



UNIVERSITÀ DEGLI STUDI DI PERUGIA
Dipartimento di Matematica e Informatica



CORSO DI LAUREA MAGISTRALE IN INFORMATICA

Tesi di Laurea

Analisi dell'efficacia del data clustering sull'addestramento di recommender system

Laureando
Andrea Ricci

Relatore/i
Prof. Valentina Poggioni
Dott. Alina Elena Baia

Anno Accademico 2019/2020

Indice

1 Introduzione	1
1.1 Fidelity Salus	1
1.2 Struttura della tesi	2
2 Recommender System	3
2.1 Il problema della raccomandazione	3
2.1.1 Raccolta delle valutazioni	5
2.1.2 Tipologie di Recommender System	6
Content-based Filtering (CBF)	6
Collaborative Filtering (CF)	7
Approcci ibridi	8
Vantaggi e limitazioni	8
2.2 Gli algoritmi	10
2.2.1 Matrix Factorization	11
2.2.2 Funk MF	12
2.2.3 Alternating Least Squares	12
2.2.4 Bayesian Personalized Ranking	14
Criterio di ottimizzazione BPR-OPT	16
2.2.5 Metriche	18
3 Clustering	21
3.1 Algoritmi di clustering	21
3.1.1 DBSCAN	22
3.1.2 Agglomerative Clustering	24
3.1.3 K-means	26
3.2 Riduzione della dimensionalità	29
3.2.1 PCA	29
3.2.2 t-SNE	32
4 Implementazione del Recommender System	35
4.1 Dataset e preprocessing dei dati	35
4.2 Costruzione della matrice degli acquisti	37
4.3 Allenamento del modello	39
4.3.1 Allenamento con ALS	39
4.3.2 Allenamento con BPR	39
4.4 Erogazione delle raccomandazioni	40

4.5	Testing	41
5	Applicazione degli algoritmi di clustering	43
5.1	Struttura dei dati	43
5.2	Selezione e preprocessing degli attributi	44
5.3	Clustering con DBSCAN	49
5.4	Clustering con K-means	53
5.5	Clustering con metodo Agglomerative	55
5.6	Raccolta dei risultati	57
5.6.1	Caso di studio 1: clustering con attributi <i>Tutor, ConsortiumCode, Zone, StartupYear, Revenue, SalesMean</i>	57
5.6.2	Caso di studio 2: clustering con attributi <i>Tutor, ConsortiumCode, Zone, Revenue</i>	66
5.6.3	Caso di studio 3: clustering con attributi <i>Tutor, Zone, StartupYear, SalesMean</i>	75
6	Conclusioni	85
	Bibliografia	87

Capitolo 1

Introduzione

L'evoluzione tecnologica caratterizzante questi ultimi decenni ha portato ad un eccesso di informazioni sulla rete. Gli utenti, oberati da una tale mole di dati, utilizzano le tecniche più disparate atte a vagliare quali di queste informazioni passano essere di loro interesse. I Recommender System si inseriscono in questo quadro, andando ad automatizzare alcune di queste strategie con l'obiettivo di catalogare un vasto insieme di oggetti per poi fornire agli utenti raccomandazioni personali e di alta qualità.

Questo lavoro di tesi si inserisce in un progetto più grande in collaborazione con Fidelity Salus, azienda perugina che fornisce servizi di marketing a farmacie. Il primo obiettivo è stato la costruzione di un Recommender System dove il ruolo degli utenti viene assunto dalle farmacie stesse. Il fine ultimo del sistema è quello di consigliare al punto vendita quali possano essere i prodotti più indicati da inserire in magazzino studiando il comportamento di esercizi simili. Il secondo obiettivo è rappresentato da un tentativo di clustering delle farmacie, condotto secondo varie metodologie, atto a verificare se le performance del suddetto Recommender System possano essere migliorate.

1.1 Fidelity Salus

Fidelity Salus è un'azienda perugina parte del gruppo Pharmaservice S.r.l. che si occupa dell'erogazione di servizi di marketing a farmacie sull'intero territorio nazionale.

L'attività di maggior interesse in questo frangente è quella della gestione delle fidelity card. Le farmacie che fanno parte del circuito Fidelity Salus possono distribuire ai propri clienti delle carte fedeltà con le quali accumulare punti a fronte di acquisti riconducibili al "non farmaco". Il cliente, oltre a godere di sconti riservati ai possessori della carta, può convertire i punti con oggetti di vario genere contenuti nel catalogo premi scaricabile dal sito internet di Fidelity Salus. Egli è in questo modo invogliato ad utilizzare la sua fidelity card ed è quindi possibile utilizzare il codice identificativo della stessa per procedere alla profilazione del consumatore. Per profilazione si intende l'insieme delle attività di raccolta ed elaborazione dei dati inerenti

agli utenti del servizio, al fine di suddividerli in gruppi a seconda del loro comportamento. Tra le varie informazioni ottenute tramite la profilazione sono ad esempio presenti il codice identificativo della carta utilizzata per l'acquisto, informazioni sul prodotto venduto e sulla sua quantità, la farmacia dove la transazione è avvenuta ecc. I dati così ottenuti possono essere utilizzati per fornire servizi personalizzati o campagne di comunicazione mirate per insiemi specifici di clienti.

1.2 Struttura della tesi

Nel capitolo 2 viene presentata una panoramica generale sui Recommender System, incluse le strategie di feedback degli utenti, gli algoritmi utilizzati e le metriche considerate. Nel capitolo 3 viene approfondito il tema del clustering, con particolare attenzione alle principali metodologie e algoritmi e la problematica della riduzione della dimensionalità, fondamentale per la visualizzazione 2-dimensionale di dati n-dimensional. Nel capitolo 4 viene presentata l'implementazione del Recommender System sviluppata per questo lavoro di tesi, con la specifica delle librerie utilizzate, e lo studio della scelta dell'algoritmo migliore da utilizzare. Nel capitolo 5 vengono condotti dei tentativi di clustering delle farmacie atti a verificare se queste tecniche possano portare ad un miglioramento delle performance del Recommender System implementato nel capitolo 4. Nel capitolo 6 si discutono le conclusioni del progetto e le possibilità di sviluppi futuri.

Capitolo 2

Recommender System

In questo capitolo viene fornita una panoramica sul problema della raccomandazione e sulle principali categorie, evidenziandone pregi e limitazioni. Nella sezione 2.1 viene formalizzato il problema della raccomandazione e ne vengono introdotti diversi approcci, definiti in base a quali sono le informazioni che utilizzano. Vengono inoltre presentati i diversi criteri di ottenimento dei rating dagli utenti, che vengono poi utilizzati per l'allenamento del Recommender System. Nella sezione 2.2 vengono presentati i principali algoritmi che vengono utilizzati nell'ambito dei Recommender System, ponendo particolare attenzione all'algoritmo Bayesian Personalized Ranking utilizzato estensivamente in questo progetto di tesi, e le metriche di valutazione delle performance utilizzate.

2.1 Il problema della raccomandazione

L'obiettivo di un Recommender System (RS) è quello di assistere l'utente nella ricerca di informazioni utili all'interno di enormi banche dati, come possono essere cataloghi musicali (iTunes, Spotify), video (Netflix, Now TV) o di e-commerce (Amazon, eBay). Un Recommender System filtra questa grande mole di oggetti e suggerisce quelli che l'utente potrebbe apprezzare tenendo in considerazione le sue preferenze, i suoi interessi e le sue priorità.

Molte tra le più grandi e rinomate aziende che operano online fanno uso di questi sistemi. Un esempio celebre è Amazon, che inserisce all'interno della pagina descrittiva di ogni prodotto una sezione nella quale vengono suggeriti oggetti che altri utenti hanno acquistato insieme a quello esaminato. Un ulteriore esempio che ha contribuito alla diffusione dello studio del problema stesso della raccomandazione è quello di Netflix, che nel 2007 con il suo "the Netflix prize" [2] offrì una ricompensa di un milione di dollari al team che avesse presentato un algoritmo di raccomandazione migliore di quello in uso all'epoca. In questo caso l'obiettivo del Recommender System è quello di consigliare all'utente film, serie TV o documentari che sono stati apprezzati da altri con preferenze simili.

L'approccio più antico al problema della raccomandazione è quello del collaborative filtering, termine coniato da Goldberg ed i suoi colleghi in un lavoro del 1992 [7] atto allo sviluppo di un sistema di filtraggio automatico delle mail. Fu in questi anni

che prese piede la formulazione più comune del problema della raccomandazione, ovvero quella che si basa sul concetto di rating. L'obiettivo è quello di assegnare un punteggio (il rating) agli oggetti che un utente non ha mai visto, che sta ad indicare quanto questi potrebbero essere di suo gradimento. Tipicamente per procedere alla stima di questi valori vengono utilizzati i rating che egli stesso ha assegnato ad oggetti differenti, con l'ausilio in alcuni casi di altre informazioni derivate dagli oggetti stessi. Una volta che questi rating sono stati assegnati è possibile suggerire all'utente quelli con punteggio stimato più alto.

Formalmente, sia U l'insieme di tutti gli utenti ed I l'insieme di tutti i possibili oggetti che possono essere raccomandati. Lo spazio I dei possibili oggetti può essere molto ampio, così come quello degli utenti: basti pensare ai numeri di iTunes, il celebre negozio online di musica di Apple, che a Giugno 2020 vanta 72 milioni di utenti ed un catalogo di oltre 600 milioni di brani [16]. Sia r una funzione che misura l'utilità dell'oggetto i per l'utente u , ovvero

$$r : U \times I \rightarrow R \quad (2.1)$$

dove R è un insieme totalmente ordinato. A questo punto per ogni utente $u \in U$ si sceglie l'oggetto $i' \in I$ che massimizza l'utilità. Formalmente

$$\forall u \in U, \quad i'_u = \arg \max_{i \in I} r(u, i) \quad (2.2)$$

Nei Recommender System l'utilità di un oggetto viene tipicamente rappresentata dal suo rating [1], che indica quanto un particolare utente abbia apprezzato un particolare oggetto. Ad ogni modo l'utilità può essere una funzione arbitraria, e può in particolare essere indicata esplicitamente dall'utente o calcolata implicitamente dal sistema stesso.

Ogni elemento dello spazio U viene definito con un profilo che indica varie caratteristiche dell'utente, come età, genere o reddito. Nel caso più semplice il profilo può contenere anche un solo elemento, ovvero l'ID dell'utente. In maniera speculare anche ogni elemento in I è definito con il suo insieme di caratteristiche, ad esempio nel caso di un brano musicale è possibile specificarne l'autore, il genere o l'album di appartenenza.

Il problema fondamentale dei recommender system risiede nel fatto che solitamente l'utilità r non è definita sull'intero spazio $U \times I$, ma solo su un suo sottoinsieme. Di conseguenza questa deve essere ricavata laddove non espressa.

Segue la tabella 2.1 contenente una porzione di una matrice dei rating user-item, che rappresenta le interazioni tra utenti e prodotti, per un servizio di streaming on-demand di serie TV. In questo esempio i rating vengono selezionati utilizzando una scala da 1 a 5. Le celle vuote sono quelle dove l'utente non ha inserito una valutazione, in quanto presumibilmente non ha visto la serie TV. Il compito del recommender system è dunque quello di prevedere i valori di questi rating ed offrire raccomandazioni appropriate di conseguenza.

	Angel	Breaking Bad	Castle	Dr.House
Alice	4		3	
Bruno			3	
Carlo	1	5		5
Dante			4	
Ettore		2		2

TABELLA 2.1: Porzione di matrice dei rating per un RS di un servizio di streaming

Una volta che i valori mancanti sono stati stimati vengono fornite all’utente le raccomandazioni vere e proprie selezionando i rating più alti tra tutti quelli stimati, come da formula 2.2.

2.1.1 Raccolta delle valutazioni

La raccolta delle valutazioni può avvenire in maniera implicita o esplicita.

1. **Feedback esplicito:** questa tipologia di feedback consiste nell’assegnazione di rating che l’utente assegna esplicitamente agli oggetti. Tipicamente vengono utilizzati valori numerici compresi in un certo intervallo, come ad esempio punteggi discreti da 1 a 5 (IMDb) o valutazioni binarie (Youtube). Nonostante il feedback esplicito offra un maggiori informazioni rispetto a quello implicito per dedurre la preferenza dell’utente verso un determinato oggetto, è comunque più difficile da ottenere. L’utente infatti si trova di fronte ad un ulteriore passaggio facoltativo, che viene spesso tralasciato anche solo per semplice indolenza. Per questa ragione l’erogatore del servizio deve rendere l’assegnazione delle valutazioni la più semplice e rapida possibile, poiché in caso contrario l’utente potrebbe risultare riluttante nel valutare gli oggetti.
2. **Feedback implicito:** Dal momento che la valutazione esplicita può essere difficile da ottenere, si è recentemente posta particolare attenzione allo studio di sistemi che possano sfruttare il comportamento dell’utente per derivare le sue preferenze e che dunque non richiedano un’azione da parte sua. I comportamenti dell’utente, come i click effettuati, le pagine web visitate, i prodotti acquistati (Amazon) o i brani riprodotti (Spotify), possono essere mappati come una valutazione positiva implicitamente assegnata verso particolari oggetti. Queste forme di comportamento dell’utente prendono il nome di feedback implicito e vengono tipicamente rappresentate utilizzando un valore binario o un valore discreto, che può magari descrivere quante volte l’utente ha usufruito dell’oggetto. Nonostante la quantità di valutazioni implicite sia solitamente maggiore di quelle esplicite, in quanto raccolte automaticamente, occorre tenere presente che le prime offrono un grado di sicurezza della preferenza dell’utente minore rispetto alle ultime.

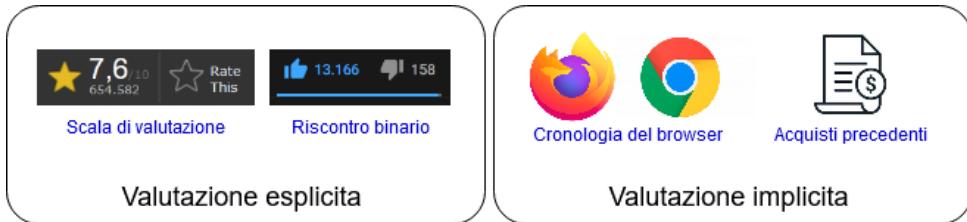


FIGURA 2.1: Valutazione esplicita ed implicita

2.1.2 Tipologie di Recommender System

In base alla modalità con cui vengono realizzate le raccomandazioni i Recommender System vengono comunemente suddivisi in tre categorie.

Content-based Filtering (CBF)

I sistemi che utilizzano questo metodo raccomandano oggetti simili ad altri che l'utente ha già apprezzato in passato. Ogni oggetto può essere rappresentato da un insieme di caratteristiche, indicate nel proseguo della trattazione con il termine inglese features. Ad esempio un film può essere caratterizzato dall'autore, il genere, la durata e così via. Le tecniche di tipo Content-Based sfruttano queste informazioni per raccomandare oggetti con caratteristiche simili a quelli di cui l'utente ha già beneficiato.

Si assuma ad esempio che un utente di Netflix abbia assegnato le seguenti valutazioni ad alcuni film:

Film	Genere	Valutazione
Die Hard	Azione	Positiva
Mad Max	Azione	Positiva
Come d'incanto	Bambini	Negativa

TABELLA 2.2: Esempio di valutazione in un sistema CBF

A partire da queste tre valutazioni è possibile costruire il vettore dell'utente. Risulta evidente come questo apprezzi particolarmente i film d'azione, mentre un po' meno quelli destinati ad un pubblico più giovane; per questa ragione, utilizzando una scala di valutazione da -10 a 10, si assegna 9 al genere "Azione" e -6 a quello "Bambini". Si assegna inoltre una valutazione neutra (0) ai film d'animazione, in quanto mai valutati dall'utente. Il vettore dell'utente sarà quindi [9,0,-6] seguendo l'ordine [Azione, Animazione, Bambini]. Si ipotizzi ora che il RS debba raccomandare dei film a questo utente e prenda in considerazione "Toy Story", con item vector [0,1,1] in quanto ricade sia nella categoria "Animazione" che in quella "Bambini", e Star Wars, con item vector [1,0,0] in quanto film d'azione. A questo punto non resta che eseguire il prodotto scalare tra item vector ed user vector per ottenere la valutazione:

$$\text{Toy Story: } [9, 0, -6] \cdot [0, 1, 1] = -6$$

$$\text{Star Wars: } [9, 0, -6] \cdot [1, 0, 0] = 9$$

Il Recommender System consiglierà all’utente l’opera con valutazione maggiore, ovvero Star Wars. Tipicamente un RS di questo tipo calcola il prodotto dell’user vector e di tutti gli item vector e consiglia gli N dei quali è stato calcolato il valore maggiore.

Collaborative Filtering (CF)

I sistemi di questo tipo sfruttano le interazioni tra utenti ed oggetti per effettuare le raccomandazioni. Utilizzando gli oggetti valutati precedentemente dall’utente obiettivo della raccomandazione e da altri utenti dalle preferenze simili, il sistema prevede i rating degli oggetti non ancora osservati. Si tratta della categoria approfondita in questo progetto di tesi. In questa tipologia di Recommender System non viene presa in considerazione alcuna caratteristica intrinseca degli oggetti, come autore, anno di produzione ecc. Formalmente, l’utilità $r(u, i)$ dell’oggetto i per l’utente u viene stimata utilizzando le utilità $r(u_j, i)$ assegnate all’oggetto i da quegli utenti $u_j \in U$ che risultano essere simili all’utente u . Ad esempio, per consigliare all’utente c dei brani da ascoltare in un servizio di musica in streaming, il recommender system può cercare utenti che ascoltano gli stessi generi per poi consigliare a c ciò che questi utenti hanno gradito.

Nel 1998 venne presentata da Breese et al. [3] una distinzione dei sistemi di collaborative filtering in due categorie, ovvero memory-based e model-based.

- **Memory-based Collaborative Filtering**

Gli algoritmi di tipo memory-based sono essenzialmente euristiche che effettuano previsioni sui rating utilizzando l’intero insieme degli oggetti precedentemente valutati da tutti gli utenti. Formalmente, il valore del rating $r_{u,i}$ per l’utente u e l’oggetto i viene calcolato come aggregazione dei rating di altri utenti, solitamente gli N più simili ad u , per l’oggetto i :

$$r_{u,i} = \text{aggr}_{u' \in \hat{U}} r_{u',i} \quad (2.3)$$

dove \hat{U} rappresenta l’insieme degli N utenti considerati più simili ad u che hanno valutato i . Seguono due esempi di funzioni di aggregazione, che possono essere tra le più disparate.

$$r_{u,i} = \frac{1}{N} \sum_{u' \in \hat{U}} r_{u',i} \quad (2.4)$$

$$r_{u,i} = k \sum_{u' \in \hat{U}} \text{sim}(u, u') \times r_{u',i} \quad (2.5)$$

dove k rappresenta un fattore di normalizzazione, solitamente scelto come

$$k = 1 / \left(\sum_{u' \in \hat{U}} |\text{sim}(u, u')| \right) \quad (2.6)$$

La formula 2.4 rappresenta il caso più banale che utilizza la semplice media aritmetica, mentre 2.5 l'approccio più comune che utilizza la media pesata dei rating. La similarità tra gli utenti u ed u' , $\text{sim}(u, u')$, è una misura di distanza che viene utilizzata come peso: più u ed u' sono simili, più il rating $r_{u',i}$ avrà peso nella stima di $r_{u,i}$. Ad ogni modo, recommender system differenti possono utilizzare funzioni di similarità differenti, a patto che il risultato venga normalizzato utilizzando k .

Questa varietà di algoritmi è molto semplice da utilizzare, in quanto non richiede l'allenamento di un modello o l'applicazione di complessi metodi di ottimizzazione. D'altro canto le sue performance calano drasticamente in presenza di dati sparsi, il che limita fortemente la scalabilità di questo approccio verso i problemi del mondo reale, così come quello trattato in questo lavoro di tesi.

- **Model-based Collaborative Filtering**

Gli algoritmi di tipo model-based utilizzano l'insieme dei rating per l'allenamento di un modello, che viene poi utilizzato per effettuare le previsioni. La grande differenza tra approcci model-based e memory-based è che nei primi il calcolo dell'utilità si basa su un modello costruito utilizzando tecniche di machine learning sui dati, mentre nei secondi vengono applicate regole euristiche costruite specificatamente per il problema in questione.

Questa varietà di algoritmi è particolarmente indicata per database ad alta sparsità ed è fortemente scalabile. Nel caso di un database molto grande il risparmio in termini di tempo è notevole, in quanto questo viene completamente esplorato una sola volta per l'allenamento del modello mentre viene utilizzato esclusivamente quest'ultimo ogni volta che si richiede una raccomandazione.

I sistemi di tipo Collaborative Filtering più comuni utilizzano la tecnica della Matrix Factorization, esposta nel capitolo 2.2.1.

Approcci ibridi

I RS di tipo ibrido combinano informazioni ottenute da varie fonti, sia basate sulle caratteristiche dell'oggetto (Content-based filtering) che sull'interazione utente-oggetto (Collaborative filtering).

Vantaggi e limitazioni

Ogni tipologia di Recommender System ha determinati punti di forza e limitazioni, pertanto la scelta riguardo quale utilizzare dipende dal contesto in cui si opera. I principali vantaggi dei Recommender System di tipo Content-based Filtering sono:

- **Indipendenza dell'utente:** I sistemi di tipo CBF, al momento del calcolo della raccomandazione, prendono in considerazione gli item vector di tutti i prodotti

del catalogo ed un solo user vector, ovvero quello corrispondente al cliente destinatario della raccomandazione. Per questa ragione sistemi di questo tipo restituiscono ottimi risultati anche in presenza di un numero esiguo di utenti nel database, in quanto essi sono tutti indipendenti fra loro.

- **Trasparenza:** Nei sistemi di tipo CF può risultare complesso risalire alle ragioni che hanno portato alla scelta di un determinato suggerimento, in quanto questa dipende dalle abitudini di altre persone, sconosciute all'utente dato. Questo problema non sussiste nei sistemi CBF, in quanto le features degli oggetti sono facilmente consultabili e di immediata comprensione.
- **Nuovi prodotti:** È possibile suggerire nuovi oggetti basandosi semplicemente sul loro insieme di features, anche prima che essi siano valutati da un numero sostanziale di utenti.

Le principali limitazioni dei Recommender System di tipo Content-based Filtering sono:

- **Numero di features disponibili insufficiente:** Se gli oggetti non posseggono un numero sufficiente di features, o il loro insieme non descrive il prodotto in maniera propriamente dettagliata, le raccomandazioni potrebbero non risultare precise.
- **Iperspecializzazione:** Un RS di tipo CBF tende a raccomandare oggetti non particolarmente inaspettati o sorprendenti, poiché per come è costruito il RS questi hanno caratteristiche simili a quelli già utilizzati dall'utente. In altre parole la novelty, ovvero una misura che indica quanto vengono raccomandati oggetti dei quali l'utente non era a conoscenza, risulta essere spesso insufficiente.

Una conseguenza di questo problema può essere il cosiddetto "portfolio effect", dove il Recommender System suggerisce ad un dato utente oggetti molto simili tra loro, fornendo una lista di raccomandazioni non abbastanza diversificata ed interessante.

- **Nuovo utente:** Le raccomandazioni possono risultare poco accurate nel caso in cui non sia disponibile un numero sufficiente di informazioni per la costruzione del profilo dell'utente.

Per quanto riguarda i Recommender System di tipo Collaborative Filtering, i principali vantaggi sono:

- **Raccomandazioni ad alta serendipity:** Il sistema può aiutare gli utenti a scoprire nuovi interessi. Infatti, mentre da un certo punto di vista un oggetto può non aver nulla a che vedere con un dato utente (e.g. non vi sono features in comune con prodotti graditi in precedenza), questo può essere comunque raccomandato se utenti simili l'hanno trovato interessante.

- **Informazione richiesta minima:** Il corretto funzionamento di un sistema di tipo CF necessita della sola matrice dei rating. In particolare non è necessario conoscere le features degli oggetti come nei sistemi di tipo Content-based.

Le principali limitazioni dei Recommender System di tipo Collaborative Filtering sono:

- **Nuovo oggetto:** In questa tipologia di sistema la raccomandazione per una coppia (utente, oggetto) consiste nel prodotto scalare dei rispettivi embedding. Di conseguenza, se un oggetto non era presente durante la fase di training del modello, il sistema non può crearne un item vector.

In generale questo problema, conosciuto con il nome di "Cold Start problem", può verificarsi anche se un oggetto era presente in fase di test ma non è stato consumato da un numero sufficiente di utenti. Dunque oggetti popolari godono di un vantaggio notevole ed è più probabile che vengano inclusi nelle raccomandazioni.

- **Nuovo utente:** Così come negli approcci content-based, finché un utente non ha fornito abbastanza valutazioni il Recommender System può non essere in grado di fornire raccomandazioni di qualità di prodotti sconosciuti ed interessanti.
- **Il problema della "grey sheep":** Dal momento che il Recommender System utilizza le preferenze di persone simili ad un dato utente per suggerire nuovi prodotti, persone dai gusti particolari o di nicchia possono incorrere in grandi difficoltà nel trovare utenti simili, e dunque nel ricevere raccomandazioni di oggetti interessanti.
- **Mancanza di informazioni aggiuntive:** Tipicamente un sistema di tipo CF lavora unicamente con gli ID di utenti e prodotti. Di conseguenza informazioni aggiuntive come l'età di un utente o l'autore di un prodotto non possono essere utilizzate per migliorare la qualità delle raccomandazioni. Ciò nonostante esistono approcci ibridi che permettono di utilizzare insieme entrambe le metodologie, mantenendo uno scheletro di CF ma facendo uso delle features degli oggetti per arricchire le raccomandazioni.

2.2 Gli algoritmi

All'interno di questo lavoro di tesi sono stati utilizzati Recommender System appartenenti alla categoria del Collaborative Filtering, poiché questi garantiscono solitamente raccomandazioni migliori e più diversificate. Entrambi gli algoritmi studiati si basano su approccio model-based che prende il nome di Matrix Factorization.

2.2.1 Matrix Factorization

La Matrix factorization (MF), o fattorizzazione di matrice, è una classe di algoritmi di tipo collaborative filtering. Il suo obiettivo è quello di descrivere sia utenti che oggetti con l'ausilio di un numero arbitrario di fattori latenti derivati dal dataset. L'algoritmo opera poi decomponendo la matrice di interazioni utente-oggetto nel prodotto di due matrici rettangolari dalla dimensionalità inferiore.

Considerando un numero arbitrario di fattori latenti f , gli utenti e gli oggetti vengono mappati come vettori $v \in \mathbb{R}^f$. Da questi, dati m utenti ed n oggetti, è possibile costruire le matrici $U \in \mathbb{R}^{m \times f}$ e $V \in \mathbb{R}^{n \times f}$ rispettivamente per utenti ed oggetti. All'interno di queste ultime un vettore riga v_i prende il nome di vettore latente e corrisponde all'utente i nel caso di U ed all'oggetto i nel caso di V , mentre le colonne corrispondono ai fattori latenti. Il prodotto scalare tra un vettore latente utente u_i ed un vettore latente oggetto v_j restituisce la preferenza stimata associata all'oggetto j dall'utente i . Il prodotto

$$A \approx UV^T$$

rappresenta dunque la preferenza predetta dal sistema per ogni utente ed ogni oggetto. L'obiettivo è quello di garantire che la matrice A rappresenti una buona approssimazione della matrice dei rating, ossia la matrice sparsa $R \in \mathbb{R}^{m \times n}$ i cui elementi sono i rating $r_{i,j}$ che l'utente i ha assegnato all'oggetto j .

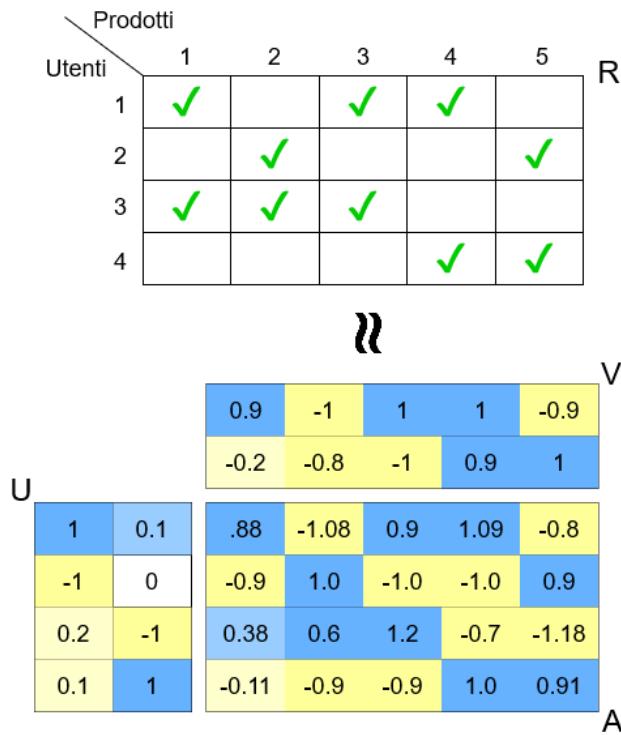


FIGURA 2.2: Esempio di matrix factorization

Questa famiglia di metodi divenne nota durante il concorso "the Netflix prize" [2], grazie alla sua efficacia dimostrata da Simon Funk in un post sul suo blog datato

2006 [6], dove condivise la sua ricerca con la comunità scientifica.

Nei capitoli seguenti viene presentato quest'ultimo algoritmo, base di quelli moderni, insieme ai due che sono stati utilizzati nel progetto di tesi, ovvero Alternating Least Squares e Bayesian Personalized Ranking.

2.2.2 Funk MF

L'algoritmo primordiale Funk MF fattorizza la matrice dei rating come prodotto di due matrici di dimensionalità inferiore, dove la prima possiede una riga per ogni utente e la seconda una colonna per ogni prodotto. Gli elementi che fanno parte di una riga o colonna associata ad uno specifico utente o prodotto prendono il nome di fattori latenti: mentre i cosiddetti fattori osservati derivano da campionamenti sul campo, i fattori latenti vengono dedotti utilizzando un modello matematico a partire dai fattori osservati.

