LUISS

Dipartimento di Impresa e Management

Corso di Laurea Magistrale in Marketing

Cattedra di Machine Learning & Object Driven Marketing

SISTEMI DI RACCOMANDAZIONE E COMPORTAMENTO DEL CONSUMATORE: CONFRONTO TRA USER-BASED E ITEM-BASED COLLABORATIVE FILTERING

RELATORE

Chiar.mo Prof. Luigi Laura

CANDIDATO

Alessia Delmedico

Matr. 701231

Correlatore

Chiar.mo Prof. Massimo Bernaschi

ANNO ACCADEMICO 2018/2019

INDICE

Alla mia famiglia

Introduzione

È sempre più complicato trovare la giusta informazione al momento più opportuno, riuscire a comprare l'abito adatto al nostro evento nel più breve lasso di tempo, avere la possibilità di scegliere, tra una miriade di film a disposizione, quello più adatto al nostro *mood*. Queste sono solo alcune delle ragioni che determinano l'importanza dei cosiddetti sistemi di raccomandazione, modelli in grado di suggerire notizie personalizzate ad un generico consumatore o utente mediante tecniche di filtraggio di informazioni in grado di soddisfare le preferenze di quel consumatore.

Questi sistemi, proprio perché suggeriscono dei prodotti o servizi a potenziali consumatori o, in generale, ad utenti, hanno un grande potere di condizionare le scelte di questi ultimi e di modificare, in alcuni casi, il ciclo di vita dei prodotti. Merita di essere citato, a tal proposito, il caso di un libro, intitolato "Touching the Void", che al momento della pubblicazione non riscosse molto successo. Anni dopo fu pubblicato un altro libro dal titolo "Into Thin Air", che trattava la stessa tematica del libro "Touching the Void". L'algoritmo di raccomandazione di Amazon notò che vi erano alcuni individui che avevano acquistato entrambi i libri e iniziò a raccomandare "Touching the Void" alle persone che avevano acquistato o avevano mostrato interesse per il libro "Into Thin Air": questo favorì il successo di "Touching The Void", un libro che senza l'algoritmo di Amazon avrebbe potuto quasi sicuramente non trovare più alcun potenziale compratore.

Alla luce di questo fondamentale ruolo svolto da tali modelli, la presente tesi mira all'analisi dei dati e all'applicazione di uno dei principali metodi di raccomandazione, i.e. il criterio del *collaborative filtering*, ai fini di comprendere e valutare le performance di questo algoritmo e confrontare i risultati ottenuti.

Poiché l'argomento che si intende trattare in questo elaborato risulta molto articolato e non di immediata comprensione, ho deciso di suddividere il mio lavoro di tesi in tre macro-sezioni, in cui le prime due sono finalizzate ad introdurre e spiegare le molteplici caratteristiche e proprietà dei sistemi di raccomandazione, anche in considerazione di metodi valutativi in grado di misurare la bontà dei suggerimenti forniti dagli algoritmi; il terzo capitolo mette in pratica quanto asserito nelle prime due parti e focalizza l'attenzione sull'analisi sperimentale di un dataset per la raccomandazione di libri.

Nel dettaglio, il primo capitolo si avvia con la descrizione delle differenti "ondate" che hanno caratterizzato lo sviluppo dei sistemi di raccomandazione con una particolare enfasi posta al rapporto di questi con l'individuo e potenziale consumatore. Si sono, inoltre, indagati i vari livelli di personalizzazione nelle raccomandazioni di un modello e le fonti di conoscenza (i.e. sociale, individuale e di contenuto) alla base del corretto funzionamento dei diversi tipi di algoritmi. Si sono successivamente passati in rassegna i principali tipi di algoritmi di raccomandazione, ossia il metodo di filtraggio collaborativo, quello basato sul contenuto e, infine, i cosiddetti sistemi ibridi: per ognuno di questi tre tipi di raccomandazione sono state analizzate le logiche sottostanti, le principali metodologie e i limiti insiti in ciascuna tecnica di raccomandazione.

Se lo scopo principale del primo capitolo consiste nel delineare le principali tipologie di raccomandazione e nel comprendere, per ciascuna tipologia, le tecniche più idonee per una corretta costruzione e implementazione degli algoritmi, la seconda parte di questo elaborato è incentrata sui metodi valutativi, che determinano se e in quale misura un algoritmo di raccomandazione ha operato correttamente oppure no.

Nel caso specifico, è stata descritta una fase di preparazione dei dati che comprende la distinzione tra diversi tipi di valutazioni (i.e. *online* e *offline*) e differenti metodi di suddivisione del dataset in analisi. In seguito, sono state presentate varie metriche tra cui quelle di accuratezza previsionale e quelle di accuratezza nella classificazione, utili per capire in che misura un algoritmo sbaglia nel fornire raccomandazioni che rispecchino gli interessi e preferenze di un utente.

Il terzo capitolo è stato dedicato all'analisi empirica di un *set* di dati per la raccomandazione di libri: *in primis* si è descritto il *dataset* ed è stata avviata un'analisi esplorativa delle variabili. Successivamente sono stati impostati i modelli di raccomandazione (i.e. *item-based* e *user-based collaborative filtering*), i quali hanno fornito delle raccomandazioni, a loro volta in seguito valutate secondo i metodi valutativi descritti nel capitolo due.

I motivi alla base della mia scelta di tesi risiedono principalmente nel desiderio di approfondire una tematica del *machine learning* a me prima poco conosciuta e, in generale, nel particolare interesse per gli algoritmi di raccomandazione e le loro potenziali applicazioni ad un numero sempre maggiore di settori.

Se, infatti, in principio queste tecniche sono state maggiormente utilizzate in ambito informatico e si sono prestate fin da subito ad essere sfruttate nel settore dell'*e-commerce*, oggi si assiste ad un'espansione più capillare di questi algoritmi, che hanno trovato applicazione anche in settori più "delicati" e rischiosi, come può essere l'ambito medico.

.

Capitolo 1 – I sistemi di raccomandazione

Il presente capitolo mira a fornire una panoramica generale del cosiddetti sistemi di raccomandazione, indagando soprattutto i motivi della loro importanza e le tecniche principali a cui si affidano. Un particolare focus è posto inoltre sul rapporto tra questi algoritmi e il comportamento degli utenti del sistema e potenziai consumatori.

1.1. Overview, ragioni e value proposition

Nell'era del Web e della digitalizzazione, i singoli utenti sono esposti ad un traffico di informazioni su Internet, che entro la fine del 2019 dovrebbe raggiungere i 2 zettabyte[1], dove un zettabyte è equivalente a un sestilione di byte: nel 2016 tale valore era "solo" pari a 1,1.
La crescita della mole di notizie sul web si accompagna ad un aumento del numero di utenti attivi: ricerche pubblicate dalla piattaforma web Hootsuite con l'agenzia di Marketing "Wearesocial" rivelano, infatti, che nel 2019 in Italia il 92,5% della popolazione naviga sul Web, con un aumento del 19% rispetto allo scorso anno. Tra le varie attività possibili in rete, l'*e-commerce* sta diventando un processo di acquisto sempre più diffuso, con due italiani su tre che preferiscono acquistare ed effettuare pagamenti online[2].
In questo contesto, dove qualsiasi tipo di bene o servizio è a portata di clic per ogni individuo e dove tutto può essere analizzato e tracciato, il singolo utente è esposto a sovrabbondanti informazioni, con la conseguenza di riscontrare difficoltà e di ritrovarsi a prendere decisioni "povere", come afferma Carlos Gomez Uribe, Vice Presidente per l'Innovazione di Prodotto in Netflix; e non perfettamente adatte a soddisfare i propri bisogni oppure di non riuscire affatto a scegliere tra alternative simili [21].
È in questo ambito che trovano campo fertile i cosiddetti sistemi di raccomandazione sui quali, implementati dall'ultimo decennio del secolo scorso, numerose ricerche e

[1] Per approfondimenti: *Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper*, disponibile al sito https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html
[2] Per approfondimenti: *Digital 2019: tre italiani su cinque attivi sui social per quasi due ore al giorno*, disponibile al sito https://wearesocial.com/it/blog/2019/01/digital-in-2019

miglioramenti sono stati apportati grazie anche a tecniche di intelligenza artificiale che includono, tra le altre, *machine learning* e *data mining*. Tali sistemi di raccomandazione sono, dunque, strumenti che filtrano informazioni rilevanti per il singolo individuo in base alle sue preferenze, interessi o atteggiamento osservato nei confronti del prodotto, oggetto di scelta.

Un sistema di raccomandazione efficiente è diretto a fornire suggerimenti personalizzati che derivano dall'unione delle notizie sull'*user* e le caratteristiche del prodotto, le quali includono anche *feedback* di altri utenti e comparazione con altri articoli [3].

Uno dei loro scopi principali consiste nel predire il comportamento dell'individuo grazie alle tecniche di *artificial intelligence* utilizzate: da questa prospettiva, essi vengono sfruttati dalle imprese per adattare le caratteristiche del prodotto da loro commercializzato alle esigenze del potenziale consumatore. Indubbiamente queste ultime vedranno aumentato il loro vantaggio competitivo e assisteranno anche ad una crescita delle vendite e, di conseguenza, dei ricavi.[3]

Alla luce di quanto detto si possono, quindi, delineare due *value proposition* dei sistemi di raccomandazione (o *recommender system*s, nell'espressione inglese): da un lato, i *recommender systems* vengono utilizzati per aiutare gli utenti a fronteggiare la mole smisurata di informazioni sul web e per indirizzarli, in seguito, verso gli articoli e le informazioni con più alta utilità; dall'altro si configurano come strumenti in grado di operare efficacemente e garantire alle imprese che ne fanno uso un incremento nelle vendite e, in generale, un livello più alto di *engagement* nei loro sevizi da parte degli stessi consumatori [32].

Sono proprio queste due macro-esigenze a rappresentare i punti cardine da cui è possibile tracciare, raggruppare e scandire differenti fasi, che segnano la nascita, gli sviluppi e i miglioramenti in questo ambito.

La prima fase si avvia negli anni '80 del secolo scorso, quando Internet inizia ad espandersi e si iniziano a studiare le prime tecniche di *information filtering* e automazione del *word-of-mouth*. Significativa è l'espressione *"personal assistant"* utilizzata dal professor Maes per descrivere la compresenza di un utente fisico, a cui si affianca un agente in grado di apprendere le preferenze, abitudini e interessi dell'utente stesso e classificarli stabilendo un ordine di priorità. Alla base del concetto espresso dal professore del MIT vi è l'idea più generale di sfruttare le valutazioni di più individui per poter predire concetti astratti ad esse connessi, validi per un insieme di utenti con preferenze simili e classificare tali concetti in base alle priorità del singolo individuo[4]. Il progetto Tapestry del 1992, ad esempio, fu uno dei primi progetti

[3] Per approfondimenti: I Big Data: alla scoperta del big fenomeno, disponibile al sito https://www.teamsystem.com/store/blog/digitalizzazione/big-data-alla-scoperta-del-big-fenomeno/

[4] Tale concetto è alla base del metodo del collaborative filtering, che verrà trattato nel paragrafo 1.5.

sviluppati al Research Center Xerox, che utilizzava la tecnica di *collaborative filtering*: in particolare, Tapestry permetteva all'utente di commentare dei documenti, come le e-mail, in modo esplicito con delle valutazioni o annotazioni; oppure implicitamente, ad esempio, rispondendo ad una mail come indicatore di rilevanza. Tutto ciò era fatto per fornire suggerimenti ad altri utenti. In particolare, Tapestry supportava sia il filtraggio basato sui contenuti che il filtraggio collaborativo (*content* e *collaborative filtering*): la collaborazione tra i soggetti consisteva nel fatto che questi ultimi registravano le loro reazioni ai documenti che leggevano (tali reazioni venivano chiamate annotazioni). Vi era, inoltre, la possibilità di accedere alle annotazioni anche mediante filtri altrui e questo permetteva agli individui di aiutarsi a vicenda nello stabilire l'importanza e la priorità nella gestione della posta elettronica. Tapestry era in grado di gestire qualsiasi flusso di documenti elettronici in entrata e serviva sia come filtro della posta che come archivio.

In questa prima fase, sviluppare e mantenere un algoritmo di raccomandazione era costoso e la maggior parte di questi erano utilizzanti con finalità commerciali e in un'ottica di guadagno per le imprese.

A titolo di esempio può essere citato Firefly Networks, prima azienda a fornire servizi di raccomandazione, che aveva sfruttato il successo nel fornire suggerimenti musicali del sistema Ringo, per raccomandare anche film e siti web secondo la tecnica del *collaborative filtering*. Net Perceptions e Likeminds furono leader per lo sviluppo di servizi di raccomandazione nel mercato *business-to-business*, con il secondo specializzato nella commercializzazione di strumenti, che utilizzavano l'approccio del *collaborative filtering* in grado di assistere altri siti web ad offrire raccomandazioni personalizzate.

Se in una fase iniziale i siti *e-commerce* e nuovi portali web erano tra loro in forte competizione per attrarre nuovi consumatori e differenziavano i loro servizi con lo sviluppo più o meno attivo di tecniche di raccomandazione, in un secondo periodo, nei primi anni 2000, il *focus* si sposta sugli *user*: il numero di utenti inizia, infatti, a crescere e si assiste alla nascita di *social network* come Facebook e MySpace, rispettivamente nel 2004 e 2003. La nascita di questi nuovi canali di comunicazione suggerisce all'utente una nuova prospettiva, in quanto quest'ultimo può leggere e informarsi su interessi e abitudini dei propri "amici "virtuali navigando nel *feed* in cui sono presenti le notizie su Facebook, per esempio: si viene, dunque, a creare un sistema indiretto di *word-of- mouth* che influenza le preferenze degli utenti.

La maggior parte degli algoritmi di raccomandazione presenti in questo periodo si configuravano come *rating-based*[5] e riuscivano a suggerire adeguate proposte solo in seguito alle valutazioni di molti *items* da parte degli utenti; comunemente i venditori offrivano un *remote application programming interface* (API), che permetteva di collegare e tracciare le attività degli utenti sulle *web pages*, come click o acquisto, e generare raccomandazioni ad esse appropriate.

Vi erano, inoltre, forti evidenze empiriche che a differenti prodotti (e.g. film, musica, notizie) corrispondessero differenti necessità di servizi di raccomandazione, anche se l'offerta di questi ultimi era scarsamente differenziata per tali specifici domini [3].

Sebbene gli utenti in questa fase assumano un ruolo importante, il fine dell'implementazione di sempre più sofisticati algoritmi di raccomandazione risiede nel profitto di breve periodo dei grandi *player online*. Se da un lato, quindi, si cerca di perfezionare questi *recommender system*, dall'altro iniziano inevitabilmente a verificarsi anche problemi legati alla qualità degli algoritmi come la scalabilità, sparsità e lo *shilling attack*[6]. Alcuni grandi siti web *e-commerce*, come Amazon e Yahoo, cercano di ovviare a queste problematiche attraverso strutture in grado di considerare contemporaneamente un sistema *rating-based* con l'andamento dei comportamenti di acquisto o noleggio degli utenti [7].

Celeberrima a tal proposito è la competizione lanciata nel 2006 da Netflix a livello internazionale e nota come Netflix Prize, volta a migliorare del 10% l'accuratezza dello algoritmo di raccomandazione da loro usato, CineMatch, e nello specifico a minimizzare l'errore connesso al suggerimento di un determinato film per uno specifico utente.

Tale concorso dà simbolicamente il via ad una fase caratterizzata dalla diffusione più capillare dei sistemi di raccomandazione: basti pensare che questi algoritmi agiscono sia a livello *mobile*, sia *online*, operando su siti *e-commerce*, di intrattenimento o anche all'interno di sistema e-mail in generale, il quale permette di filtrare la posta elettronica in base ad un criterio di priorità. Si diffondono anche applicazioni o *widget* che permettono di contare le ore di sonno o i passi fatti durante una giornata, per suggerire al singolo individuo azioni e comportamenti per uno stile di vita più sano: esempi in questo senso possono considerarsi le app MapMyRun o Fitbit.

In linea generale, questa terza e più moderna "ondata" di sviluppo di sistemi di raccomandazione si caratterizza per la cosiddetta *contexual awareness*, ossia la capacità degli

[5] Molti algoritmi seguivano il cd. criterio di similarità *rating-based*, che considerava le valutazioni (generalmente da 1= molto negativo, a 5 o 7=molto positivo) degli utenti riguardo i prodotti consumati.

