delle differenze fra i possibili valori dello stimatore e il parametro da stimare), mentre l'RMSE (è la radice quadrata della media degli errori quadratici ed è quindi più sensibile ai valori anomali).
La cross-validation è una tecnica utilizzata nel Machine Learning per valutare le prestazioni di un modello. Implica la divisione di un set di dati in più sottoinsiemi, l'addestramento di un modello su un sottoinsieme e la sua valutazione sui restanti sottoinsiemi. Esistono diversi tipi di tecniche di convalida incrociata, tra cui:
K-Fold Cross-Validation: i dati sono divisi in k sottoinsiemi e il modello viene addestrato su $k-1$ sottoinsiemi e valutato sul sottoinsieme rimanente. Questo processo viene ripetuto k volte, con ogni sottoinsieme utilizzato una volta come set di test.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
MSE (testset)	0.6579	0.7031	0.6408	0.6623	0.6710	0.6670	0.0205
RMSE (testset)	0.8111	0.8385	0.8005	0.8138	0.8191	0.8166	0.0125
Fit time	0.10	0.13	0.11	0.11	0.11	0.11	0.01
Test time	0.84	0.94	0.83	0.86	0.84	0.86	0.04

Proseguendo, per ricercare la configurazione degli iperparametri migliori (anche detto **tuning** degli hyperparameters) si è eseguita la **GridSearchCV** dove CV sta per Cross-Validation.
GridSearchCV esegue una ricerca di combinazioni di valori di parametro e addestra un modello per ogni combinazione, quindi restituisce la combinazione con il punteggio migliore misurato da una funzione di punteggio fornita (nel nostro caso, cosine oppure pearson).
In altre parole, GridSearchCV automatizza il processo di addestramento di un modello con diversi iperparametri, valutando le prestazioni del modello per ogni combinazione di iperparametri e selezionando il set di iperparametri con le migliori prestazioni. Ciò può farti risparmiare molto tempo rispetto al test manuale di diverse combinazioni di iperparametri.
Ecco un esempio di come utilizzare GridSearchCV per ottimizzare gli iperparametri di un classificatore dell'albero decisionale:
Infine, l'output ricavato dalla Grid Search è:

```
Best MSE = 0.6142
Best RMSE = 0.7836
Best configuration = {'k': 30,
                      'sim_options': {'name': 'cosine', 'user_based': False}}
```

Una volta trovata la configurazione ottimale dell'algoritmo è necessario addestrarlo sull'intero dataset e prevedere tutti i ratings mancanti per riempire la matrice di ratings. Questa volta per preparare i due dataset train e test set, uitilizzeremo le sequenti funzioni:

- trainset=df_surprise.build_full_trainset(), dove df_surprise è il dataframe surprise costruito precedentemente.
- testset=trainset.build_anti_testset(), dove trainset è il risultato dell'istruzione precedente.

Infine, eseguiamo la fit dell'algoritmo ponendo in input la configurazione ottimale ricavata attraverso la Grid Search e utilizzando il training set. Una volta eseguita, si proseguirà eseguendo il testing dell'algoritmo usando l'apposito testset.
In conclusione, è possibile prevedere i dati mancanti della rating matrix usando la funzione predict(user, recipe), la quale per ogni coppia (user, recipe) (tale che non esiste un rating pregresso assegnato) verrà predetto il valore del rating di un utente ad una ricetta. Se ne ricava dunque la seguente rating matrix fillata:

recipe_id	519	536	607	749	822	1005	1356	1985	2072	2496	...	367414	373346	376786	384800	385071	388397	400137	428004	428116	429576
user_id																					
1535	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
3288	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
4291	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
4439	5	6	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
4470	1	5	5	5	5	5	6	5	5	5	...	5	6	6	5	6	6	5	5	6	6
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2324285	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
2549237	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
1800054678	5	5	5	5	5	5	6	5	5	5	...	5	5	5	5	5	5	5	5	5	5
1802849661	5	5	5	5	5	5	5	5	5	6	...	5	5	5	5	5	5	5	5	5	5
2000431901	5	5	4	5	5	4	4	5	5	4	...	5	5	5	4	5	5	5	5	5	5

988 rows × 1696 columns

3.2.2 Top-n items

Top n items si riferisce alla selezione dei primi n elementi più rilevanti o di maggior successo da una lista di elementi. Questa tecnica viene spesso utilizzata in sistemi di raccomandazione per selezionare i migliori prodotti o servizi da raccomandare agli utenti.
Il calcolo dei Top n items può essere basato su una vasta gamma di criteri, tra cui la popolarità, le recensioni degli utenti, i prezzi o la qualità. Ad esempio, se si desidera raccomandare i migliori film a un utente, i Top n items potrebbero essere basati sulla popolarità dei film, sulle recensioni degli utenti o sulla valutazione degli esperti.
La selezione dei Top n items è un processo importante per i sistemi di raccomandazione, poiché aiuta a filtrare la quantità di informazioni e a fornire raccomandazioni più rilevanti e personalizzate agli utenti. Tuttavia, è importante considerare che le preferenze degli utenti possono variare e che la selezione dei Top n items deve essere eseguita con attenzione per evitare di escludere elementi di valore per alcuni utenti.