Siano x il numero di utenti, y quello degli oggetti e z quello dei fattori latenti. Formalmente la matrice viene fattorizzata come $R = UI$, dove $R \in \mathbb{R}^{x \times y}$ è la matrice dei rating utente-oggetti, $U \in \mathbb{R}^{x \times z}$ contiene i fattori latenti degli utenti e $I \in \mathbb{R}^{z \times y}$ contiene i fattori latenti dei prodotti. Il rating predetto che l'utente u assegnerà all'oggetto i viene calcolato come

$$r_{u,i} = \sum_{f=0}^z U_{u,f} I_{f,i} \quad (2.7)$$

Il numero di fattori latenti gioca un ruolo cruciale in questo contesto: al suo aumentare vengono restituite raccomandazioni più personalizzate, e dunque di qualità superiore. Se questo raggiunge però un numero troppo elevato si inizia ad incorrere in problemi di overfitting, e la qualità delle raccomandazioni ne risente. E' dunque cruciale trovare un giusto compromesso per il numero di fattori latenti, e per farlo vengono spesso impiegate apposite tecniche di cross-validation.

2.2.3 Alternating Least Squares

Alternating Least Squares (ALS) [9] è un metodo di Matrix Factorization che può essere utilizzato per predire i rating mancanti in una matrice utente-oggetto. Data una matrice $R \in \mathbb{R}^{m \times n}$ con m oggetti ed n utenti il metodo ALS viene utilizzato per fattorizzare la matrice in due matrici U ed I in modo tale che il loro prodotto approssimi R il più possibile:

$$R \approx U \times I \quad (2.8)$$

In primo luogo viene introdotto un insieme di variabili p_{ui} , che indicano la preferenza dell'utente u verso l'oggetto i . Questi valori vengono calcolati binarizzando i valori di r_{ui} , ovvero le osservazioni reali che descrivono le interazioni tra utenti ed

oggetti:

$$p_{ui} = \begin{cases} 1 & \text{se } r_{ui} > 0 \\ 0 & \text{se } r_{ui} = 0 \end{cases} \quad (2.9)$$

In altre parole, se un utente u ha consumato un oggetto i ($r_{ui} > 0$), abbiamo un'indicazione che u ha apprezzato i ($p_{ui} = 1$). In caso contrario, se u non ha mai consumato i , allora u non ha alcuna preferenza nei confronti di i ($p_{ui} = 0$).

La realtà non è però così binaria, possono infatti esistere una moltitudine di ragioni per cui un utente consumi un prodotto nonostante non lo apprezzi particolarmente (e.g. l'acquisto di un oggetto per un regalo ad un amico o la visione di un programma televisivo dovuta al semplice zapping). Allo stesso modo vi possono essere diverse motivazioni per cui un utente non consumi un prodotto, non necessariamente legate al basso gradimento (e.g. il non essere a conoscenza dell'esistenza del prodotto o l'esistenza di una barriera di natura economica per la sua fruizione). Per questa ragione vengono considerati diversi livelli di confidenza anche tra gli oggetti dei quali un certo utente è fruitore: al crescere di r_{ui} si ha un'indicazione più concreta che l'utente abbia effettivamente gradito il prodotto, e dunque un livello di confidenza più alto. Viene così introdotto un insieme di variabili c_{ui} che misurano proprio la confidenza del sistema nell'osservazione di p_{ui} . Il valore di c_{ui} scelto dagli autori è di

$$c_{ui} = 1 + \alpha r_{ui} \quad (2.10)$$

In questo modo si pone un certo valore minimo di confidenza per ogni coppia utente-oggetto, ma al crescere del numero di informazioni che suggeriscono un'opinione positiva la confidenza del sistema in $p_{ui} = 1$ cresce di conseguenza. La velocità di crescita viene dettata dalla costante α , ed è stato sperimentalmente osservato come la scelta di $\alpha = 40$ porti ad ottimi risultati.

L'obiettivo dell'algoritmo è quello di trovare un vettore $x_u \in \mathbb{R}^f$ per ogni utente u ed un vettore $y_i \in \mathbb{R}^f$ per ogni oggetto i tali che le preferenze dell'utente siano rappresentabili come loro prodotto: $p_{ui} = x_u^T y_i$. Questi vettori prendono il nome di user-factors ed item-factors. Il fine di questa procedura è quello di mappare utenti e prodotti in uno spazio di fattori latenti comune, in modo da poterli confrontare direttamente. I fattori vengono calcolati minimizzando la seguente funzione di costo:

$$\min_{x_u, y_i} \sum_{u,i} c_{ui} (p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right) \quad (2.11)$$

Il termine $\lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$ è necessario per la regolarizzazione del modello, in modo da evitare overfitting dei dati di allenamento. Il valore di λ è strettamente legato ai dati e viene determinato caso per caso tramite tecniche di cross-validation. Si noti che quando uno dei due vettori user-factors e item-factors vengono fissati la funzione di costo diventa quadratica, e dunque se ne può calcolare il minimo globale. Questa osservazione è alla base del processo di ottimizzazione Alternating Least

Squares, dove ad ogni iterazione si fissano gli item-factors e si ricalcolano gli user-factors per poi fissare gli user-factors e ricalcolare gli item-factors. Ogni iterazione può terminare in due modi: la funzione di costo rimane invariata o decresce.

Il primo passo è dunque il ricalcolo degli user-factors. Si assume che tutti gli item-factor vengano raccolti in una matrice Y di dimensione $n \times f$. Per ogni utente u definiamo la matrice diagonale C^u di dimensione $n \times n$ dove $C_{ii}^u = c_{ui}$ ed il vettore $p(u) \in \mathbb{R}^n$ che contiene tutte le preferenze di u , ovvero i valori p_{ui} . Tramite un processo di differenziazione è possibile trovare un'espressione per il calcolo di x_u che minimizzi la funzione di costo 2.11, ovvero

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u) \quad (2.12)$$

È possibile dimostrare come il costo di questa operazione sia lineare nella dimensione dell'input. Al ricalcolo degli user-factor segue quello degli item-factors, che si svolge in maniera analoga calcolando y_i sempre minimizzando 2.11

$$y_i = (X^T C^i Y + \lambda I)^{-1} X^T C^i p(i) \quad (2.13)$$

dove X è la matrice contenente gli user-vector, C^i la matrice diagonale dove $C_{uu}^i = c_{ui}$ e $p(i)$ il vettore contenente le preferenze verso l'oggetto i . Il ricalcolo dei fattori viene eseguito finché questi non si stabilizzano, richiedendo tipicamente una decina di iterazioni.

Al termine di questo processo è possibile raccomandare all'utente u gli N prodotti con rating maggiore, ovvero il più alto valore di

$$\hat{p}_{ui} = x_u^T y_i \quad (2.14)$$

dove \hat{p}_{ui} rappresenta la preferenza prevista dal sistema dell'utente u per il prodotto i .

2.2.4 Bayesian Personalized Ranking

Bayesian Personalized Ranking (BPR) [13] è un criterio di ottimizzazione basato sul feedback隐式. L'assunzione alla base è che se un utente u ha interagito con un oggetto i esso lo preferisce a tutti gli altri oggetti con cui non ha interagito. Non esiste invece alcun ordine di preferenza tra oggetti che hanno entrambi avuto un'interazione con l'utente, così come tra oggetti che non ne hanno avute.

Sia U l'insieme di tutti gli utenti ed I l'insieme di tutti gli oggetti. L'obiettivo dell'algoritmo è quello di trovare una classifica personalizzata $>_u \subset I^2$ per ogni utente $u \in U$ ed ogni coppia di oggetti $(i, j) \in I^2$ che soddisfi le proprietà di un ordine totale, ovvero:

- $\forall i, j \in I : i \neq j \implies i >_u j \vee j >_u i$ (totalità)
- $\forall i \in I \implies i >_u i$ (riflessività)

- $\forall i, j \in I : i >_u j \wedge j >_u i \implies i = j$ (antisimmetria)
- $\forall i, j, k \in I : i >_u j \wedge j >_u k \implies i >_u k$ (transitività)

Per convenienza, vengono definiti anche

$$I_u^+ = \{i \in I : (u, i) \in S\}$$

$$U_i^+ = \{u \in U : (u, i) \in S\}$$

dove I_u^+ rappresenta l'insieme degli oggetti con i quali l'utente u ha interagito ed U_i^+ l'insieme degli utenti che hanno interagito con l'oggetto i .

In un sistema a feedback implicito sono disponibili unicamente dati inerenti le osservazioni positive, dove dunque l'utente ha interagito con l'oggetto. Per questa ragione non è possibile distinguere se le coppie utente-prodotto non osservate siano dovute ad un feedback negativo (l'utente ha valutato l'acquisto e non ne è interessato) o conseguenza di valori mancanti (l'utente potrebbe comprarlo in futuro, magari non è ancora venuto a conoscenza della sua esistenza). Gli algoritmi tradizionali, come ad esempio ALS, ignorano questa differenza costruendo una matrice a valori binari dove 1 indica una coppia utente-prodotto positiva e 0 una non positiva.

The diagram illustrates the transformation of implicit feedback data. On the left, a sparse matrix represents user interactions with four products (i1 to i4). Rows represent users (u1 to u5) and columns represent products (i1 to i4). A '+' symbol indicates an interaction, while a '?' symbol indicates no interaction. An arrow points to the right, leading to a binary matrix used for training. This training matrix has the same structure but uses binary values (0 or 1) to represent interactions. A '+' symbol in the original matrix becomes a 1 in the training matrix, and a '?' symbol becomes a 0.

		Prodotto			
		i ₁	i ₂	i ₃	i ₄
Utente	u ₁	?	+	+	?
	u ₂	+	?	?	+
	u ₃	+	+	?	?
	u ₄	?	?	+	+
	u ₅	?	?	+	?
	→				
		Prodotto			
		i ₁	i ₂	i ₃	i ₄
Utente	u ₁	0	+	+	0
	u ₂	+	0	0	+
	u ₃	+	+	0	0
	u ₄	0	0	+	+
	u ₅	0	0	+	0

FIGURA 2.3: Trasformazione dei dati in un sistema tradizionale

Nella parte sinistra di figura 2.3 vengono mostrati i dati delle osservazioni implicite, contenenti unicamente le interazioni positive indicate con un simbolo +. Nella parte destra vengono mostrati i dati utilizzati per l'allenamento del modello, con un simbolo + che indica l'avvenuta interazione tra utente u ed oggetto i ed un simbolo – la mancata interazione tra gli stessi. L'obiettivo di questi algoritmi "classici" è quello di prevedere uno score \hat{x}_{ui} per gli oggetti che rispecchi la preferenza dell'utente u verso l'oggetto i . Una volta ottenuto il punteggio di tutti gli oggetti viene costruito il loro ranking inserendoli nella classifica in ordine decrescente in base al punteggio ottenuto.

In BPR viene utilizzato un approccio differente che utilizza come dati di allenamento le coppie di oggetti e procede al ranking di queste coppie piuttosto che assegnare un punteggio al singolo oggetto. Partendo dai dati osservati l'obiettivo è quello di costruire sezioni di $>_u$ per ogni utente. Per farlo, se un oggetto i è stato acquistato da un utente u è lecito ritenere che l'utente preferisca questo prodotto rispetto a tutti quelli non acquistati. Per quanto riguarda coppie di oggetti entrambi

acquistati o non acquistati non è invece possibile ricavare alcuna preferenza. Formalmente, si costruiscono i dati di allenamento $D_S : U \times I \times I$ a partire dai dati osservati S come:

$$D_S := \{(u, i, j) | i \in I_u^+ \wedge j \in I \setminus I_u^+\}$$

dove una tripla $(u, i, j) \in D_S$ indica che l'utente u preferisce l'oggetto i all'oggetto j .

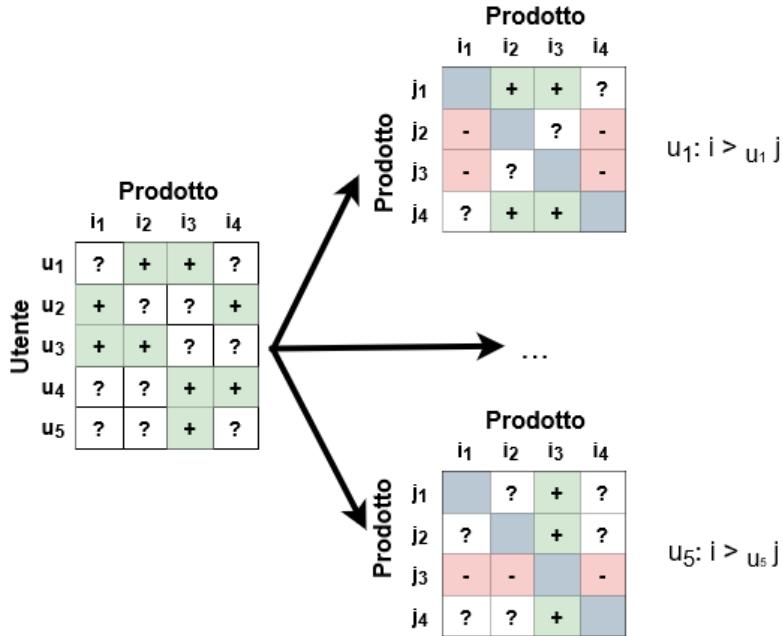


FIGURA 2.4: Trasformazione dei dati in un sistema BPR

Nella parte sinistra della figura 2.4 vengono mostrati i dati delle osservazioni implicite, contenenti unicamente le interazioni positive indicate con un simbolo +. Nella parte destra viene mostrata la preferenza a coppie di oggetti per ogni utente. Un + indica che un utente preferisce l'oggetto i all'oggetto j , un – che al contrario preferisce j ad i .

Questo approccio ha due principali vantaggi:

1. Il training set D_S ed il test set sono disgiunti, in quanto i valori tra due oggetti non osservati saranno proprio quelli che dovranno essere classificati in futuro.
2. I dati di allenamento sono creati con l'obiettivo reale di effettuare un ranking, e non adattati per questo scopo successivamente. In altre parole il sottoinsieme D_S di $>_u$ viene utilizzato come training set.

Criterio di ottimizzazione BPR-OPT

Al fine di calcolare il personalized ranking per tutti gli oggetti $i \in I$ la formulazione Bayesiana massimizza la seguente probabilità a posteriori:

$$p(\Theta | >_u) \propto p(>_u | \Theta)p(\Theta)$$

dove Θ rappresenta il vettore parametrico del modello scelto (ad es. matrix factorization). Si assume che tutti gli utenti si comportino in maniera indipendente e che l'ordinamento di ogni coppia (i, j) per uno specifico utente sia indipendente da quello di ogni altra coppia. Segue dunque che $p(>_u | \Theta)$ può essere riscritto come prodotto di densità singole e combinato per tutti gli utenti $u \in U$:

$$\prod_{u \in U} p(>_u | \Theta) = \prod_{(u,i,j) \in D_S} p(i >_u j | \Theta) \quad (2.15)$$

La probabilità che un utente preferisca realmente l'oggetto i all'oggetto j è definita come:

$$p(i >_u j | \Theta) := \sigma(\hat{x}_{uij}(\Theta)) \quad (2.16)$$

dove σ rappresenta la sigmoide logistica

$$\sigma(x) := \frac{1}{1 + e^{-x}} \quad (2.17)$$

ed il suo input $\hat{x}_{uij}(\Theta)$ è una funzione a valori reali che rappresenta la relazione tra u , i e j e che viene calcolata in base al modello che si intende utilizzare (ad es. matrix factorization). Infine, il criterio di ottimizzazione può essere definito come:

$$\begin{aligned} BPR - OPT &:= \ln p(\Theta | >_u) \\ &= \ln p(>_u | \Theta) p(\Theta) \\ &= \ln \prod_{(u,i,j) \in D_S} \sigma(\hat{x}_{uij}) p(\Theta) \\ &= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\ &= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2 \end{aligned} \quad (2.18)$$

dove λ_Θ sono valori di regolarizzazione specifici del modello utilizzato.

Questo problema di ottimizzazione può essere risolto tramite discesa del gradiente su Θ . Dal momento che può risultare computazionalmente costoso considerare tutte le triple $(u, i, j) \in D_S$, viene utilizzato un approccio di discesa stocastica del gradiente che sceglie casualmente alcune triple da D_S in maniera uniforme.

Ponendo

$$\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj} \quad (2.19)$$

è possibile calcolare i due termini utilizzando le tecniche esistenti per la previsione dei rating. In [13] vengono presentati come possibili modelli sottostanti l'algoritmo BPR sia la Matrix Factorization che il K-nearest neighbors. A differenza dei modelli stand-alone di MF e kNN, che minimizzano l'errore della previsione dei rating, BPR-OPT assicura che il ranking degli oggetti sia ottimizzato.

2.2.5 Metriche

Le metriche utilizzate per la valutazione delle performance nei Recommender System a feedback implicito sono tipicamente prese in prestito dal campo dell'Information Retrieval. In questo lavoro di tesi sono state utilizzate le misure seguenti:

- **Precision:** rappresenta la percentuale di oggetti raccomandati che sono rilevanti. Per definirla formalmente sono necessari alcuni concetti fondamentali:

- Vero positivo: L'algoritmo ha raccomandato un oggetto rilevante
- Vero negativo: L'algoritmo non ha raccomandato un oggetto irrilevante
- Falso positivo: L'algoritmo ha raccomandato un oggetto irrilevante
- Falso negativo: L'algoritmo non ha raccomandato un oggetto rilevante

La precision viene definita come il rapporto tra il numero di oggetti rilevanti raccomandati ed il numero totale di oggetti raccomandati. Formalmente,

$$\text{Precision} = \frac{\text{Oggetti rilevanti raccomandati}}{\text{Oggetti raccomandati}} = \frac{\text{Veri positivi}}{\text{Veri positivi} + \text{Falsi positivi}}$$

La precision viene solitamente valutata nella forma di *Precision@K*, che calcola la metrica sui primi K oggetti raccomandati, con K che rappresenta un valore predeterminato (nel nostro caso fissato a 5). I valori della precision variano in un intervallo da zero, dove nessuno degli oggetti raccomandati è rilevante, ad uno, dove tutti gli oggetti raccomandati sono rilevanti.

- **Mean Average Precision:** Per definire la Mean Average Precision (mAP) è necessario presentare prima l'Average Precision (AP). L'AP di una lista di oggetti viene calcolata considerando la *Precision@K* di ogni elemento rilevante della lista. Sia m il numero di oggetti rilevanti della lista di raccomandazione, si ha:

$$AP = \frac{\sum_{k=1}^n \text{Precision}@k \cdot Rel(k)}{m} \quad (2.20)$$

dove $Rel(k) = 1$ se l'elemento in posizione k è rilevante, 0 altrimenti.

La mAP viene semplicemente calcolata tramite la media delle Average Precision delle N liste di raccomandazione prodotte. Formalmente,

$$mAP = \frac{\sum_{i=1}^N AP(i)}{N} \quad (2.21)$$

dove N è il numero di liste di raccomandazione prodotte. Così come la *Precision@K*, anche i valori di *MAP* variano nell'intervallo $[0, 1]$.

- **Normalized Discounted Cumulative Gain:** In presenza di liste di oggetti dotati di un certo punteggio che indica la loro rilevanza, solitamente rappresentata da un valore continuo, gli elementi con un punteggio elevato sono considerati di maggior valore per l'utente. Il Normalized Discounted Cumulative Gain (nDCG) è una misura che tiene in considerazione questa importante caratteristica.

Una prima metrica pensata per valutare liste di oggetti dotati di un certo punteggio fu il Cumulative Gain, che potrebbe essere considerato come un antenato del moderno nDCG. Il CG calcolato per una posizione p si definisce come

$$CG_p = \sum_{i=1}^p rel_i \quad (2.22)$$

dove rel_i indica la rilevanza dell'oggetto in posizione p . Il grande limite di questa misura è il fatto che non tiene in considerazione la collocazione degli oggetti all'interno della lista dei risultati, in quanto un eventuale cambio di posizione non altera il valore calcolato dal GC. Per questa ragione l'nDCG viene spesso preferito al CG.

Prima di definire l'nDCG è necessario presentare il concetto di Discounted Cumulative Gain (DCG). La premessa del DCG è che oggetti particolarmente rilevanti che appaiono in basso nella lista dei risultati debbano essere penalizzati riducendo il loro punteggio di rilevanza in maniera logaritmicamente proporzionale alla posizione del risultato. Il DCG calcolato alla posizione p viene definito come

$$DCG_p = \sum_{i=1}^p \frac{Rel_i}{\log_2(i+1)} \quad (2.23)$$

Infine, il valore dell'nDCG calcolato alla posizione P è dato da

$$nDCG_p = \frac{DCG_p}{IDCG_p} \quad (2.24)$$

dove l>IDCG (Ideal Discounted Cumulative Gain) viene calcolato valutando tutti i possibili ordinamenti degli oggetti rilevanti e considerando il massimo valore possibile per il DCG. Formalmente,

$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{Rel_i}{\log_2(i+1)} \quad (2.25)$$

dove $|REL_p|$ rappresenta il numero di oggetti rilevanti fino alla posizione p .

Anche i valori di quest'ultima misura variano nell'intervallo $[0, 1]$.

Capitolo 3

Clustering

In questo capitolo viene presentata una panoramica sul clustering e sulle tecniche di riduzione della dimensionalità utilizzate per la visualizzazione. Nella sezione 3.1 viene presentata una descrizione del clustering, incluso un approfondimento sugli algoritmi utilizzati in questo lavoro di tesi, ovvero DBSCAN, K-means e Agglomerative Clustering. Nella sezione 3.2 vengono trattate le principali tecniche di dimensione della dimensionalità, in particolare PCA e t-SNE.

3.1 Algoritmi di clustering

Il clustering è una tecnica che consiste nel raggruppamento di punti di dati basato sulla loro similarità, in modo che punti appartenenti allo stesso gruppo siano più simili tra loro rispetto ad ogni altro punto contenuto in altri gruppi. Gli insiemi risultanti da questo compito prendono il nome di cluster.

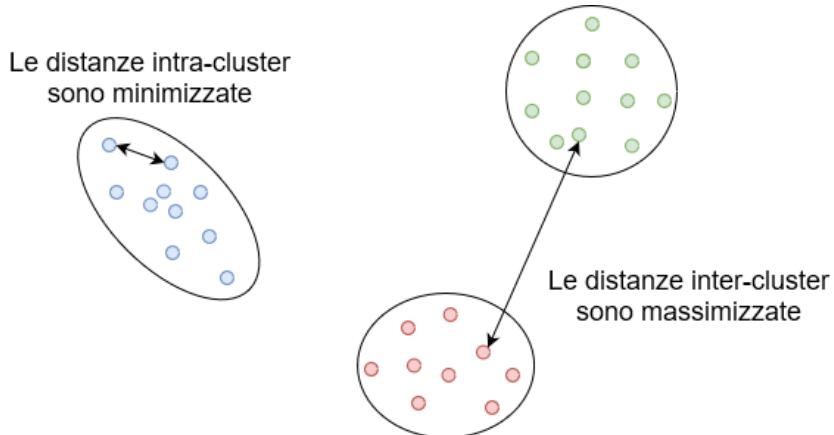


FIGURA 3.1: Schema del concetto di clustering

La differenza principale tra il clustering ed altre tecniche di machine learning come la classificazione risiede nel tipo di dati utilizzati nell’allenamento del modello. Gli elementi dell’insieme di dati utilizzati per i modelli di classificazione infatti sono dotati di un’etichetta che descrive la loro classe di appartenenza. È per questa ragione che questo approccio prende il nome di "apprendimento supervisionato", in quanto si basa su etichette fornite inizialmente al sistema. Al contrario il clustering

non necessita di alcuna etichetta e basa il suo funzionamento sulle caratteristiche degli elementi da raggruppare. Questo approccio prende il nome di "apprendimento non supervisionato".

Gli algoritmi di clustering possono essere separati in tre categorie [15]:

- Algoritmi partizionali: suddividono i dati in un numero determinato di cluster basandosi sulla similarità o la distanza tra gli oggetti.
- Algoritmi gerarchici: mirano alla costruzione di una gerarchia di cluster.
- Algoritmi density-based: per formare i cluster ricercano regioni ad alta densità di oggetti separate da regioni a bassa densità.

3.1.1 DBSCAN

L'algoritmo DBSCAN è un algoritmo di clustering density-based tra i più utilizzati ed il più citato in letteratura, presentato nel 1996 [4].

DBSCAN stima la densità attorno a ciascun punto contando il numero di punti in un suo intorno di raggio eps ed applicando una soglia chiamata minPts per suddividere i punti in core points, border points o noise points. L'algoritmo prosegue riunendo i core points in un unico cluster se questi sono density-reachable, ovvero se esiste una sequenza di core points dove ognuno di essi ricade all'interno dell'intorno "eps" del successivo. I border points sono infine assegnati ai cluster. I due parametri richiesti "eps" e "minPts" vengono specificati dall'utente.

Dato un insieme di punti nello spazio, DBSCAN li classifica come core point, density-reachable o outlier seguendo le regole seguenti:

- Un punto p è un core point se almeno minPts punti sono contenuti nel suo intorno di raggio eps , incluso p stesso. Questi punti vengono definiti come direttamente raggiungibili da p .
- Un punto q è direttamente raggiungibile da p se q si trova entro la distanza eps dal punto p con p core point.
- Un punto q è raggiungibile da p se esiste un percorso p_1, \dots, p_n , con $p_1 = p$ e $p_n = q$, dove ogni $p_i + 1$ è direttamente raggiungibile da p_i . Tutti i punti nel percorso devono essere core points ad eccezione di q , che può esserlo o meno.
- Tutti i punti non raggiungibili da altri sono definiti outlier.

Se un punto p è un core point allora questo forma un cluster con tutti gli altri punti da esso raggiungibili, che essi siano core points o meno. Ogni cluster deve dunque contenere almeno un core point, mentre i punti non core che appartengono ai cluster si definiscono border point in quanto non possono essere utilizzati per raggiungere altri punti.

Si consideri l'esempio in figura 3.2 nel quale $\text{minPts} = 4$. A e gli altri punti rossi sono core points, in quanto contengono almeno quattro punti nel loro intorno di raggio

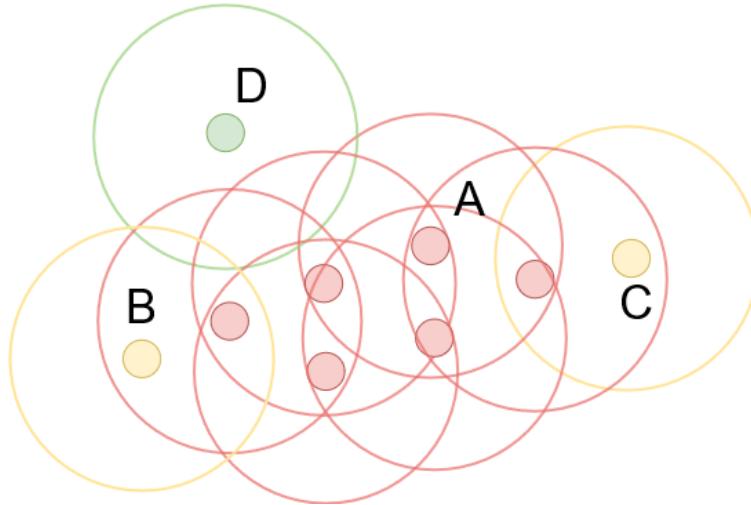


FIGURA 3.2: Esempio di assegnazione delle etichette ai punti in DBSCAN

eps . Dal momento che i punti rossi sono tutti raggiungibili tra loro essi fanno tutti parte dello stesso cluster. B e C non sono core points, ma essendo raggiungibili da A appartengono a loro volta al cluster. Il punto D è invece un punto di rumore, poiché non è un core point e non è direttamente raggiungibile.

La relazione di raggiungibilità non è simmetrica, poiché un punto non può essere raggiungibile da un non-core point a prescindere dalla distanza. In altre parole un punto non-core può essere raggiungibile, ma da esso non si può raggiungere alcun punto (ad esempio, tornando alla figura 3.2, B è raggiungibile da A ma A non è raggiungibile da B). È pertanto necessaria un'ulteriore nozione di connessione, stavolta simmetrica, per definire formalmente i cluster trovati da DBSCAN. Due punti p e q sono "density-connected" se esiste un punto r tale che sia p che q siano raggiungibili da r . In conclusione un cluster soddisfa le due proprietà seguenti:

- Tutti i punti del cluster sono mutualmente density-connected
- Se un punto è density-raggiungibile da un qualunque punto del cluster, allora è anch'esso parte del cluster.

Lo pseudocodice dell'algoritmo è il seguente:

Algorithm: DBSCAN

Calcola gli eps -vicini di ogni punto ed identifica i core point con più di minPts vicini;

Assegna core point connessi allo stesso cluster;

foreach non-core point **do**

Assegna al cluster di un core point eps -vicino se possibile;
Assegna al rumore altrimenti;

end

DBSCAN è particolarmente indicato per l'identificazione di cluster di forme e dimensioni differenti. L'algoritmo è inoltre resistente al rumore e capace di gestire

eventuali outliers del dataset. D'altro canto non funziona bene con cluster di densità variabile e dataset con dimensionalità particolarmente elevata.

3.1.2 Agglomerative Clustering

Ci sono due tipologie principali di clustering gerarchico:

- **Agglomerative Clustering:** inizialmente ogni oggetto forma un singoletto, ovvero un cluster di un solo elemento. L'algoritmo unisce iterativamente le coppie di cluster più vicine tra loro finché non rimane un unico cluster contenente tutti gli elementi.
- **Divisive Clustering:** inizialmente tutti gli oggetti sono contenuti in un unico cluster. L'algoritmo divide iterativamente ogni cluster in due in modo che questi siano il più distanziati possibile.