[6] Tali problematiche si riferiscono principalmente al metodo del *collaborative filtering*. Questi argomenti verranno trattati nel paragrafo 1.7.

algoritmi di generare suggerimenti sempre più adatti a soddisfare i bisogni degli utenti in un dato momento, grazie a un numero maggiore di informazioni estrapolate dai dati utilizzando, tra gli altri, dispositivi mobile o rilevatori GPS. Anche il *cloud computing* permette, inoltre, di erogare sistemi di raccomandazione come servizi *on demand* mediante l'utilizzo della rete Internet e di ridurre così notevolmente i costi di gestione connessi all'acquisto di programmi software o di *server* per la memorizzazione dei dati[7]. La presenza online di procedure API pubbliche e di biblioteche informatiche consente il trattamento e l'utilizzo di più sofisticati algoritmi e dati: tutto ciò permette di offrire all'utente finale un servizio sempre più preciso e dettagliato. In questa prospettiva, i moderni sistemi di raccomandazione iniziano ad operare in un'ottica di lungo periodo e a fornire suggerimenti, che in una visione di breve periodo sarebbero stati svantaggiosi. Esempi in questo senso possono essere considerati i vincoli di budget che l'utente dovrebbe rispettare per un'ottimizzazione delle sue risorse; ossia viene accantonata una visione di breve periodo legata a tecniche finalizzate al consumo immediato e d'impulso come l'*up-selling* e il *cross-selling*.

1.2. Definizioni e concetti generali

Nel paragrafo precedente si è voluto fornire una panoramica anche su un piano storico dell'oggetto di trattazione della presente tesi. In questa sezione verranno definiti in maniera più dettagliata tali algoritmi, con un particolare *focus* sulle caratteristiche dei principali settori in cui questi trovano maggiore applicazione [23].

In una prospettiva che pone l'enfasi sul beneficio ottenuto dall'utente mediante l'utilizzo delle tecniche di raccomandazione, si può dedurre la seguente definizione.

Dati C l'insieme di tutti gli utenti per i quali il sistema provvederà ad una raccomandazione, S l'insieme di tutti i possibili prodotti (*item*), che possono essere raccomandati e u la funzione di utilità, che misura l'adeguatezza del prodotto s per il singolo user c, per cui si avrà che:

$$u: C \; x \; S \Rightarrow R \quad [1.1]$$

[7] Libero Tecnologia, Che cos'è il cloud computing, disponibile al sito https://tecnologia.libero.it/che-cose-il-cloud-computing-14843

dove *R* è l'insieme totale raccomandato. Dunque, per ogni user c ∈ C si vuole scegliere il prodotto s' ∈ S che massimizza l'utilità dell'utente, che è generalmente indicata da una valutazione dell'utente stesso[8].

Ogni utente è indentificato dal sistema in base ad un suo profilo, che può contenere sia informazioni di carattere demografico e, quindi, strettamente connesse alla sua persona, sia dati relativi alle sue preferenze: in questo ultimo caso si farà, quindi, riferimento a notizie connesse alla visualizzazione, valutazione ed eventualmente acquisto di particolari *item*. Allo stesso modo, anche il singolo prodotto sarà definito alla luce di peculiarità e proprietà esplicitamente o implicitamente aggiornate in base all'interazione rispettivamente esplicita (e.g. valutazione o acquisto di un *item*) o implicita (e.g. click o visualizzazione frequente e ripetuta di un articolo) dell'utente con il sistema, dove per sistema si intende il portafoglio di tutti i possibili *item* soggetti a raccomandazione.

La figura sottostante riassume quanto affermato precedentemente e sintetizza in processo di raccomandazione (Figura 1.1).

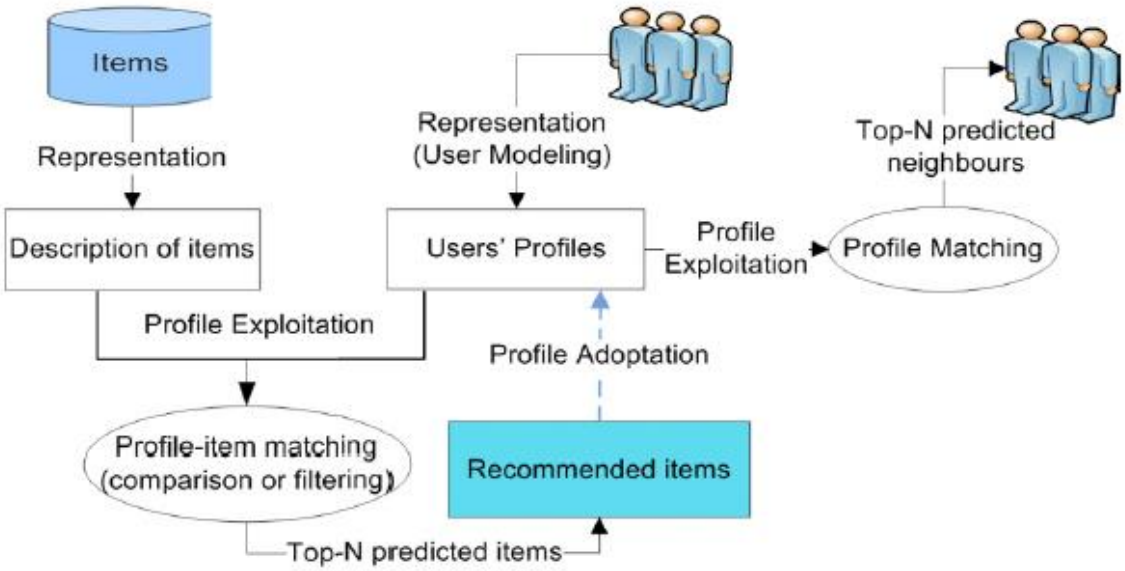


Figura 1.1 Modello generale del processo di raccomandazione[9]

[8] Come si è già affermato nel paragrafo precedente, generalmente per la valutazione degli *item* si fa riferimento ad una scala ad intervalli 1-5 o 1-7, dove 1 rappresenta il totale disinteresse dell'*user* nei confronti di quel particolare *item*, viceversa il 5 o il 7. Oltre ad un *rating* espresso in forma scalare, ci possono essere *rating* binari, in cui l'utente esprime valutazioni come Accordo/Disaccordo, Bene/Male; e *rating* unari, che permettono agli utenti di assegnare gli *item* ad una singola classe, in genere positiva (e.g. bottone "Like" su Facebook).

[9] Khusro, S., Ullah, I., Ali, Z., (2016). "Recommender Systems: Issues, Challenges and Research Opportunities", Springer Singapore

Per far funzionare un sistema di raccomandazione, assumendo di avere una piattaforma in un sito web o un'app dove si vuole implementarlo, si inizia con la definizione di un obiettivo quale può essere, per un *site provider*, aumentare le vendite. Successivamente si raccolgono dati comportamentali necessari a costruire un algoritmo, che a sua volta dà vita ad un modello, il quale calcola raccomandazioni. In genere il modello viene sperimentato su dati storici per verificare la sua efficacia sul comportamento dell'utente. Se il test ha esito positivo, il modello può essere esposto ad una parte degli utenti per capire se è possibile tracciare dei miglioramenti: è solo in questo caso che esso può essere prodotto ed utilizzato su individui reali.

1.3. Livelli di personalizzazione e comportamento degli utenti

Gli *output* che restituisce il sistema grazie all'utilizzo di un algoritmo, ossia le raccomandazioni, possono collocarsi su differenti livelli di personalizzazione, in base all'utilizzo di più generiche statistiche o di dati strettamente associati all'individuo.

Quando un sistema suggerisce una lista di *item* che può essere di gradimento per molti *user* si è di fronte a raccomandazioni non personalizzate. Fanno parte di questo gruppo di raccomandazioni anche quelle che mostrano informazioni ordinate temporalmente, ossia che presentano per primi notizie più recenti e ogni utente che interagirà con questo sistema riceverà lo stesso elenco di raccomandazioni.

Il secondo livello di personalizzazione, definito *semi/segment personalized*, fornisce indicazioni agli utenti raggruppati in base ad un criterio, quale può essere età, professione, nazionalità. In questo caso, il sistema di raccomandazione non conosce niente di strettamente personale del singolo individuo, ma solo notizie della persona come parte di un insieme o segmento, i cui membri riceveranno le stesse raccomandazioni [7].

Una raccomandazione personalizzata, invece, prende in considerazione i dati dell'utente in questione, che mostrano le modalità con cui l'individuo ha interagito con il sistema in precedenza, in modo da poter generare informazioni *ad hoc* per quell'utente.

La maggior parte dei sistemi di raccomandazione combina i primi due livelli di personalizzazione per dare vita a raccomandazioni personalizzate. Un esempio a tal proposito, è rappresentato dalla *starting page* di Netflix, in cui i film consigliati si basano da un lato su quanto precedente guardato dall'utente sulla piattaforma, dall'altro secondo un più generalizzato raggruppamento dei film che prende in considerazione il genere e il livello di attualità.

Molto di frequente all'interno di uno stesso sito si possono trovare applicati più livelli di personalizzazione: Amazon, ad esempio, presenta una sezione che contiene gli "articoli più venduti" (in questo caso si è di fronte ad una raccomandazione non personalizzata), così come una parte in cui "chi ha comprato questo prodotto ha anche comprato questa lista di articoli" (*segment personalized*).

Il livello di personalizzazione di un sito dipende anche dal traffico di utenti che il medesimo registra. Naturalmente più *user* visiteranno e interagiranno in un determinato sito, più l'algoritmo al suo interno avrà la possibilità di elaborare una mole maggiore di dati e fornire raccomandazioni più personalizzate.

Per questa ragione, prima di comprendere le modalità di funzionamento dei *recommender system*, è necessario delineare i vari *step* che compongono il "ciclo di vita" del potenziale consumatore.

In un primo momento, il potenziale consumatore naviga nel Web o in un sito solo per vedere cosa offre, senza un particolare obiettivo. Egli cercherà in maniera *random* tra diversi contenuti e articoli, soffermandosi su quelli che mostrano *insight* per lui interessanti e rilevanti.

In questa fase, è molto importante per il sito tracciare i punti in cui il potenziale consumatore si ferma a guardare (*pageview*) o sui quali clicca, poiché questo potrebbe essere sintomo di interesse. Bisogna, però, tenere in considerazione che se il tempo speso a cliccare su un prodotto ripetutamente o a visualizzarlo risulta eccessivo (più di qualche minuto), questo molto probabilmente non porterà a conversioni, ossia il potenziale consumatore non andrà avanti nel suo *funnel*. In generale, possono essere considerati segnali di interesse dell'individuo verso un prodotto o servizio: cercare ulteriori informazioni sull'articolo di interesse, magari cliccando sull'*expansion link*, che normalmente si trova al termine di una breve descrizione del prodotto, leggere recensioni o dettagli tecnici dello stesso, condividere sui *social network*, scaricare una brochure o guardare un video su un contenuto particolare [7].

Il ciclo di vita del processo di acquisto del consumatore prosegue, dunque, quando egli mostrerà interesse verso uno o più articoli e li aggiungerà al suo carrello o all'interno di una lista con l'intenzione di acquistarli subito o in seguito.

Successivamente, egli comprerà i prodotti, li utilizzerà e li valuterà, a volte nello stesso sito in cui ha effettuato l'acquisto. Anche se l'atto dell'acquisto rappresenta l'obiettivo principale per i venditori nei siti *e-commerce*, un algoritmo di raccomandazione deve valutare la possibilità di eventuali *outlier*, ossia che un acquisto differente per contenuto o tipologia dai precedenti, lo sia perché indirizzato ad una persona terza rispetto all'individuo che ha effettuato la transazione

o poichè è segnale di un suo nuovo gusto [7]. In linea generale, è arduo e non sempre possibile identificare un utente e capire quando questi agiscono per se stessi o per terzi: esistono delle modalità, come eseguire il *log-in* in una pagina web o degli strumenti, quali i *cookie*[10], che cercano di arginare questo problema, anche se non lo eliminano del tutto.

Anche il *rating* di un prodotto, quando è presente, non è del tutto indicativo delle preferenze di un consumatore: questo andrà connesso anche ad altri sistemi di valutazione più dettagliati, quali possono essere le recensioni e i commenti che gli utenti lasciano dopo aver visto, comprato o usato un prodotto o servizio. Poiché non è sempre possibile assegnare un *rating* a tutti gli *item*, un sistema di raccomandazione deve utilizzare adeguati metodi di filtraggio di informazioni, per fare in modo che alcuni prodotti siano comunque valutati esplicitamente o implicitamente e, di conseguenza, suggeriti all'utente.

Tali tecniche di filtraggio si suddividono in tre principali tipologie: *collaborative filtering*, *content- based filtering* e *hybrid filtering*, che verranno trattati dal paragrafo 1.5.

La parte conclusiva del *funnel* di acquisto di un prodotto da parte di un utente prevede che quest'ultimo rivenda o in qualche modo si liberi del prodotto: in questo caso, molto probabilmente il prodotto ripeterà lo stesso ciclo.

1.4. Fonti di conoscenza e tipi di raccomandazione

Gli algoritmi di raccomandazione sono sistemi di intelligenza artificiale, per i quali è importante comprendere la fonte e il tipo di conoscenza impiegati per attivarli correttamente. Nel caso dei *recommendation system* risulta molto importante conoscere gli *user* e le caratteristiche degli *item* consigliati, proprio per generare suggerimenti sempre più personalizzati (cfr. par. 1.3.).

Le informazioni sugli utenti si basano su dati relativi all'*user* in target e sulla sua *community* di riferimento. Nello specifico, si possono delineare tre categorie di conoscenza che sono alla base del corretto funzionamento dei diversi tipi di algoritmi di raccomandazione [5]. Esse sono:

- Sociale: conoscenza di una vasta comunità di utenti oltre all'individuo target;
- Individuale: conoscenza del target *user*;

[10] Generalmente i siti impostano dei *cookie* che raccolgono e archiviano informazioni sull'utente. Quando poi l'utente effettua eventualmente il *log-in* o crea un suo profilo in quel sito, tutte le notizie raccolte dai *cookie* vengono trasferite in quello specifico *account*.

- Contenuto: conoscenza delle caratteristiche dei prodotti che vengono raccomandati e, più in generale, dei loro usi.

Tra i fattori che differenziano la conoscenza sociale da quella individuale vi è il contesto: un'adeguata raccomandazione è funzione molto spesso dal contesto nel quale questa si colloca [33].

A titolo di esempio, uno stesso individuo che ricerca un ristorante della stessa tipologia (e.g. giapponese) dovrebbe ricevere suggerimenti differenti a seconda che questo ristorante debba essere prenotato per una cena di lavoro o per un anniversario personale.

Mentre la conoscenza sociale rappresenta la somma totale di tutti i profili memorizzati in un sistema, è su quella individuale che si instaura la relazione con l'algoritmo di raccomandazione. Si può avere una conoscenza individuale implicita, nel senso che il profilo di utente viene ricercato, richiamato alla memoria e utilizzato per avviare il processo di raccomandazione, o esplicita, se l'individuo specifica i suoi interessi prima o durante lo stesso procedimento.

La conoscenza di contenuto si manifesta in differenti forme, tra le quali la più basilare prevede che il sistema riconosca le caratteristiche degli *item* che sta andando a suggerire, prevedendo quelle che gli utenti potrebbero preferire. Una struttura più complessa di questa categoria è la conoscenza del settore (*domain knowledge*), che si configura come una fonte di conoscenza mezzi-fini, ossia quali mezzi sono appropriati per quel determinato obiettivo che l'individuo ha in mente. Vincoli di settore possono aiutare un sistema di raccomandazione ad evitare di suggerire un prodotto o servizio che risulta non coerente con i confini del medesimo settore [5].

Un'altra peculiarità della conoscenza di contenuto consiste nell'ontologia delle caratteristiche, che permette di raggruppare proprietà simili tra di loro, in modo tale da identificare più facilmente similarità o differenze tra gli *item*.

In merito a un settore, in cui operano i sistemi di raccomandazione, si possono delineare sei caratteristiche: eterogeneità, livello di rischio, abbandono e interazione con il sistema, stabilità delle preferenze e "scrutabilità".

La prima caratteristica, ossia l'eterogeneità, si riferisce ad uno spazio in cui sono presenti molti *item* con differenti peculiarità e diversi obiettivi da soddisfare; uno spazio di raccomandazione omogeneo, invece, implica che la conoscenza di contenuto relativa la settore sia più semplice da acquisire e mantenere. Si possono, inoltre, distinguere settori ad alto e basso rischio: il livello di rischio determina la tolleranza deli *user* per raccomandazioni cosiddette false positive, di conseguenza tanto più un settore presenta un alto livello di rischio tanto più la tolleranza per i falsi positivi sarà bassa.