	0	1	2	3	4	5	6	7	8	9
user_id										
1535	100408	100417	95606	95823	95416	95311	95808	95222	95711	96514
3288	101104	99570	99476	99430	99272	99156	99103	99086	98869	102073
4291	99156	99272	99103	99430	99476	98869	99570	99086	519	99709
4439	100526	100513	100480	100474	100417	95416	100408	100005	99870	95311
4470	69214	55556	56652	57062	57772	58823	59148	59311	54715	53876
...	...	...	...	...	...	...	...	...	...	...
2324285	99272	100005	99870	99843	99709	99570	99476	99430	95808	94211
2549237	98415	98370	97831	97523	97764	97589	97573	97531	97496	97439
1800054678	90674	18725	57679	40983	137370	27733	35805	9327	8739	71933
1802849661	99156	98724	99272	98447	98636	98867	98844	99103	99086	98869
2000431901	109884	94031	94964	94624	111212	94532	94528	94520	94469	93886

988 rows × 10 columns

Come possiamo notare i top n items del content based e del collaborative filtering sono diversi, questo perché li selezioniamo in modi differenti come spiegato nelle relative sezioni.

3.2.3 Matrix Factorization

In questo punto vengono svolte le medesime operazioni analizzate precedentemente riguardo la realizzazione dell'algoritmo kNN, sostituendolo con l'applicazione di **SVD** (Singular Value Decomposition)
Prima cosa, viene eseguito il fitting e testing dell'algoritmo sul dataframe trasformato di surprise con lo splitting usuale in trainset e testset del tipo 80% e 20%

MSE: 0.6067
RMSE: 0.7789

Successivamente, eseguiamo la k-crossfold validation con 5 splits

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
MSE (testset)	0.6461	0.6100	0.5990	0.6024	0.6085	0.6132	0.0169
RMSE (testset)	0.8038	0.7811	0.7740	0.7761	0.7800	0.7830	0.0107
Fit time	2.61	0.89	0.89	0.87	0.86	1.23	0.69
Test time	0.18	0.08	0.20	0.09	0.08	0.13	0.05

Infine, lanciamo la GridSearchCV per il tuning degli hyperparameters:

```
Best MSE = 0.5993
Best RMSE = 0.7741
Best configuration = {'n_epochs': 30, 'lr_all': 0.002, 'reg_all': 0.4}
```

Utilizziamo questa configurazione ottimale per ricavare la matrice dei rating fillata.
Matrice dei rating fillata usando l'algoritmo **SVD**:

recipe_id	519	536	607	749	822	1005	1356	1985	2072	2496	...	367414	373346	376786	384800	385071	388397	400137	428004	428116	429576
user_id																					
1535	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
3288	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
4291	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
4439	5	6	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
4470	1	5	5	5	5	5	6	5	5	5	...	5	5	5	5	5	5	5	5	6	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2324285	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
2549237	5	5	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5
1800054678	5	5	5	5	5	5	6	5	5	5	...	5	5	5	5	5	5	5	5	5	5
1802849661	5	5	5	5	5	5	5	5	5	6	...	5	5	5	5	5	5	5	5	5	5
2000431901	5	4	5	5	5	5	5	5	5	5	...	5	5	5	5	5	5	5	5	5	5

988 rows × 1696 columns

Notiamo inoltre che la previsione eseguita è leggermente più precisa rispetto a quella utilizzando l'algoritmo kNN.

4 Multi-objective Optimization

4.1 Introduzione

Una volta aver sviluppato la parte base del progetto, proseguiamo con un'analisi più dettagliata riguardante la **Multi-objective Optimization**.
La libreria utilizzata è **Pymoo**, la quale fornisce una vasta gamma di algoritmi di ottimizzazione multi-obiettivo, inclusi algoritmi genetici.
Inoltre, Pymoo fornisce diversi strumenti di visualizzazione per comprendere e analizzare il processo di ottimizzazione. Questi strumenti includono grafici a dispersione, grafici a coordinate parallele, grafici di convergenza e altri diversi.
L'output che si dovrà produrre, utilizzando questi metodi, sarà una matrice di raccomandazione del tipo:

Top-L recommendation matrix

	Item 1	Item 2	...	Item L
User 1	15	20	...	74
User 2	20	11	...	5
			...	
User M	87	23	...	81

In questo caso le funzioni che dovranno essere ottimizzate saranno:

$$\max f_1(s) = \frac{1}{M \cdot L} \sum_{u_i \in U} \sum_{o_j \in S_L(u_i)} r(u_i, o_j)$$

$$\max f_2(s) = \frac{1}{N} \left| \bigcup_{u_i \in U} S_L(u_i) \right|$$

$$\max f_3(s) = \frac{1}{M} \sum_{u_i \in U} \sum_{j \in S_L(u_i)} \frac{N_j}{L}$$

Dove in questo caso:

- f_1 corrisponde alla **Novelty**
- f_2 corrisponde alla **Accuracy**
- f_3 corrisponde alla **Coverage**
- M rappresenta il numero degli utenti

- N rappresenta il numero delle ricette
- L sono il numero di ricette da raccomandare a ciascun utente
- $S_L(u_i)$ è l'i-esima riga della matrice di raccomandazione
- $N_j = log_2(\frac{M}{d_j})$ dove d_j è il grado dell'item j, ovvero, il numero di volte che la ricetta j è stata valutata.

Per attuare queste operazioni si necessita utilizzare esclusivamente la matrice dei rating generata precedentemente (naturalmente per ogni metodo verrà usata la matrice dei rating più performante, ad esempio nel Collaborative Filtering si era affermata essere quella generata con l'algoritmo SVD, la quale rimarca un errore minore).
Inoltre, Pymoo è progettato per essere facilmente estendibile e personalizzabile, questa caratteristica è molto utile per poter ridefinire le funzioni genetiche problem-dependent quali: **Sampling**, **Crossover** e **Mutation**. Infatti, sono state costruite queste funzioni genetiche in modo tale che all'interno di ciascuna fossero implementati i vincoli del problema (ad esempio che ciascun utente debba avere esattamente L ricette consigliate e che tra tutte queste, non ci siano quelle già valutate dal suddetto utente).
I nostri individui, componenti della popolazione, che andremo a generare nel Sampling saranno le singole matrici di raccomandazione, il nostro obiettivo sarà trovare la matrice di raccomandazione che meglio ottimizza le tre funzioni sopra descritte. Per facilitare le operazioni e calcoli da svolgere, ciascuna matrice di raccomandazione non è altro che una matrice di dimensione $M \times N$ (ricordiamo che Pymoo permette di processare esclusivamente array, quindi la matrice sarà vista come array di array) i cui singoli elementi (detti geni) saranno esclusivamente i valori 0 e 1, banalmente:

- 1 se viene raccomandata una certa ricetta a un certo utente
- 0 altrimenti

Naturalmente nel Sampling verrà generata un'intera popolazione di individui, ma come vedremo Pymoo computerà un singolo individuo alla volta (singola matrice di raccomandazione) come un vettore di dimensione $M \times N$, dove ciasuna cella di tale vettore corrisponderà al singolo elemento identificato dalla coppia (user, recipe) della matrice di raccomandazione.
Per iniziare, si è effettuata la traduzione delle tre funzioni obiettivo in codice Python, in particolare, poiché Pymoo per ottimizzare le funzioni utilizza il metodo **minimize**, bisogna trasformare gli operandi delle suddette funzioni, da massimo a minimo (per fare ciò basta moltiplicare per -1, successivamente i risultati ottenuti dovranno essere riportati in positivo).

```
class OptimizationProblem(ElementwiseProblem):
    def __init__(self):
        super().__init__(n_var=M*N, n_obj=3, n_constr=0, xl=np.zeros(M*N), xu=np.ones(M*N))

    def _evaluate(self, x, out, *args, **kwargs):

        x = np.matrix(x.reshape((M, N)))

        f1 = -func1(x)
        f2 = -func2(x)
        f3 = -func3(x)

        out["F"] = [f1, f2, f3]


# create the actual problem to be solved
problem = OptimizationProblem()
```

Successivamente si è andati a strutturare la formulazione del problema appena descritto utilizzando la classe **ElementwiseProblem**, la quale si riferisce a un problema di ottimizzazione in cui la funzione obiettivo e i vincoli sono definiti per operare su ciascun elemento di un vettore soluzione individualmente, piuttosto che sul vettore nel suo insieme. In altre parole, la funzione obiettivo e i vincoli vengono applicati "per elemento" alle variabili decisionali. Sono state inoltre definite le funzioni realizzate, il numero di variabili (nel nostro caso $M \times N$), il numero delle funzioni obiettivo (nel nostro caso 3), i valori di lower e upper bound e infine il numero dei vincoli del problema (abbiamo impostato questo valore a 0, in quanto i vincoli sono stati introdotti direttamente all'interno delle nostre funzioni genetiche, questo per poterle riutilizzare nell'algoritmo MOEA/D, il quale non permette di includere la definizione dei vincoli). Come ultimo passaggio, istanziamo la classe del nostro problema.