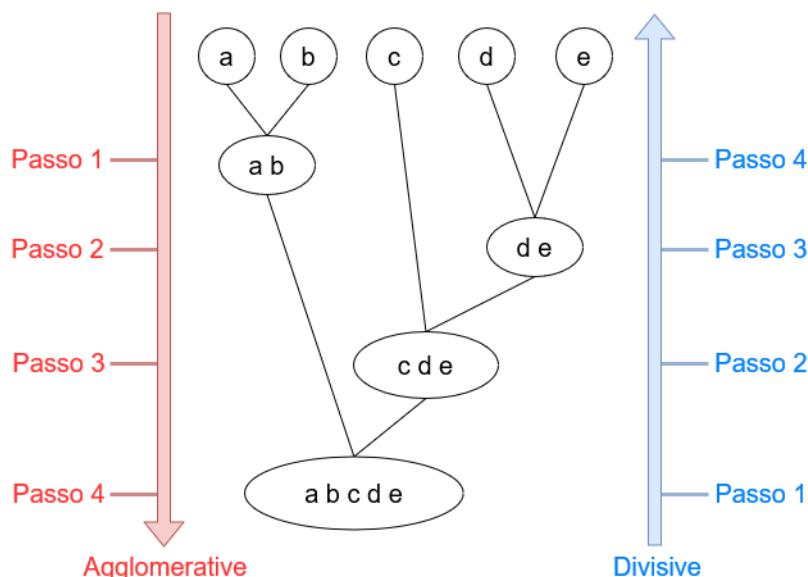


FIGURA 3.3: Esempio di Agglomerative e Divisive Clustering

In questo lavoro di tesi è stato utilizzato l'algoritmo di Agglomerative Clustering. Il primo passaggio per la clusterizzazione di n elementi è la costruzione di una matrice delle similarità simmetrica $n \times n$. Possono essere utilizzate diverse metriche per calcolare queste similarità, come la distanza euclidea, di Manhattan o la cosine similarity.

I passi principali dell'algoritmo sono i seguenti:

1. Si assegna ogni elemento ad un cluster diverso, ottenendone n contenenti un solo punto (un singoletto)
2. Si trova la coppia di cluster più vicini; si uniscono formando un nuovo cluster
3. Si calcola la distanza tra il nuovo cluster ottenuto nel passo 2. ed ognuno dei cluster rimanenti

4. Si iterano i passi 2. e 3. finché tutti gli n punti nel dataset vengono uniti in un singolo cluster

Per decidere quali cluster devono essere combinati è necessario definire una misura di dissimilarità tra cluster. Nella maggior parte dei metodi di clustering gerarchico si fa uso di metriche specifiche che quantificano la distanza tra coppie di elementi e di un criterio di collegamento che specifica la dissimilarità di due cluster come funzione della distanza a coppie tra elementi nei due insiemi. Dati due insiemi di elementi A e B , dove $\Delta(A, B)$ indica la distanza tra di essi e $d(i, j)$ la distanza tra due punti, alcuni criteri comunemente utilizzati sono:

- **Complete linkage:** la distanza tra i cluster è posta pari alla più grande delle distanze istituibili a due a due tra tutti gli elementi dei due cluster.

$$\Delta(A, B) = \max\{d(i, j) : i \in A, j \in B\} \quad (3.1)$$

- **Single linkage:** la distanza tra i cluster è posta pari alla più piccola delle distanze istituibili a due a due tra tutti gli elementi dei due cluster.

$$\Delta(A, B) = \min\{d(i, j) : i \in A, j \in B\} \quad (3.2)$$

- **Average linkage:** la distanza tra i cluster è posta pari alla media aritmetica di tutte le distanze tra gli elementi dei due cluster.

$$\Delta(A, B) = \frac{1}{|A||B|} \sum_{i \in A} \sum_{j \in B} d(i, j) \quad (3.3)$$

- **Metodo di Ward:** differisce in parte dai precedenti in quanto suggerisce di riunire, ad ogni tappa del processo, i due cluster dalla cui fusione deriva il minimo incremento possibile della varianza intra-cluster. È possibile dimostrare che questo criterio corrisponde all'unire i due cluster che hanno la minima distanza pesata tra i centroidi, dove il peso è dato da $\frac{|A||B|}{|A|+|B|}$.

$$\Delta(A, B) = \frac{|A||B|}{|A| + |B|} d(\bar{x}_A, \bar{x}_B) \quad (3.4)$$

dove \bar{x}_A ed \bar{x}_B sono rispettivamente i centroidi dei cluster A e B .

In questo lavoro di tesi è stato utilizzato il metodo di Ward facendo uso come metrica per la distanza tra i punti della distanza euclidea.

La struttura ad albero generata dal clustering gerarchico può essere rappresentata graficamente tramite l'utilizzo di un dendrogramma. Tagliando il dendrogramma ad una determinata altezza è possibile specificare il numero di cluster che verranno generati dall'algoritmo. Nell'esempio in figura 3.4 tagliando il dendrogramma al livello indicato dalla linea tratteggiata si ottiene un numero di cluster pari a quattro.

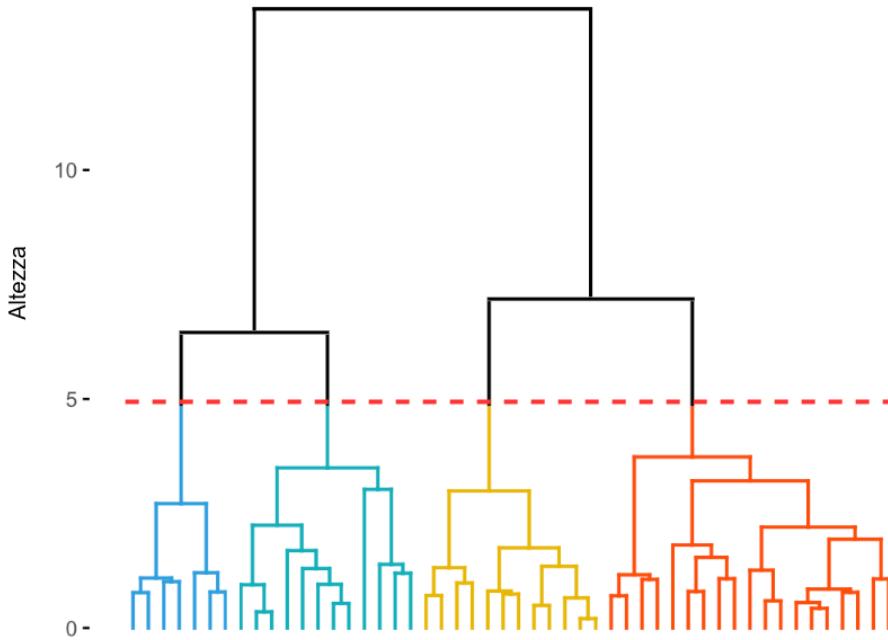


FIGURA 3.4: Esempio di dendrogramma

3.1.3 K-means

L'obiettivo dell'algoritmo partizionale K-means è quello di minimizzare la distanza euclidea tra i centroidi ed i punti ad essi associati. Il valore di K , ovvero il numero di cluster risultanti, viene specificato manualmente dall'utente. Formalmente, sia X un insieme di oggetti. K-means assegna ogni elemento x_i ad un cluster c_j (con $j \leq k$) in modo da minimizzare la somma dei quadrati dei residui (RSS). In altre parole la suddivisione viene effettuata in modo da minimizzare la funzione obiettivo 3.5.

$$\sum_{j=1}^k \sum_{x_i \in c_j} \| x_i - \mu_j \|^2 \quad (3.5)$$

dove μ_j rappresenta il centroide del cluster c_j .

L'algoritmo lavora nel modo seguente:

1. K punti vengono inizializzati casualmente come centroidi dei cluster
2. Per formare i K cluster ogni punto dati del dataset viene assegnato al centroide più vicino rispetto ad una qualche misura di distanza. In questo lavoro di tesi è stata utilizzata la più comunemente utilizzata, ovvero la distanza euclidea.
3. Conseguentemente al cambio dei membri il centroide di ogni cluster viene aggiornato diventando il punto medio di tutti i punti del cluster.

Questo processo di riassegnazione dei membri dei cluster ed aggiornamento dei centroidi viene ripetuto finché non si raggiunge la convergenza, ovvero non vengono riscontrate più modifiche al valore dei centroidi.

Nonostante K-means venga spesso preferito ad altri algoritmi per via della sua semplicità e scalabilità, esso ha difficoltà nell'identificazione di insiemi di densità e dimensione differenti. Inoltre, K-means non può in alcun modo identificare outliers e punti di rumore. Un caso in cui K-means non è in grado di trovare la corretta clusterizzazione dei dati è quella in figura 3.5, dove la struttura del dataset è tale da non poter essere identificata da un algoritmo basato sul concetto di media come K-means (mentre un algoritmo basato sulla densità come DBSCAN, con i giusti parametri, riesce perfettamente).

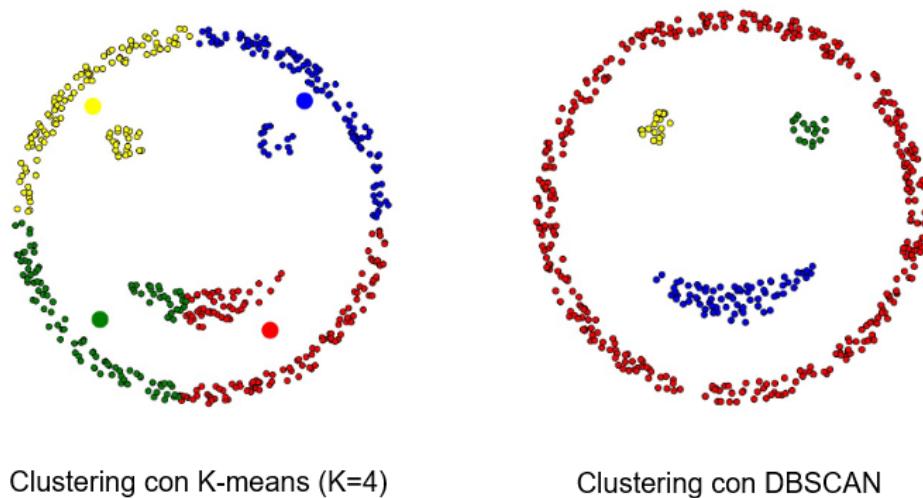


FIGURA 3.5: Clustering con K-means e DBSCAN di forme particolari

	Vantaggi	Svantaggi
K-means	<ul style="list-style-type: none"> - Semplice, facilmente implementabile - Computazionalmente molto veloce - Fortemente scalabile 	<ul style="list-style-type: none"> - Il numero di cluster K deve essere specificato manualmente - Difficoltà nel clusterizzare insiemi di diversa dimensione e grandezza - Impossibilità di identificare outliers e punti di rumore
Agglomerative Clustering	<ul style="list-style-type: none"> - Semplice, facilmente implementabile - Fornisce la struttura gerarchica dell'albero (dendogramma), che può aiutare a decidere il corretto numero di cluster 	<ul style="list-style-type: none"> - Complessità temporale elevata - Difficoltà nel clusterizzare insiemi di dimensione differente e di forma convessa
DBSCAN	<ul style="list-style-type: none"> - Resistente al rumore e gestisce bene gli outliers - Il numero di cluster non deve essere definito in precedenza 	<ul style="list-style-type: none"> - I parametri <i>minPts</i> ed <i>eps</i> possono risultare complessi da stimare - Non funziona bene con insiemi di densità variabile - Non gestisce bene dataset ad alta dimensionalità

TABELLA 3.1: Vantaggi e svantaggi degli algoritmi di clustering considerati

3.2 Riduzione della dimensionalità

Lavorando con dataset aventi un elevato numero di features può risultare difficile comprendere o esplorare le relazioni che intercorrono tra esse. Per questa ragione viene spesso utilizzata nell'ambito del machine learning la riduzione della dimensionalità, ovvero un processo nel quale si riduce il numero di variabili da prendere in considerazione, ottenendo un insieme di variabili principali che conservano in una certa misura la struttura e le informazioni trasportate dalle variabili originali. Riducendo la dimensione dello spazio delle variabili è presente un numero inferiore di relazioni tra le variabili stesse che possono essere dunque studiate e visualizzate facilmente.

La riduzione della dimensionalità può essere ottenuta nei modi seguenti:

- **Eliminazione delle features:** Viene ridotto lo spazio delle features eliminandone alcune. Questo metodo ha un grande svantaggio, ovvero non viene ottenuta alcuna informazione dalle variabili rimosse.
- **Selezione delle features:** Vengono applicati dei test statistici per classificare le features in base alla loro importanza, per poi selezionare un sottoinsieme delle più significative. Anche questo metodo risente del problema precedente ed è meno stabile in quanto test differenti possono assegnare un diverso punteggio di importanza alle variabili.
- **Estrazione delle features:** Vengono create delle nuove features, ognuna combinazione di tutte le features originali. Le tecniche di questo tipo possono essere divise in lineare o non lineari.

Le tecniche di interesse in questo progetto di tesi sono unicamente quelle di estrazione delle features. I metodi di riduzione della dimensionalità di questo tipo convertono il dataset ad alta dimensionalità $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ in un insieme di dati di dimensionalità inferiore $\mathcal{Y} = \{y_1, y_2, \dots, y_n\}$. Se le dimensioni ottenute sono due o tre il dataset può in questo modo essere rappresentato attraverso un grafico di dispersione. L'obiettivo di questa famiglia di tecniche è dunque quello di preservare quanto più possibile la struttura dei dati ad alta dimensionalità. Mentre le tradizionali tecniche lineari di riduzione della dimensionalità come PCA mirano a tenere lontani nella rappresentazione a bassa dimensionalità punti dissimili nella rappresentazione originale, t-SNE punta alla preservazione delle informazioni dei vicini locali. In altre parole t-SNE crea una mappatura di dimensionalità inferiore dove i punti appartenenti allo stesso vicinato sono notevolmente più vicini rispetto a punti in vicinati diversi.

3.2.1 PCA

Principal Component Analysis (PCA) [12], è un metodo statistico di riduzione della dimensionalità utilizzato per ridurre la complessità di un dataset minimizzando

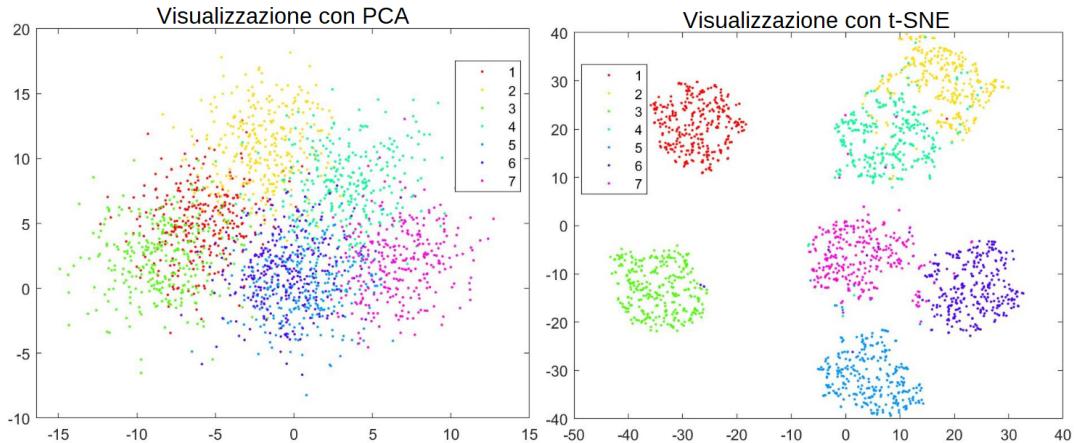


FIGURA 3.6: Confronto di visualizzazione con PCA e t-SNE di sette dataset sei-dimensionali

la perdita d'informazione. PCA trasforma un insieme di dati composto da un numero elevato di variabili correlate in un nuovo insieme di variabili indipendenti, le Principal Components (PC), ordinate sequenzialmente in modo che la prima riesca a catturare la maggior quantità di varianza possibile. In altre parole l'obiettivo di PCA è dunque quello di diminuire il numero di variabili nel dataset preservando quanta più informazione possibile. Le PC devono rispettare i seguenti requisiti:

1. sono combinazione lineare degli attributi originali
2. sono ortogonali tra loro
3. la quantità di varianza da esse catturata diminuisce man mano che ci si sposta dalla prima componente all'ultima. Per questa ragione le prime PC ricoprono un'importanza maggiore rispetto alle successive.

La variabilità dei dati può spesso essere catturata da un numero esiguo di PC, ed è per questo motivo che PCA è particolarmente indicato per la visualizzazione in due o tre dimensioni di dati ad alta dimensionalità. Il primo passo è la normalizzazione o standardizzazione dei dati. Ogni punto viene sostituito con la sua controparte standardizzata utilizzando la formula

$$\frac{x_i - \bar{x}}{\sigma} \quad (3.6)$$

per $1 \leq i \leq n$, dove n è il numero di elementi nel dataset e

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad (3.7)$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}} \quad (3.8)$$

In altre parole ad ogni punto viene sottratta la media e quel valore viene diviso per la deviazione standard per ottenere il nuovo punto. Successivamente viene calcolata la matrice di covarianza utilizzando i dati normalizzati appena ottenuti. La covarianza è una misura che esprime quanto una variabile varia rispetto ad un'altra. La formula per il calcolo della covarianza tra due variabili normalizzate è la seguente:

$$\text{cov}(x, y) = \frac{\sum_{i=1}^n x_i y_i}{n - 1} \quad (3.9)$$

La matrice di covarianza è una matrice contenente tutti i possibili valori della covarianza tra tutte le possibili dimensioni. Considerando un esempio dove il dataset è dotato di tre dimensioni x, y, z la matrice di covarianza verrebbe calcolata come:

$$C = \begin{bmatrix} \text{cov}(x, x) & \text{cov}(x, y) & \text{cov}(x, z) \\ \text{cov}(y, x) & \text{cov}(y, y) & \text{cov}(y, z) \\ \text{cov}(z, x) & \text{cov}(z, y) & \text{cov}(z, z) \end{bmatrix} \quad (3.10)$$

dove la diagonale principale consiste nella covarianza di una variabile con se stessa, cioè semplicemente la sua varianza. Una volta che la matrice di covarianza è stata calcolata possono essere determinati i suoi autovalori ed autovettori. Un autovettore è un vettore che, quando soggetto ad una particolare trasformazione lineare, produce un multiplo scalare del vettore originale. Il valore scalare in questione corrisponde all'autovalore. Autovettori ed autovalori possono essere espressi in termini di A , una matrice quadrata, ed x , un autovettore non nullo di A se esiste un autovalore λ tale che

$$Ax = \lambda x \quad (3.11)$$

Nel caso di PCA la matrice quadrata A è la matrice di covarianza C . Da notare come gli autovettori possono essere trovati solo per matrici quadrate, anche se non necessariamente ogni matrice quadrata ha autovettori. Se però una matrice quadrata $n \times n$ ha autovettori allora ce ne saranno n . Tutti gli autovettori di una matrice sono inoltre perpendicolari tra loro, a prescindere dal numero di variabili coinvolte. Una volta trovati gli autovettori, questi vengono scalati in modo da avere lunghezza unitaria. L'autovettore con autovalore maggiore sarà la componente primaria del dataset. Questo risulta essere facilmente visibile nel momento in cui gli autovettori trovati nella matrice di covarianza vengono ordinati per autovalore dal maggiore al minore, ponendoli dunque in ordine di importanza. Le componenti di minore importanza possono a questo punto venire ignorate. Questo causa una certa perdita di informazione, ma se gli autovalori sono piccoli questa è minima. La rimozione di alcune componenti porta pertanto alla riduzione del numero di dimensioni dei dati originali. A questo punto viene costruita una feature matrix, ovvero una matrice formata dagli autovettori rimasti posti in colonna. L'ultimo passo consiste nel calcolo delle Principal Components. Per farlo viene effettuato il prodotto scalare di ogni

colonna della feature matrix con ogni riga del dataset normalizzato. Questo procedimento fornisce una nuova matrice che contiene le Principal Components: la prima colonna rappresenta la prima Principal Component, la seconda colonna rappresenta la seconda Principal Component e così via. Questa matrice rappresenta i dati originali in termini dei vettori scelti. Nel caso in cui il numero di dimensioni ottenute sia due o tre è possibile infine visualizzare i dati originali utilizzando come assi quelli del dataset finale, essendo le Principal Components tutte ortogonali tra loro. Nel caso della visualizzazione in due dimensioni le prime due Principal Components PC1 e PC2 vengono utilizzate come assi, ruotando i punti di conseguenza.

3.2.2 t-SNE

t-SNE [11] (t-Distributed Stochastic Neighbor Embedding) è una tecnica di riduzione della dimensionalità non lineare, originata dalla preesistente SNE [8], particolarmente indicata per la visualizzazione di dataset ad alta dimensionalità.

Sia $\mathcal{X} = \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$ il dataset di input d-dimensionale. Dato un intero $s \ll d$, l'algoritmo t-SNE crea una mappatura s-dimensionale $\mathcal{Y} = \{y_1, y_2, \dots, y_n\} \subset \mathbb{R}^s$ dei punti in \mathcal{X} . La scelta più comune per s è 2 o 3, in modo che i punti possano essere visualizzati in un grafico di dispersione a due o tre dimensioni.

In primo luogo t-SNE converte le distanze euclidee tra i punti nello spazio ad alta dimensionalità, chiamato anche H-space, in probabilità condizionate che rappresentano valori di similarità. La similarità del punto x_j al punto x_i è la probabilità condizionata $p_{j|i}$ che x_i scelga x_j come suo vicino se i vicini venissero scelti in proporzione alla loro densità di probabilità sotto una Gaussiana centrata in x_i . Per punti vicini $p_{j|i}$ è relativamente alta, mentre nel caso di punti ampiamente separati sarà quasi infinitesimale. Matematicamente, la probabilità condizionata $p_{j|i}$ è calcolata come

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)} \quad (3.12)$$

dove i σ_i rappresentano la varianza della Gaussiana centrata nel punto x_i . Questi vengono scelti in modo che la perplexity della distribuzione condizionata P_i fra tutti i punti x_i , corrisponda ad un valore passato come parametro all'algoritmo dall'utente. La perplexity della distribuzione di probabilità P_i si definisce come

$$\text{Perp}(P_i) = 2^{H(P_i)} \quad (3.13)$$

dove

$$H(P_i) = - \sum_j p_{j|i} \log_2 p_{j|i} \quad (3.14)$$

prende il nome di entropia della distribuzione. La perplexity rappresenta il numero di vicini che esercitano un'influenza attiva sui singoli punti e ricopre un ruolo fondamentale, in quanto valori adeguati di σ_i garantiscono che tutti i punti, compresi

outlier e punti in zone particolarmente sparse, possano trovare i propri vicini più prossimi.

Dopo aver determinato tutte le probabilità condizionate l'algoritmo inizializza una soluzione, ovvero un insieme di n punti nel V-space, generata casualmente sotto una distribuzione Gaussiana. Successivamente vengono calcolate le distanze tra questi punti. In questo caso la varianza è distribuita sotto una distribuzione t di Student, e non sotto una distribuzione Gaussiana come nel caso dei punti dell'H-space. L'utilizzo della distribuzione t di Student con un grado di libertà, dotata dunque di code più pesanti rispetto alla Normale, fa sì che sia i punti più isolati che quelli moderatamente distanti vengano rappresentati ad una distanza minore dai propri vicini.

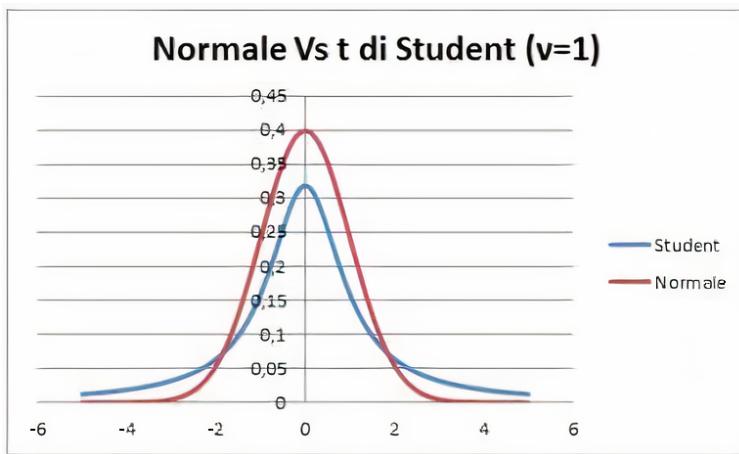


FIGURA 3.7: Confronto tra distribuzione Gaussiana e distribuzione t di Student ad un grado di libertà.

Per le controparti a bassa dimensionalità y_i e y_j dei punti ad alta dimensionalità x_i ed x_j la probabilità condizionata, denotata con $q_{j|i}$, viene calcolata come

$$q_{j|i} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|y_i - y_k\|^2)^{-1}} \quad (3.15)$$

Dal momento che interessa unicamente la similarità tra due punti distinti, si pone $p_{i|i} = 0$ e $q_{i|i} = 0 \forall i$. Occorre evidenziare come $p_{i|j}$ e $p_{j|i}$ non siano uguali, e lo stesso vale per q . t-SNE, a differenza dell'SNE originale, prevede un passaggio ulteriore nel quale ognuna di queste coppie di probabilità condizionate vengono mediate. Questa modifica semplifica la discesa del gradiente della funzione di costo C , rendendo la computazione più veloce ed i risultati migliori, come dimostrato empiricamente dagli autori. Le formule 3.12 e 3.15 vengono ridefinite come

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \quad q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}} \quad (3.16)$$

rispettivamente.

Se i punti $y_i, y_j \in \mathcal{Y}$ modellano correttamente la similarità tra i punti ad alta dimensionalità $x_i, x_j \in \mathcal{X}$, allora le probabilità condizionate p_{ij} e q_{ij} saranno uguali. Motivato da questa osservazione t-SNE punta a trovare una rappresentazione dei dati a bassa dimensionalità che minimizzi la differenza tra p_{ij} e q_{ij} . Per farlo viene utilizzata come funzione di costo C la divergenza di Kullback-Leibler [10], una misura che indica il grado di informazione persa quando la distribuzione Q è utilizzata per approssimare P .

$$C = KL(P \parallel Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \quad (3.17)$$

Ad ogni iterazione la posizione dei punti in V-space viene aggiornata in modo che il gradiente della funzione di costo C

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij}) (1 + \|y_i - y_j\|^2)^{-1} (y_i - y_j) \quad (3.18)$$

sia minimo.

Capitolo 4

Implementazione del Recommender System

In questo capitolo viene presentata l'implementazione del Recommender System, realizzata nel linguaggio di programmazione Python 3 con l'ausilio della libreria Implicit [5]. Quest'ultima fornisce un'implementazione di diversi algoritmi di raccomandazione per dataset a feedback implicito, come i celebri Alternating Least Squares e Bayesian Personalized Ranking esposti nel dettaglio nel capitolo 2. Nella sezione 4.1 viene presentata la struttura dei dataset forniti da Fidelity Salus, così come le operazioni di preprocessing svolte sui dati. Nella sezione 4.2 viene esposta la procedura di creazione della matrice degli acquisti. Nella sezione 4.3 si descrive la procedura di allenamento del modello, specificandone i parametri di inizializzazione. Nella sezione 4.4 vengono infine presentati i test eseguiti per determinare l'algoritmo migliore da utilizzare con il dataset in nostro possesso.

4.1 Dataset e preprocessing dei dati

La lettura e manipolazione dei file .csv forniti da Fidelity Salus è stata condotta con l'ausilio di Pandas [17], ovvero una libreria open source che fornisce strutture e strumenti per l'analisi di dati in linguaggio Python. In particolare ogni dataset viene convertito in un DataFrame, ovvero una struttura dati etichettata a due dimensioni con colonne di tipo potenzialmente eterogeneo.

Il dataset fornito da Fidelity Salus, 001_2019_RecSys_Dataset_v2.5.csv, contiene i dati di vendita di ogni farmacia nel sistema per l'anno solare 2019. La sua struttura è la seguente:

- *UserID*: Codice identificativo del cliente.
- *ItemID* e *ItemName*: Codice identificativo assegnato dal Ministero della Salute e nome del prodotto acquistato.
- *CatCode* e *CatName*: Codice identificativo e nome della categoria alla quale appartiene il prodotto acquistato.
- *Quantity*: Quantità totale di prodotto acquistata dall'utente nell'arco dell'anno solare 2019.

- *PharmacyID*: Codice identificativo della farmacia nella quale sono stati effettuati gli acquisti.

UserID	1397958
ItemID	971989823
ItemName	LFP FERMENTIFLUID 10X10ML
CatCode	4AA2F35
CatName	FERMENTI LATTICI
Quantity	1
PharmacyID	512

TABELLA 4.1: Struttura del dataset
001_2019_RecSys_Dataset_v2.5.csv

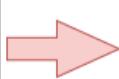
Una prima fase di preprocessing è consistita nella rimozione dei record inerenti oggetti appartenenti ad alcune categorie selezionate precedentemente. Questo è reso necessario dal fatto che ci sono alcuni generi di prodotti che vendono molto a prescindere da qualunque fattore. Alcuni esempi possono essere oggetti di prima necessità, come siringhe ed aghi per insulina, oppure semplicemente articoli di uso comune come cerotti, test di gravidanza o contenitori di vario genere. Considerare questi elementi nel sistema sarebbe un errore, in quanto non identificano particolari abitudini di consumo specifiche per il punto vendita e le liste di raccomandazione sarebbero composte quasi esclusivamente da essi, in quanto venduti in quantità smisurata. L'obiettivo del Recommender System è invece quello di consigliare prodotti vari ed inaspettati, dei quali la farmacia può risultare sprovvista. Sono stati inoltre rimossi record con quantità ≤ 0 , possibile conseguenza di prodotti resi o di malfunzionamenti, e con prodotti particolari venduti specificatamente da un solo punto vendita (ad es. buoni sconto/regalo). Il dataset viene poi alleggerito di tutti i campi non necessari alla costruzione del Recommender System. Questo nuovo dataset, chiamato nel seguito sales_dataset, mantiene solo i campi *PharmacyID*, *ItemID* e *Quantity*. Dopo aver rimosso ogni record con valori mancanti il numero di righe del dataset si attesta su ≈ 9.5 milioni.

4.2 Costruzione della matrice degli acquisti

Una volta terminata la fase di preprocessing si procede alla preparazione dei dati per la creazione della matrice degli acquisti.