Gli algoritmi di raccomandazione possono essere utilizzati in domini, in cui sono presenti prodotti con un ciclo di vita stabile e duraturo (e.g. libri), così come per suggerire *item* che

hanno, al contrario, una vita più breve (e.g. news) e che sono soggetti ad un alto livello di abbandono, nel senso che con il passare del tempo perdono la loro rilevanza [7].

In sistemi in cui l'utente interagisce con il sito in maniera implicita, le valutazioni e successivamente le raccomandazioni vengono fornite in seguito ad inferenze sul comportamento del consumatore: ad esempio, le raccomandazioni possono essere effettuate in base alle visualizzazioni di determinati prodotti o allo storico delle transazioni di un utente. Quando, invece, un soggetto interagisce in maniera esplicita, formula sue opinioni o aggiunge dati personali al sistema; in questo modo rende esplicite le sue preferenze. Quando le preferenze sono stabili, gli *opinion data* raccolti nel passato avranno un'alta probabilità di essere validi anche nel presente: in questo modo, sarà più facile mantenere alta qualità delle fonti di conoscenza che usano tali dati. Se, viceversa, vi è instabilità di preferenze, i dati raccolti in passato potrebbero non avere più validità: tale problema può essere parzialmente arginato attraverso la raccolta di più dati.

L'ultima caratteristica dei settori consiste nella "scrutabilità", ossia nel dare informazioni all'utente circa le modalità in cui sono state effettuate le raccomandazioni, al fine di accrescere la *confidence* dell'*user* e la consapevolezza che quella raccomandazione è appropriata.

Ritornando alle fonti di conoscenza, distinzione tra le varie fonti risulta molto utile per avere una visione più chiara sui principali tipi di raccomandazione. La figura 1.2. sottostante chiarisce quanto affermato finora e mostra il collegamento tra le fonti di conoscenza e le differenti tipologie di sistemi di raccomandazione.

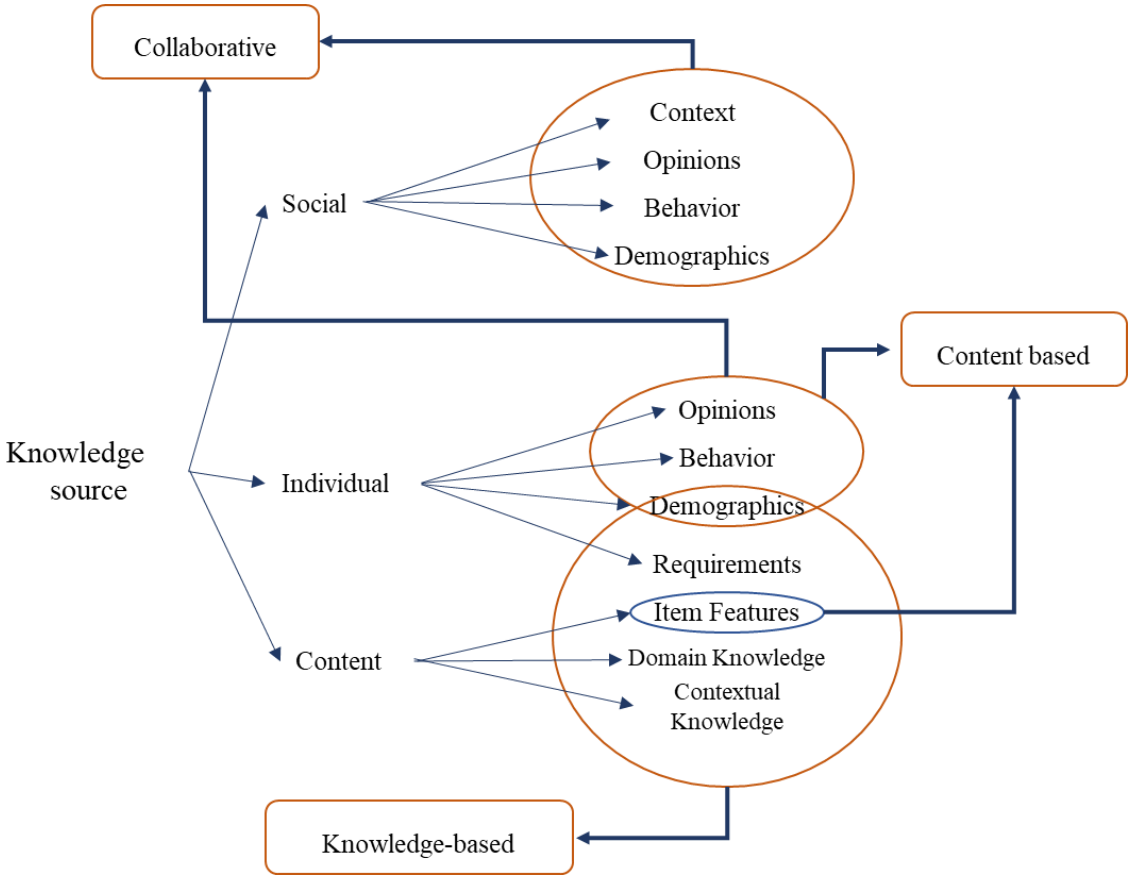


Figura 1.2: Fonti di conoscenza e tipi di raccomandazioni[11]

Il metodo del *collaborative filtering* connette la fonte di conoscenza individuale con quella sociale e fornisce suggerimenti per il target di utente sulla base di preferenze di individui considerati simili a lui.

Raccomandazioni basate sul contenuto si implementano, invece, sulle fonti di conoscenza individuale e utilizzano le caratteristiche di un *item* e le opinioni di un individuo per impostare classificazioni, in modo tale da poter predire le preferenze degli utenti o nuovi prodotti.

Sistemi *knowledge-based* raccolgono informazioni più strettamente connesse all'individuo, come, ad esempio, i dati demografici, e le connettono conoscenze del settore e contesto. Questi sistemi vengono utilizzati in settori in cui vi è poco storico degli acquisti degli utenti. In questo caso, prima di provvedere con la raccomandazione, l'algoritmo considera le informazioni riguardo agli *item*, come le preferenze degli individui o le caratteristiche dei prodotti.

Per quanto riguarda sistemi di raccomandazioni ibridi, dalla prospettiva delle fonti di conoscenza, non è possibile delineare in modo univoco le fonti a cui queste sono connesse.

[11] Burke, R., Ramezani, M., (2011). "Matching Recommendation Technologies and Domains" In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA [18].

Molto spesso, tali sistemi di raccomandazioni si creano adattando un algoritmo utilizzato per un tipo di raccomandazione in modo tale che questo possa comprendere un'ulteriore fonte di conoscenza, tipicamente associata ad un altro tipo di algoritmo.

1.5. Collaborative filtering system

L'idea alla base dell'approccio *collaborative filtering* considera che se due utenti hanno manifestato lo stesso interesse nel passato, come, ad esempio, amare lo stesso film, essi mostreranno simili preferenze anche in futuro. Nello specifico, se due utenti A e B hanno effettuato acquisti simili negli anni e di recente l'individuo A ha comprato un film che B non ha ancora visto, allora il sistema di raccomandazione suggerirà quel film anche a B.
In questo caso, per suggerire un prodotto o servizio da un più ampio *set* di alternative si attinge dalle preferenze degli utenti e non si tengono in considerazione la natura, le caratteristiche o i contenuti degli *item* raccomandati. Tanto più saranno le informazioni circa le preferenze degli utenti, tanto più accurati saranno i risultati.
Tra gli strumenti più usati per l'analisi e l'implementazione degli algoritmi appartenenti a questo sistema di raccomandazione vi è la matrice *user-rating*, in cui vengono rappresentate sottoforma di *rating* le preferenze di vari utenti rispetto a vari *item* presenti. Questa è una matrice mxn, in cui ad ogni colonna *m* corrisponde un prodotto e le *n* righe rappresentano i vari utenti, che assegnano un *rating* per ogni determinato elemento [7]. La cella $r_{u,i}$ corrisponderà, quindi, alla valutazione assegnata all'*item i* da parte dell'*user u*. La matrice in questione è in genere una matrice sparsa, ossia caratterizzata da valori mancanti, poiché nella quasi totalità dei casi gli *user* si rivelano riluttanti nel valutare ogni elemento presente in un certo dataset.

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{1i} & \cdots & r_{1n} \\ r_{u1} & r_{u2} & r_{ui} & \cdots & r_{un} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ r_{m1} & r_{m2} & r_{mi} & \cdots & r_{mn} \end{pmatrix}$$

Sarà compito degli algoritmi di raccomandazione calcolare mancante presente in una cella per un determinato utente.

È, inoltre, possibile suddividere gli algoritmi di *collaborative filtering* in due sezioni: gli algoritmi *memory-based* e quelli *model-based*. Questi ultimi producono periodicamente un modello di *rating* dei vari utenti e calcolano il valore atteso delle previsioni di individuo, sulla base delle valutazioni di quell'*user* rispetto ad altri *item*. Tra le tecniche utilizzate per questo tipo di metodologie vi sono tecniche di regressione e classificazione.

Per gli algoritmi *memory-based*, invece, è necessario che tutti gli *item*, gli utenti e i *rating* siano presenti nel sistema: essi utilizzano tecniche di *nearest-neighbors* per predire il *rating* di un *user* su un insieme di *item*. Gli algoritmi correntemente utilizzati possono essere classificati o come *model- based* o come un ibrido tra alcuni dati pre-calcolati e valutazioni presenti in memoria.

Per quanto riguarda l'organizzazione degli algoritmi di *collaborative filtering*, è possibile individuare tecniche probabilistiche e non probabilistiche (cfr. Figura 1.3).

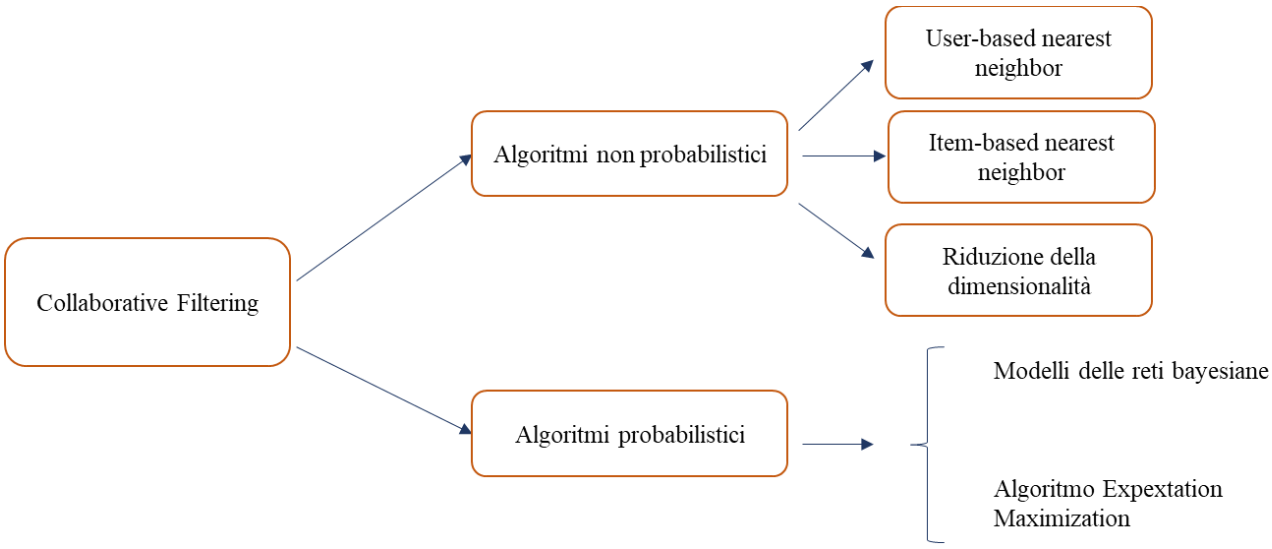


Figura1.3: Suddivisione dei sistemi di collaborative filtering

Algoritmi probabilistici riportano distribuzioni di probabilità nel calcolo dei *rating* previsti o delle liste di raccomandazioni classificate, mentre modelli non probabilistici sono largamente utilizzati dagli esperti in materia.

1.5.1 Algoritmi non probabilistici

Tra i modelli non probabilistici più noti si annovera l'algoritmo *nearest neighbor*.
Prima di affrontare questo argomento, è opportuno definire i *neighbors*, ossia i "vicini", che sono degli utenti, simili agli *user* in target, grazie ai quali *rating* è possibile effettuare delle previsioni circa i *rating* degli utenti target per un insieme di prodotti.
L'obiettivo del *k-nearest neighbors*, dunque, consiste nel prevedere il *rating* dell'utente attivo per i prodotti non ancora valutati. In particolare, si individua un sottoinsieme di *k* individui più vicini e simili all'utente in questione e si raccomandano a quest'ultimo i prodotti per i quali i suoi vicini hanno espresso un *rating* positivo. I *rating* dell'individuo target non saranno altro che combinazioni pesate delle valutazioni fornite per gli stessi elementi dai *k* individui più simili. Questo metodo ha alla base l'idea che se alcuni individui hanno valutato positivamente degli stessi tipi di prodotto, probabilmente sono simili e mostreranno simili preferenze anche in futuro [30].
Da un punto di vista operativo, è necessario, come già accennato, in primo luogo individuare utenti simili all'utente attivo; in seguito si calcola la media ponderata delle valutazioni che questi hanno dato per i prodotti e questa diventa la valutazione degli utenti attivi. Nello specifico, per misurare una somiglianza si costruisce una matrice *user-item*, in cui si riportano le valutazioni dei prodotti e degli articoli per i vari *n* utenti. La similarità è calcolata principalmente con la distanza euclidea tra due *rating* degli utenti, ma vi sono anche altre misure come il calcolo del coseno o la distanza di Manhattan.
La tecnica del *nearest neighbor* può essere sua volta suddivisa in *user- based* o *item-based.*
Il primo algoritmo si basa sul concetto di generare una previsione per un prodotto *i* di un utente *u* analizzando le valutazioni di utenti che si trovano "nelle vicinanze" rispetto a *u* [24]. Ad un primo approccio, questa previsione potrebbe essere data dalla media delle valutazioni degli altri individui "vicini", anche se questo calcolo non tiene in considerazione che alcuni vicini potrebbero avere un livello di similarità più elevato rispetto ad altri. Alla luce di ciò, è possibile, quindi, avere una più accurata previsione introducendo una misura di similarità $userSim(u, n)$, che è in grado di pesare i *rating* degli utenti più simili ad *u*.
Inoltre, per avere una previsione più accurata del *rating* per l'item *i*, si normalizza dividendo per la somma delle similarità dei *neighbors*.

$$pred(u, i) = \frac{\sum_{n \subset \text{neighbors(u)}} userSim(u, n) \times r_{ni}}{\sum_{n \subset \text{neighbors(u)}} userSim\ (u, n)} \qquad [1.2]$$

Dati *u* l'utente di cui si vuole predire il *rating* per il prodotto *i* e *n* i vicini, ossia gli utenti con preferenze simili rispetto all'utente in analisi *u.*

Uno dei modi più comuni per calcolare la similarità tra gli utenti è attraverso il coefficiente di correlazione di Pearson, che compara i *rating* per tutti i prodotti valutati sia dal target *user* sia dal "vicino" *n*. Di seguito la formula della correlazione di Pearson tra l'utente *u* e il vicino *n*:

$$userSim(u,n) = \frac{\sum_{i \subset CR_{u,n}} (r_{ui} - \overline{r_u})(r_{ni} - \overline{r_n})}{\sqrt{\sum_{i \subset CR_{u,n}} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \subset CR_{u,n}} (r_{ni} - \overline{r_n})^2}} \qquad [1.3]$$

Dove r_u *e* r_n sono i rating rispettivamente dell'utente in target u e dell'utente con simili preferenze *n*, $\overline{r_u}$ ed $\overline{r_n}$ rappresentano rispettivamente la media delle valutazioni espresse dall'utente *u* e l'utente n e CR indica un *set* di *item* valutati sia da *u* che da *n.* Il coefficiente di Pearson assume sempre valori compresi tra -1 e 1, dove 1 indica completa similarità tra i due *user*, viceversa -1. Per questa ragione, le correlazioni negative non sono generalmente utili per aumentare l'accuratezza delle previsioni e si può scegliere quindi di non utilizzarle.