4.1.1 NSGA-II

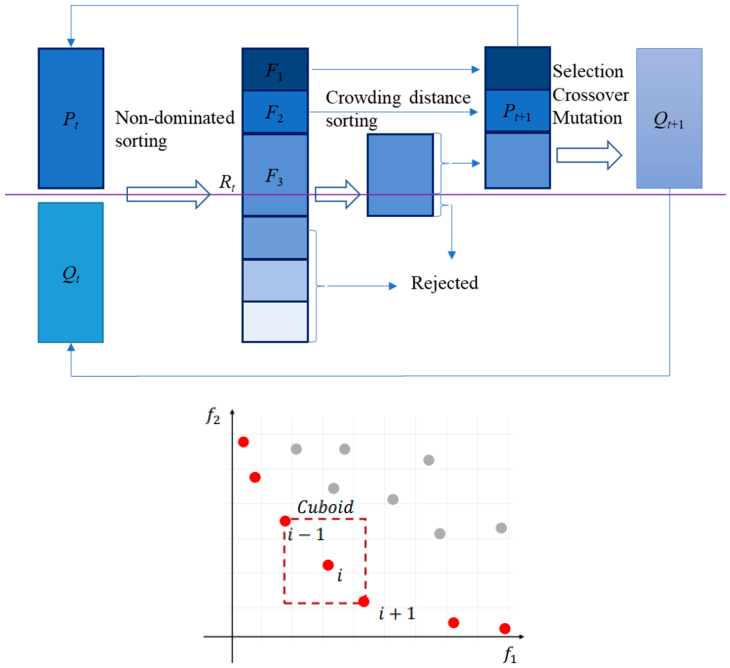


Figure: L'algoritmo NSGA-II (Non-dominated Sorting Genetic Algorithm) segue lo schema generale di un algoritmo genetico in cui gli individui (le soluzioni) vengono selezionati in base alla crowding distance. Utilizza la Manhatten Distance.

Per poter eseguire l'algoritmo occorre specificare anzitutto diversi parametri (tra i quali troviamo anche gli iperparametri)

```
algorithm = NSGA2(pop_size=80,
                  offspring=2,
                  sampling=MySampling(),
                  crossover=BinaryCrossover(),
                  mutation=MyMutation(),
                  eliminate_duplicates=True)

res_nsga = minimize(problem,
                  algorithm,
                  ('n_gen', 50),
                  seed=1,
                  verbose=True)

print("Function value: %s" % res_nsga.F[0])
```

Tra i più noti troviamo sicuramente:

- *pop_size* fa riferimento al numero di individui della popolazione
- *offspring* indica il numero di nuovi individui che verranno generati.
- *sampling*, *crossover* e *mutation* sono le funzioni genetiche che si intende specificare ed utilizzare nell'algoritmo
- *eliminate_duplicates* permette di eliminare i duplicati se posto a *True*

Una volta aver definito questi parametri, occorre infine impostare la chiamata alla funzione *minimize* che permetterà l'esecuzione dell'effettiva ottimizzazione delle funzioni obiettivo. A questa passeremo il *problema* descritto, l'*algoritmo*, il criterio di *terminazione* e con *verbose* indicheremo la nostra volontà o meno nel stampare i risultati di ciascuna generazione effettuata.

4.1.2 MOEA/D

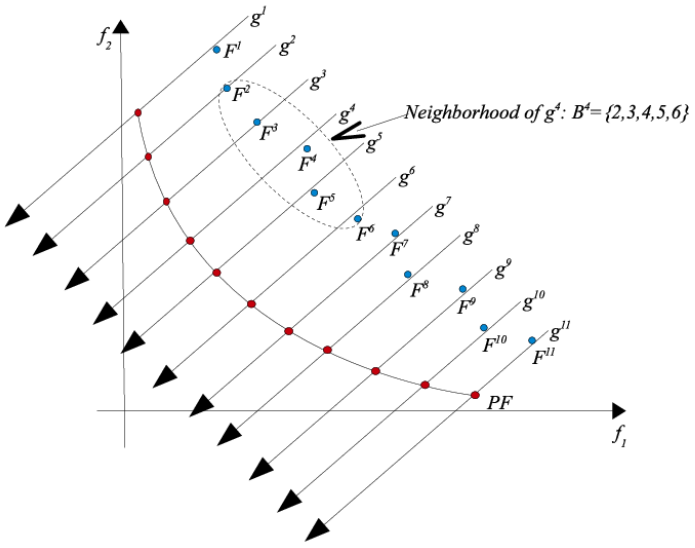


Figure: MOEA/D scompone il problema originale in un numero n di singoli sotto-problemi associati un vettore dei pesi di aggregazione e a diversi punti della Pareto Front.