Il primo passaggio necessario è la creazione in `sales_dataset` di due nuove colonne contenenti una mappatura delle colonne `PharmacyID` e `ItemID` a valori interi nell'intervallo [0, numero di elementi distinti]. La ragione è da ricercare nella codifica degli ID all'interno del database: gli ID degli oggetti sono stati assegnati dal Ministero della Salute utilizzando numeri a nove cifre. In altre parole questi assumono un valore intero dell'ordine di centinaia di milioni. Ipotizzando un numero di farmacie pari a 1000, al momento della creazione della matrice degli acquisti il sistema cercherebbe di allocare la quantità di spazio necessaria alla memorizzazione di una matrice di dimensione $\approx 10^8 \times 10^3$, fallendo restituendo un messaggio di errore per via del numero eccessivo di celle. Per questo motivo i valori delle colonne `PharmacyID` e `ItemID` vengono convertiti nel tipo `category` di Pandas, ovvero una lista finita di valori testuali, che vengono poi mappati a numeri interi. Questi vengono immagazzinati in due nuove colonne del dataset, `PharmacyCAT` e `ItemCAT`. In questo modo i valori degli ID delle farmacie assumono un valore in [0, numero di farmacie – 1] mentre quelli degli oggetti in [0, numero di oggetti – 1], risolvendo così la problematica sudetta. I valori originali degli ID vengono comunque mantenuti nel dataset in modo da poter essere recuperati alla conclusione del processo per la stampa della lista di raccomandazione.

A questo punto si esegue un raggruppamento in `sales_dataset` aggregando rispetto alla coppia di attributi `PharmacyID/ItemID` sommando la `Quantity` in modo da ottenere un nuovo dataset, `sales_dataset_grouped`, che abbia una riga per ogni coppia. Ogni record del nuovo dataset indica quante unità di un prodotto sono state acquistate complessivamente in un certo punto vendita. Il dataset appena ottenuto è composto da ≈ 2.8 milioni di righe.



sales_dataset		
PharmacyCAT	ItemCAT	Quantity
15	1017	7
137	1017	3
137	96	1
15	1017	10
137	96	6

sales_dataset_grouped		
PharmacyCAT	ItemCAT	Quantity
15	1017	17
137	1017	3
137	96	7

FIGURA 4.1: Esempio di raggruppamento di `sales_dataset`

Dopo aver eseguito queste operazioni preliminari si procede alla costruzione della matrice degli acquisti. Ogni farmacia ha in vendita un numero di prodotti relativamente ristretto rispetto al totale: 10588 in media a fronte di 134850 articoli totali. Gran parte della matrice sarà pertanto composta da celle contenenti il valore 0. Una struttura con queste caratteristiche prende il nome di matrice sparsa. Per la gestione efficiente di matrici sparse la libreria `scipy` mette a disposizione diverse funzioni che

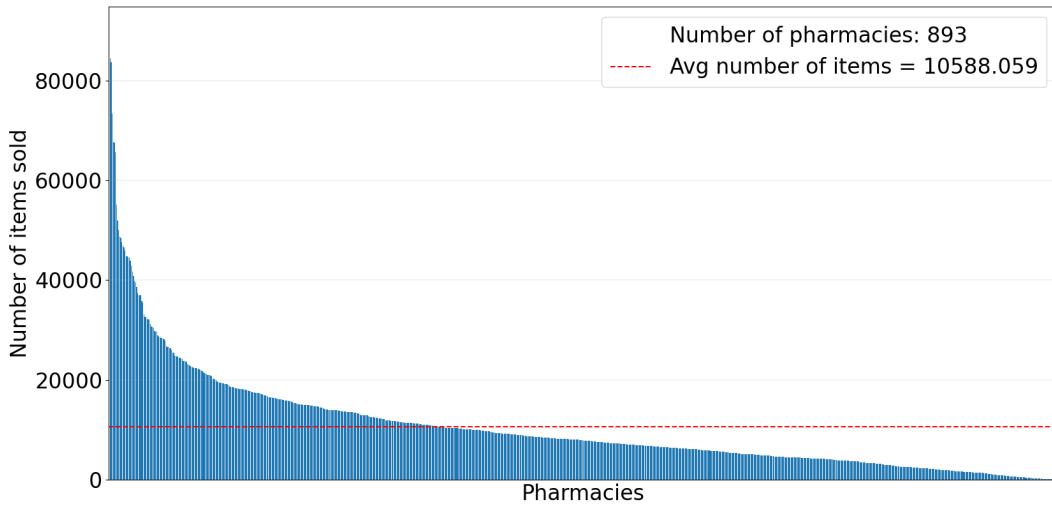


FIGURA 4.2: Distribuzione del numero di oggetti venduti nelle farmacie

permettono di memorizzare i soli valori diversi da zero, consentendo un notevole risparmio di memoria ed un forte miglioramento dei tempi di calcolo. Una di queste è l'algoritmo Compressed Sparse Row (CSR), il quale rappresenta una matrice come tre vettori unidimensionali (V , COL_INDEX , ROW_INDEX) che contengono rispettivamente i valori non nulli di *Quantity* e gli indici di colonna e di riga corrispondenti. La funzione prende in input due liste: una contenente i valori ed una contenente coppie di indici, rispettivamente quello di riga e quello di colonna. Implicit utilizza due matrici sparse differenti in due diverse fasi del processo di raccomandazione:

- una matrice $\text{item} \times \text{user}$ in fase di allenamento del modello, dove le righe rappresentano i possibili prodotti e le colonne le farmacie. In questo caso si passano dunque a `csr_matrix` le colonne di `sales_dataset_grouped` (*Quantity*, (*Item-CAT*, *PharmacyCAT*)).
- una matrice $\text{user} \times \text{item}$ in fase di stesura della lista di raccomandazione, dove le righe rappresentano le farmacie e le colonne i possibili prodotti. In questo caso si passano dunque a `csr_matrix` le colonne di `sales_dataset_grouped` (*Quantity*, (*PharmacyCAT*, *ItemCAT*)).

Si mostra in figura 4.3 un esempio di come una matrice sparsa venga rappresentata nei formati tradizionale e CSR.

$\begin{pmatrix} 0 & 0 & 0 & 9 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$	Matrice tradizionale	Matrice CSR
	(0,3)	9
	(1,1)	8
	(2,2)	1
	(3,1)	6

FIGURA 4.3: Rappresentazione tradizionale e CSR di una matrice

4.3 Allenamento del modello

La struttura del modello utilizzato per la raccomandazione varia in base all'algoritmo che si intende utilizzare. Quelli considerati in questo progetto di tesi sono ALS e BPR, entrambi esposti estensivamente nei capitoli 2.2.3 e 2.2.4.

4.3.1 Allenamento con ALS

Il metodo `implicit.ALS.AlternatingLeastSquares()` si occupa dell'inizializzazione del modello ALS. I parametri principali sono i seguenti:

- **factors** (f): indica il numero di fattori latenti da calcolare
- **regularization** (λ): indica il valore del fattore di regolarizzazione
- **iterations** (it): indica il numero di iterazioni ALS da eseguire al momento del fitting dei dati

In questo lavoro sono stati impostati i parametri $f = 350, \lambda = 0.01, it = 150$, in quanto hanno garantito sperimentalmente i risultati migliori in tempi ragionevoli. Questo metodo restituisce un oggetto *model*, contenente il modello pronto per essere adattato ai dati.

È il metodo `model.fit()` ad occuparsi del fitting del modello. Una volta che questo metodo viene chiamato gli user-factors e gli item-factors vengono inizializzati con un modello a fattori latenti dei dati in input. La matrice item-user ricopre un duplice ruolo: oltre a definire quali oggetti sono piaciuti a quali utenti (p_{iu} nell'articolo originale), specifica anche il livello di confidenza che si ha del fatto che l'utente abbia davvero gradito l'oggetto (c_{iu}), ovvero il valore di *Quantity* all'interno delle celle. I valori nulli vengono trattati in maniera neutra assumendo $p_{iu} = 0$ e $c_{iu} = 1$. L'unico parametro richiesto dal metodo è la matrice dei dati in input, ovvero la matrice item×user.

4.3.2 Allenamento con BPR

Se si desidera utilizzare l'algoritmo BPR il modello viene inizializzato con il metodo `implicit.BPR.BayesianPersonalizedRanking()`. I parametri principali sono i seguenti:

- **factors** (f): indica il numero di fattori latenti da calcolare
- **regularization** (λ): indica il valore del fattore di regolarizzazione
- **Learning rate** (lr): indica il tasso di apprendimento, ovvero l'ampiezza di ogni passo nella procedura di discesa stocastica del gradiente (SGD)
- **iterations** (it): indica il numero di epoche da utilizzare al momento del fitting dei dati

In questo lavoro sono stati impostati i parametri $f = 350$, $\lambda = 0.1$, $lr = 0.01$, $it = 150$. Questo metodo, come per l'ALS, restituisce un oggetto *model* contenente il modello pronto per essere adattato ai dati.

Da un punto di vista sintattico il metodo `model.fit()` si comporta nella stessa maniera dell'algoritmo precedente, ricevendo in input la sola matrice `item × user`. La differenza sta nel fatto che il modello generato da BPR ignora il peso dei valori contenuti nelle celle della matrice, considerando semplicemente i valori $\neq 0$ come un segnale binario del fatto che l'utente abbia apprezzato l'oggetto.

4.4 Erogazione delle raccomandazioni

Il metodo `model.recommend()` si occupa di generare la lista di raccomandazione. I parametri principali sono i seguenti:

- **userid**: l'ID dell'utente per il quale calcolare le raccomandazioni
- **user × items**: la matrice sparsa di dimensione (numero di utenti \times numero di oggetti) creata in precedenza. Questa viene utilizzata per consultare gli oggetti piaciuti all'utente e le *Quantity* corrispondenti e filtrare gli oggetti già piaciuti all'utente se lo si desidera
- **number** (N): valore intero che indica il numero di risultati da restituire
- **filter_already_liked_items**: valore booleano che indica se escludere dalla lista dei risultati oggetti presenti nel training set già piaciuti all'utente

In questo lavoro di tesi viene impostato $N = 10$ e `filter_already_liked_items = True`. Il metodo restituisce una lista di tuple (item, score) di N elementi che rappresenta gli N oggetti con score calcolato più alto.

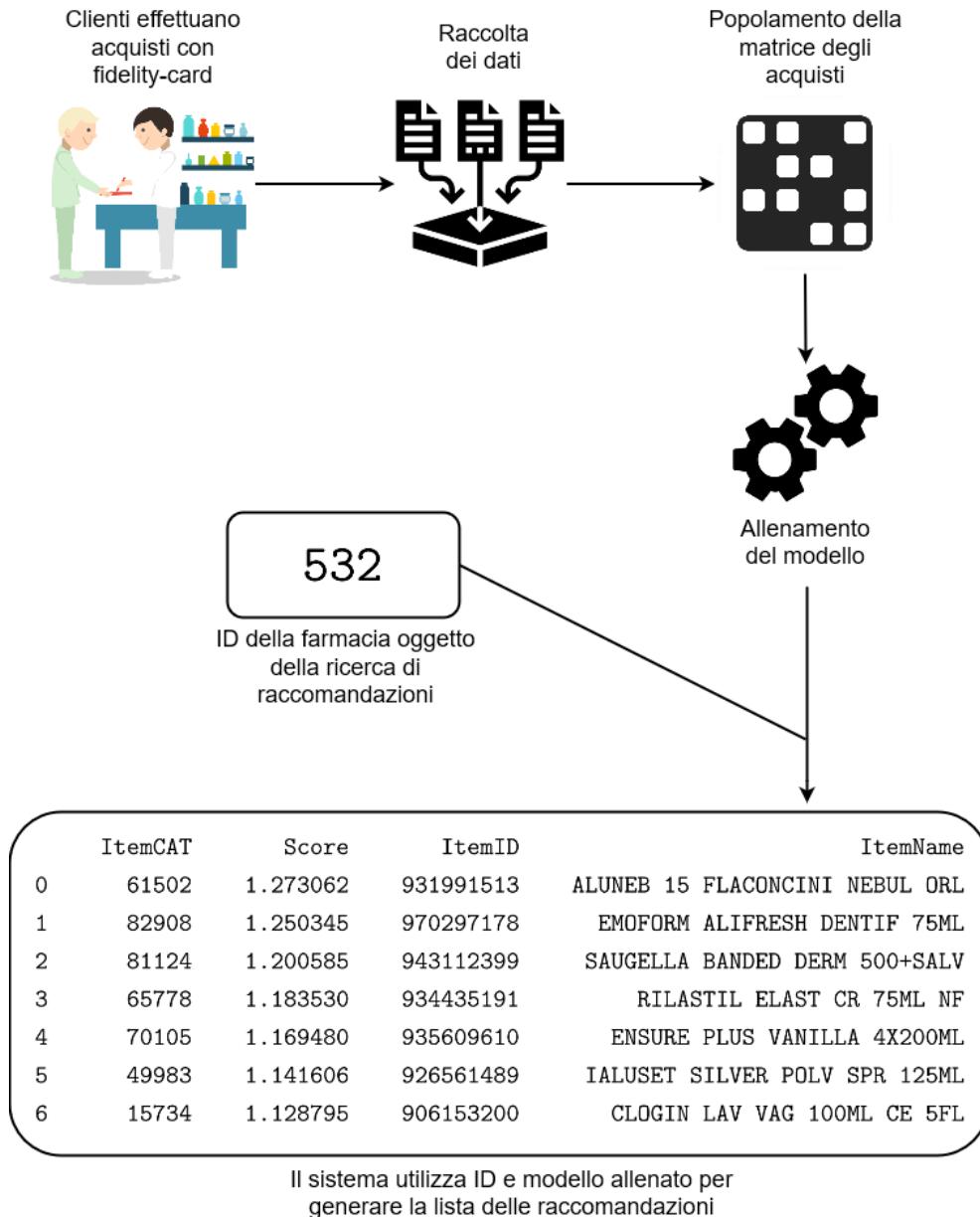


FIGURA 4.4: Schema di funzionamento del Recommender System

4.5 Testing

Al termine della stesura del sistema di raccomandazione è seguita una fase di testing atta a determinare quale dei due algoritmi, ALS o BPR, avesse il comportamento migliore nel caso specifico del dataset fornito da Fidelity Salus. Per misurare l'efficienza degli algoritmi sono state utilizzate le metriche descritte estensivamente nel capitolo 2.2.5, ovvero Precision@K, MAP@K ed NDCG@K. Per tutte le misure sono state considerate le top-5 e le top-10 raccomandazioni. In questa fase il dataset sales_dataset_grouped è stato partizionato in due insiemi disgiunti:

- L'80% dei record sono assegnati al training set, ovvero l'insieme di dati utilizzato per l'allenamento del modello.

- Il 20% dei record sono assegnati al test set, ovvero l'insieme di dati utilizzato per valutare le prestazioni del sistema, o in altre parole per generalizzarlo. Il sistema infatti assume di non conoscere il contenuto delle celle nella matrice di test ed utilizza il modello allenato col training set per calcolarle. Queste previsioni vengono poi comparate con i reali valori delle celle per determinare l'accuratezza del modello.

Gli elementi che appartengono a l'uno o l'altro insieme vengono determinati casualmente. Come è possibile notare dai risultati esposti in tabella 4.2 con l'algoritmo BPR si ottengono i risultati migliori.

	ALS	BPR
Sparsity	97.68%	
Precision@5	23.01%	76.45%
Precision@10	23.42%	73.18%
MAP@5	14.45%	71.16%
MAP@10	12.04%	65.57%
nDCG@5	23.08%	77.39%
nDCG@10	23.34%	74.78%

TABELLA 4.2: Risultati dei test di comparazione degli algoritmi di raccomandazione

Capitolo 5

Applicazione degli algoritmi di clustering

Una volta aver determinato quale algoritmo di raccomandazione garantisca i migliori risultati ci si è chiesto se si potessero applicare algoritmi di clustering per migliorare le già ottime performance del Recommender System. Nella sezione 5.1 viene presentata la struttura del dataset utilizzato. Nella sezione 5.2 vengono presentati gli attributi del dataset considerati per il lavoro di clustering e descritta la fase di preprocessing che li coinvolge. Nelle sezioni 5.3, 5.4 e 5.5 vengono mostrate le applicazioni degli algoritmi descritti nel capitolo 3 al dataset e presentati i risultati migliori. Nella sezione 5.6 vengono mostrati i risultati ottenuti in vari test di esempio che utilizzano diverse combinazioni di attributi.

5.1 Struttura dei dati

Per poter suddividere le farmacie in base alle loro caratteristiche è stato studiato un secondo dataset fornito da Fidelity Salus, 002_2019_Mapping_Farmacie_v2.5.csv, che contiene varie informazioni riguardanti i punti vendita. La struttura del dataset è la seguente:

- *PharmacyID* e *PharmacyName*: Codice identificativo e nome della farmacia.
- *GroupID* e *GroupDES*: Codice identificativo e nome del gruppo al quale la farmacia appartiene. Tipicamente appartengono allo stesso gruppo farmacie di proprietà dello stesso individuo o ente. Sono anche ammessi gruppi composti da una sola farmacia.
- *ConsortiumCode* e *ConsortiumDES*: Codice identificativo e nome del consorzio al quale la farmacia appartiene. Un consorzio è un'organizzazione a cui più farmacie possono associarsi e che ha lo scopo di realizzare gli interessi finanziari di queste ultime. Farmacie dello stesso consorzio potrebbero avere fornitori in comune o utilizzare strategie simili di fidelizzazione/marketing. Questo campo è opzionale, in quanto una farmacia può non far parte di alcun consorzio.

- *City, CAP e Province*: Informazioni geografiche della farmacia.
- *StartupDate*: Data di ingresso della farmacia nel sistema Fidelity Salus.
- *Tutor*: Soggetto che rappresenta Fidelity Salus nell’interfacciamento quotidiano con le farmacie. Il tutor si occupa della scelta delle promozioni da offrire ai clienti, della gestione e pubblicizzazione delle fidelity card e in generale dell’erogazione alle singole farmacie di tutti i servizi che Fidelity Salus offre.
- *Revenue*: Valore numerico che rappresenta il fatturato della farmacia nell’anno solare 2019, considerando unicamente gli acquisti effettuati con fidelity card.
- *UsersNumber*: Valore numerico che rappresenta il numero di utenti dotati di fidelity card che hanno compiuto acquisti nell’anno solare 2019.

PharmacyID	1172
PharmacyName	FARMACIA DR. PAGANELLI
GroupID	61
GroupDES	FARMACIA DR. PAGANELLI
ConsortiumCode	5
ConsortiumDES	UNIFARCO
City	ROMA
CAP	00132
Province	RM
StartupDate	2014-01-13
Tutor	martas
Revenue	734427.50
UsersNumber	938

TABELLA 5.1: Struttura del dataset
002_2019_Mapping_Farmacie_v2.5.csv

Nel dataset ricevuto dall’azienda erano presenti alcuni record con valori mancanti. Mentre è stato possibile inserirli manualmente in alcuni campi come *CAP* o *Province* tramite una ricerca in un qualunque search engine, punti vendita con valori mancanti in attributi non pubblicamente disponibili come *Revenue* o *Tutor* sono stati rimossi. A seguito di questo filtraggio il numero di farmacie totali nel dataset è di 893.

5.2 Selezione e preprocessing degli attributi

Non tutti gli attributi del dataset sono adeguati alle operazioni di clustering. Per quanto riguarda le variabili categoriche, identificate dunque da un’etichetta testuale scelta da un insieme limitato di elementi, il numero di classi non deve essere troppo elevato. In questo caso infatti il numero di record aventi un certo valore dell’attributo è troppo basso, e dunque non sono presenti abbastanza informazioni per descrivere adeguatamente la classe. In tabella 5.2 viene mostrato il numero di label diverse

	Numero di etichette
CAP	610
City	611
Province	94
GroupID	615
ConsortiumCode	9
Tutor	7
StartupDate	575

TABELLA 5.2: Numero di etichette uniche per ogni attributo

per ogni attributo nel dataset. Gli attributi *CAP*, *City*, *Province*, *GroupID*, *StartupDate* si rivelano inadatti ad essere considerati per il clustering e vengono pertanto scartati. In sostituzione di due degli attributi esclusi ne vengono creati due nuovi, *Zone* e *StartupYear*:

- *Zone*: indica la zona geografica della farmacia. In figura 5.1 vengono mostrate le tre zone in cui il territorio è stato suddiviso: blu per il nord Italia, verde per il centro e rosso per il sud.

FIGURA 5.1: Territorio italiano suddiviso nelle tre zone geografiche
North, Center e South

- *StartupYear*: indica il solo anno dell'attributo *StartupDate*

Al fine di aumentare il numero di variabili da poter prendere in considerazione è stato utilizzato il dataset 01_2019_RecSys_Dataset_v2.5.csv, ovvero quello contenente le informazioni riguardanti gli acquisti nell'arco dell'anno solare 2019 utilizzato per l'allenamento del modello del Recommender System, per derivare quattro attributi ulteriori:

- *ItemsNumber*: numero totale di prodotti distinti venduti
- *TotalQuantity*: somma della quantità cumulativa di tutti i prodotti venduti

- *SalesMean*: valore medio delle quantità cumulative vendute di tutti i prodotti
- *SalesMax*: valore massimo della quantità cumulativa venduta di un prodotto

Per decidere quali attributi prendere in considerazione per le operazioni di clustering è stata utilizzata la cosiddetta matrice di correlazione. Dato il numero di attributi p , la matrice di correlazione è una matrice simmetrica quadrata di dimensioni $p \times p$ che contiene i coefficienti di correlazione fra tutti gli attributi del dataset. Il coefficiente di correlazione r è un valore reale compreso nell'intervallo $[-1, 1]$ che esprime il grado di dipendenza tra due variabili:

- Più r si avvicina a zero, più la correlazione lineare è debole.
- Un valore r positivo è indice di una correlazione positiva, in cui i valori delle due variabili tendono ad aumentare in parallelo.
- Un valore r negativo è indice di una correlazione negativa, in cui il valore di una variabile tende ad aumentare quando l'altra diminuisce.
- I valori 1 e -1 rappresentano le correlazioni "perfette", una positiva e l'altra negativa. Due variabili perfettamente correlate mutano insieme a velocità fissa. In questo caso, si dice che hanno una relazione lineare perché, se inseriti in un grafico a dispersione, tutti i punti di dati possono essere collegati tra loro tramite una linea retta. Sono un esempio di questo le variabili *Revenue* ed *UsersNumber*, dotate di un coefficiente di correlazione pari a 0.79: il grafico a dispersione di figura 5.2 evidenzia questa diretta proporzionalità.

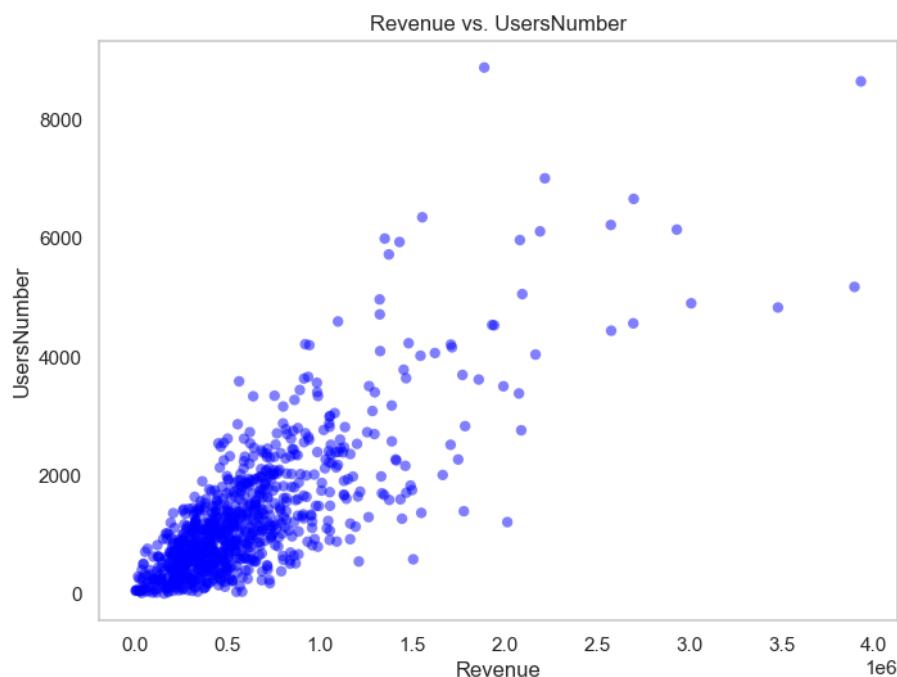


FIGURA 5.2: Grafico a dispersione delle variabili correlate *Revenue* ed *UsersNumber*

Dalla tabella di correlazione del nostro dataset, mostrata in figura 5.3, appare evidente come la quasi totalità delle variabili numeriche siano fortemente correlate. Eccezione fa la variabile *SalesMean*, la cui correlazione lineare verso le altre variabili risulta essere pressoché nulla. Per questa ragione, per minimizzare l'overlapping delle informazioni all'interno dei vari attributi le uniche variabili numeriche prese in considerazione sono *Revenue* e *SalesMean*.

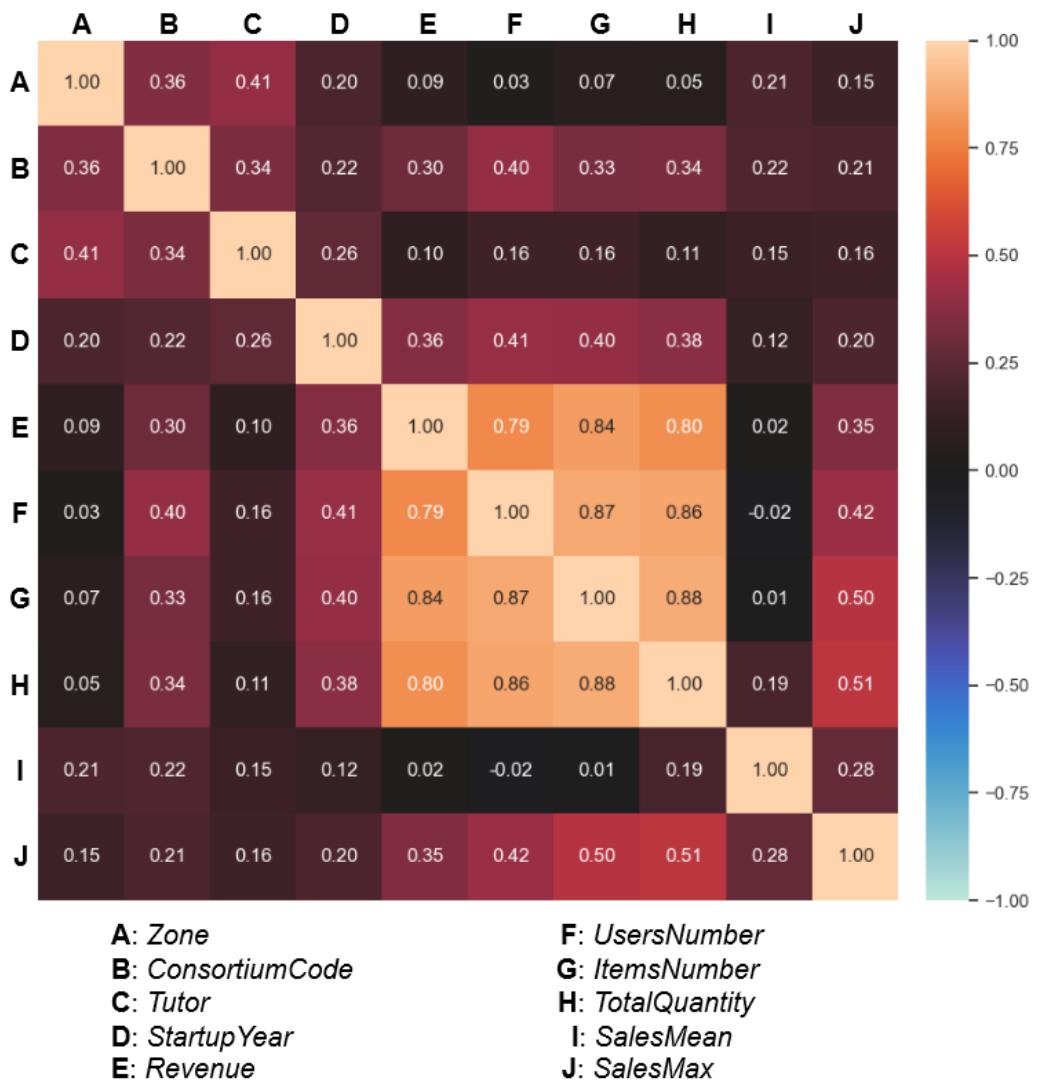


FIGURA 5.3: Matrice di correlazione degli attributi del dataset

Il dataset risulta dunque essere composto sia da variabili numeriche che categoriche, che richiedono diverse attività di preprocessing. Gli algoritmi utilizzati in fase di clustering, ovvero DBSCAN, K-means e Agglomerative Clustering, accettano come input esclusivamente variabili numeriche. Viene dunque applicata una conversione delle variabili categoriche in variabili numeriche, la quale può essere applicata utilizzando due diverse tecniche:

1. **Integer Encoding:** ad ogni etichetta viene assegnato un valore intero. Nel caso in cui tra i possibili valori dell'attributo intercorra un naturale ordinamento

(come ad esempio *primo*, *secondo*, *terzo*), questo metodo potrebbe essere sufficiente. Per attributi dove questa relazione non sussiste non lo è e potrebbe anzi risultare dannoso, in quanto l'algoritmo potrebbe assumere l'ordinamento anche se non esiste, alterando significativamente i risultati.

2. **One-Hot Encoding:** le variabili categoriche vengono sostituite da nuove variabili binarie, una per ogni valore distinto, che prendono il nome di *dummy variables*. Ad ogni record del dataset viene assegnato il valore 1 nella variabile binaria che indica l'etichetta originale di appartenenza, 0 in tutte le altre.

Vista la natura delle variabili del nostro dataset, dove non sussiste un ordinamento, la tecnica utilizzata è quella del One-Hot Encoding.