Tornando alla formula [1.2], questa si aggiusta per la media dei rating di *neighbors* $\overline{r_n}$, in quanto non tiene in considerazione il diverso atteggiamento dei vari individui nell'utilizzo delle scale di valutazione. Data una stessa scala, infatti, utenti più ottimisti potrebbero valutare assegnando un punteggio maggiore ad un prodotto rispetto ad altri individui più pessimisti, anche se entrambe le tipologie di persone hanno la stessa idea nei confronti di quel prodotto (ad esempio, è uno dei loro preferiti). Si avrà, dunque, la seguente formula:

$$pred(u,i) = \overline{r_u} + \frac{\sum_{n \subset \text{neighbors}} userSim(u,n) \times (r_{ni} - \overline{r_n})}{\sum_{n \subset \text{neighbors(u)}} userSim(u,n)} \qquad [1.4]$$

Sebbene il metodo dell'*user- based nearest neighbor* aiuti a fornire suggerimento ad un utente sulla base di valutazioni di individui simili, questo presenta delle criticità date, in primo luogo, dal fatto che i dati con le valutazioni degli utenti sono sparsi e incompleti e questo porta spesso ad avere coppie di utenti con pochi *rating* in comune, con la conseguenza che questi molto frequentemente registreranno una similarità quasi perfetta. Ciò comporterà una distorsione dei risultati nella valutazione complessiva.

Inoltre, il coefficiente di Pearson non tiene conto della popolarità di un elemento nel calcolo delle correlazioni: ad esempio, *rating* molto alti di utenti nei confronti di un film che è notoriamente molto apprezzato "valgono" meno rispetto a valutazioni allo stesso modo positive di film più criticati: esistono a questo proposito algoritmi *user-based* che includono pesi inversamente proporzionali alla popolarità di un elemento nel calcolo delle correlazioni utente.
Un ulteriore problema connesso a questo modello consiste nel fatto che calcolare con esattezza lo spazio che delimita la zona in cui sono presenti i "vicini" è molto dispendioso perché richiederebbe un confronto tra tutti gli utenti presenti. Per questa ragione, secondo un'implementazione più semplice, requisiti di tempo e memoria dell'algoritmo *user-based* erano direttamente proporzionali al numero di utenti e alle loro valutazioni [14]. Poiché sarebbe immensamente dispendioso in termini di risorse analizzare le valutazioni di milioni di clienti per restituire delle adeguate raccomandazioni in un tempo limitato, si sono susseguite alcune ricerche che hanno provato molte tecniche per ridurre il tempo di elaborazione e la memoria. Alcune di queste sono il sotto campionamento e il *clustering*. Il sotto campionamento consiste nel selezionare un sottoinsieme all'interno del campione di utenti, prima di effettuare il calcolo previsionale. Il tempo per calcolare il *set* di utenti che saranno considerati "vicini" rimane fisso e vengono proposti schemi per scegliere intelligentemente un utente simile.
Gli algoritmi di *clustering*, invece, sono utilizzati per individuare rapidamente informazioni di un *user* con simili preferenze rispetto all'individuo in target. Quest'ultimo viene comparato a gruppi di utenti, rispetto che ad individui unici, in modo da scoprire più velocemente cluster di utenti simili per poter poi selezionare quelli a lui più vicini.
Il metodo del clustering utilizza funzioni di distanza, come la correlazione di Pearson, sia per formare i cluster che per misurare la distanza da un cluster [30]. Tuttavia, a causa della mancanza di dati, le funzioni di distanza generalmente non obbediscono alla proprietà della disuguaglianza triangolare[12] e non possono essere considerate vere e proprie metriche matematiche. Ciò può portare ad un clustering poco intuitivo e instabile.
L'algoritmo *item-based nearest neighbor*, invece, riprende in parte la tecnica precedentemente vista per i sistemi *user-based*, solo che mentre quest'ultimo algoritmo generava suggerimenti sulla base di similarità tra gli utenti, l'algoritmo *item-based* li fornisce considerando similarità tra i prodotti. La previsione di un prodotto avviene grazie alle valutazioni di elementi simili da parte dell'utente. Di seguito la formulazione matematica di tale previsione per l'user *u* e l'item *i* data dalla somma pesata dei *rating* dell'utente *u* per i prodotti più simili ad *i*.

[12] La distanza tra due punti gode di quattro proprietà: non negatività, identità, simmetrie e disuguaglianza triangolare.

$$pred(u,i) = \frac{\sum_{j \in ratedItems(u)} itemSim(i,j) \times r_{ui}}{\sum_{j \in ratedItems(u)} itemSim(i,j)} \qquad [1.5]$$

Tale formula non presenta la correzione per la media come per le formule [1.3] e [1.4], proprio perché i *rating* provengono dall'utente.

In questo caso, $itemSim(i,j)$ rappresenta la misura di similarità tra gli *item*, che può essere calcolata mediante l'*adjusted-cosine similarity*. Questa metrica, valida per una coppia di elementi *i* e *j*, si ritiene che sia, oltre che la più popolare, anche la più accurata e si calcola usando tutti gli individui che hanno espresso una valutazione sia per *i* che per *j*.

$$itemSim\ (i,j) = \frac{\sum_{u \subset RB_{I,J}} (r_{ui} - \overline{r_u})(r_{uj} - \overline{r_u})}{\sqrt{\sum_{u \subset RB_{i,j}} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{u \subset RB_{i,j}} \left(r_{uj} - \overline{r_u}\right)^2}} \qquad [1.6]$$

Dove RB è il *set* di utenti che hanno espresso una valutazione sia per i prodotti *i* che per *j*.

Come per il coefficiente di Pearson, anche questa misura varia tra -1 e 1.

In merito all'algoritmo *item- based*, si può asserire che in linea teorica tale modello può essere tanto grande quanto il quadrato del numero di *item*, anche se in pratica si può ridurre in modo sostanziale questa dimensione memorizzando solo le correlazioni per le coppie di item con più di un numero *k* di *corating*. Bisogna, però, tenere anche in considerazione che eliminare tante correlazioni potrebbe portare delle difficoltà nelle previsioni per un dato *item* e utente, in quanto i prodotti correlati con le valutazioni dell'individuo potrebbero non contenere l'*item* in target.

Nonostante queste criticità, ci sono evidenze che gli algoritmi *item-based* sono più accurati rispetto a quelli *user-based*[13].

L'ultimo tipo di algoritmo non probabilistico è il modello della riduzione della dimensionalità (*dimensionality reduction*). In generale, questi sono algoritmi che riducono il volume dei dati e la complessità del settore attraverso la mappatura dello spazio, in cui è presente l'*item* in un numero minore di dimensioni sottostanti. Alla base di ciò vi è l'idea che questo sottospazio dimensionale risulta più compatto e potrebbe presentare temi o caratteristiche "nascoste" presenti in quel prodotto. Per questo motivo si adopera questa tecnica di *mapping* tra i *rating* dell'utilizzatore e le dimensioni sottostanti il prodotto, ovvero i potenziali "gusti" latenti del

[13] Sawar, B., Karypis, G., Konstan, J.A., Riedl, J., 2001: Item-Based Collaborative Filtering Recommendation Algorithms. Proceedings oft he 10th international conference on World Wide Web, Hong Kong, ACM Press, pp. 285-295

consumatore: la previsione di un *item*, quindi, potrebbe essere generata dai "gusti" nascosti dell'utente. La funzione di mappatura consiste generalmente in operazioni vettoriali, tra le quali ci sono l'analisi fattoriale e l'analisi delle componenti principali.
Se, però, da un lato queste tecniche riducono la mole di dati a disposizione e agevolano la complessità computazionale dell'algoritmo di apprendimento, dall'altro è necessario tenere in considerazione che le informazioni, che derivano da questo modello possono risultare compromesse o incomplete e in questo modo ne risentirebbero anche le prestazioni predittive dell'algoritmo[14].

1.5.2. Algoritmi probabilistici

Gli algoritmi probabilistici riprendono esplicitamente le distribuzioni di probabilità nel calcolo delle previsioni dei *rating* o delle liste di raccomandazioni classificate. Nello specifico, molti di questi calcolano la probabilità che, dato un utente *u* e un *item i*, per cui si è espressa un'opinione in termini di rating, quell'utente assegni all'*item* una valutazione di *r*: ossia $p(r|u,i)$. In questo caso, la previsione della valutazione sarà basata o sul valore più probabile del *rating* oppure sul valore atteso di *r* per quel prodotto *i*. Per quest'ultima circostanza si avrà la seguente formula [1.7]:

$$E(r|u,i) = \sum_{r} r \cdot p(r|u,i) \qquad [1.7]$$

Ossia il valore atteso di *r* dato l'utente *u* e l'*item i* è uguale alla sommatoria di tutti i valori di *r* per la probabilità di *r* condizionata al fatto che un *user* attribuisca al prodotto *i* una valutazione di *r*: in questo modo si misura la probabilità che un utente per un determinato prodotto dia una valutazione *r*.
I modelli più utilizzati in questo ambito sono le reti bayesiane (*Bayesian networks*), che calcolano la dipendenza probabilistica per gli utenti o gli *item*.
In generale, una rete bayesiana è rappresentata da grafi diretti aciclici (DAG), che danno informazioni riguardo alle dipendenze condizionali di variabili stocastiche [30]. Su un piano più formale, i nodi di questo modello rappresentano variabili casuali in senso bayesiano; ossia,

[14] Ulteriori informazioni al seguente sito: http://www.andreaminini.com/ai/machine-learning/riduzione-dimensionalita-dati

quantità osservabili, variabili latenti, parametri sconosciuti o ipotesi; ad ognuno di essi (tranne per quelli iniziali: A e C nella Figura 1.4) è associata una funzione di probabilità che assegna alla variabile del nodo un valore influenzato dai valori delle variabili presenti nei nodi soprastanti. Gli archi, invece, esprimono condizioni di dipendenza tra due nodi e i nodi che non sono connessi rappresentano variabili condizionalmente indipendenti tra di loro.

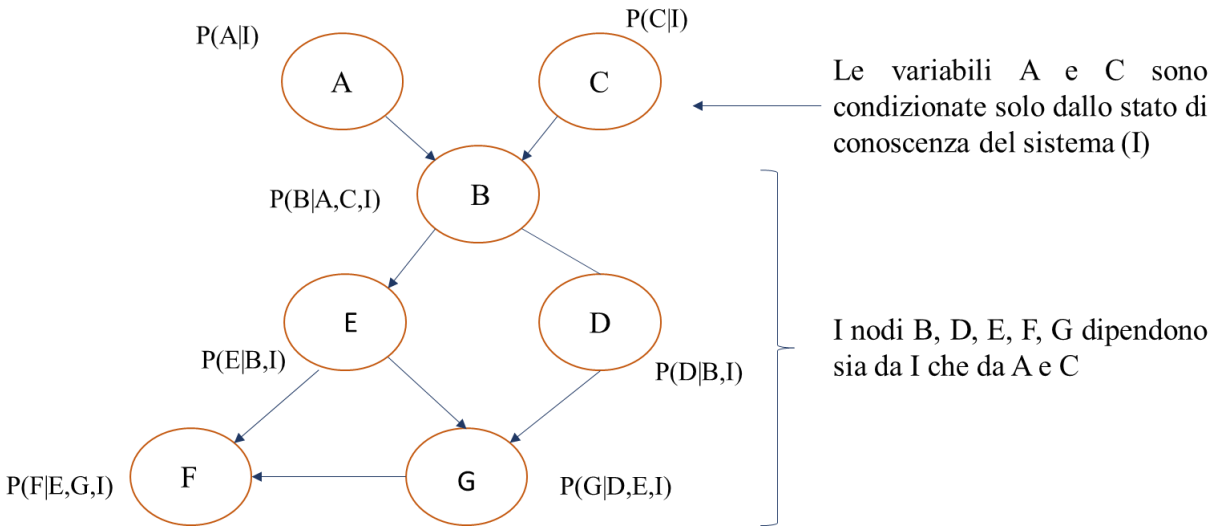


Figura 1.4: Esempio di rete bayesiana

I primi algoritmi probabilistici[15] di *collaborative filtering* applicavano le reti bayesiane attraverso l'utilizzo di *decision tree*. In questo caso, il ramo dell'albero dipende dal *rating* o dall'assenza di *rating* espressa dall'utente per un particolare *item* e i nodi memorizzano un vettore di probabilità per le valutazioni del prodotto oggetto di raccomandazione da parte dell'*user*.

Accanto allo studio e uso delle reti bayesiane, sono state condotte anche numerose ricerche per sviluppare tecniche probabilistiche di clustering o riduzione della dimensionalità.

Queste ultime introducono una variabile latente $p(z|u)$, che indica la probabilità che un utente appartenga alla classe latente *z*. La probabilità che un user *u* dia ad un item *i* un rating del valore di *r*, è data dalla seguente formula [1.8]:

[15] Breese, J.S., Heckerman, D., Kadie, C.: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In Proceeding of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI). (1998) Madison, Wisconsin. Morgan Kaufmann p. 43-52

$$p(r|u,i) = \sum_{z} p(r|i,z)p(z|u) \qquad [1.8]$$

La corrispondente previsione di *r* non è altro che il valore atteso del *rating* (Formula [1.9])

$$E(r|u,i) = \sum_{r} \left(r \cdot \sum_{z} p(r|z,i)p(z|u) \right) \qquad [1.9]$$

Mediante l'algoritmo *expectation maximization* (EM) è possibile stimare le classi latenti z applicando la distribuzione di probabilità di Gauss. Per fare ciò, è necessario prima di tutto avere un modello di dati completi: in questo modello, si ipotizza di osservare realmente le varabili latenti. Per questo motivo, si assume che ogni coppia osservata *user-item (u,y)* corrisponde realmente ad una tripletta *(u,y,z)* e il modello di dati completo sarà dato da $p(y,z|u) = P(y|z)P(z|u)$.

Nello specifico, si ipotizza che i dati siano stati generati da un mix di distribuzioni, per cui ogni classe presenterà dei dati in base alla specifica distribuzione che li ha generati, anche se al termine della generazione i *pattern* si presentano come prodotti da un'unica distribuzione multi-modale. Il clustering con l'algoritmo EM ha l'obiettivo di risalire (a partire dai pattern del *training set*) ai parametri delle singole distribuzioni; per questo motivo si considera conosciuta la forma delle distribuzioni e si suppone, per semplicità, che queste siano tutte dello stesso tipo. Il caso più comune è quello di mix di *s* distribuzioni multinormali (gaussiane), di cui si vogliono stimare i parametri di definizione. Per tale stima si utilizza il criterio di massima verosimiglianza (*maximum likehood*)[16].

Tra i vantaggi degli algoritmi probabilistici vi è la possibilità di produrre una distribuzione di probabilità tra i possibili valori di *rating*, in modo tale da avere informazioni sulla probabilità di tutti questi valori. Da ciò ne deriva che non solo si può calcolare il *rating* più probabile, ma anche la probabilità che quel *rating* sia corretto, ossia di comprendere la fiducia dell'algoritmo.

[16] Ulteriori informazioni al seguente sito: http://bias.csr.unibo.it/maltoni/ml/DispensePDF/6_ML_Clustering.pdf

1.6. Content-based filtering

Accanto al *collaborative filtering*, algoritmi *content-based* costituiscono un ulteriore tipo di sistema di raccomandazione.

Tale sistema suggerisce dei prodotti ad un utente, simili rispetto a quelli per cui quest'ultimo ha dimostrato interesse in passato. Questo tipo di sistema di raccomandazione parte dall'assunto di base che, se ad esempio un individuo ha assegnato un *rating* positivo ad un film che appartiene al genere commedia, gli algoritmi di questo tipo andranno a raccomandargli ulteriori film che appartengono allo stesso genere [30].

In particolare, sistemi di raccomandazione *content-based* suggeriscono ad un utente un *item* sulla base della descrizione e caratteristiche del prodotto e al profilo di interessi dell'individuo. Alcuni primi lavori che impiegavano questo tipo di raccomandazioni utilizzavano le *query*[17] per costruire modelli per gli utenti. In questo modo, infatti, un modello utente si basava su una o un insieme di *query*, in grado di recuperare informazioni aggiuntive o nuove per l'utente. Successivamente sono stati adottati metodi fondati proprio sul recupero di informazioni (*information retrieval*), quali l'algoritmo di Rocchio e tf*idf, che saranno successivamente oggetto di trattazione.

Su un piano teorico si ha, quindi, un sistema che presenta una serie di differenti prodotti ad un utente e quest'ultimo seleziona tra gli *item* per ricevere più dettagli o per interagire con gli stessi. I sistemi di raccomandazione basati sul contenuto effettuano, dunque, un'analisi delle caratteristiche di un *item* per identificare quei prodotti che potrebbero calzare con gli interessi dell'utente.

Gli *item*, però, si mostrano in modo differente a seconda del sistema e del *dataset* in cui si trovano: essi, infatti, possono presentarsi sottoforma di dati strutturati o non strutturati. Nel primo caso, ogni *item* è caratterizzato dallo stesso *set* di attributi, variabili o campi e sono noti i valori che possono assumere le variabili. Inoltre, un codice ID permette di distinguere tra gli *item* che potrebbero avere alcuni attributi uguali. A titolo di esempio, la tabella 1.1 riassume quanto appena asserito.