L'algoritmo MOEA/D si basa sulle indicazioni di riferimento (*ref_dirs* nella nostra implementazione) che devono essere fornite durante l'inizializzazione dell'oggetto algoritmo. Il numero di punti è determinato da un parametro p (il quale corrisponde a *n_partitions*), che indica il numero di spazi tra due punti consecutivi lungo un asse. Dunque, il numero totale di punti sarà determinato da:

$$n = C_p^{M+p-1} \tag{1}$$

Nel nostro caso, avendo scelto come parametri $M = 3$ (corrispondente al numero di funzioni obiettivo) e $n\_partitions = 12$, il numero totale n di punti sarà

$$C_{12}^{3+12-1} = C_{12}^{14} = \frac{14!}{12! * (14-12)!} = \frac{14!}{12! * 2!} = 91 \tag{2}$$

```
# In this case (M=3, p=12) -> n=91
ref_dirs = get_reference_directions("uniform", 3, n_partitions=12)

algorithm1 = MOEAD(
    ref_dirs,
    n_neighbors=5,
    prob_neighbor_mating=0.7,
    sampling=MySampling(),
    crossover=BinaryCrossover(),
    mutation=MyMutation(),
)
```

```
res_moaed = minimize(problem,
                     algorithm1,
                     ('n_gen', 50),
                     seed=1,
                     verbose=True)
```

Come è possibile vedere, sono stati indicati ulteriori due parametri (hyperparameters) quali *n_neighbors* e *prob_neighbor_mating* i quali fanno riferimento al numero di vicini da analizzare e la probabilità di accoppiamento dei vicini, questo perché per risolvere i sottoproblemi in a modo collaborativo, MOEA/D definisce un rapporto di vicinato tra tutti i sotto-problemi basati sulla somiglianza. Notiamo che le funzioni genetiche e il problema sono i medesimi rispetto a quelli utilizzati in precedenza. Una volta impostato l'algoritmo, facciamo partire l'esecuzione. Ora che entrambi gli algoritmi sono stati eseguiti, confrontiamo i dati ricavati

4.2 Content-Based

4.2.1 NSGA-II

n_gen	n_eval	n_nds	eps	indicator
1	80	5	–	–
2	160	20	1.0000000000	ideal
3	240	26	0.5000000000	ideal
4	320	30	0.2222222222	ideal
5	400	31	0.1578947368	ideal

.
.
.

46	3680	80	0.0196078431	ideal
47	3760	80	0.0377358491	ideal
48	3840	80	0.0277777778	ideal
49	3920	80	0.0183486239	ideal
50	4000	80	0.0180180180	ideal

Function value: [−5.1925 −0.20872642 −0.50111278]

4.2.2 MOEA/D

n_gen	n_eval	n_nds	eps	indicator
1	91	6	–	–
2	182	23	1.0000000000	ideal
3	273	34	0.5000000000	ideal
4	364	24	0.7333333333	ideal
5	455	36	0.5609441688	ideal
.				
.				
.				
46	4186	58	0.4000000000	ideal
47	4277	47	0.0625000000	ideal
48	4368	41	0.2500000000	ideal
49	4459	37	0.2000000000	ideal
50	4550	28	0.0625000000	ideal

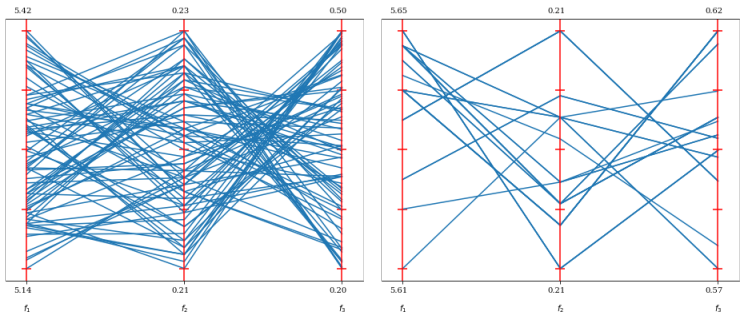


Figure: I grafici di coordinate parallele sono una tecnica essenziale per analizzare come le soluzioni siano distribuite in diversi intervalli rispetto a ciascuna coordinata