PharmacyCAT	ConsortiumCode
593	1
124	5
600	1
331	6

PharmacyCAT	ConsortiumCode_1	ConsortiumCode_5	ConsortiumCode_6
593	1	0	0
124	0	1	0
600	1	0	0
331	0	0	1

FIGURA 5.4: Esempio di One-Hot Encoding sulla variabile categorica ConsortiumCode

Gli attributi numerici devono essere standardizzati prima di poter essere utilizzati per il clustering. La standardizzazione dei dati è infatti un requisito comune per molti algoritmi di machine learning, atta ad impedire che le diverse unità di misura delle variabili influiscano sui risultati. Il metodo utilizzato è StandardScaler, offerto dalla libreria scikit-learn. L'idea alla base è quella di trasformare le variabili individualmente in modo che esse si comportino pressappoco come una distribuzione normale standard, con media $\mu = 0$ e varianza $\sigma = 1$. Nella pratica StandardScaler ignora la forma della distribuzione, limitandosi a centrare i valori sottraendo ad ognuno la media e dividendoli per la deviazione standard. Matematicamente, sia N il numero di elementi della colonna da standardizzare. Per ogni valore x della colonna il suo valore standardizzato z viene calcolato come:

$$z = \frac{x - \mu}{\sigma} \quad (5.1)$$

dove

$$\mu = \frac{1}{N} \sum_{i=1}^N (x_i) \quad (5.2)$$

$$\sigma = \sqrt{\frac{1}{N} \sum_{i=1}^N N(x_i - \mu)^2} \quad (5.3)$$

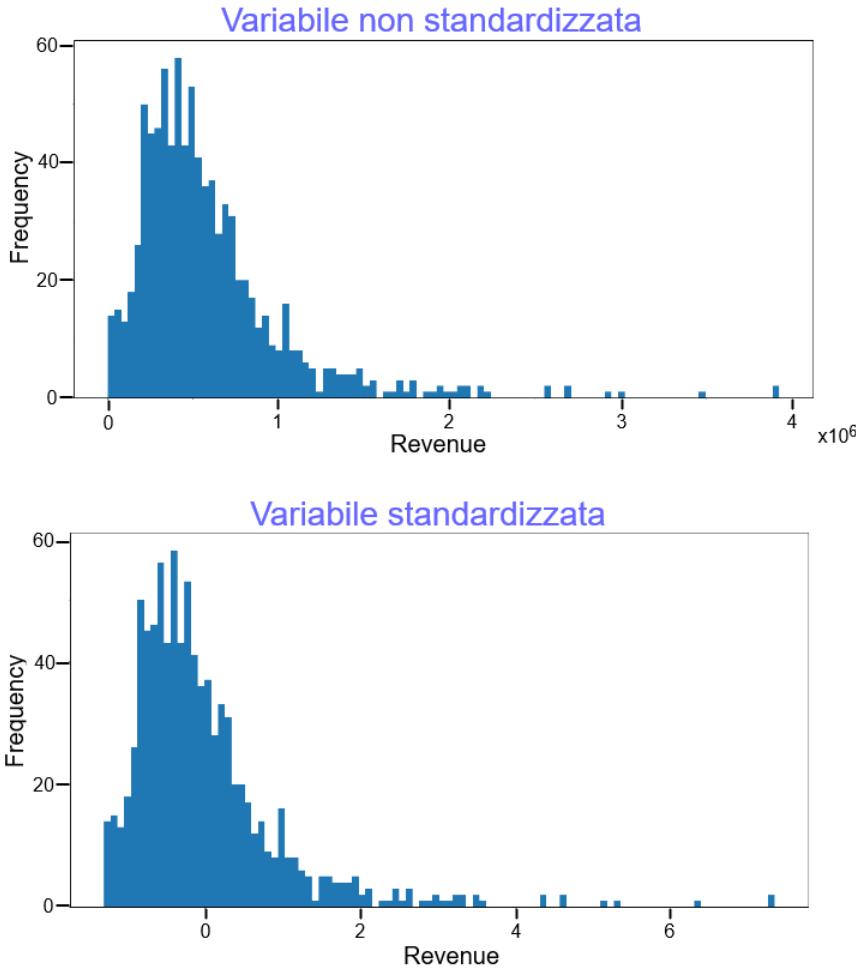


FIGURA 5.5: Esempio di standardizzazione di una variabile numerica

Nell'esempio in figura 5.5 è possibile notare come la distribuzione della variabile *Revenue* non modifichi la sua forma, ma venga traslato lungo l'asse orizzontale e l'intervallo di dominio ridimensionato.

5.3 Clustering con DBSCAN

Il primo algoritmo considerato è stato DBSCAN, il quale fornisce clusterizzazioni basate sulla densità. Il passo più delicato dell'utilizzo di questo algoritmo è la scelta dei due parametri *eps* e *minPts*, determinanti per la riuscita del processo. Questi sono stati scelti seguendo il procedimento seguente:

- Per *minPts* un buon punto di partenza è rappresentato dal doppio del numero di dimensioni del dataset [14].

- Una volta fissato il valore di $minPts$ è possibile utilizzare una tecnica che prende il nome di Elbow Method per determinare il valore di eps ottimale. Sia $k = minpts$. Il funzionamento di questo metodo è il seguente:
 1. Una volta fissato il valore di k viene calcolata per ogni punto nel dataset la distanza tra esso ed il suo k -esimo vicino.
 2. Viene tracciato un diagramma cartesiano avente sull'asse x gli indici dei punti e sull'asse y le distanze calcolate al passo precedente, ordinate in ordine crescente.
 3. Il grafico che si ottiene sarà concavo verso l'alto e la curva avrà un punto che rappresenta il punto di massima curvatura, dove il tasso di incremento aumenta drasticamente. Questo punto prende il nome di elbow point e la sua coordinata y rappresenta il valore di eps cercato.

I valori di $minPts$ ed eps così trovati rappresentano comunque solo un punto di partenza: variando leggermente i parametri si ottengono clusterizzazioni differenti che vanno valutate e confrontate per ottenere il risultato migliore.

Un primo tentativo di clustering density-based utilizzando DBSCAN è stato eseguito sull'intero insieme di attributi, ovvero *Revenue*, *SalesMean*, *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*. I parametri impostati sono $eps = 1.85$ e $minPts = 10$.

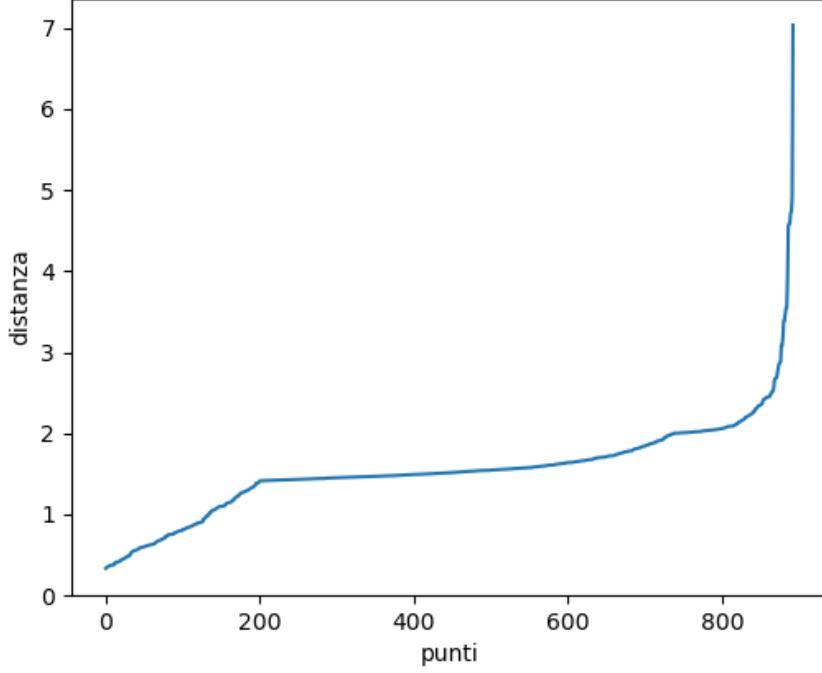


FIGURA 5.6: Elbow method con attributi *Revenue*, *SalesMean*, *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear* e valore di $minPts = 10$

DBSCAN rileva una zona ad alta densità che contiene la quasi totalità dei punti ed una decina di punti di rumore, identificati dall'algoritmo con l'etichetta -1, come evidenziato in figura 5.7.

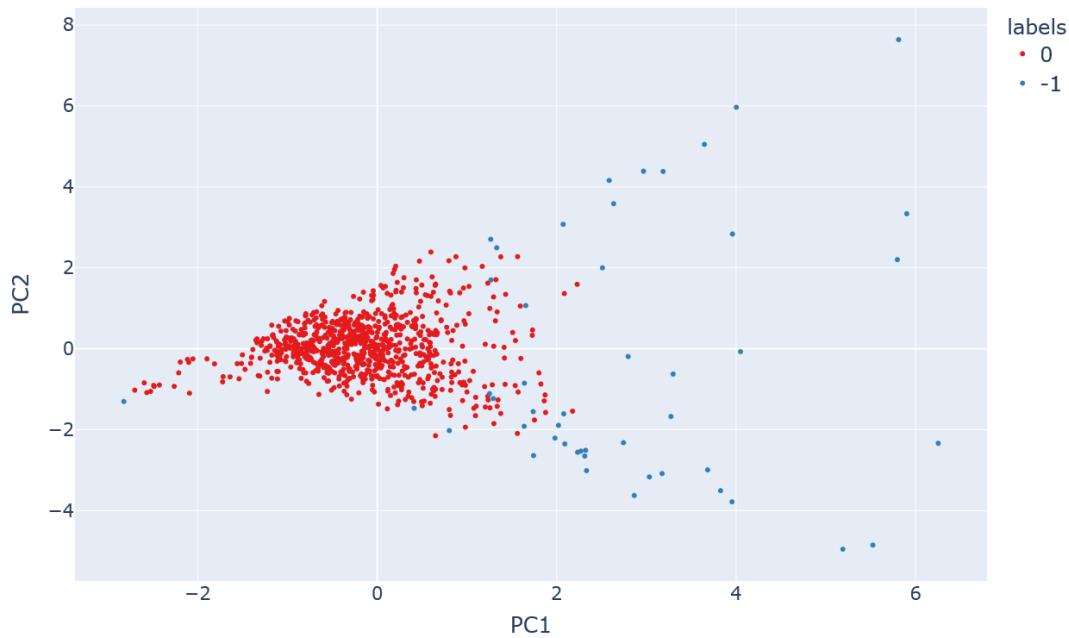


FIGURA 5.7: Grafico PCA con attributi *Revenue*, *SalesMean*, *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear* e parametri $\text{eps} = 1.85$ e $\text{minPts} = 10$

Sono state testate varie combinazioni di attributi, per vedere se fosse possibile trovare cluster ben definiti. Viene riportata la prova effettuata su una di queste combinazioni, composta dagli attributi *Revenue*, *SalesMean*, *Tutor*, *Zone*, utilizzando come parametri dell’algoritmo $\text{eps} = 1.4$ e $\text{minPts} = 8$. In questo caso, come mostrato dalla visualizzazione t-SNE di figura 5.8, DBSCAN rileva un gran numero di zone ad alta densità composte da un numero relativamente esiguo di punti. A seguito di ulteriori analisi risulta che i cluster identificati non sono altro che insiemi composti da elementi aventi gli stessi valori degli attributi categorici. In altre parole non vengono mai inseriti punti con etichette differenti all’interno dello stesso cluster. Le variabili numeriche sembrano essere completamente dominate da quelle categoriche, in quanto non hanno alcun effetto sulla definizione dei cluster. Ad ulteriore riprova della dominanza delle variabili categoriche si porta un esempio ulteriore, generato con i soli attributi *Revenue*, *SalesMean* e *Tutor*. Eseguendo una run di DBSCAN su questo dataset ed analizzando la visualizzazione t-SNE ottenuta in figura 5.9 appare evidente come i sette cluster non siano altro che i record suddivisi per tutor, l’unica variabile categorica presente. Questo stesso comportamento viene ottenuto con una qualunque combinazione di una variabile categorica e di quelle numeriche. L’approccio basato sulla densità si rivela dunque inadatto al clustering del nostro dataset.

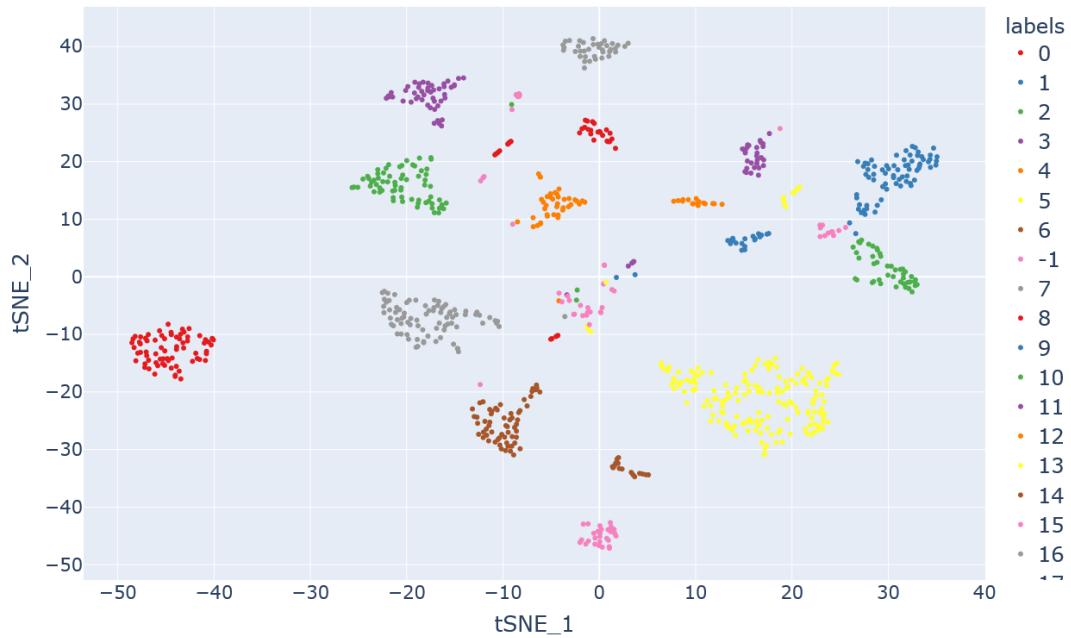


FIGURA 5.8: Grafico t-SNE con attributi *Revenue*, *SalesMean*, *Tutor*, *Zone*, e parametri $\text{eps} = 1.4$ e $\text{minPts} = 8$

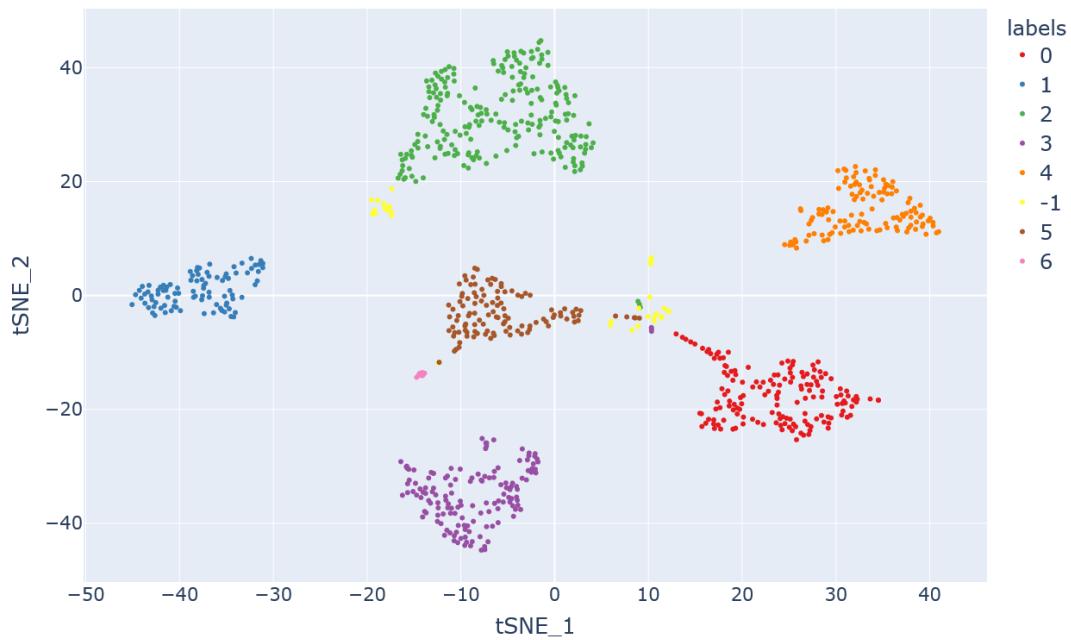


FIGURA 5.9: Grafico t-SNE con attributi *Revenue*, *SalesMean*, *Tutor* e parametri $\text{eps} = 1.1$ e $\text{minPts} = 7$

5.4 Clustering con K-means

Il secondo algoritmo di clustering considerato è K-means. A differenza di DBSCAN questo algoritmo richiede l'inserimento del numero di cluster K desiderato da parte dell'utente. Per decidere il valore ottimale di K è stato utilizzato il metodo del silouhette score. Il valore di silouhette è una misura di quanto un oggetto è simile ad altri all'interno del suo stesso cluster (cohesion) e dissimile ad elementi in cluster differenti (separation). Questo valore varia nell'intervallo $[-1, 1]$, dove un valore alto indica che l'oggetto è ben abbinato con gli altri elementi del suo cluster e mal abbinato con elementi di cluster differenti. Il silouhette score dell'intero dataset è dato dalla media dei valori di silouhette di tutti i punti. Se questo è elevato la configurazione di clustering risulta essere appropriata, nel caso contrario è presente un numero di cluster insufficiente o eccessivo. Al momento della scelta del valore di K essa ricadrà dunque su quello con silouhette score più elevato. Ad ogni modo, essendo presenti nel database un numero di elementi relativamente esiguo, sono stati considerati solo valori di $K \leq 5$.

Un aspetto da tenere in considerazione sia per K-means che per Agglomerative Clustering è il fatto che, a differenza di DBSCAN, questi restituiscono sempre un risultato. Questi algoritmi infatti suddividono sempre il dataset in K cluster, anche se in realtà questo non presenta pattern interessanti. Per questo motivo occorre porre particolare attenzione nell'analisi dei risultati, in modo tale da distinguere clustering reali derivanti dalle caratteristiche dei dati e semplici artefatti degli algoritmi.

La suddivisione che ha portato ai migliori risultati in termini di performance del Recommender System è stata quella che fa uso del dataset completo, ovvero con attributi *Revenue*, *SalesMean*, *Tutor*, *ConsortiumCode*, *StartupYear* e *Zone*, della quale è riportata la visualizzazione PCA in figura 5.10. Il parametro K è stato impostato = 2 in quanto porta al maggior valore del silouhette score.

```
silhouette_score (numero di cluster:2) = 0.289
silhouette_score (numero di cluster:3) = 0.239
silhouette_score (numero di cluster:4) = 0.158
silhouette_score (numero di cluster:5) = 0.192
```

Il clustering produce due insiemi contenenti rispettivamente 127 e 766 farmacie. Analizzando le distribuzioni degli attributi all'interno dei cluster è possibile notare come, a meno di una piccola sovrapposizione, i record con *Revenue* maggiore siano inseriti nel primo cluster e la restante parte nel secondo, come mostrato in tabella 5.3. Gli attributi rimanenti risultano invece essere distribuiti in maniera abbastanza uniforme.

Una volta identificato il clustering vengono effettuati dei test atti a verificare se le performance del recommender incrementino come conseguenza di questa suddivisione. Per farlo vengono creati due dataset degli acquisti, contenenti rispettivamente i soli dati di vendita inerenti farmacie nei cluster 1 e 2. Il programma di test viene poi eseguito separatamente su questi dati e si confrontano i valori di performance

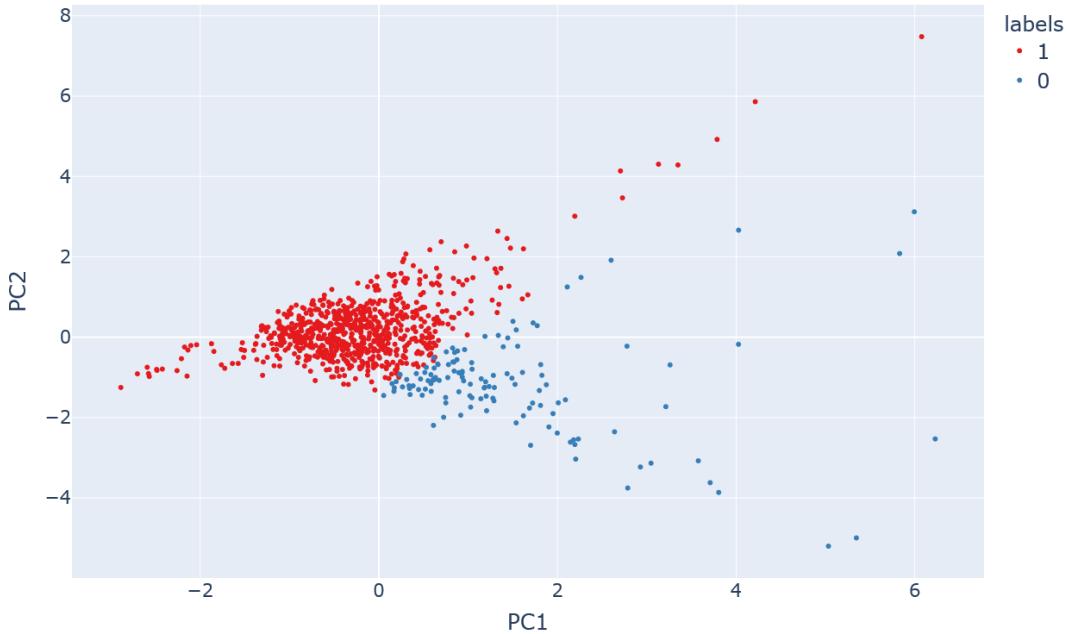


FIGURA 5.10: Grafico PCA con attributi *Revenue*, *SalesMean*, *Tutor*, *ConsortiumCode*, *StartupYear* e parametro $K = 2$

	Min	Max	Media	Std
Cluster 1	$\approx 850K$	$\approx 4M$	$\approx 1.5M$	$\approx 580K$
Cluster 2	$\approx 3K$	$\approx 980K$	$\approx 450K$	$\approx 210K$

TABELLA 5.3: Distribuzione del parametro *Revenue* nel clustering con attributi *Revenue*, *SalesMean*, *Tutor*, *ConsortiumCode*, *StartupYear* e parametro $K = 2$

ottenuti. Trattandosi di un sistema di raccomandazione a feedback implicito esso ha bisogno di un numero elevato di record per l’allenamento del modello. Fortunatamente, nonostante il numero di farmacie nel cluster 2 sia esiguo, questo non comporta problemi particolari in questo senso. Essendo infatti questi punti vendita quelli con *Revenue* maggiore essi hanno naturalmente un numero di vendite più elevato: basti pensare come di tutti i record contenuti nel dataset degli acquisti 001_2019_RecSys_Dataset_v2.5.csv solo il 65% coinvolge una farmacia del primo cluster e ben il 35% una del secondo. Vengono riportati in tabella 5.4 i dati inerenti il solo algoritmo BPR, in quanto garantisce risultati migliori nel nostro sistema, come mostrato nello studio condotto nel capitolo 4.

	Unclustered	Cluster 1	Cluster 2
Sparsity	97.68%	93.56%	97.65%
Precision@5	76.45%	89.16%	75.58%
Precision@10	73.18%	87.63%	71.58%
MAP@5	71.16%	84.67%	69.25%
MAP@10	65.57%	81.73%	63.08%
NDCG@5	77.39%	88.92%	76.28%
NDCG@10	74.78%	87.93%	73.23%

TABELLA 5.4: Performance del Recommender System con algoritmo di allenamento BPR sui cluster prodotti con algoritmo K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 2$

Le performance del RS sul dataset ottenuto con i soli punti vendita nel cluster 1 mostrano un miglioramento del $\approx 12\%$, mentre quello ottenuto con i punti rimanenti un peggioramento marginale intorno al punto percentuale.

5.5 Clustering con metodo Agglomerative

Il terzo ed ultimo algoritmo di clustering considerato è quello gerarchico dell’Agglomerative Clustering. Anche questo algoritmo, come K-means, richiede l’inserimento del numero di cluster K desiderato da parte dell’utente. Per decidere il valore ottimale di K è possibile servirsi, oltre che del silouhette score, anche del dendrogramma. Per i clustering riportati in questo lavoro di tesi è stato utilizzato il metodo di linkage Ward e come metrica la distanza euclidea. La suddivisione che ha portato ai migliori risultati in termini di performance del Recommender System è stata quella che fa uso del dataset completo, ovvero con attributi *Revenue*, *SalesMean*, *Tutor*, *ConsortiumCode*, *StartupYear* e *Zone*, della quale è riportata la visualizzazione PCA in figura 5.11. Il numero di cluster selezionato è $K = 2$. Nonostante il silouhette score inerente la suddivisione in tre insiemi sia superiore, uno dei cluster prodotti è composto da soli 12 elementi, insufficienti per un buon allenamento del modello.

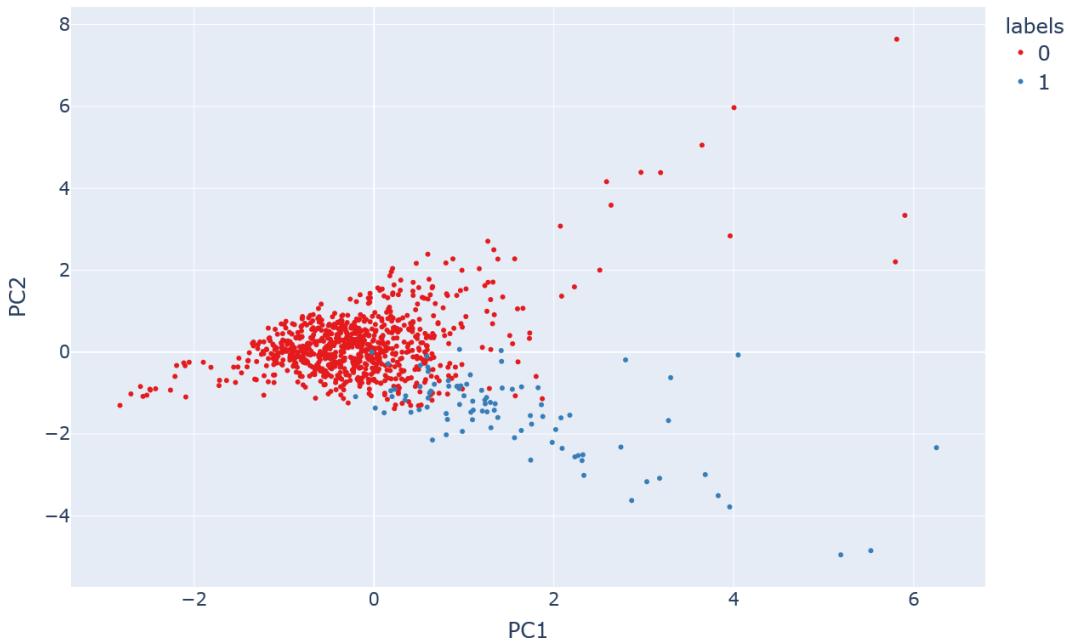


FIGURA 5.11: Grafico PCA con attributi *Revenue*, *SalesMean*, *Tutor*, *ConsortiumCode*, *StartupYear* e parametro $K = 2$

```

silhouette_score (numero di cluster: 2) = 0.2255096481679159
silhouette_score (numero di cluster: 3) = 0.24101659655505955
silhouette_score (numero di cluster: 4) = 0.11425113881311375
silhouette_score (numero di cluster: 5) = 0.10668108104836828

```

Il clustering produce due insiemi contenenti rispettivamente 90 e 803 farmacie. La maggior parte dei punti vengono distribuiti nei due cluster allo stesso modo di quanto accadeva col K-means, ed anche i risultati delle performance del Recommender System sono approssimativamente equivalenti. Questi vengono riportati in tabella 5.5.

	Unclustered	Cluster 1	Cluster 2
Sparsity	97.68%	93.56%	97.65%
Precision@5	76.45%	89.16%	75.58%
Precision@10	73.18%	87.63%	71.58%
MAP@5	71.16%	84.67%	69.25%
MAP@10	65.57%	81.73%	63.08%
NDCG@5	77.39%	88.92%	76.28%
NDCG@10	74.78%	87.93%	73.23%

TABELLA 5.5: Performance del Recommender System con algoritmo di allenamento BPR sui cluster prodotti con metodo agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 2$

5.6 Raccolta dei risultati

In questa sezione sono presentati alcuni dei test svolti nella fase sperimentale di questo lavoro di tesi. Per ogni combinazione di attributi che segue sono esposti:

- I risultati delle operazioni di clustering utilizzando gli algoritmi DBSCAN, K-means ed Agglomerative Clustering, inclusi i grafici t-SNE e PCA ed informazioni riguardanti la scelta del numero di cluster o i parametri dell'algoritmo (silhouette score, dendrogramma, elbow method)
- Laddove sia stato ottenuto un numero di cluster adeguato, i risultati dei test delle performance del Recommender System per ognuno di loro (valore medio dei risultati di venti esecuzioni del programma di test)
- Osservazioni eventuali

5.6.1 Caso di studio 1: clustering con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean*

Questa combinazione di attributi rappresenta il dataset nella sua interezza a seguito delle fasi di preprocessing e filtraggio tramite la matrice di correlazione. Sperimentalmente restituisce i migliori risultati in termini di performance del Recommender System.

- **DBSCAN:** questo algoritmo di clustering risulta inefficace per il nostro dataset, infatti l'algoritmo rileva al più pochi insiemi di punti di modeste dimensioni circondati da punti di rumore. Gli elementi all'interno di uno stesso cluster hanno semplicemente gli stessi valori degli attributi categorici, dunque questa suddivisione non porta ad alcuna informazione utile. Vengono presentati due casi con diversi valori di eps : nelle figure 5.13 e 5.14 i parametri sono settati a $\text{minPts} = 10$ e $\text{eps} = 1.85$ e l'algoritmo identifica la grande maggioranza dei punti come appartenente ad un unico cluster; nelle figure 5.15 e 5.16 i parametri sono settati a $\text{minPts} = 10$ e $\text{eps} = 1.25$ e vengono identificati i piccoli cluster descritti in precedenza.