[17] Il termine *query* (letteralmente interrogazione) permette di ricercare ed estrarre dei dati da un *database* secondo determinati filtri e regole.

Tabella 1.1 – Database di un ristorante

ID	NOME	CUCINA	SERVIZIO	COSTO
120001	Albert Pizza	Italiana	Tavolo	Medio
120002	Hiro-Chan	Cinese	Buffet	Basso
120003	Jean Bistro	Francese	Tavolo	Alto

Molto spesso si ha a che fare, però, con dati non strutturati, dove non sono etichettati degli attributi, né sono riportati valori ben definiti di essi: è il caso di articoli o *review*, in cui si è di fronte alla complessità del linguaggio e alla presenza di parole polisemantiche o di sinonimi. Moli settori presentano dati semi-strutturati caratterizzati da alcuni attributi, a cui è associato un *set* ristretto di valori e alcuni campi di testo, molto spesso difficili da analizzare dagli algoritmi così come si presentano; di conseguenza, si cerca di trasformarli in dati strutturati attraverso l'utilizzo di varie tecniche. Si può, infatti, considerare ciascuna parola presente in un articolo come un attributo a cui è connesso un valore booleano, che indica se quella parola è presente o meno nell'articolo, oppure un valore intero, il quale segna il numero di volte in cui si trova il termine nel testo.

Molti sistemi di personalizzazione che analizzano testi usano una tecnica per creare una rappresentazione strutturata proveniente da sistemi di ricerca testuale. Nello specifico, si considerano le radici delle parole attraverso il processo di *stemming*, con l'obiettivo di creare un termine che rifletta il comune significato di termini, che si presentano con desinenze o suffissi differenti, ma che sono accumunati dalla stessa radice, quindi condividono lo stesso significato di fondo. Il valore della variabile associata al termine, calcolato mediante la funzione di peso *tf*idf*, è un numero reale, che denota l'importanza o la rilevanza del termine.

L'algoritmo *tf-idf*, acronimo di frequenza del termine-frequenza inversa nei documenti, permette di estrarre informazioni di un testo da una base dati. Questo algoritmo è composto da due parti: la prima (*tf*) determina il rapporto tra il numero di volte in cui il termine *t* è presente nel documento *d* e il numero totale delle parole presenti nello stesso documento; la seconda parte (*idf*) è data dal logaritmo del numero totale di documenti presenti nel *database* sul numero di documenti in cui è presente la parola *t*.

La funzione *tf*idf* (*term frequency–inverse document frequency*) di un termine *t* in un documento *d*[18] è, dunque, una funzione della frequenza di *t* nel documento $(tf_{t,d})$, dove (df_t) rappresenta il numero di documenti che contengono quel termine, N è il numero totale di

[18] Nel contesto dei sistemi di raccomandazione, i documenti corrispondono alla descrizione del testo associata ad un item oggetto di raccomandazione.

documenti all'interno del dataset e $max_z f_{z,d}$ è la frequenza massima di tutti i termini *z* presenti nel documento.
Di seguito la formula della funzione *tf*idf*:

$$TF - IDF(t,d) = \frac{tf_{t,d}}{max_z f_{z,d}} \cdot log \frac{N}{df_t} \qquad [1.10]$$

I termini con il peso maggiore sono quelli che ricorrono più spesso all'interno di un determinato documento rispetto agli altri (documenti) e per questo possono essere considerati più rappresentativi e centrali per l'argomento del documento stesso; anche se la funzione non è in grado di comprendere le relazioni logiche tra i termini all'interno della descrizione, né di dare informazioni circa il contesto in cui il termine è usato.
Il parametro *tf-idf*, quindi, assume un valore più elevato quando una parola è presente spesso in un documento, ma non è frequente nella totalità dei documenti del *dataset*: in questo modo, si permette di valorizzare meno parole molto comuni come articoli, preposizioni e congiunzioni, sempre presenti nella base dati.
Per quanto riguarda il profilo di interessi dell'utente, è necessario focalizzarsi principalmente su due tipi di informazione: *in primis* bisogna considerare le preferenze dell'individuo e quindi avere una descrizione dei tipi di *item* a cui questo è interessato [42]. Analiticamente questo è dato da una funzione che per ogni *item* restituisce la probabilità che quest'ultimo rientri nelle preferenze dell'utente. Assume, inoltre, molta importanza avere uno storico delle informazioni delle interazioni dell'*user* con il sistema di raccomandazione: questo include, ad esempio, avere memoria all'interno del sistema di ciò che l'individuo ha ricercato, visualizzato, espresso un *rating* oppure acquistato. Tutto ciò è molto utile, perché permette al sistema di rendere visibile all'utente i prodotti appena visualizzati oppure che l'individuo ha acquistato: l'algoritmo di *machine learning* sfrutterà queste informazioni per creare un modello per quell'utente (*user model*).
Di tale modello si serviranno quindi i sistemi di raccomandazione per fornire suggerimenti finali agli utenti: tale *user model* sarà basato sull'approccio di customizzazione dell'utente (*user customization*) oppure su un criterio *rule-based*. Per il primo metodo, il sistema di raccomandazione fornisce agli utenti un'interfaccia composta di box, in cui gli individui possono selezionare tra valori noti degli attributi o di *form*, all'interno dei quali poter inserire una descrizione di testo degli *item*: questo permette agli utenti di costruire una rappresentazione dei propri interessi e preferenze [30]. Una volta che gli utenti hanno inserito queste informazioni, l'algoritmo effettua un *match* tra gli elementi presenti nel *database* e le preferenze

espresse dagli utenti per raccomandare i prodotti più adatti per quest'ultimi. Questi sistemi di customizzazione degli utenti richiedono, però, uno sforzo notevole da questi ultimi, che molto spesso non sono disposti a rivelare le loro preferenze. Inoltre, è da tenere in considerazione che gli interessi degli *user* potrebbero cambiare nel tempo e per di più questi sistemi non prevedono un modo di determinare l'ordine in cui presentare tali *item*.

I sistemi di raccomandazione basati su regole, invece, suggeriscono altri prodotti agli utenti secondo regole che tengono in considerazione scelte passate dell'utente: ad esempio, un sistema può prevedere una regola, la quale raccomanda il *sequel* di un libro o un film ad utenti che in precedenza avevano acquistato il primo "capitolo" della serie di quel libro o pellicola. Tali sistemi potrebbero non fornire, tuttavia, raccomandazioni con un alto livello di personalizzazione.

La creazione di un modello di preferenze dell'utente basate sulle scelte e interazioni passate di quest'ultimo è una forma di apprendimento di classificazione (*classification learning*). Gli algoritmi di *classification learning* sono fondamentali per i sistemi di raccomandazione basati sul contenuto, in quanto prevedono una funzione che dà informazioni circa l'interesse dell'utente nei confronti di un nuovo *item*. Tale funzione può fornire una stima della probabilità che a un individuo potrebbe interessare un prodotto non ancora visionato, oppure l'algoritmo genera una funzione che predice direttamente un valore numerico, indice, ad esempio, del grado di interesse dell'user. Alcuni di questi algoritmi lavorano con dati strutturati; nel caso in cui questi si trovino ad operare in *dataset* con elementi non strutturati come fattori di testo, questi possono essere convertiti in dati strutturati selezionando un piccolo sottoinsieme di quei termini e considerandoli come attributi.

Gli algoritmi di *classification learning* sono i seguenti: *Decision Trees*, metodi del *Nearest Neighbor*, i *feedback* di rilevanza e l'algoritmo di Rocchio, Classificatori Lineari (*linear classifiers*) e *Naïve Bayes*.

Gli alberi di decisione (*decision trees*) sono costruiti secondo la logica di suddividere (split) i dati presenti nel sistema in sottogruppi fino a quando questi conterranno solo istanze di una singola classe e secondo un criterio *top down*, basato sulla ricerca e scelta degli attributi senza tornare a rivalutare le precedenti decisioni.

Inizialmente si ha un albero con una sola radice a cui sono assegnate tutte le istanze di addestramento; successivamente l'algoritmo, in base ad alcuni criteri[19], seleziona un attributo

[19] Tra i criteri che l'algoritmo sfrutta per scegliere l'attributo da assegnare al nodo dell'albero vi sono: *information gain* o criterio di massima riduzione di incertezza e il *gain ratio*. Il primo seleziona l'attributo, che è in grado di partizionare "meglio" le istanze che appartengono a classi differenti. Il *gain ratio*, invece, corregge l'*info gain*, in quanto prende in considerazione il numero e la dimensione delle partizioni quando sceglie un attributo. Ulteriori

e da questo attributo si avrà un numero di nodi pari ai valori dell'attributo che è stato selezionato [7]. Si procede, infine, ricorsivamente considerando come nuove radici i nodi figli dell'originaria radice. Nel momento in cui le istanze in un nodo appartengono alla medesima classe, l'algoritmo si arresta. Inoltre, per non incorrere in situazioni di *overfitting*, ossia di eccessivo adattamento[20] durante il processo di apprendimento induttivo, l'algoritmo determina gli attributi irrilevanti per la suddivisione delle istanze e rimuove i rispettivi nodi in modo da unire le istanze al livello superiore (fase di potatura).

Tale metodo si presta ad essere utilizzato con dati strutturati e non è efficace quando si hanno numerose classi e pochi esempi. Nonostante ciò, quando vi sono pochi attributi strutturati, le prestazioni e la comprensibilità degli alberi decisionali per i modelli basati sui contenuti rendono questi algoritmi molto vantaggiosi.

Per quanto riguarda il *Nearest Neighbor*, già trattato nel paragrafo precedente in merito ai sistemi di raccomandazioni *collaborative filtering*, si può asserire che questo algoritmo utilizza i dati presenti nella memoria del sistema, che in questo caso sono descrizioni testuali di *item* etichettati in maniera implicita o esplicita dall'utente. L'algoritmo in questione procede, dunque, a classificare un nuovo *item* attraverso il confronto di quest'ultimo con quelli già memorizzati, mediante una funzione di similarità e determina, in questo modo, l'*item* o i *k item* più vicini a quello oggetto di studio, che verrà etichettato e assumerà valori dei *k item* più vicini.

La funzione di similarità utilizzata per l'algoritmo del *nearest neighbor* dipende dalla tipologia di dati: se si parla di dati strutturati, si usa molto spesso la distanza euclidea; quando si impiega il modello vettoriale dello spazio, la tecnica euristica della similarità del coseno risulta più adatta. Nella funzione di distanza euclidea, non si distingue tra caratteristiche di due esempi con un valore alto e quelle con un valore più basso. Al contrario, la funzione di similarità del coseno non avrà un valore alto, se le caratteristiche corrispondenti ai due esempi hanno valori piccoli. Di conseguenza, questo è appropriato per il testo quando si vogliono due documenti che devono essere simili, in quanto riguardano lo stesso argomento, ma non quando entrambi non si riferiscono ad un argomento.

Tale approccio spaziale vettoriale e la funzione di similarità del coseno sono stati applicati a diversi algoritmi di classificazione dei testi e, nonostante la semplicità nel calcolo, registrano risultati molto simili ad algoritmi più complessi.

---

informazioni nei seguenti siti: https://vitolavecchia.altervista.org/caratteristiche-degli-alberi-di-decisione-decision-tree-in-informatica/, http://staff.icar.cnr.it/manco/Teaching/2005/datamining/lezioni/lezione5.pdf.

[20] Il concetto di *overfitting* fa riferimento, in linee generali, ad un quasi perfetto adattamento di un modello statistico ai dati oggetto di osservazione; nel caso di algoritmi di apprendimento questo si traduce in una forte relazione tra un attributo e il risultato finale dell'analisi. Questa situazione è dovuta ad un numero di attributi molto superiore rispetto al numero di osservazioni.

I *feedback* di rilevanza sono, invece, metodi che aiutano gli utenti a migliorare efficacemente il reperimento di documenti mediante l'utilizzo di *queries* di ricerca composte da un *set* di parole chiave. Il principio di fondo di tali criteri consiste, quindi, nel consentire agli utenti di valutare i documenti restituiti dal sistema di recupero rispetto alle loro esigenze informative. Questa forma di *feedback* può essere utilizzata anche per affinare in modo incrementale la *query* iniziale; così come per la valutazione tramite *rating* degli *item*, esistono mezzi espliciti e impliciti per raccogliere i dati di *feedback* di rilevanza. A tal proposito, l'algoritmo di Rocchio è un algoritmo di *feedback* di rilevanza ampiamente utilizzato, che opera nel modello dello spazio vettoriale e si basa sulla modifica della *query* iniziale mediante insiemi di documenti rilevanti e non rilevanti con pesi diversi. La seguente formula riassume l'algoritmo formalmente [1.11]:

$$\overrightarrow{q_m} = \alpha\, \overrightarrow{q_0} + \beta \frac{1}{|D_r|} \sum_{\overrightarrow{d_j} \in D_r} \overrightarrow{d_j} - \gamma \frac{1}{|D_{nr}|} \sum_{\overrightarrow{d_j} \epsilon\, D_{nr}} \overrightarrow{d_j} \quad [1.11]$$

Dove $\overrightarrow{q_m}$ rappresenta il valore della query modificata, $\overrightarrow{q_0}$ il vettore originario della query, D_r e D_{nr} sono rispettivamente i documenti ritenuti rilevanti e non rilevanti e α, β, γ costituiscono i pesi scelti manualmente o in modo empirico.

La formula [1.11] permette in sostanza di spostare in modo incrementale il vettore di *query* verso cluster di documenti rilevanti e di allontanarsi, di conseguenza, dai documenti irrilevanti, in modo da avere un *feedback* solo positivo (per cui il valore di γ sarebbe 0). Sebbene questo obiettivo costituisca una giustificazione intuitiva dell'algoritmo di Rocchio, non esistono fondamenti teorici alla base della formula e questo porta a non garantire le prestazioni. A livello empirico, però, sono stati eseguiti diversi esperimenti in grado di dimostrare che l'approccio porta a miglioramenti significativi nelle prestazioni di recupero: in uno di questi lavori, vari studiosi hanno utilizzato una variante dell'algoritmo di Rocchio in un contesto di apprendimento automatico e, nello specifico, per impostare un profilo utente da un testo non strutturato. L'obiettivo di queste applicazioni è quello di indurre automaticamente un classificatore di testo che possa distinguere tra classi di documenti. In questo contesto, si presume che non esista alcuna *query* iniziale, e l'algoritmo formi prototipi e insiemi per classi in modo analogo all'approccio di Rocchio, ossia come somme vettoriali sui documenti appartenenti alla stessa classe. L'algoritmo avrà come risultato un insieme di vettori pesati, la cui vicinanza a documenti non ancora etichettati può essere utilizzata per assegnare questi ultimi ad una classe di appartenenza.

In merito ai classificatori lineari, si può affermare che questi sono algoritmi che operano negli iperpiani che separano le istanze in uno spazio multidimensionale. Esistono molti classificatori lineari, ma tutti sono descritti da elementi comuni. In generale, infatti, il risultato del processo di apprendimento è un vettore pesato *n*-dimensionale *w*, il cui prodotto di punti con un'istanza *n*-dimensionale (e.g. un documento di testo rappresentato nel modello spaziale vettoriale) produce una previsione numerica. Il mantenimento della previsione numerica porta ad un approccio di regressione lineare, anche se può essere utilizzata una soglia per convertire le previsioni continue in etichette di classe discrete. Mentre questo quadro generale vale per tutti i classificatori lineari, gli algoritmi differiscono nei metodi di formazione utilizzati per derivare il vettore di peso *w*. Tra i criteri utilizzati in questo senso vi è la regola di Widrow-Hoff o regola delta.

Il Classificatore *Naïve* di Bayes utilizza un approccio probabilistico per la classificazione dei testi molto utilizzato per la sua efficacia dai ricercatori.

In particolare, il Classificatore Bayesiano naif calcola, come a livello generale aveva previsto il teorema di Bayes[21], la probabilità che un documento *d* appartenga alla classe *c* [1.12]:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \qquad [1.12]$$

In base a tale approccio, la classe di appartenenza del documento sarà quella che presenterà maggiore probabilità secondo la [1.13]:

$$c = argmax_{c_j} \frac{P(c_j)P(d|c_j)}{P(d)} \qquad [1.13]$$

Dove c_j rappresenta una generica classe.