Figure 1: NSGA-II

Figure 2: MOEA/D

4.2.3 Hypervolume

L'hypervolume, in poche parole, calcola l'area/volume che è dominato dall'insieme di soluzioni fornito, rispetto a un punto di riferimento. Per calcolare l'hypervolume

è sufficiente fornire un punto di riferimento. Nel nostro caso, si è indicato il punto: $(6, 2, 2)$

HV of true Pareto Front NSGA-II: 2.7105853048487596
HV of true Pareto Front MOEA/D: 0.9935439578226695

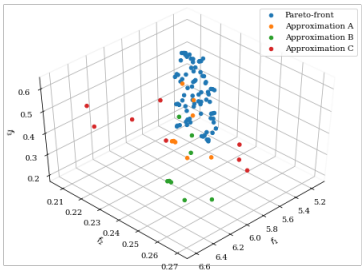
4.2.4 C-Metric

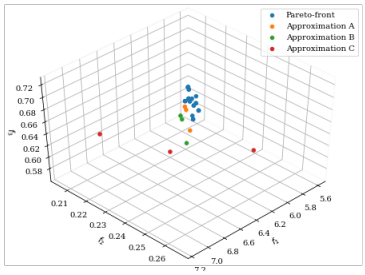
Siano A e B due approssimazioni paretiane. La C-metric mappa la coppia ordinata (A, B) all'intervallo $[0; 1]$ ed è definito da:

$$C(A, B) = \frac{|\{b \in B, \text{ there exists } a \in A \text{ such that } a \preceq b\}|}{|B|}$$

- Se $C(A, B) = 1$, tutti gli elementi di B sono dominati da (o uguali a) gli elementi di A.
- Se $C(A, B) = 0$, tutti gli elementi di B dominano strettamente gli elementi dell'insieme A.
- Entrambi gli ordinamenti devono essere calcolati, poiché $C(A, B)$ non è sempre uguale a $1 - C(A, B)$.

Questa metrica cattura la proporzione di punti in un'approssimazione dell'insieme di Pareto A dominata dall'approssimazione dell'insieme di Pareto B.