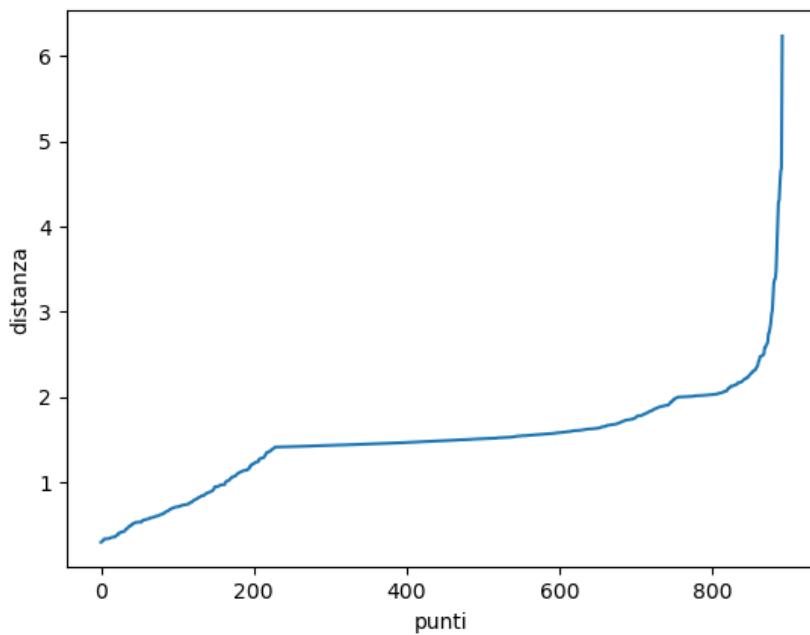


FIGURA 5.12: Elbow method con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $\text{minPts} = 10$

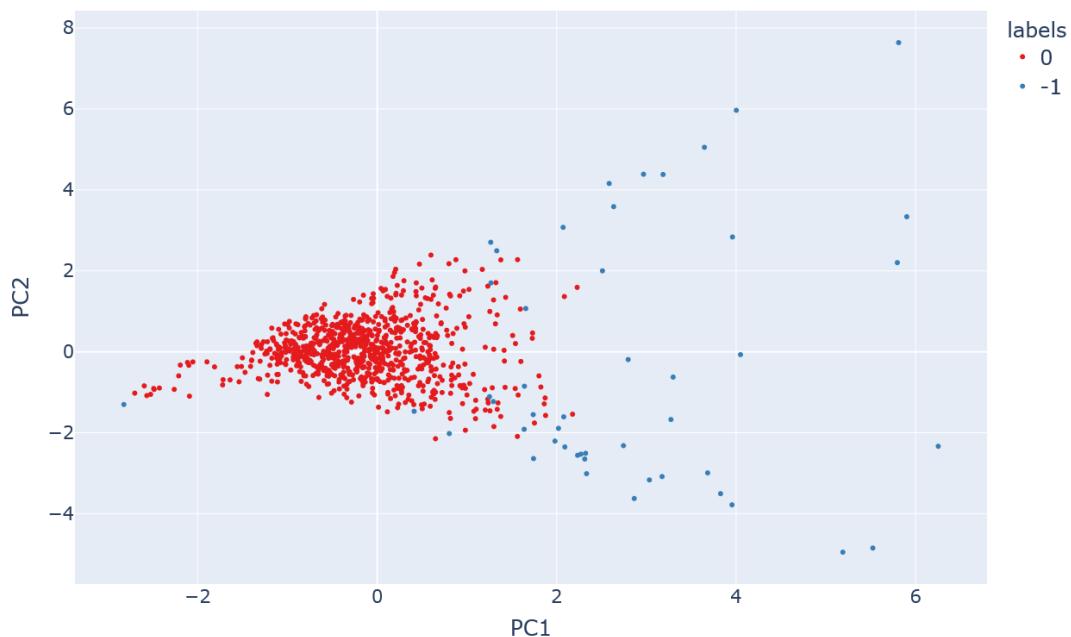


FIGURA 5.13: Grafico PCA con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametri $\text{minPts} = 10$, $\text{eps} = 1.85$

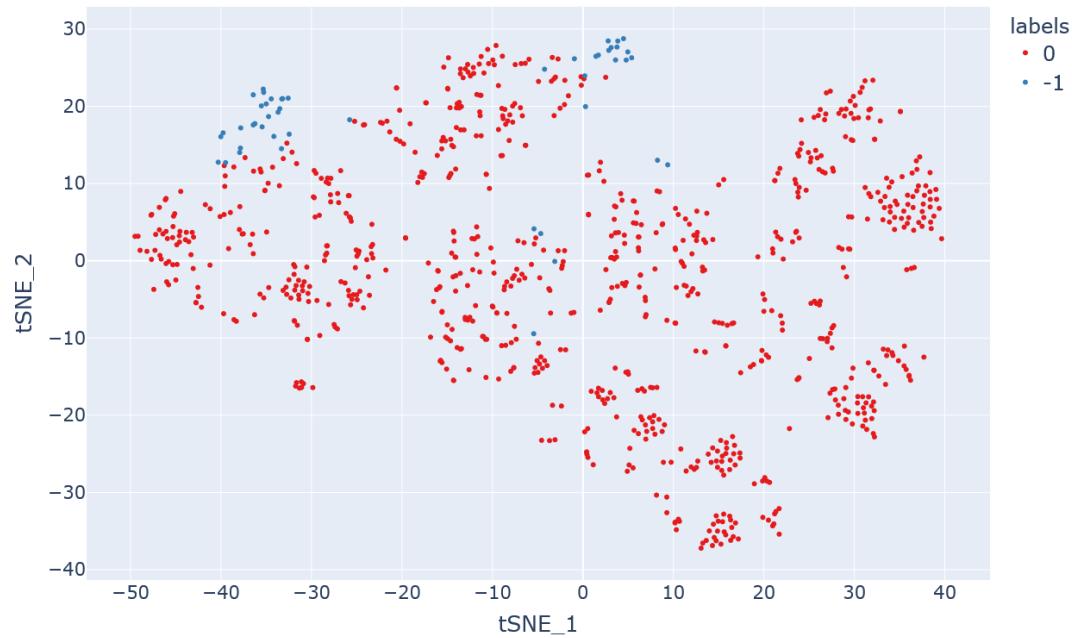


FIGURA 5.14: Grafico t-SNE con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametri $\text{minPts} = 10$, $\text{eps} = 1.85$

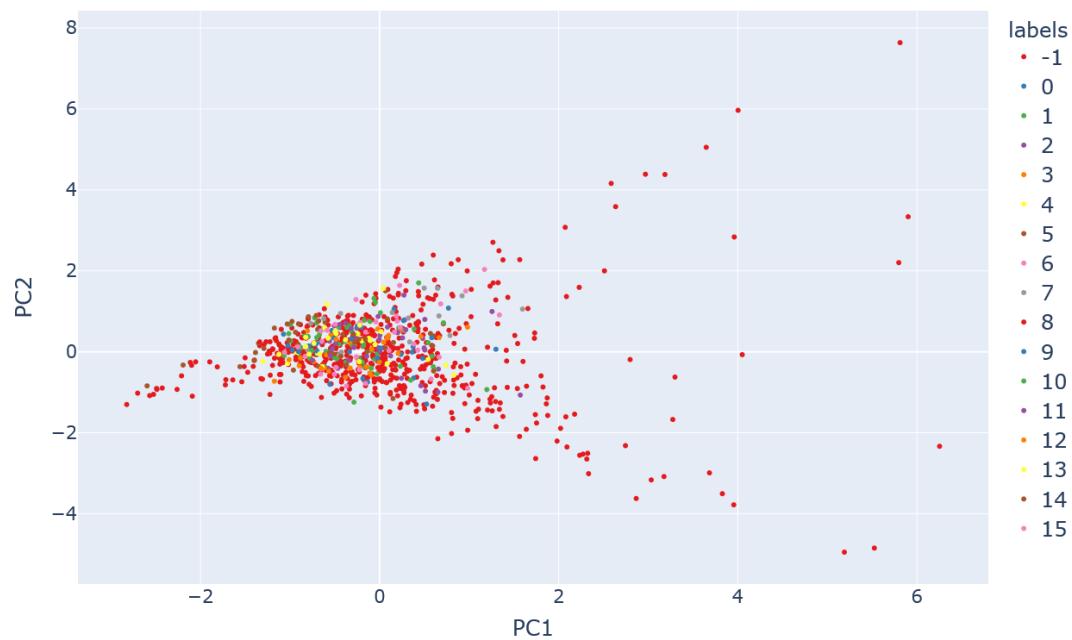


FIGURA 5.15: Grafico PCA con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametri $\text{minPts} = 10$, $\text{eps} = 1.25$

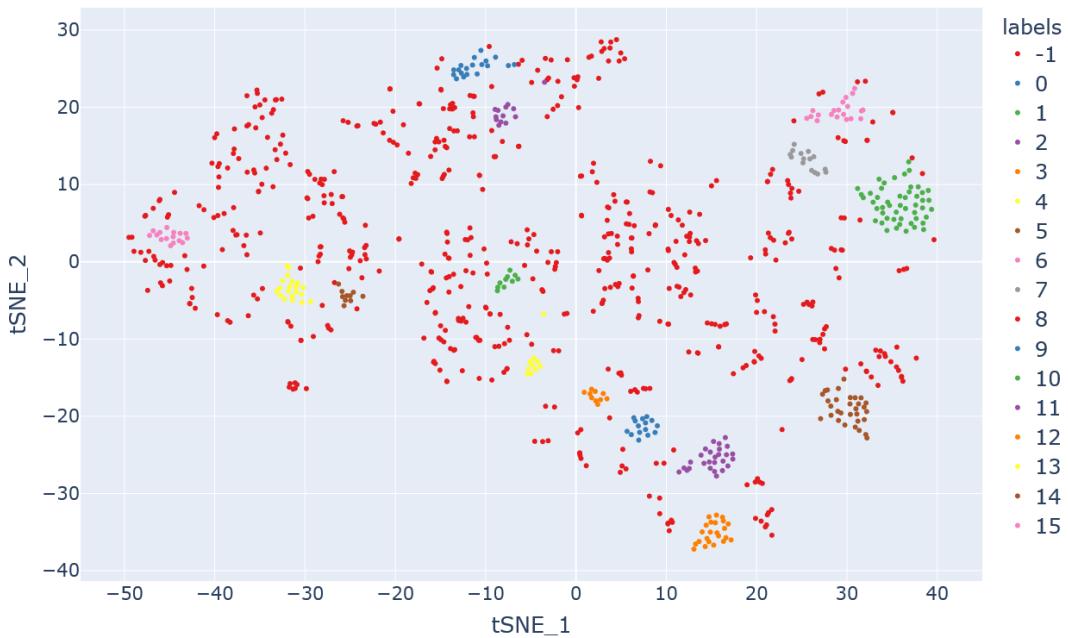


FIGURA 5.16: Grafico t-SNE con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametri $\text{minPts} = 10$, $\text{eps} = 1.25$

- **K-means:** questo algoritmo di clustering porta a risultati parzialmente positivi. Vengono presentati due casi con diversi valori di K : sono stati scelti quelli che garantiscono il valore maggiore del silouhette score, ovvero $K = 2$ e $K = 3$.

```

silhouette_score (numero di cluster: 2) = 0.21435988356822597
silhouette_score (numero di cluster: 3) = 0.20425394436821984
silhouette_score (numero di cluster: 4) = 0.12500894815086985
silhouette_score (numero di cluster: 5) = 0.1259772822294658

```

Nelle figure 5.17 e 5.18 vengono presentati i grafici inerenti il caso $K = 2$. Questa suddivisione, che produce due insiemi da 131 e 762 elementi, risulta essere la migliore trovata in termini di performance del Recommender System. Come mostrato nella sezione precedente la suddivisione si rivela essere effettuata principalmente sull'attributo *Revenue*: a meno di una piccola sovrapposizione il primo cluster contiene esclusivamente gli elementi con *Revenue* più alta.

Nelle figure 5.19 e 5.20 vengono presentati i grafici inerenti il caso $K = 3$. Questa suddivisione, che produce tre insiemi da 116, 700 e 77 elementi, vede un peggioramento delle performance per uno dei cluster trovati.

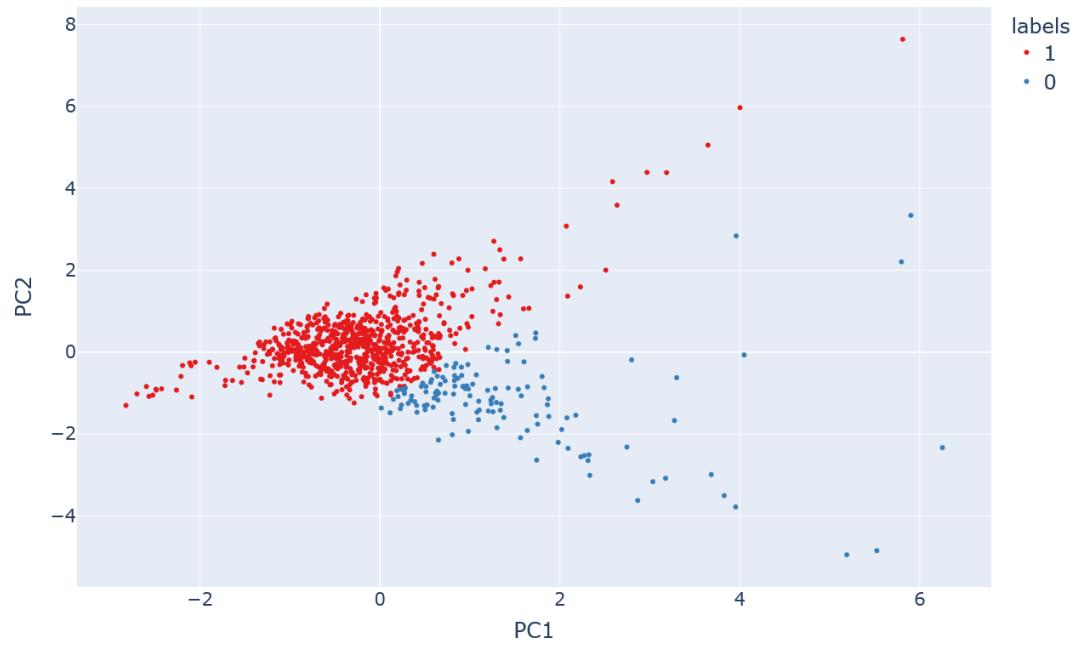


FIGURA 5.17: Grafico PCA del clustering K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 2$

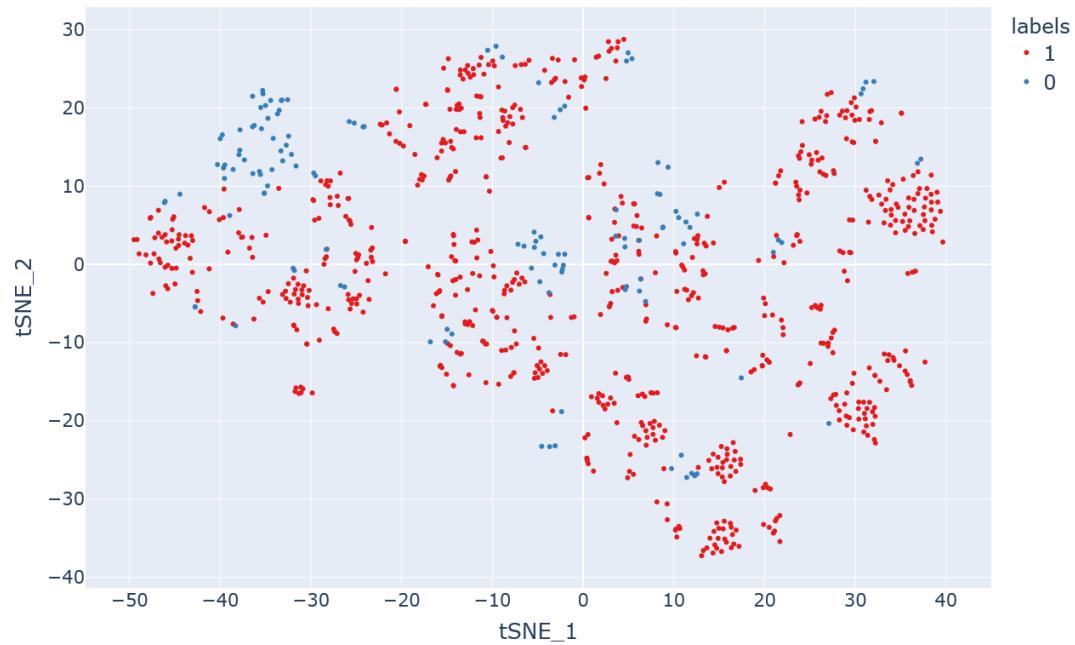


FIGURA 5.18: Grafico t-SNE del clustering K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 2$

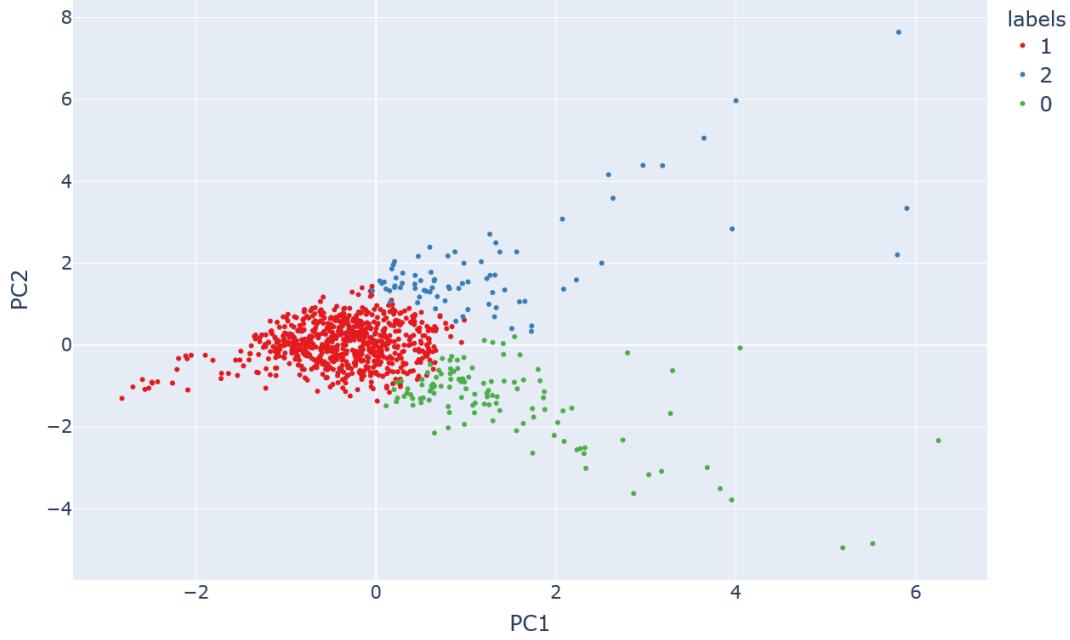


FIGURA 5.19: Grafico PCA del clustering K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 3$

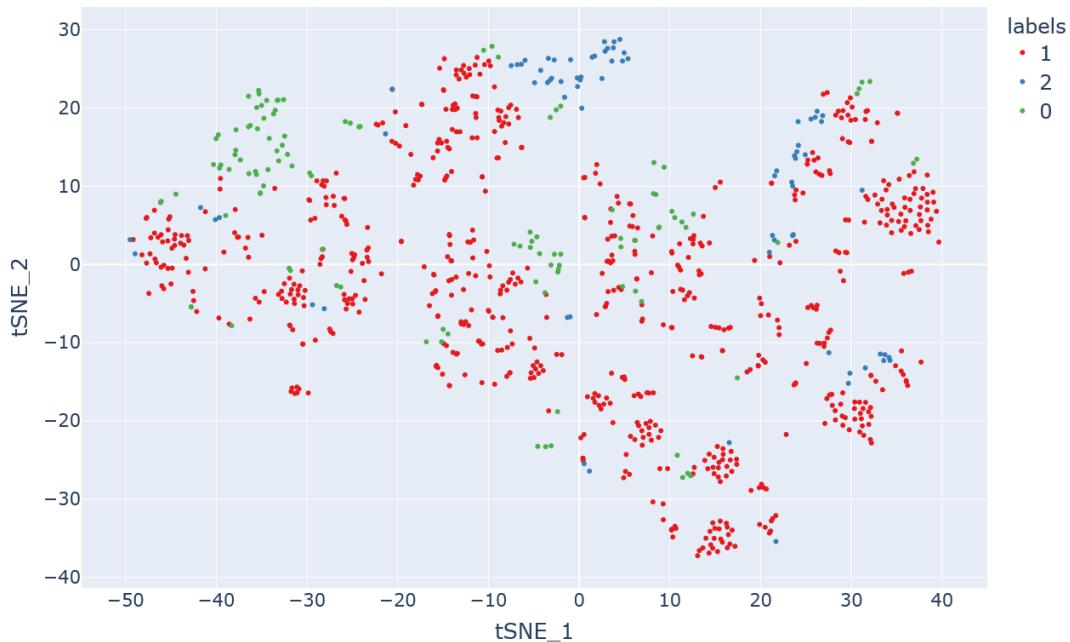


FIGURA 5.20: Grafico t-SNE del clustering K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 3$

	Unclustered	Cluster 1	Cluster 2
Sparsity	97.68%	93.56%	97.65%
Precision@5	76.45%	89.16%	75.58%
Precision@10	73.18%	87.63%	71.58%
MAP@5	71.16%	84.67%	69.25%
MAP@10	65.57%	81.73%	63.08%
NDCG@5	77.39%	88.92%	76.28%
NDCG@10	74.78%	87.93%	73.23%

TABELLA 5.6: Performance del Recommender System sui cluster prodotti con algoritmo K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 2$

	Unclustered	Cluster 1	Cluster 2	Cluster 3
Sparsity	97.68%	93.1%	97.52%	94.34%
Precision@5	76.45%	88.96%	75.42%	56.81%
Precision@10	73.18%	86.64%	72.82%	55.77%
MAP@5	71.16%	84.97%	69.08%	49.46%
MAP@10	65.57%	80.95%	64.27%	44.29%
NDCG@5	77.39%	83.63%	76.04%	57.44%
NDCG@10	74.78%	87.76%	74.04%	56.34%

TABELLA 5.7: Performance del Recommender System sui cluster prodotti con algoritmo K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 3$

- **Agglomerative clustering:** questo algoritmo di clustering porta a risultati molto simili a quelli del K-means, sia dal punto di vista del clustering vero e proprio sia, conseguentemente, delle performance del Recommender System. Il silhouette score suggerisce di prendere in considerazione i valori $K = 2$ e $K = 3$.

```
silhouette_score (numero di cluster: 2) = 0.2255096481679159
silhouette_score (numero di cluster: 3) = 0.24101659655505955
silhouette_score (numero di cluster: 4) = 0.11425113881311375
silhouette_score (numero di cluster: 5) = 0.10668108104836828
```

Nelle figure 5.22 e 5.23 vengono presentati i grafici inerenti il caso $K = 2$. Questa suddivisione, che produce due insiemi da 90 ed 803 elementi, porta a risultati in termini di performance del Recommender System simili a quelli ottenuti con l'algoritmo K-means.

	Unclustered	Cluster 1	Cluster 2
Sparsity	97.68%	92.29%	97.66%
Precision@5	76.45%	88.56%	75.67%
Precision@10	73.18%	87.44%	71.84%
MAP@5	71.16%	84.84%	69.56%
MAP@10	65.57%	81.84%	63.72%
NDCG@5	77.39%	88.5%	76.06%
NDCG@10	74.78%	87.75%	73.27%

TABELLA 5.8: Performance del Recommender System sui cluster prodotti con algoritmo agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 2$

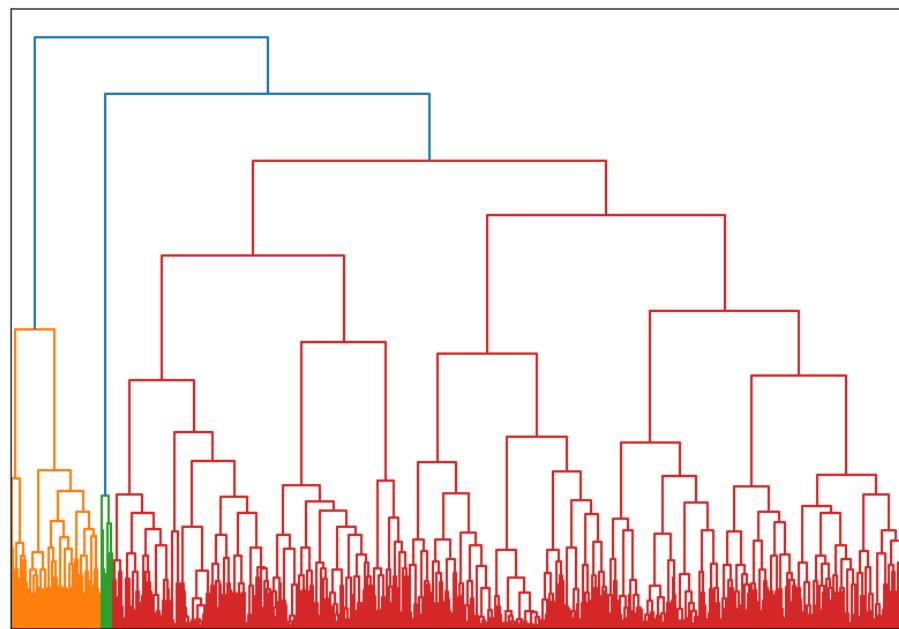


FIGURA 5.21: Dendrogramma del clustering agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean*

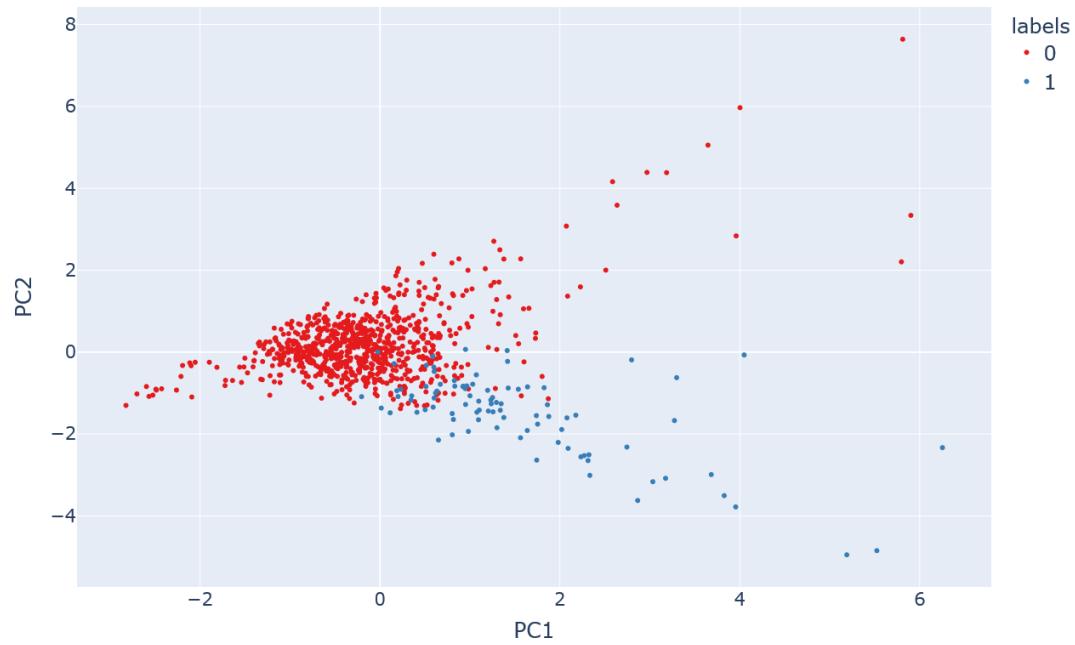


FIGURA 5.22: Grafico PCA del clustering agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 2$

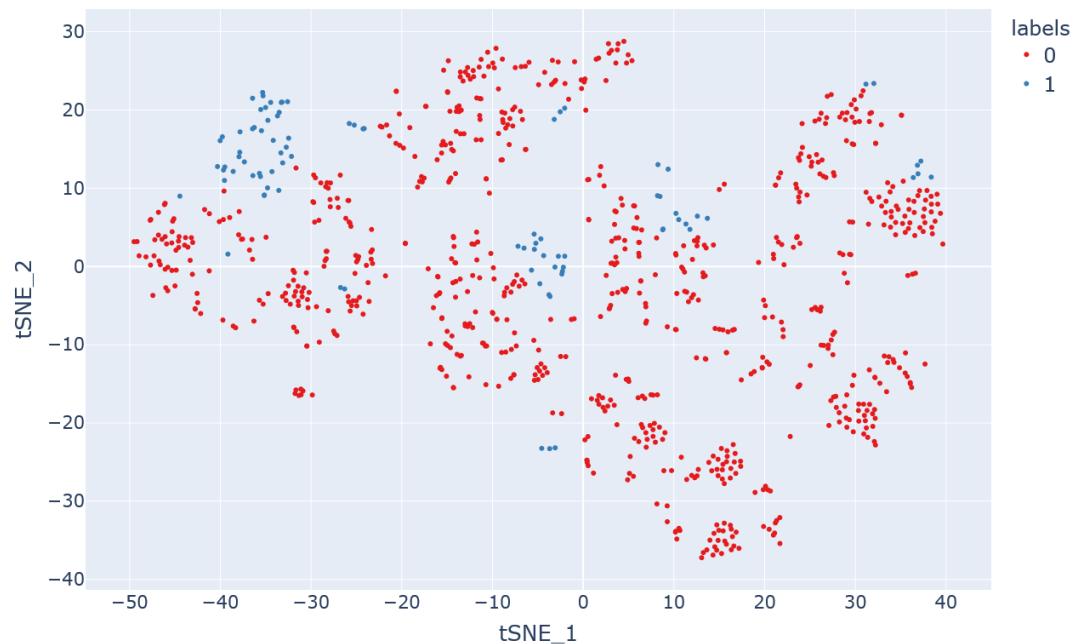


FIGURA 5.23: Grafico t-SNE del clustering agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *StartupYear*, *Revenue*, *SalesMean* e parametro $K = 2$

Nonostante sia corrispondente al valore del silouhette score più alto, il clustering con $K = 3$ produce tre insiemi da 791, 90 e 12 elementi. Il terzo cluster ha visibilmente un numero di elementi insufficiente per l'allenamento del modello, fatto comprovato anche dai risultati dei test delle performance del Recommender System, secondo i quali il valore della *Precision@5* di questo insieme si aggira intorno al 30%. Per questa ragione i grafici relativi a questa configurazione, e ad ognuna con $K \geq 3$, vengono tralasciati.