L'algoritmo Naïve Bayes opera, tuttavia, una semplificazione (da qui l'appellativo "naïve"), in quanto assume che, data la classe *c* di appartenenza, ciascuna parola contenuta in un dato documento *d* sia condizionatamente indipendente: questo perché risulta arduo stimare $P(d|c)$ facendo solo riferimento ai dati osservati nel *training dataset*. Le probabilità individuali delle parole presenti nel documento sono, di conseguenza, stimate singolarmente piuttosto che considerando il documento nel suo insieme. Sebbene questa condizione risulti contrastante con

[21] Il teorema di Bayes prevede, infatti, di calcolare la probabilità condizionata di un evento A dato B, se si conoscono le probabilità a priori P(A) e P(B) e la probabilità condizionata P(B|A), come nella seguente formula $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$.

quanto accade nella realtà, tale metodo si è dimostrato empiricamente molto efficace per la classificazione di documenti testuali.

Le due formulazioni comunemente utilizzate del classificatore naïve di Bayes e studiate in particolare dai ricercatori Mc Callum e Nigam [22] sono il modello multivariato di Bernoulli e il modello multinomiale.

Per entrambi i modelli un documento è rappresentato da un vettore di valori sull'insieme di vocaboli, V, dove ogni voce nel vettore indica se una parola si trova all'interno del documento. Nello specifico, il modello multivariato di Bernoulli codifica ogni termine come attributo binario, e quindi riporta se una parola è apparsa o meno; mentre il modello multinomiale conta il numero di volte in cui il termine è apparso nel documento. Su un piano empirico, la formulazione multinomiale del classificatore naïve Bayes è preferita al modello multivariato di Bernoulli, soprattutto per grandi dimensioni del vocabolario di termini. Di seguito la formula del modello multinomiale per calcolare $P(c_j|d_i)$, che tiene conto soltanto di un sottoinsieme dei vocaboli *V* contente le parole che compaiono nel documento d_i [1.14]:

$$P(c_j|d_i) = P(c_j) \prod_{w \in V_{d_i}} P(t_k|c_j)^{N_{(d_i,t_k)}} \quad [1.14]$$

Dove $N_{(d_i,t_k)}$ è il numero di volte in cui una parola appare nel documento d_i.

Per una stima delle probabilità della parola $P(t_k|c_j)$ si utilizza un metodo di *smoothing* che non si limita al semplice conteggio, ma valorizza meno le parole poco presenti, andando in questo modo a modificare il valore delle probabilità. Inoltre, questo criterio evita di assegnare valori di probabilità pari a zero alle parole non presenti nel *training dataset* per una particolare classe. Tra gli altri metodi di *smoothing* vi è anche quello basato sulle comuni stime di Laplace (cioè, aggiungere una parola a tutti i termini contati per una classe).

I classificatori basati sul modello multinomiale registrano dei limiti in merito alle prestazioni, quando i documenti nel *training set* presentano una differente lunghezza e nel momento in cui vi sono pochi documenti formativi disponibili, quindi si è in presenza di categorie rare. Tali limiti emergono anche quando si esegue la profilazione dell'utente, poichè non è possibile ipotizzare la lunghezza dei documenti e in quanto insiemi di esempi negativi (ossia quando un

[22] Andrew McCallum e Kamal Nigam sono due ricercatori specializzati in Natural Language Processing, Information Extraction e Social Network Analysis.

utente invia un *feedback* negativo al sistema) risultano spesso molto minori rispetto a *set* di esempi positivi.

Nonostante, come affermato prima, alcuni metodi statistici, come il classificatore che utilizza il criterio del *nearest-neighbor*, risultino più efficienti a livello di prestazioni rispetto al classificatore naif di Bayes, quest'ultimo risulta più facile da implementare rispetto ad altri metodi di apprendimento.

Gli algoritmi precedentemente elencati suggeriscono differenti metodi di apprendimento per individuare interessi dell'utente utilizzando raccomandazioni basate sul contenuto; bisogna, tuttavia, tenere in considerazione che tali raccomandazioni risultano realmente valide, se il contenuto che si va ad analizzare è in grado di distinguere e dare informazioni circa gli *item* che piacciano all'*user* e prodotti per cui questo non dimostra interesse. Vi sono, inoltre, situazioni, come raccomandare dei film, libri o ristoranti, in cui la presenza di informazioni strutturate (e.g. genere di un film o tipo di un ristorante) fa sì che i sistemi basati sul contenuto siano più adatti all'analisi. Quando questo non è possibile, si ricorre ad altre tecniche di raccomandazione, quali, ad esempio, i sistemi di raccomandazioni *collaborative filtering*.

Sistemi di raccomandazione *content-based* possono, infine, essere utilizzati per filtrare risultati che derivano da altri metodi come il *collaborative filtering*. A titolo di esempio, se un algoritmo di *collaborative filtering* suggerisce che individui che acquistano giocattoli comprano anche film horror, l'ulteriore applicazione di un algoritmo basato sul contenuto potrebbe portare a non raccomandare all'utente giocattoli in alcune situazioni. Vi sono, inoltre, alcuni casi in cui sistemi non raccomandano *item* che risultano terminati e non più disponibili.

1.7. Problematiche degli algoritmi Collaborative Filtering e Content-based

Gli algoritmi utilizzati per fornire raccomandazioni basate sulla collaborazione e sul contenuto possono presentare delle problematiche, che verranno esposte di seguito.

Tra i principali limiti di questi sistemi vi è il cosiddetto problema del *cold start*: tale problematica si verifica quando nuovi utenti entrano nel sistema o nuovi *item* sono aggiunti al catalogo di quelli disponibili. In questa situazione, non è, quindi, possibile prevedere le preferenze dei nuovi individui, né gli utenti possono valutare o acquistare i nuovi *item*. Per arginare questo problema è possibile utilizzare vari modi: si può chiedere all'inizio all'utente di valutare alcuni *item* oppure di fornire esplicitamente informazioni aggregate circa le sue

preferenze [7]. Si possono, inoltre, suggerire all'utente prodotti in base alle informazioni demografiche raccolte, che possono essere impiegate per localizzare l'individuo, per comprendere mediante *zip-code* le interazioni di questo con il nuovo sistema e fornire in questo modo raccomandazioni di *item* sulla base di *rating* di altri individui simili che mostrano simili informazioni demografiche. In alcuni settori, vi sono degli item chiamati "sleepers", che potrebbero rappresentare potenzialmente dei validi suggerimenti per l'utente, ma sono privi di *rating*, per cui possono essere gestiti mediante l'utilizzo di metadati o metodi *content-based*.

Anche il problema della sparsità (*sparsity* in inglese) rappresenta un limite al corretto ed efficace funzionamento degli algoritmi di raccomandazione. Tale limite consiste nell'avere una grande disponibilità di *item*, ma ogni utente è riluttante ad assegnare un *rating* agli oggetti presi in considerazione; di conseguenza si avranno molti profili di utenti "dispersi", con molti valori mancanti, e questo porterà a raccomandazioni meno accurate. Tecniche SVD e modelli di raccomandazione multidimensionale possono ridurre tale problematica.

Un ulteriore problema è rappresentato dalla sinonimia, ossia quando un *item* è rappresentato in due o più differenti varianti o con voci che presentano significato simile. In questo caso, il sistema di raccomandazione non riesce a identificare e comprendere se i termini rappresentano lo stesso *item* oppure no; per cui questa eccessiva variazione nell'utilizzo di differenti termini descrittivi porta gli algoritmi di raccomandazione a prestazioni inferiori. Infatti, quando i contenuti degli *item* vengono ignorati, la raccomandazione fornita dal sistema non considera l'associazione latente tra gli *item* e questa rappresenta una ragione per cui nuovi elementi non vengono raccomandati fin quando non sono valutati dagli utenti. Alcune tecniche come il Single Value Decomposition (SVD) e il Latent Semantic Indexing (LSI) possono risolvere questo tipo di problematica.

La scalabilità (*scalability*) si ritrova nei sistemi che utilizzano un approccio di *collaborative filtering*, quando è presente un dataset molto ampio, in cui il numero degli utenti e degli *item* cresce e il sistema richiede, quindi, più risorse per dare raccomandazioni più accurate all'utente. Al conseguente aumentare del numero dei *rating* il costo di aggiornamento del sistema diventa computazionalmente più alto.

Sistemi di tipo *collaborative* soffrono anche dei cosiddetti *shilling attacks*: si è in una situazione in cui un utente o un competitor entrano in un sistema e iniziano ad assegnare falsi *rating* ad *item*, al fine di aumentare (*push attack*) o ridurre (*nuke attack*) la popolarità di questi ultimi. Alcuni utenti, definiti *shiller* o *attacker*, falsificano dei profili inserendo *rating* distorti, per influenzare e manipolare le decisioni degli altri *user*. Tali attacchi mirano a intaccare la fiducia degli individui nei confronti dei sistemi di raccomandazione e allo stesso tempo riducono le

performance e la qualità dei suggerimenti forniti dagli algoritmi. Tra le metriche e le modalità per indagare, comprendere ed infine eliminare tale problema esistono degli indici di credibilità, che confrontano tutti i *rating* di un profilo di un utente con la varianza dei *rating* di specifici *item*. Vi sono anche metriche come il *detection rate* e i *false positive rate*[23], che cercano di individuare questi attacchi. In una prospettiva più generale, prima di risolvere tale problema è necessario comprendere le dimensioni e lo scopo alla base dello *shilling attack*.

Anche il *grey sheep* si verifica in sistemi *collaborative filtering*, quando le opinioni di un utente non incontrano quelle di nessun gruppo e quindi quell'utente non potrà usufruire dei benefici degli algoritmi di raccomandazione. Gli *user grey sheep* possono essere individuati e separati dagli altri utenti mediante l'applicazione di tecniche i clustering, quali il *k-mean*: ciò permette di ottenere migliori performance e di ridurre l'errore di fornire suggerimenti poco adeguati. Tale problematica può anche essere risolta quando gli *item* vengono suggeriti sulla base del profilo personale dell'utente. Inoltre, integrare tecniche di *collaborative filtering* con quelle *content-based* contribuisce a garantire suggerimenti più efficaci e in linea con gli interessi dell'utente.

Sistemi che adottano metodi di filtraggio collaborativo presentano anche il problema dell'inattività (*latency problem*): quando nuovi prodotti sono aggiunti più frequentemente ad un *dataset* e l'algoritmo utilizzato continua a suggerire solo gli *item* già valutati e tratta i prodotti appena inseriti come ancora non valutati. L'utilizzo di sistemi basati sul contenuto può essere conveniente in termini di tempo, ma potrebbe condurre al problema della super-specializzazione (*overspecialization*), che viene trattata nel paragrafo successivo.

Tipico, invece, di sistemi basati sul contenuto è la problematica dell'*overspecialization*: in molti settori sono, infatti, presenti alcuni rilevanti *item* che non possono essere suggeriti, finchè il contenuto analizzato non contiene informazioni sufficienti a distinguere le preferenze di un utente, oppure vi sono prodotti caratterizzati da uno scarso contenuto e quindi il sistema, in mancanza di adeguate informazioni, non riesce ad associarli ad altri contenuti simili e, dunque, a raccomandarli. Di conseguenza, l'algoritmo di raccomandazione suggerirà agli utenti *item* strettamente connessi ai loro profili, a volte ridondanti e noiosi, impedendo così di far scoprire agli individui qualcosa di veramente nuovo e differente.

Un ulteriore problema che caratterizza i sistemi di raccomandazione è legato alla *privacy*: un sistema deve garantire tramite meccanismi criptografici e di sicurezza, che le informazioni fornite dagli utenti siano protette e non visibili a terze parti o competitor.

[23] Il detection rate è dato dal rapporto tra il numero di profili rilevati e il numero di attacchi, mentre il false positive rate è calcolato sui profili falsificati degli utenti in rapporto ai profili non intaccati da questo tipo di problematica.

1.8. Sistemi ibridi di raccomandazione

Al fine di arginare alcune problematiche ricorrenti negli algoritmi dei sistemi *collaborative filtering* e basati su contenuto, sono stati implementati sistemi ibridi di raccomandazione, i quali uniscono tecniche proprie dei precedenti sistemi per ottenere performance più soddisfacenti. In linee generali, i criteri adottati dai sistemi ibridi possono prevedere l'applicazione dei criteri di *collaborative filtering* e *content based* in maniera separata, incrociando solo successivamente i risultati; oppure all'approccio collaborativo vengono incorporate caratteristiche degli algoritmi che forniscono raccomandazioni basate su contenuto e viceversa [42]. Un ulteriore metodo ibrido di raccomandazione consiste nella costruzione di un modello più complesso che cerca di unire l'approccio collaborativo con quello *content-based*.

Tra i tipi di sistemi ibridi di raccomandazione se ne identificano sette: pesato, *switching*, misto, *feature combination*, *feature augmentation*, *cascade* e *meta-level*.

I metodi ibridi appena citati possono, inoltre, essere suddivisi in due categorie in base al fatto che tali criteri tengano o meno in considerazione l'ordine con cui vengono combinati i due originari algoritmi: nello specifico, per i metodi *weighted*, *mixed*, *switching* e *feature combination* l'ordine degli approcci utilizzati non influisce; viceversa per i sistemi *feature augmentation*, *cascade* e *meta-level*.

Il più semplice tra i sistemi ibridi è quello pesato (*weighted*), in cui ogni componente del sistema segna un dato *item* e i punteggi di questi ultimi sono in seguito combinati secondo dei pesi. In particolare, vi è una fase comune a vari sistemi ibridi, in cui ogni raccomandazione individuale opera nei cosiddetti *"training data"*. Successivamente, si procede con la generazione dei candidati, molto importante per comprendere quali *item* prendere in considerazione: quando, infatti, si genera una previsione per il cosiddetto *test user*, i due sistemi di raccomandazione utilizzati proporranno congiuntamente dei candidati. Il sistema ibrido può trattare l'insieme dei candidati risultanti da ciascun tipo di sistema di raccomandazioni in due modi differenti: in un primo caso, si effettua l'intersezione dei candidati totali, quindi si gestirà un numero più piccolo di casi; un secondo scenario prevede, invece, l'unione dei candidati generati dai due sistemi di raccomandazione. In quest'ultima circostanza, il sistema ibrido deve decidere come gestire casi in cui uno dei due tipi di raccomandazione (che per comodità nella trattazione si chiameranno A e B) non riesce a valutare un dato candidato[24]. Una volta espressi i *rating* per ogni candidato sia da A che da B, si calcola una combinazione lineare tra i due punteggi, che andranno a

[24]Per risolvere questo problema, è possibile assegnare un valore neutro al candidato, né positivo né negativo.

costituire il *rating* del item previsto. Questo tipo di sistema si basa sull'assunzione implicita che ogni tipo di sistema di raccomandazione utilizzato sia uniforme nello spazio degli oggetti, per cui se si adopera un criterio *collaborative* in un dominio con poche valutazioni, questo metodo avrà assegnato un peso inferiore rispetto alle altre tecniche impiegate. Un esempio che sintetizzi e chiarisca quanto affermato finora è stato presentato dallo studioso Mobasher e i suoi colleghi, i quali hanno trovato che un sistema di raccomandazione ibrido che combinava i risultati di un sistema *collaborative filtering* e *content-based* rispettivamente con un peso[25] di 60/40 generava previsioni molto accurate.
Di seguito la formula esplicativa [1.15]:

$$\hat{r} = 0.6 \times \hat{r}_{collaborative} + 0.4 \times \hat{r}_{content} \quad [1.15]$$

Sistemi ibridi di raccomandazione mista, come per i sistemi misti pesati, si compongono di tre fasi: la prima di training, perfettamente identica per ogni sistema di raccomandazione ibrido; un processo di generazione dei candidati e infine la fase di *scoring*. Nella seconda fase, i due sistemi di raccomandazione[26] partono dal profilo dell'utente per poter generare un *set* di candidati. Il risultato finale del sistema ibrido per un dato *item* sarà dato dalla combinazione dei risultati ottenuti mediante l'utilizzo dei due metodi, i quali risultati seguiranno un criterio di *ranking* delle liste dei *rating*, basato su un livello di personalizzazione dei rating stessi. Quest'ultima situazione, in cui si opera un *ranking* dei *rating*, si attua nel caso in cui i due sistemi utilizzati non portano a suggerimenti che possono essere combinati: di conseguenza, si stabilisce un metodo principale. Nel caso opposto, partendo sempre dal presupposto che i due tipi di algoritmi di raccomandazione utilizzati siano *collaborative* e *content-based*, si avrà un suggerimento che terrà conto rispettivamente della valutazione degli *user* e di un giudizio espresso in *rating* rispetto alle descrizioni testuali degli *item*.
Un sistema ibrido di raccomandazione cosiddetto *switching* seleziona, invece, un singolo tipo di sistema di raccomandazione (quindi *content-based* o *collaborative filtering*) sulla base dei differenti profili degli utenti. Vengono seguiti alcuni criteri di scelta basati su valori di confidenza dei componenti di raccomandazione ritenuti affidabili per determinare la

[25] Tali pesi possono essere calcolati in vari modi: si può adottare un approccio non propriamente scientifico di utilizzare differenti pesi e infine selezionare quelli che performano meglio per quel sistema, oppure si usa la regressione lineare.
[26] Per tutto il corso della trattazione sui sistemi ibridi di raccomandazione, i due tipi di sistemi di raccomandazione che devono essere combinati si considerano essere: *collaborative filtering* e *content based*.

componente da utilizzare: ad esempio, il sistema NewsDude, il quale raccomanda notizie, si basa su tre componenti di raccomandazioni: il *primo content-based nearest-neighbor*, il secondo di tipo collaborativo e il terzo *content-based*, che utilizza il classificatore naif di Bayes. Per prima viene utilizzata la tecnica del *nearest-neighbor*, che, se non produce una raccomandazione con un'alta confidenza, viene accantonata e si procede con il sistema di *collaborative filtering*. Tale tipo di selezione di un tipo di raccomandazione è dovuta al fatto che suggerire un determinato *item* ad un *user* sulla base di un metodo di raccomandazione sbagliato potrebbe portare a risultati non ottimali. Una volta che questa scelta è stata fatta, la componente "scartata" non ricoprirà alcun ruolo nel processo di raccomandazione.