NSGA–II
C(A, B) = 1.0
C(B, A) = 0.0

C(A, C) = 0.25
C(C, A) = 0.375

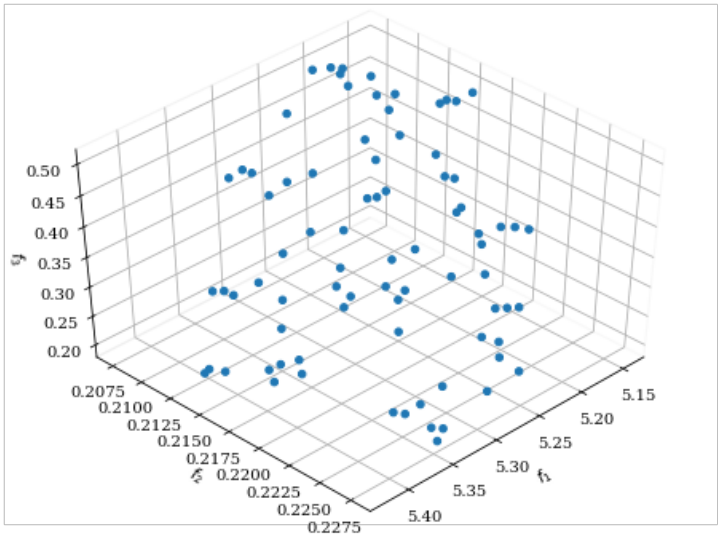
C(B, C) = 0.125
C(C, B) = 1.0

MOEA/D
C(A, B) = 1.0
C(B, A) = 0.0

C(A, C) = 0.66
C(C, A) = 0.0

C(B, C) = 0.0
C(C, B) = 0.0

4.2.5 Frontiere Paretiane



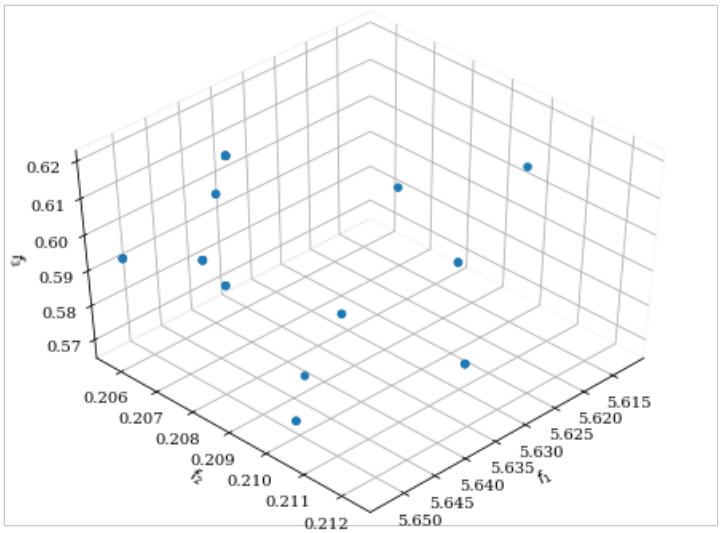


Figure 1: NSGA-II
Figure 2: MOEA/D

4.3 Collaborative Filtering

I passaggi introdotti e applicati con il metodo Content-Based, risultano essere i medesimi anche per il metodo Collaborative Filtering, l'unico elemento distinto è la matrice dei rating fillata, difatti, qui si utilizzerà la matrice dei rating generata attraverso l'algoritmo SVD (in quanto la predizione è più precisa). Ci limiteremo esclusivamente a riportare i risultati ottenuti dunque.

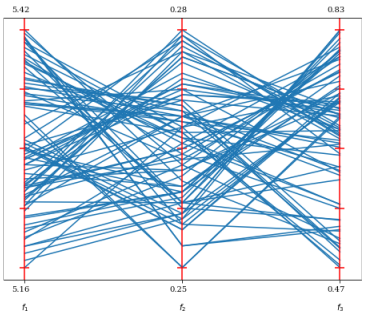
4.3.1 NSGA-II

n_gen	n_eval	n_nds	eps	indicator
1	80	8	–	–
2	160	24	1.0000000000	ideal
3	240	25	0.4285714286	ideal
4	320	22	0.3000000000	ideal
5	400	27	0.2307692308	ideal
		.		
		.		
		.		
46	3680	80	0.0232558140	nadir
47	3760	80	0.0157480315	ideal
48	3840	80	0.0325203252	nadir
49	3920	80	0.0381679389	ideal
50	4000	80	0.0223880597	ideal

Function value: [−5.164 −0.25471698 −0.83239495]

4.3.2 MOEA/D

n_gen	n_eval	n_nds	eps	indicator
1	91	11	–	–
2	182	26	0.5909090909	ideal
3	273	22	0.7894736842	ideal
4	364	19	0.4681953988	ideal
5	455	46	0.5235201333	ideal
		.		
		.		
		.		
46	4186	34	0.2857142857	ideal
47	4277	54	0.1470588235	ideal
48	4368	51	0.1000000000	ideal
49	4459	46	0.1176470588	ideal
50	4550	37	0.0122881803	ideal



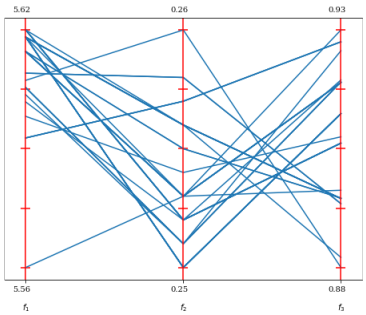
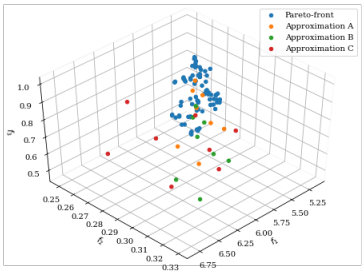


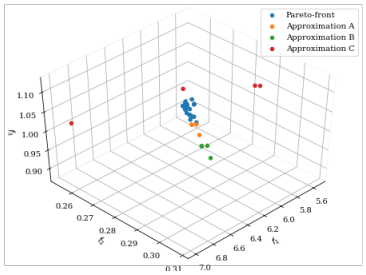
Figure 1: NSGA-II
Figure 2: MOEA/D

4.3.3 Hypervolume

HV of true Pareto Front NSGA-II: 2.173750465550889
HV of true Pareto Front MOEA/D: 0.8592809940923787