5.6.2 Caso di studio 2: clustering con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue*

Una volta ottenuti i risultati del dataset completo si è passati alla valutazione di varie combinazioni degli attributi, per appurare se una di queste potesse portare ad un ulteriore miglioramento delle prestazioni del Recommender System. Segue una di queste prove, nella quale sono stati rimossi gli attributi *StartupYear* e *SalesMean*. I risultati ottenuti avvalorano la tesi secondo la quale l'attributo *Revenue* è il più caratterizzante i dati per fini di clustering, in quanto la rimozione di altri attributi sia categorici che numerici non porta ad un deterioramento significativo delle prestazioni.

- **DBSCAN:** questo algoritmo di clustering risulta inefficace per il nostro dataset, infatti l'algoritmo rileva insiemi di modeste dimensioni circondati da punti di rumore. Gli elementi all'interno di uno stesso cluster hanno semplicemente gli stessi valori degli attributi categorici, dunque questa suddivisione non porta ad alcuna informazione utile. Nelle figure 5.25 e 5.26 vengono mostrate le visualizzazioni PCA e t-SNE con i parametri di DBSCAN settati a $minPts = 10$ e $eps = 1.2$.

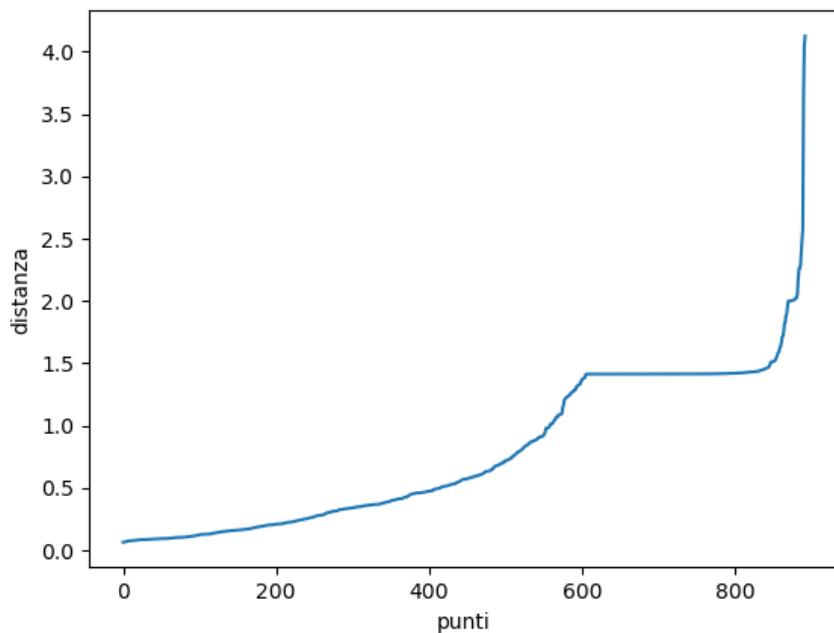


FIGURA 5.24: Elbow method con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $\text{minPts} = 10$



FIGURA 5.25: Grafico PCA con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametri $\text{minPts} = 10$, $\text{eps} = 1.2$

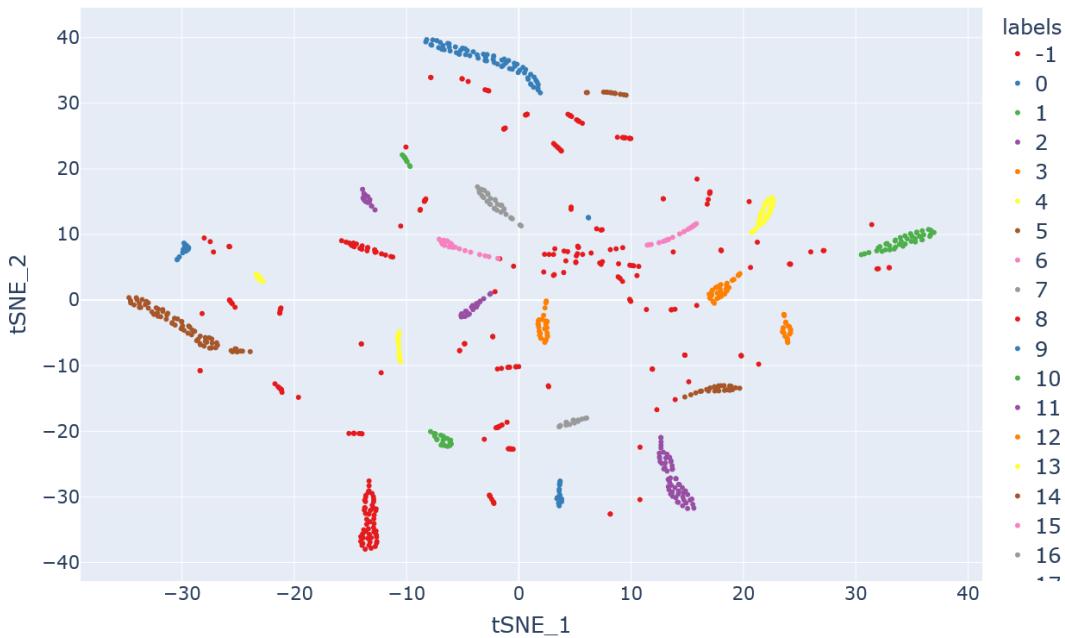


FIGURA 5.26: Grafico t-SNE con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametri $\text{minPts} = 10$, $\text{eps} = 1.2$

- **K-means:** questo algoritmo di clustering porta a risultati solo leggermente inferiori a quelli ottenuti utilizzando il dataset completo. Vengono presentati due casi con diversi valori di K : sono stati scelti quelli che garantiscono il valore maggiore del silouhette score, ovvero $K = 2$ e $K = 4$.

```

silhouette_score (numero di cluster: 2) = 0.28580606014372123
silhouette_score (numero di cluster: 3) = 0.18030311291055567
silhouette_score (numero di cluster: 4) = 0.22426475266731566
silhouette_score (numero di cluster: 5) = 0.21460850044558727

```

Nelle figure 5.27 e 5.28 vengono presentati i grafici inerenti il caso $K = 2$. Questa suddivisione produce due insiemi da 123 e 770 elementi.

	Unclustered	Cluster 1	Cluster 2
Sparsity	97.68%	93.33%	97.65%
Precision@5	76.45%	85.85%	75.53%
Precision@10	73.18%	84.15%	71.01%
MAP@5	71.16%	81.74%	69.97%
MAP@10	65.57%	77.72%	63.24%
NDCG@5	77.39%	87.02%	76.53%
NDCG@10	74.78%	85.5%	73.05%

TABELLA 5.9: Performance del Recommender System sui cluster prodotti con algoritmo K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $K = 2$

Nelle figure 5.29 e 5.30 vengono presentati i grafici inerenti il caso $K = 4$. Questa suddivisione, che produce quattro insiemi da 68, 402, 225 e 198 elementi, vede un peggioramento delle performance per uno dei cluster trovati (Cluster 4).

	Unclustered	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Sparsity	97.68%	91%	96.81%	95.74%	95.67%
Precision@5	76.45%	89.12%	77.41%	75.2%	66.16%
Precision@10	73.18%	83.82%	75.04%	73.11%	63.43%
MAP@5	71.16%	85.86%	70.62%	69.69%	57.99%
MAP@10	65.57%	78.59%	66.53%	65.48%	52.59%
NDCG@5	77.39%	90.28%	77.57%	75.82%	67.33%
NDCG@10	74.78%	86.23%	75.91%	74.17%	64.93%

TABELLA 5.10: Performance del Recommender System sui cluster prodotti con algoritmo K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $K = 4$

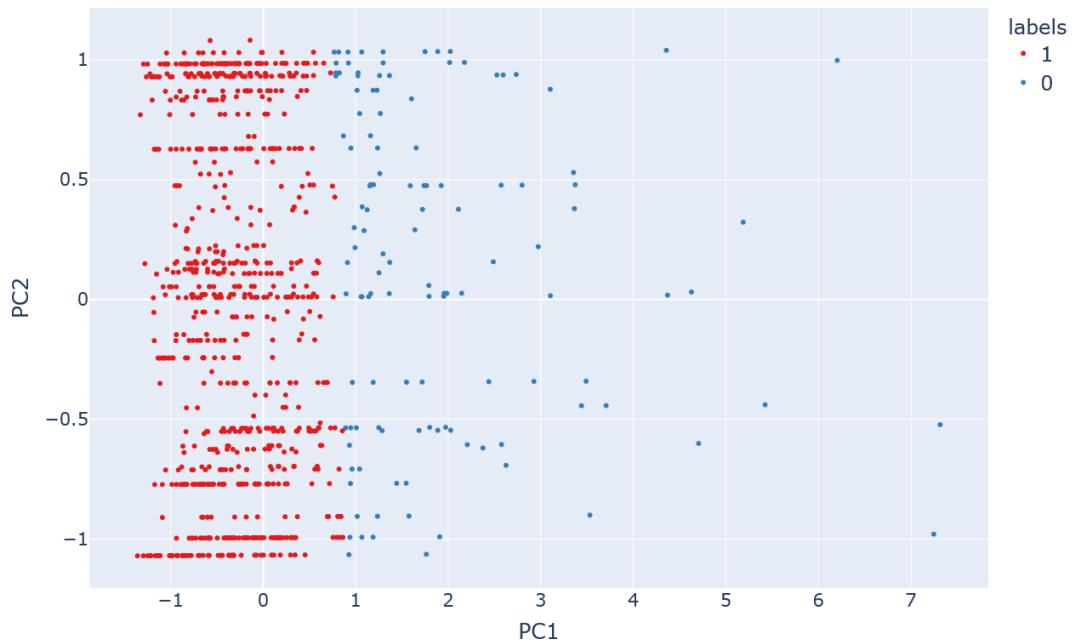


FIGURA 5.27: Grafico PCA del clustering K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $K = 2$

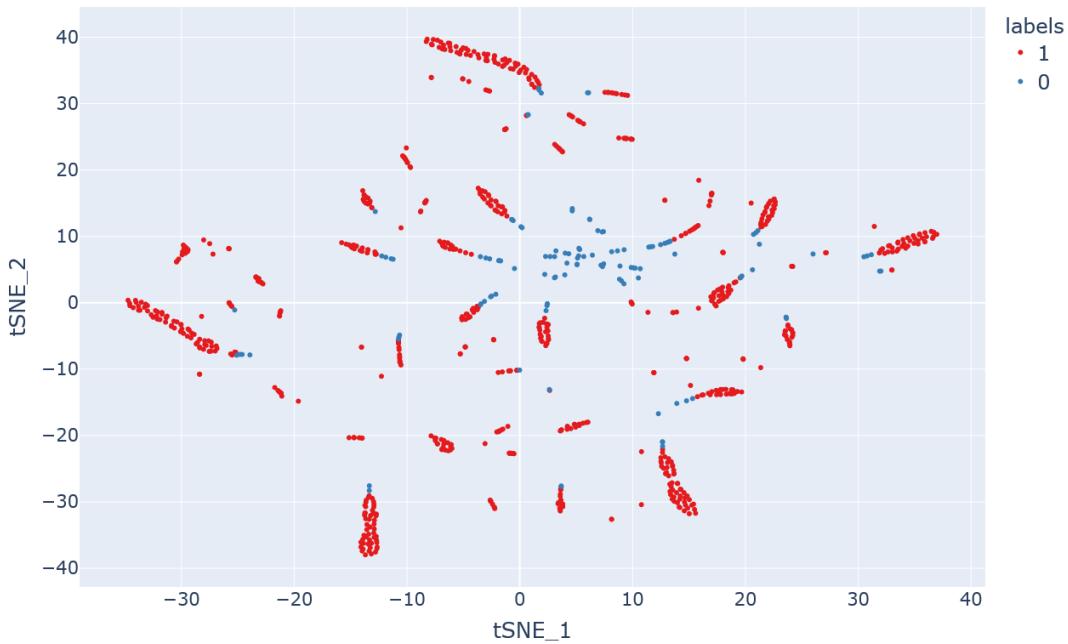


FIGURA 5.28: Grafico t-SNE del clustering K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $K = 2$



FIGURA 5.29: Grafico PCA del clustering K-means con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $K = 4$

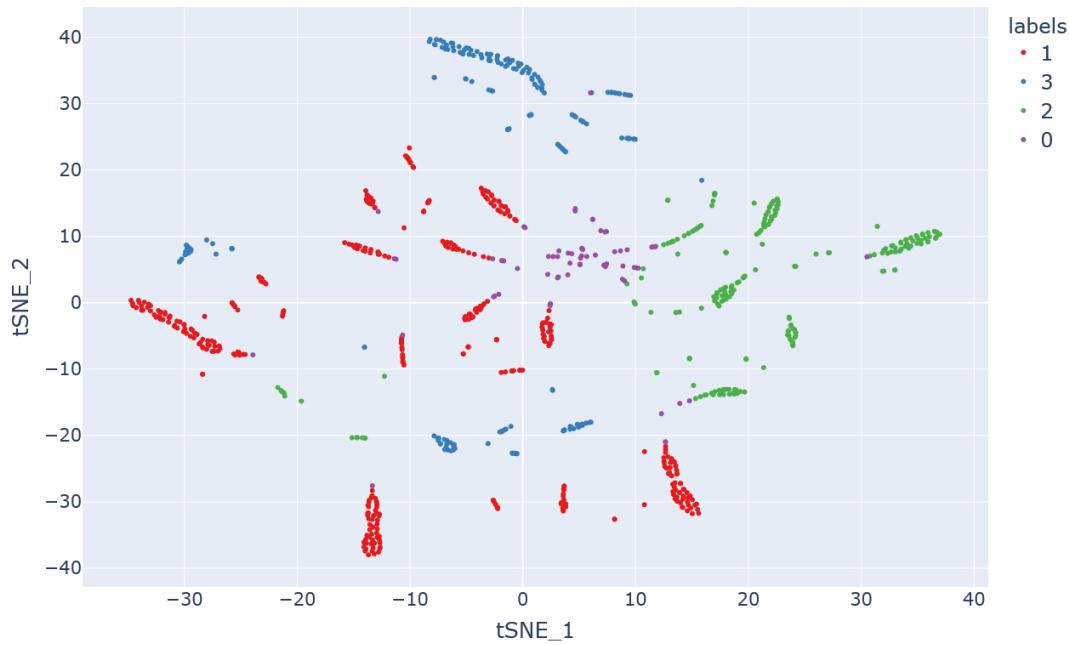


FIGURA 5.30: Grafico t-SNE del clustering K-means con attributi *Tutor, ConsortiumCode, Zone, Revenue* e parametro $K = 4$

- **Agglomerative clustering:** questo algoritmo di clustering porta a risultati lievemente inferiori rispetto a quelli del K-means. Vengono presentati due casi con diversi valori di K : sono stati scelti quelli che garantiscono il valore maggiore del silouhette score, ovvero $K = 2$ e $K = 4$.

```
silhouette_score (numero di cluster: 2) = 0.31064339810874597
silhouette_score (numero di cluster: 3) = 0.17132494783423288
silhouette_score (numero di cluster: 4) = 0.21010602285737556
silhouette_score (numero di cluster: 5) = 0.1901967615635958
```

Nelle figure 5.32 e 5.33 vengono presentati i grafici inerenti il caso $K = 2$. Questa suddivisione, che produce due insiemi da 88 ed 805 elementi, porta a risultati in termini di performance del Recommender System lievemente inferiori rispetto a quelli ottenuti con l'algoritmo K-means.

	Unclustered	Cluster 1	Cluster 2
Sparsity	97.68%	92.12%	97.66%
Precision@5	76.45%	82.95%	76.57%
Precision@10	73.18%	81.81%	72.71%
MAP@5	71.16%	76.87%	71.2%
MAP@10	65.57%	73.66%	65.06%
NDCG@5	77.39%	82.82%	77.66%
NDCG@10	74.78%	82.06%	74.55%

TABELLA 5.11: Performance del Recommender System sui cluster prodotti con algoritmo agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone Revenue* e parametro $K = 2$

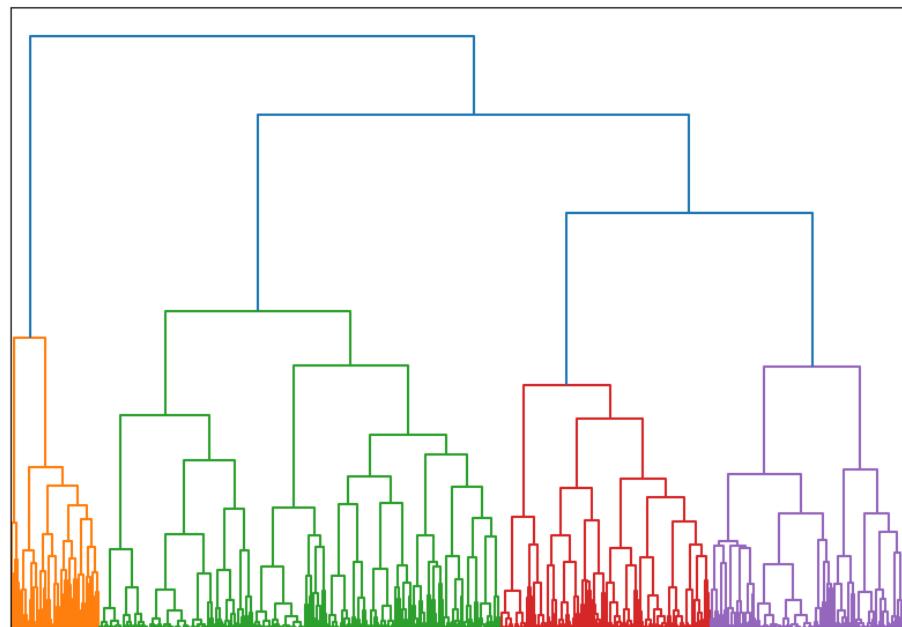


FIGURA 5.31: Dendrogramma del clustering agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue*



FIGURA 5.32: Grafico PCA del clustering agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $K = 2$

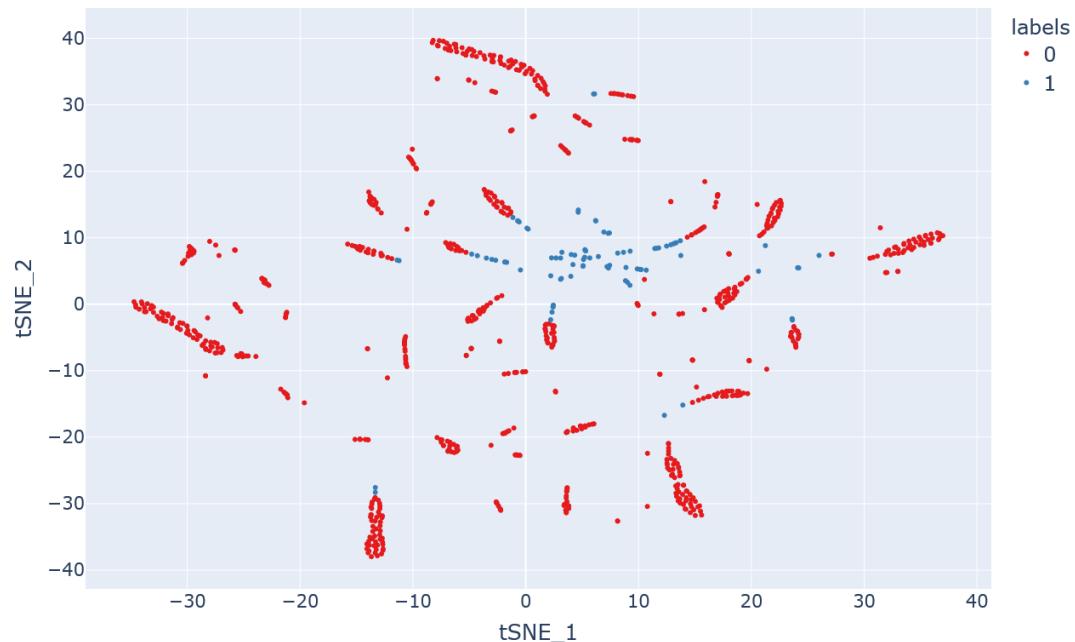


FIGURA 5.33: Grafico t-SNE del clustering agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $K = 2$

Nelle figure 5.34 e 5.35 vengono presentati i grafici inerenti il caso $K = 4$. Questa suddivisione, che produce quattro insiemi da 88, 401, 195, 209 elementi, porta a risultati in termini di performance del Recommender System lievemente inferiori rispetto a quelli ottenuti con l'algoritmo K-means.

	Unclustered	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Sparsity	97.68%	92.12%	96.84%	95.66%	95.65%
Precision@5	76.45%	83.64%	79.41%	69.23%	73.5%
Precision@10	73.18%	81.7%	75.56%	63.57%	69.38%
MAP@5	71.16%	76.95%	73.48%	62.13%	68.05%
MAP@10	65.57%	73.25%	67.54%	53.41%	61.86%
NDCG@5	77.39%	84.12%	80%	70.15%	74.76%
NDCG@10	74.78%	82.57%	77%	65.83%	71.44%

TABELLA 5.12: Performance del Recommender System sui cluster prodotti con algoritmo agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone Revenue* e parametro $K = 4$



FIGURA 5.34: Grafico PCA del clustering agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $K = 4$

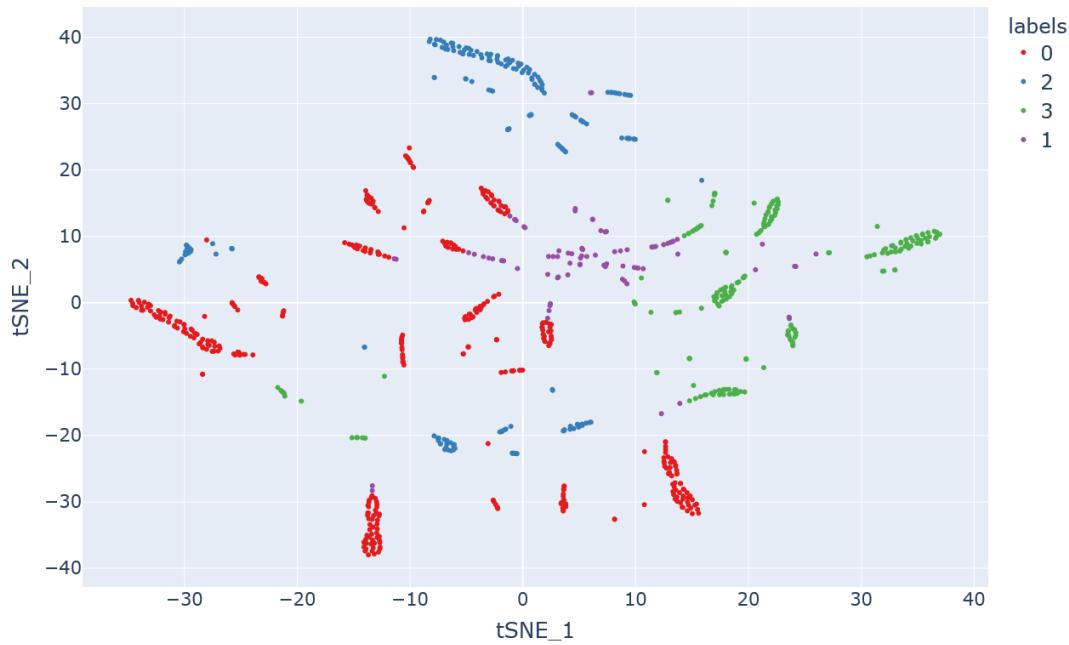


FIGURA 5.35: Grafico t-SNE del clustering agglomerativo con attributi *Tutor*, *ConsortiumCode*, *Zone*, *Revenue* e parametro $K = 4$

5.6.3 Caso di studio 3: clustering con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean*

Dai test precedenti risulta evidente come l'attributo *Revenue* sia il più significativo per il clustering. Per avvalorare questa ipotesi sono state condotte ulteriori prove, utilizzando varie combinazioni di attributi che non prevedessero l'utilizzo di tale attributo. Come ci aspettavamo i risultati delle performance del Recommender System ottenuti si sono rivelati peggiori rispetto alle prove precedenti. Nell'esempio che segue sono stati utilizzati gli attributi *Tutor*, *Zone*, *StartupYear* e *SalesMean*.

- **DBSCAN:** questo algoritmo di clustering risulta nuovamente inefficiente, per le stesse ragioni delle prove precedenti. Nelle figure 5.37 e 5.38 vengono mostrate le visualizzazioni PCA e t-SNE con i parametri di DBSCAN settati a $minPts = 8$ e $eps = 1.25$.

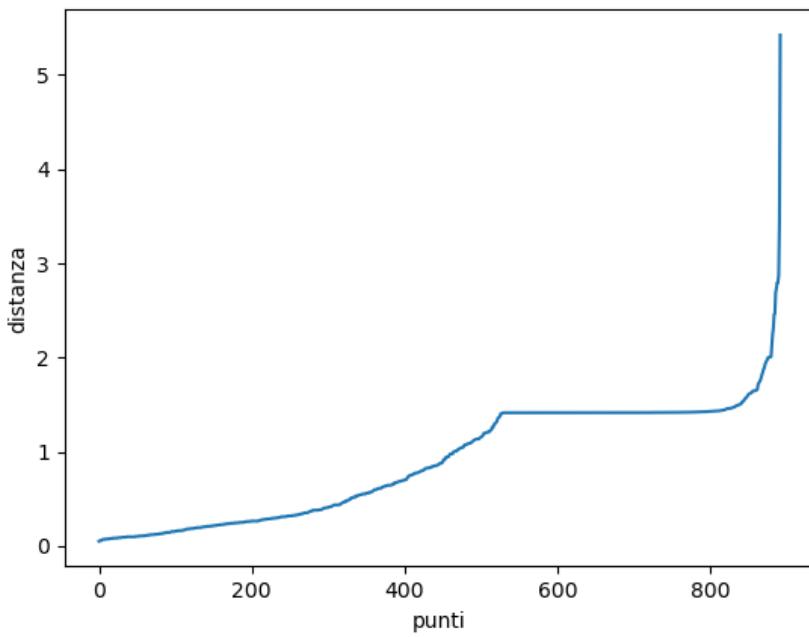


FIGURA 5.36: Elbow method con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $\text{minPts} = 10$

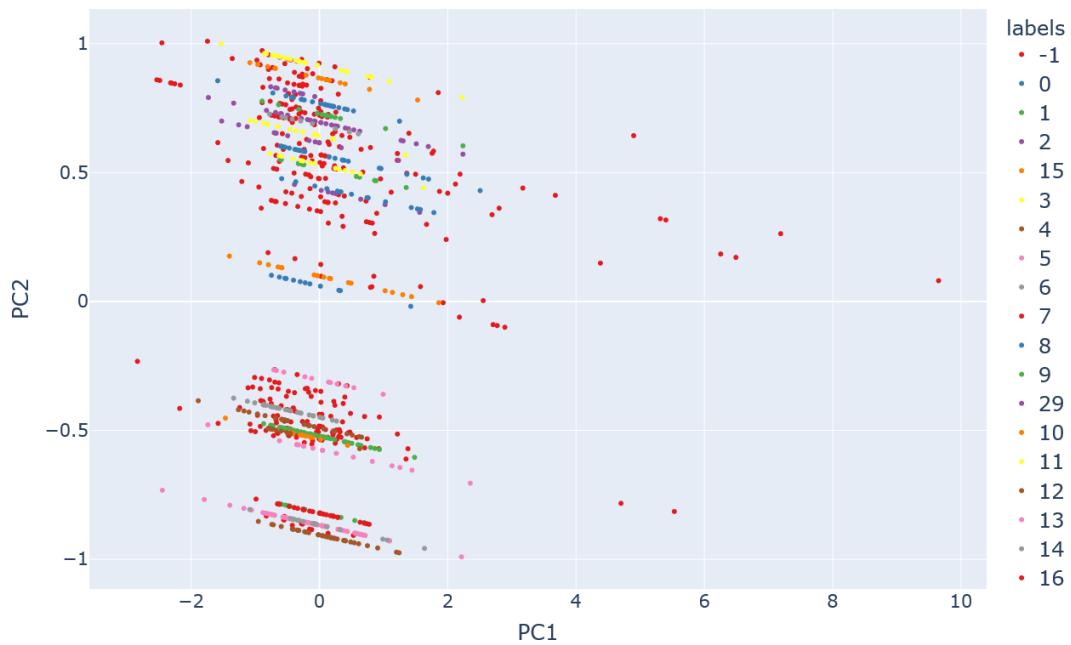


FIGURA 5.37: Grafico PCA con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametri $\text{minPts} = 10$, $\text{eps} = 1.25$

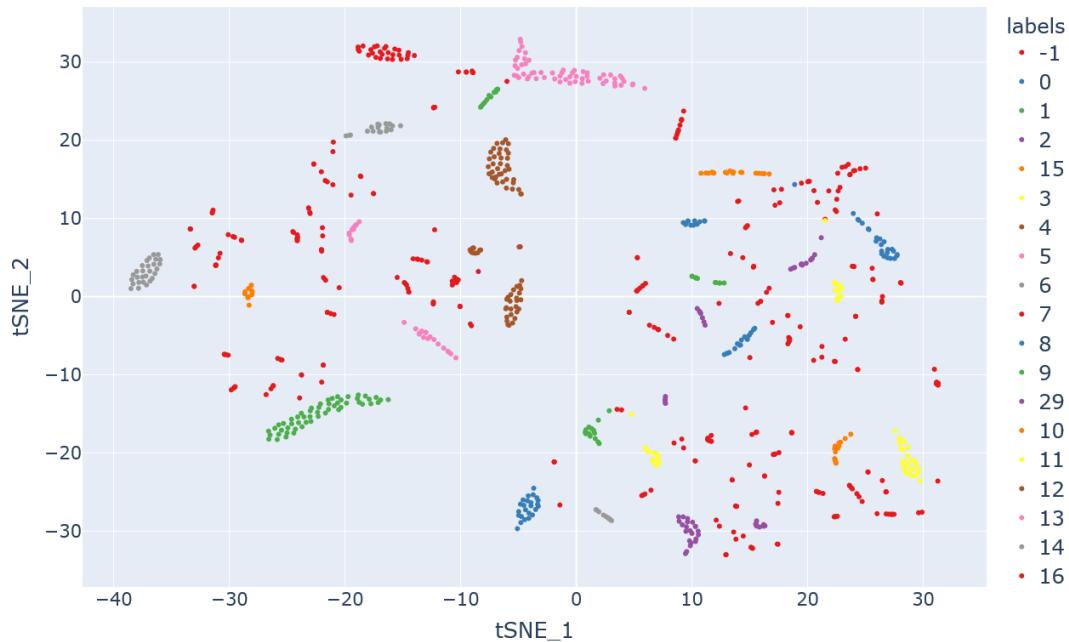


FIGURA 5.38: Grafico t-SNE con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametri $\text{minPts} = 10$, $\text{eps} = 1.25$

- **K-means:** questo algoritmo di clustering porta a risultati peggiori rispetto a quelli ottenuti utilizzando il dataset completo. Vengono presentati due casi con diversi valori di K : sono stati scelti quelli che garantiscono il valore maggiore del silouhette score, ovvero $K = 2$ e $K = 3$.

```

silhouette_score (numero di cluster: 2) = 0.3029787562495433
silhouette_score (numero di cluster: 3) = 0.20349013461995788
silhouette_score (numero di cluster: 4) = 0.16368357778575235
silhouette_score (numero di cluster: 5) = 0.18663976669458279

```

Nelle figure 5.39 e 5.40 vengono presentati i grafici inerenti il caso $K = 2$. Questa suddivisione produce due insiemi da 806 e 87 elementi.