Non può esattamente considerarsi un sistema ibrido di raccomandazione la combinazione di caratteristiche (*feature combination*), che prevede l'aggiunta di caratteristiche di una fonte, come raccomandazioni del tipo *collaborative filtering*, all'interno di un algoritmo che processa i dati con una fonte differente (e.g. raccomandazioni basate sul contenuto). Questa tecnica non può essere considerata un sistema ibrido di raccomandazione, poiché vi è solo un tipo di sistema di raccomandazione; si reputa ibrido in quanto adotta fonti di conoscenza e logiche diverse da quelle della tecnica applicata.

Simile alla tecnica del *feature combination* è la cosiddetta *feature augmentation*. In questo caso, si generano nuove caratteristiche per ogni *item* mediante l'utilizzo di logiche di raccomandazione del sistema non utilizzato principalmente. In ogni fase, il sistema secondario, ossia quello che "supporta" l'algoritmo principale che analizza i dati, intercetta gli elementi trattati dall'algoritmo principale e li arricchisce di fonti di conoscenza addizionali. In particolare, questa tecnica si affida ad un metodo di raccomandazione per produrre una valutazione di un prodotto e incorpora tale informazione nel calcolo del secondo sistema di raccomandazione. Questo metodo si differenzia dal *feature combination*, perché aggiunge un numero più piccolo di caratteristiche al sistema primario, che risulta molto più predominante e ben sviluppato rispetto al criterio di combinazione delle caratteristiche.

Il sistema ibrido a cascata (*cascade*) opera, invece, in maniera gerarchica: nello specifico, sono presenti due tipi di sistemi di raccomandazione, uno considerato come principale e l'altro come secondario, che può solamente rifinire i risultati ottenuti con il primo. L'approccio *cascade* ha il vantaggio di risultare molto semplice a livello computazionale e il procedimento che adotta permette di avere un *focus* solo sugli *item* ritenuti interessanti dal primo metodo di raccomandazione.

Il criterio *meta-level*, infine, utilizza un modello di raccomandazione come *input* per un'ulteriore tecnica di raccomandazione. Questo criterio può per certi versi essere paragonato

alla tecnica ibrida del *feature augmentation*, in quanto, anche in quel caso, vi è un primo sistema, che contribuisce ai risultati ottenuti però con il secondo tipo di sistema di raccomandazione. Il modello *meta-level* si differenzia, tuttavia, da quest'ultimo per il fatto che il primo sistema usato contribuisce con un modello di apprendimento, che a sua volta viene impiegato nei calcoli del secondo algoritmo utilizzato. Nell'approccio *feature augmentation* le performance risultati dall'applicazione del primo metodo di raccomandazione vengono considerate come informazioni più generali da trasmettere come *input* al secondo algoritmo di raccomandazione. Tra i vantaggi del sistema ibrido *meta-level* vi è la possibilità, per il secondo tipo di algoritmo utilizzato, di fornire suggerimenti più mirati ed efficaci, poiché quest'ultimo avvia la propria analisi da *input* che già in parte individuano interessi e preferenze dell'utente.
Lo scopo principale di utilizzare un sistema ibrido di raccomandazione consiste nel risolvere, o quantomeno ridurre, i problemi e i limiti presenti nei singoli tipi di algoritmi, perciò è importante selezionare i metodi di raccomandazione in modo tale che si compensino a vicenda.

Capitolo 2 - Valutazione dei sistemi di raccomandazione

Nel capitolo precedente si è fornita una panoramica dei principali tipi di sistemi di raccomandazione, con particolare *focus* sull'approccio collaborativo e di contenuto. In questa sezione si esamineranno, invece, alcune tecniche e metriche valutative in grado di fornire informazioni sull'affidabilità, accuratezza ed efficienza degli algoritmi di raccomandazione.

2.1 Overview

Come già asserito nel capitolo precedente, i sistemi di raccomandazione possono essere considerati come delle tecniche statistiche di filtraggio, sviluppate per arginare il problema degli utenti di disposizione di una mole troppo elevata di informazioni e per suggerirli, di conseguenza, prodotti maggiormente in linea con i loro interessi.
A livello pratico, un algoritmo di raccomandazione opera in un'iniziale matrice *user-rating*, dove le righe rappresentano gli utenti $U = \{u_1, u_2, \dots u_m\}$ e le colonne indicano gli item $I = \{i_1, i_{2,} \dots i_n\}$; in genere, libri, film o musica. Ogni *user* ha espresso la propria valutazione tramite un valore di *rating r*, in linea con la scala di *rating* del sistema: una scala di *rating* può, infatti, essere sia binaria, ossia si identifica con 1 la situazione in cui un utente ha espresso una valutazione, con 0 viceversa; oppure non binaria, nel caso in cui le opinioni sono espresse sottoforma di *rating* rappresentati da valori fissi (in genere, si fa riferimento ad una scala da 1 a 5, dove 0 denota l'assenza di *rating*) [41]. Per quanto riguarda i sistemi di raccomandazione di tipo collaborativo (i.e. *collaborative filtering*), oggetto dell'analisi empirica del capitolo successivo, questi suggeriscono prodotti che individui simili all' user attivo, ossia l'user per il quale si fa la previsione, avevano preferito nel passato.
In merito agli approcci valutativi utilizzati per verificare l'affidabilità e l'efficienza di un algoritmo di raccomandazione, si può, in primo luogo, affermare che quest' ultimo è valutato e classificato sulla base del suo potere predittivo, ossia sulla sua abilità di predire in modo accurato le scelte e preferenze degli utenti. Tuttavia, sebbene previsioni accurate siano ancora

fondamentali per comprendere la validità di un sistema di raccomandazione, queste ad oggi risultano insufficienti per garantire l'efficienza di una buona fonte di raccomandazione.
Vi sono, infatti, molti settori e applicazioni in cui individui e potenziali consumatori si servono di questi sistemi non solo per poter avere e richiedere delle "anticipazioni" sui loro gusti, ma anche perché questi sono interessati, ad esempio, a scoprire nuovi prodotti, ad interagire con differenti *item* o a preservare la loro *privacy*. Per questa ragione è importante identificare per ogni settore in analisi le proprietà che caratterizzano l'algoritmo e il settore stesso, in modo tale da poter influenzare positivamente il risultato del sistema di raccomandazione usato.

2.2. Valutazioni offline vs. online

In linea generale, possono distinguersi due tipi di approcci valutativi, in base al fatto che si possa realizzare un'analisi *online* o una valutazione *off-line* dei sistemi di raccomandazione.
La valutazione *offline* avviene su un insieme di dati strutturati che registrano le scelte degli utenti, nella maggior parte dei casi sottoforma di valutazioni espresse in *rating* per dei prodotti. Il *dataset* in questione è, dunque, già esistente, diviso in *training* e *test set* e sono coinvolti utenti che hanno fornito la loro opinione; inoltre, diversi approcci di raccomandazione sono comparati senza interazione dell'utente [6].
I *rating* presenti nel *training set* vengono utilizzati dagli algoritmi di raccomandazione per predire i *rating* reali nel *test set*. In altre parole, il modello è costruito sui dati del *training set* ed è valutato sui dati che si trovano nel *test set*. L'algoritmo di raccomandazione è valutato comparando i *rating* predetti e reali nel *test set*.
Esistono diversi metodi di partizionamento dei dati tra *training* e *test set*: tra i più importanti vi sono il metodo *holdout*, *bootstrap* e la validazione dei dati *k-fold*. Il metodo *holdout* (o anche chiamato *leave-k-out*) divide il *dataset* in *training set* e *test set* secondo differenti proporzioni. In particolare, nella configurazione del sistema di raccomandazione, la suddivisione avviene attraverso la selezione random di alcuni *rating* da tutti o alcuni utenti. Questi *rating* selezionati faranno parte del *test set*, mentre gli altri apparterranno al *training set*. In letteratura (cfr. Sarwar et al.), si usa il seguente split di suddivisione del *dataset*: 80% *training* e 20% *test* [32].
La seconda è una tecnica statistica di campionamento con reimmissione, che prevede la suddivisione dei dati a disposizione in *training* e *test set* e all'interno di ciascun *dataset* gli

stessi *user* o *item*, a seconda che si parli, ad esempio di *user-based* o *item-based collaborative filtering*, possono essere campionati più di una volta.

Infine, la convalida incrociata *k-fold* consiste nella suddivisione dei dati totali in *k* sottoinsiemi di uguale numerosità e tra di loro indipendenti, in modo da non creare sovrapposizioni. Di questi sottoinsiemi, una parte è utilizzata come *training* e un'altra come *test set*. Inoltre, lo stesso processo è ripetuto *k* volte e ogni volta con un differente *test set*. In particolare, iterativamente si scarta un sottoinsieme per volta, il quale andrà a far parte del *test set* e si cerca di predire con i gruppi non esclusi il campione escluso. Questo evita problemi di campionamento asimmetrico e di *overfitting* del *training test*.

Tale tecnica è, inoltre, idonea per valutare la capacità di raccomandazione quando nuovi utenti entrano nel sistema.

La valutazione degli algoritmi *offline* presuppone la simulazione del processo *online*, di cui si tratterà al termine di questo paragrafo, in cui i sistemi fanno delle previsioni o raccomandazioni e gli utenti correggono queste previsioni oppure utilizzano i suggerimenti del sistema [12]. Tale processo si avvia con la registrazione dei dati storici degli utenti e successivamente si nascondono alcuni di essi in modo da simulare lo apprendimento di come un individuo valuterà un prodotto. Tra i criteri per selezionare gli *item* e i *rating* da nascondere, è preferibile operare una scelta in grado simulare in modo più veritiero possibile le condizioni del target di riferimento. Per *dataset* molto ampi, si predilige il semplice approccio di campionare in modo casuale gli utenti del *test set*, campionare in modo casuale un momento appena prima dell'azione dell'*user* (se si hanno dati di *timestamp*), nascondere tutte le selezioni (di tutti gli utenti) dopo quell'istante e infine tentare di raccomandare gli elementi a quell'utente. Questo protocollo richiede, però, di modificare l'insieme di informazioni date prima di ogni raccomandazione, il che può essere molto costoso da calcolare.

Le valutazioni *offline* sono computazionalmente veloci e semplici da condurre su un gran numero di dati, non richiedono un'interazione con un utente reale e ciò permette di comparare un ampio intervallo di algoritmi a basso costo [6].

Lo svantaggio di questo tipo di valutazioni consiste nella capacità di risposta ad un limitato gruppo di richieste, che tipicamente si circoscrivono al potere predittivo dell'algoritmo.

Si parla, invece, di valutazioni *online* quando gli utenti interagiscono con un sistema di raccomandazione in esecuzione e ricevono effettivamente delle raccomandazioni. Vengono successivamente raccolti *feedback* degli utenti mediante l'osservazione dei loro comportamenti o attraverso una esplicita richiesta. In molte applicazioni di raccomandazione "realistiche" lo scopo del sistema consiste nell'influenzare il comportamento degli utenti e, nella maggior parte dei casi, si è interessati a misurare il cambiamento nel comportamento degli stessi quando

interagiscono con diversi sistemi di raccomandazione. A titolo di esempio, se degli individui di un sistema seguono più spesso le raccomandazioni, o se l'utilità raccolta dagli utenti di un sistema supera l'utilità raccolta dagli utenti dell'altro sistema, allora si può concludere che un sistema è superiore all'altro, a parità di tutti gli altri. Ci sono numerosi fattori che influiscono sulle intenzioni di un utente e, di conseguenza, determinano l'effetto reale di un sistema di raccomandazione: per esempio, il grado di specificità delle esigenze informative, il livello di novità di un suggerimento e la propensione al rischio di un individuo [12].

In alcuni casi, tali esperimenti *online* sono rischiosi, poiché, ad esempio, può verificarsi la situazione in cui un sistema di test fornisce raccomandazioni irrilevanti: questo potrebbe scoraggiare gli utenti del *test set* dall'utilizzare nuovamente il sistema reale e, in questo caso, si registrerebbe un effetto negativo sul sistema; il che può essere inaccettabile in alcuni campi, quali quelli delle applicazioni commerciali.

Per questi motivi, è meglio eseguire una valutazione *online* per ultima, dopo un ampio studio *offline* che dimostra che gli approcci del candidato sono ragionevoli, e in seguito a uno studio dell'utente che misura l'atteggiamento dei quest'ultimo nei confronti del sistema: in questo modo si ridurrà il rischio di una significativa insoddisfazione dell'individuo.

Le valutazioni *online* sono, inoltre, molto utili, in quanto consentono di misurare direttamente gli obiettivi generali del sistema, come il profitto a lungo termine o la fidelizzazione dell'utente. Per di più, tali valutazioni possono essere utilizzate per comprendere il modo in cui questi scopi generali sono influenzati dalle proprietà del sistema, quali l'accuratezza e la diversità delle raccomandazioni; e per cercare di trovare dei compromessi tra queste proprietà. Poiché variare tali proprietà in modo indipendente è difficile, e confrontare molti algoritmi attraverso prove *online* è costoso, molto spesso è difficile comprendere appieno queste relazioni.

Sebbene questo tipo di valutazioni richieda del tempo e risulta più complesso delle valutazioni *offline*, questo rappresenta l'unico modo per misurare la reale soddisfazione dell'individuo.

2.3. Metriche di accuracy

A prescindere dal tipo di valutazione, *online* o *offline*, che si va ad effettuare sull'algoritmo di raccomandazione e partendo dal presupposto che, successivamente, verrà presentata un'analisi empirica di dati, per cui verrà eseguita una valutazione *offline*, esistono delle metriche che misurano empiricamente la precisione (*accuracy*) degli algoritmi di raccomandazione. Nello specifico, tali metriche forniscono informazioni della misura in cui una classifica di

raccomandazioni predette da un sistema di raccomandazione differisce da un effettivo *ranking* elaborato dall'utente. Inoltre, le metriche di *accuracy* misurano la bontà di riproduzione delle valutazioni e liste di classificazioni di un individuo da parte del sistema di raccomandazione. Si fa tipicamente riferimento a *rating* divisi in *training set* e *test set*, dove le valutazioni che si trovano nel primo gruppo vengono utilizzate per addestrare la funzione in grado di predire il *rating* di un utente verso un nuovo *item*, mentre il *test set* viene utilizzato per valutare l'accuratezza di una previsione.

Lo studioso Herlocker suddivide tali metriche in due classi principali: metriche di accuratezza previsionale (*predictive accuracy metrics*) e metriche di accuratezza nella classificazione (*classification accuracy metrics*) [13].

Le metriche di accuratezza previsionale riferiscono in che misura un sistema di raccomandazione è in grado di prevedere i vari *rating* degli utenti.

Fanno parte di questa classe l'errore medio assoluto (*mean absolute error*-MAE), l'errore quadratico medio (*mean squared error*-MSE) e la radice dell'errore quadratico medio (*root mean squared error*-RMSE).

L'errore medio assoluto è calcolato come la somma della differenza tra il *rating* espresso da un utente e il *rating* previsto dal sistema, il tutto diviso per il numero di *item* considerati. Scopo principale di tale metrica consiste nel misurare quanto la previsione fornita dal sistema ben approssimi il *rating* reale di un prodotto. Di seguito la formula riassuntiva [2.1].

$$MAE = \frac{1}{|B_i|} \sum_{b_k \in B_i} |r_i(b_k) - p_i(b_k)| \qquad [2.1]$$

Dove B_i rappresentano degli *item* valutati da un utente *u*.