4.3.4 C-Metric





NSGA
C(A, B) = 1.0
C(B, A) = 0.0

C(A, C) = 0.625
C(C, A) = 0.25

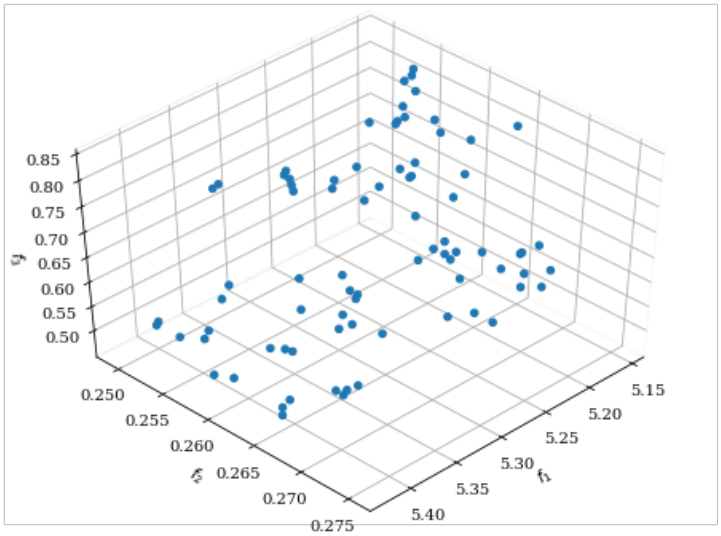
C(B, C) = 0.125
C(C, B) = 1.0

MOEA/D
C(A, B) = 1.0
C(B, A) = 0.0

C(A, C) = 0.25
C(C, A) = 0.0

C(B, C) = 0.0
C(C, B) = 1.0

4.3.5 Frontiere Paretiane



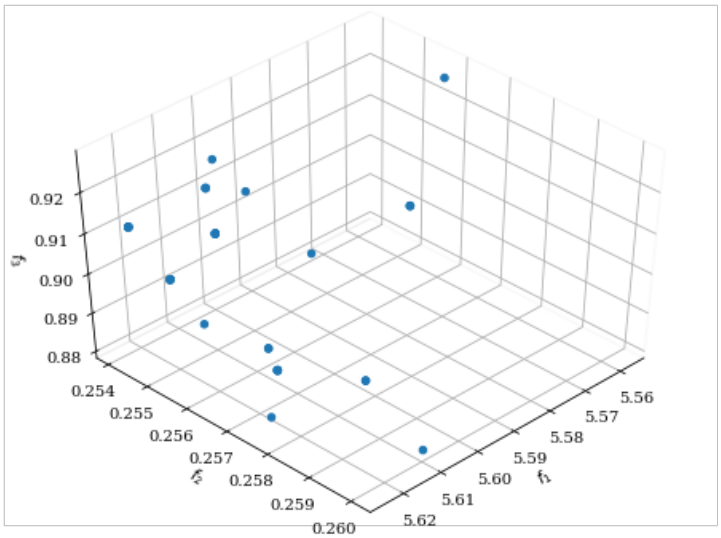


Figure 1: NSGA-II
Figure 2: MOEA/D

5 Conclusioni

Con questo report ci si è focalizzati sullo sviluppo di un Recommender System basato sul sito Food.com, dal lavoro svolto si sono riscontrati i seguenti risultati.

Come primo punto è stata svolta un analisi dei dati, da questa abbiamo ricavato in primo luogo che la frequenza delle ricette (in base al numero di steps) segue un andamento normale, mentre la frequenza delle ricette (in base al numero di ingredienti) segue invece un andamento sinusoidale. In secondo luogo si è riscontrata una correlazione molto positiva (pari a 0.98) degli zuccheri nei carboidrati.

Successivamente è stato eseguito l'algoritmo kNN sia sulla parte content based, sia sulla parte collaborative filtering (per quest'ultimo, si è eseguito anche l'algoritmo SVD per attuare la tecnica Matrix Factorization), effettuando una comparazione tra di esse si nota come i valori degli errori tra Content Based e Matrix Factorization siano molto vicini tra loro e risultano inferiori al kNN basic della parte collaborative filtering.

Per l'esecuzione dell'algoritmo k-Means sono state usate tre stategie differenti, matrice dei rating fillata i valori non presenti con media generale dei rating, media dei rating calcolata user per user e media dei rating calcolata recipe per recipe. Tra i valori della Silhouette calcolate, risulta migliore quella relativa alla matrice fillata con la media dei rating calcolata user per user.

Dopodiché sono stati estrapolati i top n items da consigliare ad ogni user, come si può notare gli items consigliati sono differenti per uno stesso user, questo avviene in quanto nel content based si consiglia in base al contenuto dell'item, mentre nel collaborative filtering si consiglia in base alla somiglianza tra diversi users.

Nell'ultima parte, invece, si è effettuata una Multi-Objective Optimization di tre funzioni (Accuracy, Novelty e Coverage). Mediante l'utilizzo degli algoritmi NSGA-II e MOEA/D sono state generate le Pareto Front, le quali, sono state confrontate ed analizzate utilizzando le due diverse metriche (quali Hypervolume e C-Metric), tra le due metodologie Content-Based e Collaborative Filtering. Tra i diversi risultati ricavati, si può notare che il valore più elevato dell'Hypervolume è dato dall'algoritmo NSGA-II utilizzando il metodo Content-Based, il suddetto valore è: 2.71