	Dataset intero	Cluster 1	Cluster 2
Sparsity	97.68%	97.53%	94.58%
Precision@5	76.45%	75.69%	61.45%
Precision@10	73.18%	72.48%	57.33%
MAP@5	71.16%	70.15%	53.79%
MAP@10	65.57%	64.62%	47.53%
NDCG@5	77.39%	76.5%	62.04%
NDCG@10	74.78%	73.99%	58.88%

TABELLA 5.13: Performance del Recommender System sui cluster prodotti con algoritmo K-means con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 2$

Nelle figure 5.41 e 5.42 vengono presentati i grafici inerenti il caso $K = 3$. In questa suddivisione, che produce tre insiemi da 429, 394 e 70 elementi, le prestazioni rimangono costanti o hanno un peggioramento per tutti i cluster trovati.

	Dataset intero	Cluster 1	Cluster 2	Cluster 3
Sparsity	97.68%	96.75%	96.61%	93.93%
Precision@5	76.45%	77.11%	74.29%	56%
Precision@10	73.18%	73.85%	70.34%	51.86%
MAP@5	71.16%	71.34%	69.22%	44.94%
MAP@10	65.57%	65.93%	63.08%	40.41%
NDCG@5	77.39%	77.42%	75.26%	54.28%
NDCG@10	74.78%	74.99%	72.23%	52.60%

TABELLA 5.14: Performance del Recommender System sui cluster prodotti con algoritmo K-means con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 3$

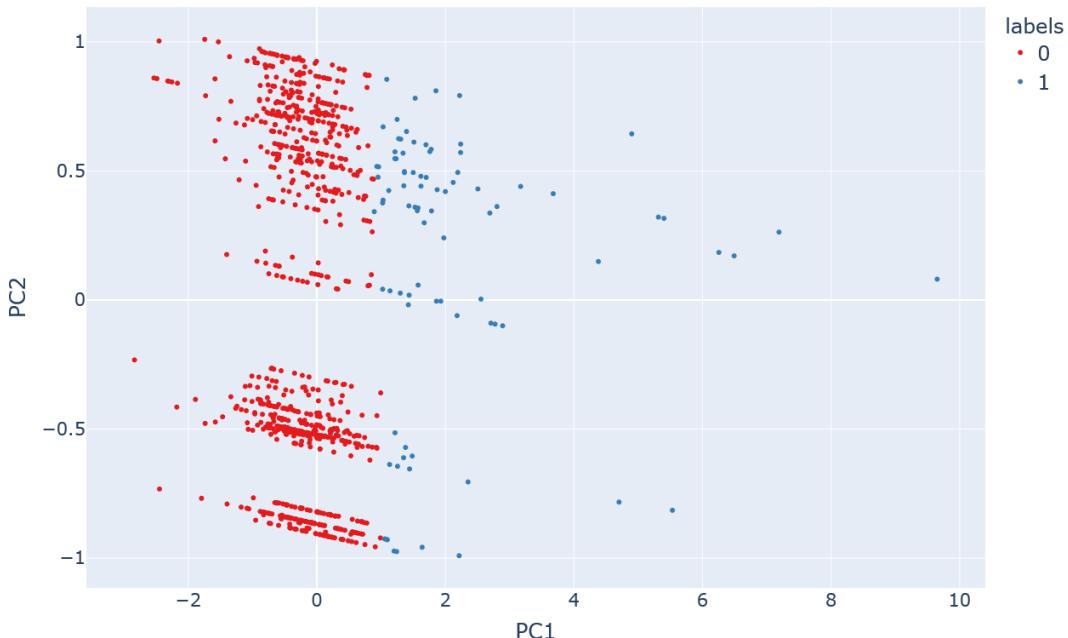


FIGURA 5.39: Grafico PCA del clustering K-means con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 2$

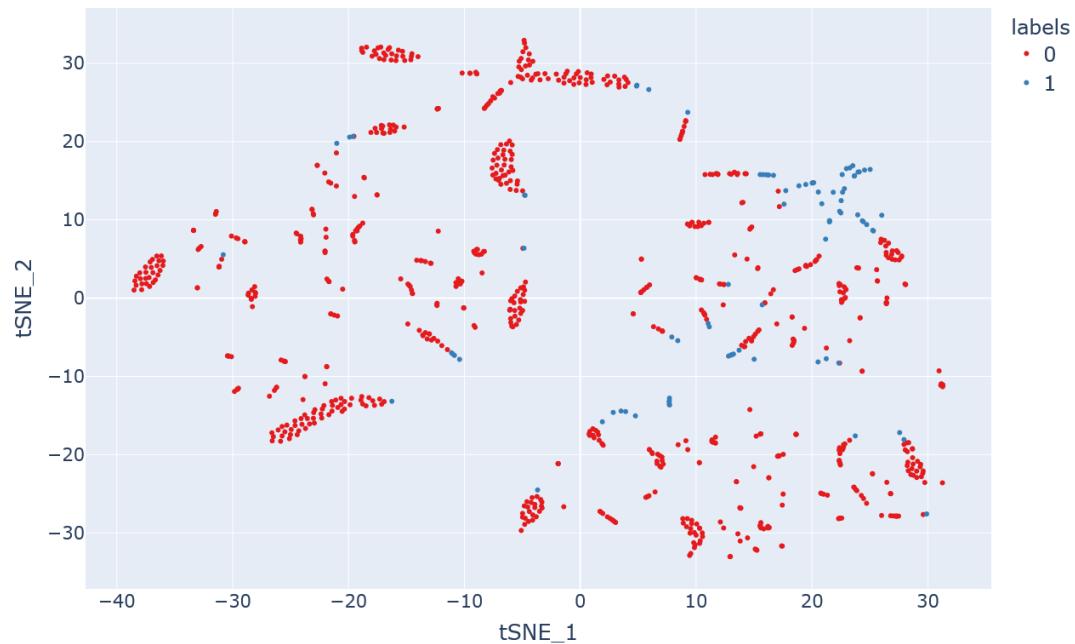


FIGURA 5.40: Grafico t-SNE del clustering K-means con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 2$

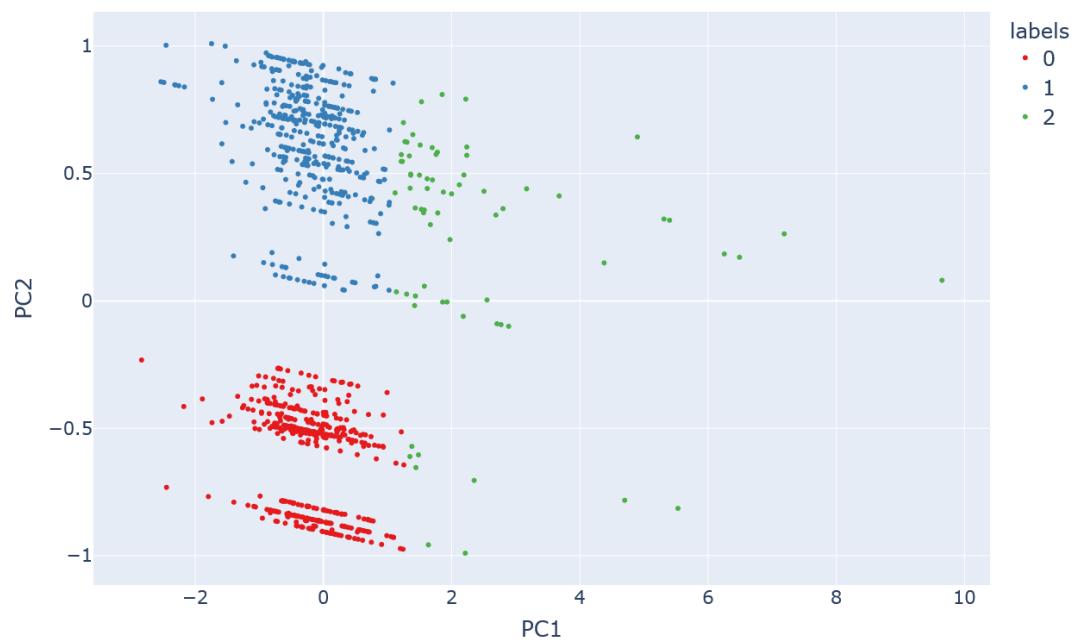


FIGURA 5.41: Grafico PCA del clustering K-means con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 3$

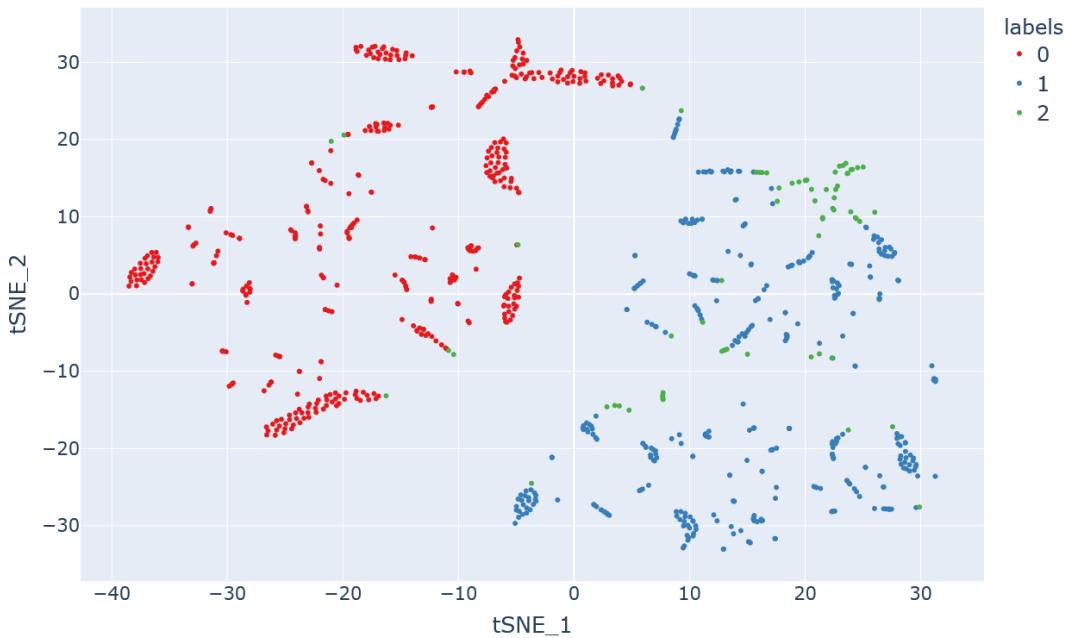


FIGURA 5.42: Grafico t-SNE del clustering K-means con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 3$

- **Agglomerative clustering:** anche utilizzando questo algoritmo i risultati non sono entusiasmanti. Vengono presentati due casi con diversi valori di K : sono stati scelti quelli che garantiscono il valore maggiore del silhouette score, ovvero $K = 2$ e $K = 4$.

```

silhouette_score (numero di cluster: 2) = 0.3124450965507309
silhouette_score (numero di cluster: 3) = 0.1562828854188454
silhouette_score (numero di cluster: 4) = 0.2096163334921243
silhouette_score (numero di cluster: 5) = 0.1796397263046389

```

Nelle figure 5.43 e 5.44 vengono presentati i grafici inerenti il caso $K = 2$. Questa suddivisione, che produce due insiemi da 817 e 76 elementi, porta a risultati in termini di performance del Recommender System peggiori rispetto all'esempio precedente.

Nelle figure 5.45 e 5.46 vengono presentati i grafici inerenti il caso $K = 4$. Questa suddivisione, che produce quattro insiemi da 76, 236, 425, 156 elementi, porta a risultati in termini di performance del Recommender System peggiori rispetto all'esempio precedente.

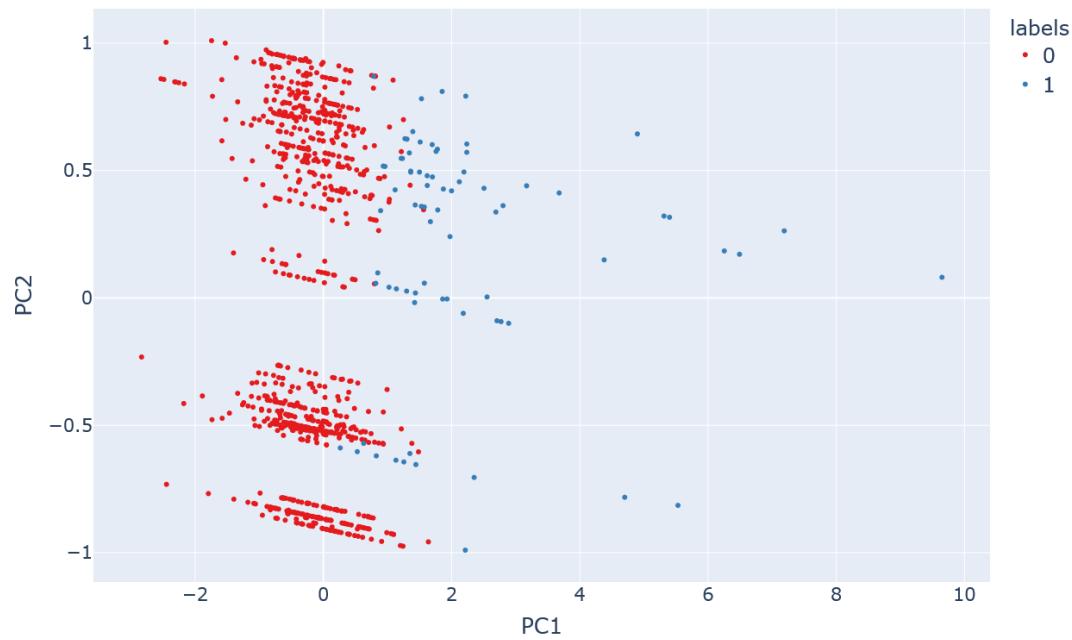


FIGURA 5.43: Grafico PCA del clustering agglomerativo con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 2$

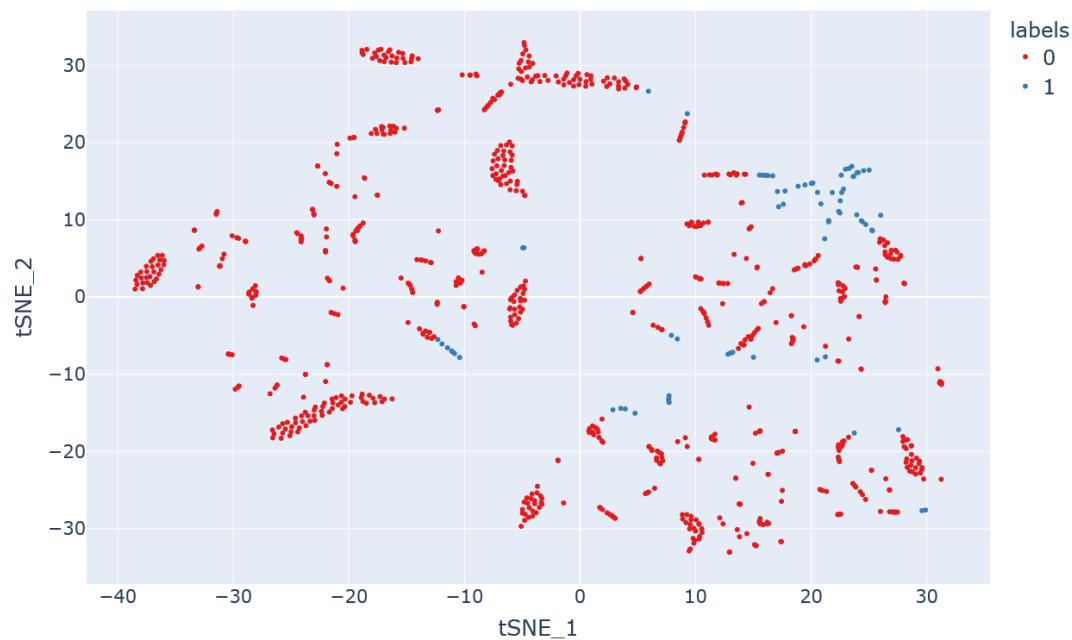


FIGURA 5.44: Grafico t-SNE del clustering agglomerativo con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 2$

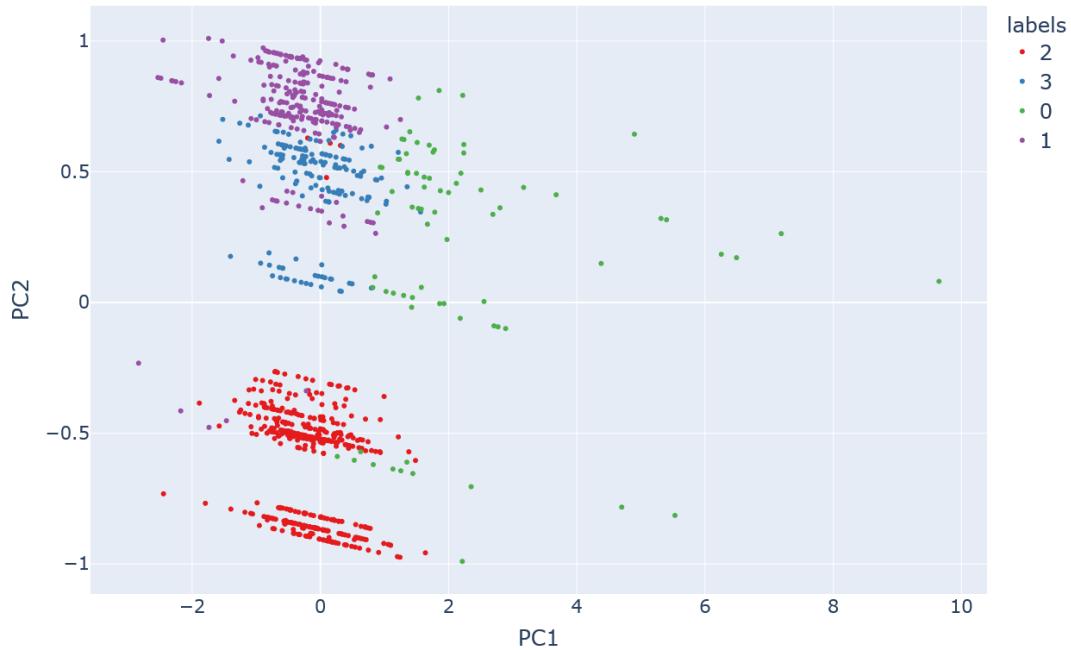


FIGURA 5.45: Grafico PCA del clustering agglomerativo con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 4$

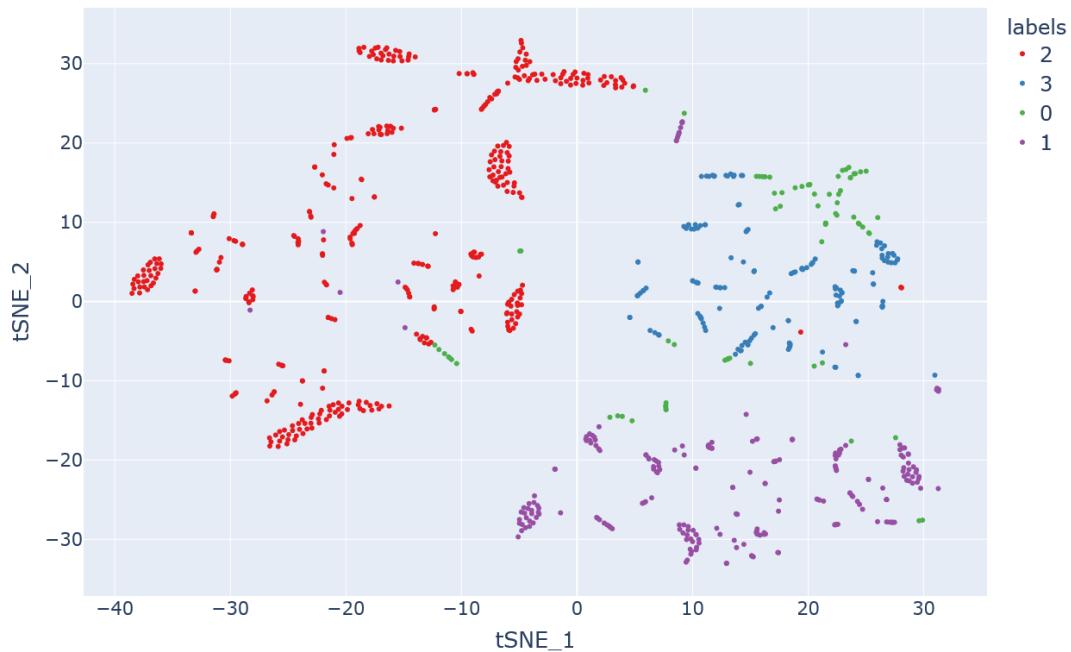


FIGURA 5.46: Grafico t-SNE del clustering agglomerativo con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 4$

	Dataset intero	Cluster 1	Cluster 2
Sparsity	97.68%	97.55%	94.06%
Precision@5	76.45%	77.23%	64.32%
Precision@10	73.18%	72.6%	58.79%
MAP@5	71.16%	73.25%	58.17%
MAP@10	65.57%	65.97%	50.21%
NDCG@5	77.39%	78.88%	65.97%
NDCG@10	74.78%	75.05%	61.59%

TABELLA 5.15: Performance del Recommender System sui cluster prodotti con algoritmo agglomerativo con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 2$

	Dataset intero	Cluster 1	Cluster 2	Cluster 3	Cluster 4
Sparsity	97.68%	94.06%	95.69%	96.74%	95.14%
Precision@5	76.45%	58%	73.85%	76.49%	66.21%
Precision@10	73.18%	56.23%	71.12%	74.51%	62.04%
MAP@5	71.16%	49.39%	68.3%	70.31%	58.16%
MAP@10	65.57%	44.89%	63.57%	66.17%	52.37%
NDCG@5	77.39%	59.12%	74.09%	77.01%	67.04%
NDCG@10	74.78%	57.47%	72%	75.43%	63.85%

TABELLA 5.16: Performance del Recommender System sui cluster prodotti con algoritmo agglomerativo con attributi *Tutor*, *Zone*, *StartupYear*, *SalesMean* e parametro $K = 4$

Capitolo 6

Conclusioni

L'elaborato ha permesso di approfondire la tematica dei Recommender System, sempre più rilevante in un mondo in cui l'e-commerce ha ormai preso il sopravvento sul commercio tradizionale, e dei principali algoritmi. Nonostante l'algoritmo ALS sia il più citato utilizzato in letteratura, esso non si è rivelato adatto per il nostro scopo, offrendo delle performance insufficienti. L'algoritmo BPR, basato sulla creazione di una classifica di oggetti piuttosto che sull'assegnazione di punteggi, ha invece offerto ottimi risultati con un valore di *Precision@5* del sistema superiore al 75%. Questa implementazione del RS potrà pertanto venire inserita all'interno del pacchetto, unione di varie componenti, che verrà fornito a Fidelity Salus per l'automazione di alcune operazioni di analisi dei dati. Questo potrà portare ad una notevole riduzione del tempo necessario per compierle, specialmente considerando che ad oggi molte di queste analisi sono condotte manualmente dai tutor dell'azienda.

La seconda parte di questo lavoro di tesi, ovvero l'applicazione di algoritmi di clustering di diverso genere atta al miglioramento delle performance del recommender system, si è invece rivelata inefficace. Le cause possono essere molteplici, in primis la conformazione dei dati inadatta alla suddivisione in cluster. Il numero di attributi del dataset delle farmacie è infatti limitato ad approssimativamente una dozzina ma, come mostrato nella matrice di correlazione, molte delle variabili numeriche sono fortemente correlate ed è quindi privo di senso utilizzarle contemporaneamente. Gran parte delle variabili nominali sono inoltre semplici ripetizioni di altre con una codifica differente o dotate di un numero eccessivo di possibili etichette. Di conseguenza la prima operazione da compiere sarebbe l'arricchimento del dataset con attributi che possano rivelarsi più significativi per le operazioni di clustering. Un esempio potrebbero essere statistiche basate sulla spesa nei punti vendita, come ad esempio la spesa media degli utenti nella farmacia (mentre al momento l'unico attributo che descrive l'andamento economico è la *Revenue*). Un'altra soluzione molto interessante da esplorare sarebbe quella di testare il funzionamento di un algoritmo di clustering con una funzione obiettivo personalizzata, che magari utilizzi come funzione obiettivo da massimizzare proprio il valore di *Precision@5* restituito come output in fase di test delle performance del Recommender System. Realisticamente questo approccio potrebbe non essere praticabile, considerato che

occorrerebbe richiamare il programma di test ad ogni passo dell'algoritmo di clustering rendendo il processo computazionalmente complesso. Ad ogni modo potrebbe essere una strada di ricerca interessante da percorrere, soprattutto alla luce del fatto che l'algoritmo di Agglomerative Clustering è predisposto proprio all'utilizzo di funzioni obiettivo personalizzate.

Con il dataset attualmente a disposizione K-means ed Agglomerative Clustering sono gli unici algoritmi di clustering che hanno portato ad un qualche miglioramento delle performance del Recommender System, limitato però solo ad alcuni dei cluster identificati. L'applicazione dei clustering presentati nelle sezioni 5.4 e 5.5 portano un miglioramento non indifferente delle performance per i punti vendita ad alto profitto, e può pertanto essere utilizzata nel pratico. È inoltre possibile utilizzare il suddetto modello specializzato per le farmacie ad alta revenue e quello generico, allenato sull'intero dataset, per tutte le altre. In questo modo il sistema clusterizzato non presenta mai prestazioni inferiori rispetto a quello generale, al costo di un lieve incremento del tempo necessario per l'allenamento.

Bibliografia

- [1] G. Adomavicius e A. Tuzhilin. «Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions». In: *IEEE Transactions on Knowledge and Data Engineering* 17.6 (2005), pp. 734–749. DOI: [10.1109/TKDE.2005.99](https://doi.org/10.1109/TKDE.2005.99).
- [2] J. Bennett e S. Lanning. «The Netflix Prize». In: *KDD Cup* (2007). URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.115.6998>.
- [3] J. S. Breese, D. Heckerman e C. Kadie. «Empirical Analysis of Predictive Algorithms for Collaborative Filtering». In: UAI'98. 1998, pp. 43–52. URL: <https://arxiv.org/ftp/arxiv/papers/1301/1301.7363.pdf>.
- [4] M. Ester et al. «A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise». In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. AAAI Press, 1996, pp. 226–231.
- [5] Ben Frederickson. *Fast Python Collaborative Filtering for Implicit Datasets*. URL: <https://github.com/benfred/implicit>.
- [6] Simon Funk. *Netflix Update: Try This at Home*. URL: <https://sifter.org/~simon/journal/20061211.html>.
- [7] D. Goldberg et al. «Using collaborative filtering to weave an information tapestry». In: *Communications of the ACM* 35 (dic. 1992), pp. 61–70. URL: <https://doi.acm.org/doi/10.1145/138859.138867>.
- [8] G. Hinton e S. Roweis. «Stochastic Neighbor Embedding». In: *Advances in Neural Information Processing Systems*. Vol. 15. 2003, pp. 833–840. URL: <https://proceedings.neurips.cc/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf>.
- [9] Y. Hu, Y. Koren e C. Volinsky. «Collaborative Filtering for Implicit Feedback Datasets». In: dic. 2008, pp. 263–272. DOI: [10.1109/ICDM.2008.22](https://doi.org/10.1109/ICDM.2008.22).
- [10] S. Kullback e R. A. Leibler. «On Information and Sufficiency». In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: [10.1214/aoms/1177729694](https://doi.org/10.1214/aoms/1177729694). URL: <https://doi.org/10.1214/aoms/1177729694>.
- [11] L. van der Maaten e G. Hinton. «Visualizing Data using t-SNE». In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: <http://jmlr.org/papers/v9/vandermaaten08a.html>.

- [12] K. Pearson. «LIII. On lines and planes of closest fit to systems of points in space». In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: [10.1080/14786440109462720](https://doi.org/10.1080/14786440109462720).
- [13] S. Rendle et al. «BPR: Bayesian Personalized Ranking from Implicit Feedback». In: UAI '09. Montreal, Quebec, Canada, 2009, pp. 452–461. URL: <https://dl.acm.org/pdf/10.5555/1795114.1795167>.
- [14] J. Sander et al. «Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications». In: *Data Min. Knowl. Discov.* 2.2 (giu. 1998), pp. 169–194. DOI: [10.1023/A:1009745219419](https://doi.org/10.1023/A:1009745219419).
- [15] D. Sisodia, L. Singh e S. Sisodia. «Clustering Techniques: A Brief Survey of Different Clustering Algorithms». In: 2012.
- [16] Statista.com. *Number of Apple Music subscribers worldwide*. URL: <https://www.statista.com/statistics/604959/number-of-apple-music-subscribers/>.
- [17] *The Pandas library*. URL: <https://pandas.pydata.org/>.