Tra le varianti dell'errore medio assoluto si annoverano l'errore quadratico medio (MSE) e la radice dell'errore quadratico medio (RMSE). L'errore quadratico medio calcola la differenza quadratica media tra le valutazioni sottoforma di *rating* degli utenti e i valori dei *rating* stimati. In questo modo, si enfatizzano errori molto grandi elevando al quadrato ogni singolo errore. La formula [2.2] riassume quanto asserito.

$$MSE = \frac{1}{B_i} \sum_{b_k \in B_i} (r_i(b_k) - p_i(b_k))^2 \qquad [2.2]$$

L'RMSE non è altro che la radice quadrata dell'MSE.

Secondo Herlocker e altri studiosi questa tipologia di metriche risulta poco significativa per situazioni in cui bisogna ricercare una lista di top-N raccomandazioni [13].

In ogni caso, le metriche appena citate sono largamente utilizzate da studiosi in questo settore per avere una panoramica generale circa l'accuratezza degli algoritmi utilizzati; anche se una visione più precisa, che tenga conto degli algoritmi che operano anche con *rating* impliciti è garantita da ulteriori metriche: *precision, recall, f-measure* e curva ROC, appartenenti al gruppo delle cosiddette *classification accuracy metrics*.

Questo secondo gruppo di metriche rivela in che misura un sistema di raccomandazione è in grado di classificare correttamente prodotti come interessanti o meno per un utente. Queste metriche sono molto efficaci in settori con *rating* binari, per cui è necessario adottare schemi binari per utilizzarle. Tali misure non riescono, però, a indagare l'entità della discrepanza tra il reale *rating* espresso da un individuo e il *rating* previsto dall'algoritmo.

Le misure di *precision* e *recall*, sebbene appartengano al campo dell*'Information Retrieval*[27], assumono molta rilevanza anche in ambito dei sistemi di raccomandazione. La *precision*, anche nota come valore predittivo positivo, rappresenta una misura dell'efficienza di una rilevante selezione di prodotti. A livello computazionale, è espressa dal rapporto tra gli *item* rilevanti selezionati e il numero di *item* presi in considerazione.

$$Precision = \frac{|B_{rs}|}{|B_s|} \qquad [2.3]$$

Nella formula [2.3] B_s rappresenta il gruppo degli *item* selezionati, mentre B_{rs} è il sottoinsieme di B_s che individua i prodotti rilevanti. In altre parole, la formula misura la probabilità che i dati realmente classificati siano rilevanti.

La misura *recall*, anche conosciuta come sensitività, è, invece, calcolata come il numero degli *item* rilevanti selezionati sul numero totale di prodotti rilevanti presenti nel set di tutti gli *item B*: questo misura l'ampiezza del risultato [2.4]. Tale metrica dà informazioni circa la probabilità che dati rilevanti siano correttamente classificati.

$$Recall = \frac{|B_{rs}|}{|B_r|} \qquad [2.4]$$

[27] L'Information Retrieval è un settore informatico che fa riferimento a tecniche di rappresentazione, memorizzazione ed organizzazione di documenti testuali, quali pagine *web* o file multimediali, che permettono ad un utente di accedere ad informazioni in linea con i suoi interessi e bisogni informativi.

Per le metriche di *precision* e *recall* è necessario che i *rating* siamo considerati secondo una scala binaria che distingue tra valutazioni "rilevanti e "non rilevanti". In una scala di *rating* da 1 a 5, in genere i prodotti valutati da 1 a 3 sono considerati non rilevanti, viceversa per gli *item* con *rating* 4 e 5.

Entrambe le misure di *precision* e *recall* si basano sul concetto di rilevanza, la quale è determinata alla luce delle preferenze di un individuo in un certo periodo di tempo; il che rende impossibile stabilire una soglia fissa che classifichi un prodotto per un certo *user* come rilevante o meno [36]. Ciononostante, un sistema di raccomandazione suggerisce *item* sulla base della probabilità che l'utente ne sia interessato, anche se è l'utente che in ultima istanza decide se un prodotto è realmente notevole per lui.

Tra questo gruppo di metriche è possibile classificare ogni raccomandazione come: vero positiva (TP), che riferisce la situazione in cui un *item* di interesse per l'utente è suggerito a quest'ultimo; vero negativa (TN), quando un *item* non di interesse per l'individuo non gli viene raccomandato; falso negativa (FN), nel momento in cui un prodotto che incontra l'interesse dell'utente non è all'interno dei suggerimenti per quell'individuo; e falso positiva (FP), quando un prodotto che non incontra i gusti di un consumatore viene raccomandato a quest'ultimo.

Questi valori sono riassunti dalla cosiddetta matrice di confusione che è costruita utilizzando i risultati in termini di valori TP, TN, FN, FP, che derivano dai modelli di classificazione (cfr. Tabella 2.1)

Tabella 2.1. - Matrice di confusione

		Raccomandazioni	
		Raccomandati	Non raccomandati
Realtà	Positivi	**TP**	**FP**
	Negativi	**FN**	**TN**

In considerazione di quanto appena asserito, si può calcolare l'*accuracy* come il rapporto tra le osservazioni correttamente predette sul totale delle osservazioni [formula 2.4].

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \qquad [2.4]$$

Più l'*accuracy* riporta un valore elevato più il modello performa meglio, anche se questo è valido soprattutto quando si è in presenza di un *dataset* simmetrico, ossia il valore dei falsi positivi è circa lo stesso dei falsi negativi.

La metrica complementare rispetto l'*accuracy* è il tasso di errore o tasso di errata classificazione (ERR). Tale misura rappresenta il numero di campioni valutati erroneamente che provengono sia dalla classe positiva sia negativa ed è calcolato come segue [2.5]:

$$ERR = 1 - Acc = \frac{TP + TN}{TP + TN + FP + FN} \qquad [2.5]$$

Come già affermato in precedenza per l'*accuracy*, anche la metrica del tasso di errore è sensibile in presenza di dati sbilanciati, ossia quando campioni appartenenti ad una classe di un *dataset* risultano più numerosi rispetto a campioni in altre classi.
Alla luce di questa classificazione è possibile individuare un'ulteriore possibile modalità di calcolo delle metriche di *precision* e *recall* [cfr. formule 2.6 e 2.7]

$$precision = \frac{TP}{TP + FP} \qquad [2.6]$$

$$recall(true\ positive\ rate) = \frac{TP}{TP + FN} \qquad [2.7]$$

Queste metriche sono adatte a valutare casi di top-N raccomandazioni; quando un algoritmo predice i top-N prodotti che un utente si aspetta siano di interesse per lui, attraverso la metrica del *recall* è possibile calcolare la percentuale di *item* rilevanti conosciuti presi dal *test set* che compaiono negli N *item* predetti.
Un modo per calcolare meglio *precision* e *recall* considerando le top-N raccomandazioni consiste nel considerare solo gli *item* valutati dagli utenti.
Con la consapevolezza che, in generale, il numero di prodotti valutati da ogni utente è molto più piccolo degli *item* disponibili nell'intero *dataset* e in considerazione che la quantità di prodotti rilevanti presenti nel *test set* è molto minore di quelli nell'intero gruppo di dati, in letteratura [28] si è giunti alla conclusione che il valore delle metriche di *precision* e *recall* dipende fortemente dal numero di prodotti valutati per ogni utente, ragion per cui i loro valori non dovrebbero essere interpretati in maniera assoluta; ma risultano utili solo per comparare differenti algoritmi nello medesimo *dataset*.

[28] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1):5–53, 2004.

Una misura che combina le metriche *precision* e *recall* è la cosiddetta *f-measure*, calcolata come segue [2.8]

$$F - measure = \frac{2 \times recall \times precision}{recall + precision} = \frac{2 \cdot TP}{2 \cdot TP + FB + FP} \qquad [2.8]$$

Tale metrica rappresenta la media pesata delle misure di *precision* e *recall* e, per questo, il suo risultato tiene in considerazione sia i falsi positivi che i falsi negativi. Intuitivamente non è semplice da comprendere nel modo in cui lo è l'accuracy (cfr. formula 2.4), ma è molto più utile rispetto quest'ultima quando si ha a che fare con una classe di distribuzione irregolare: l'*accuracy*, infatti, performa meglio quando i valori falsi positivi e falsi negativi hanno costo simile; in caso contrario, è meglio tenere in considerazione questa misura che riassume sia *precision* che *recall*.

2.4. Curva ROC e curva Precision-recall

Per visualizzare graficamente il *trade-off* tra le metriche di *precision* e *recall*, quest'ultima anche chiamata sensitività, è possibile computare una curva *precision-recall (PR)*, in grado di fornire l'andamento delle due grandezze.

In genere, la curva PR ha un andamento a zig-zag e una perfetta performance di classificazione è rappresentata dalla figura sottostante (cfr. Figura 2.1). Una perfetta prestazione è descritta, infatti, dalla curva verde che si avvia nel punto (0,1), dove il classificatore raggiunge il 100% di *precision* e 0% di *recall*, per poi raggiungere il punto (1,1), il punto ideale nella curva PR in cui sia la sensitività che la *precision* sono al 100%; e per arrivare, infine, nel punto (1,0), caratterizzato dal 100% di *recall* e 0% di *precision*. In questo caso, si può affermare che più è vicina la curva PR rispetto all'angolo in alto a destra, migliore saranno i risultati e le performance della classificazione [36].

Bisogna, inoltre, tenere in considerazione che la curva *precision-recall* dipende solo dalle due metriche che le danno il nome; per questa ragione ignora le prestazioni degli esempi cosiddetti *true negative*. La figura 2.1 riassume quanto detto finora.

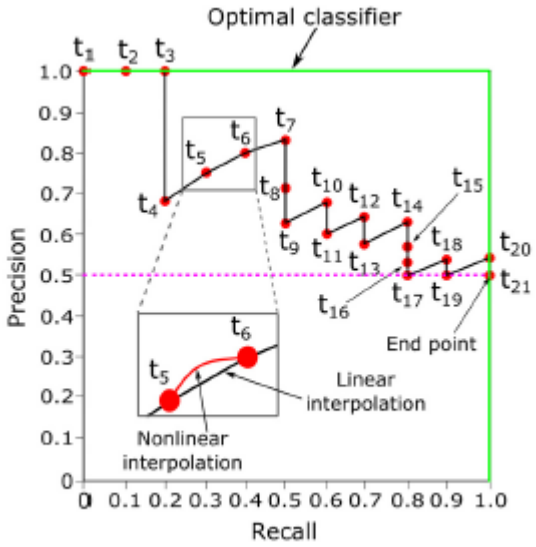


Figura 2.1 – Curva precision-recall

Esiste, però, un'ulteriore curva, chiamata ROC (*Receiver Operating Characteristic*), che compara tassi di valori vero-positivi (*true positive rate*) con tassi di valori falso-positivi (*false positive rate*), i quali sono rispettivamente calcolati come segue [cfr. formule 2.9. e 2.10]:

$$TPR = \frac{TP}{TP + FN} \qquad [2.9]$$

$$FPR = \frac{FP}{FP + TN} \qquad [2.10]$$

Il *true positive rate* non è altro che il rapporto tra i prodotti suggeriti dal sistema che sono stati in seguito acquistati dai consumatori e la somma degli *item* scelti dagli individui, a prescindere che questi siano stati raccomandati dal sistema.

Il *false positive rate* (FPR), anche detto *Fallout*, è dato dal rapporto tra i campioni negativi classificati non correttamente, ossia il numero di *item* che sono stati raccomandati, ma che l'utente alla fine non ha acquistato e il numero totale di campioni negativi, ovvero *item* non acquistati ma raccomandati e prodotti non suggeriti dall'algoritmo di raccomandazione e non acquistati dagli utenti.

La curva ROC viene utilizzata per valutare molti sistemi, tra cui sistemi di diagnostica e di *machine learning* e serve per comprendere il *trade-off* tra benefici, rappresentati dai valori *true positive*, e i costi, ovvero i valori falsi positivi [36].

A livello grafico, si possono individuare quattro punti importanti per comprendere la curva ROC.

Nello specifico, si osserva il punto A, in basso a sinistra e con coordinate (0,0), il quale rappresenta un classificatore che descrive la situazione in cui non vi sono classificazioni positive, mentre tutti i campioni "negativi" sono stati correttamente classificati; di conseguenza, si avrà che sia il tasso di veri positivi, che il tasso di falsi positivi sarà uguale a 0. Il punto C, al contrario, è collocato in una posizione diametralmente opposta rispetto al punto A con coordinate (1,1) e dà una rappresentazione di un classificatore che ha "catalogato" correttamente tutti i campioni positivi, mentre i campioni negativi sono stati classificati erroneamente. Il punto B situato nell'angolo in alto a sinistra indica la circostanza in cui tutti i campioni positivi e negativi sono stati correttamente classificati; di conseguenza, questo è il punto di perfetta classificazione o il cosiddetto *Ideal operating point* [36].

Graficamente si può notare nella figura 2.2 che una classificazione performa perfettamente lungo la linea verde che si avvia nel punto A per poi arrivare al punto B e terminare in C: in tutti questi punti, il classificatore ha ordinato perfettamente i campioni positivi in relazione a quelli negativi.

In generale, un punto nello spazio della curva ROC indica una scarsa performance del classificatore se si trova in basso a destra, ossia è caratterizzato da bassa TRP, alta FRP o entrambe.

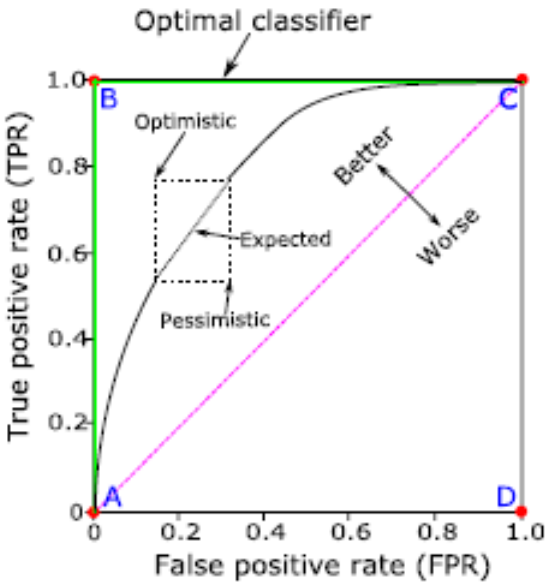


Figura 2.2. Curva ROC

All'intero della curva ROC risulta difficile comparare differenti classificatori, in quanto non esiste un valore scalare che fornisce informazioni circa la performance attesa. Vi è, però, la cosiddetta AUC, ossia l'area sotto la curva ROC, che fornisce una misura quantitativa della

performance di un classificatore e registra un valore che va da 0 a 1: nessun classificatore realistico registra un valore più basso di 0,5.

Mentre entrambe le curve di *precision-recall* e ROC misurano la proporzione di item preferiti, che sono realmente consigliati dal sistema, la curva *precison-recall* pone l'accento sulla proporzione di *item* raccomandati che entrano nelle preferenze dell'utente; la curva ROC, al contrario, enfatizza la proporzione di prodotti non preferiti che finiscono tra quelli consigliati ai consumatori.

La scelta di utilizzare la curva *precision-recall* o la curva ROC avviene in base alle proprietà del dominio e allo scopo dell'applicazione. Se, ad esempio, un servizio di noleggio video *online* raccomanda i DVD agli utenti, la misura di precisione descrive la proporzione delle loro raccomandazioni effettivamente adatta all'utente. Se le raccomandazioni non appropriate per l'individuo rappresentano una piccola o grande parte dei DVD non adatti che avrebbero potuto essere raccomandati (cioè il tasso di falsi positivi) può non essere così rilevante la percentuale degli elementi rilevanti che il sistema ha raccomandato all'utente, quindi una curva di *precision-recall* sarebbe adatta a questa applicazione [12]. Al contrario, per un sistema di raccomandazione utilizzato al fine di selezionare gli articoli da commercializzare agli utenti, ad esempio inviando un articolo all'individuo che lo può restituire gratuitamente se non lo acquista, la curva *precision-recall* non risulta adatta ad una rappresentazione esaustiva della situazione. In questo caso, dove si è interessati a realizzare il maggior numero possibile di vendite potenziali riducendo al minimo i costi di marketing, la curva ROC sarebbe, infatti, più rilevante rispetto ad una curva di *precision-recall*.

In applicazioni in cui un numero fisso di raccomandazioni è fatto ad ogni utente (e.g. quando un numero fisso di titoli sono mostrati ad un utente che visita un portale di notizie), è possibile calcolare *precision* e *recall* (o tasso di veri positivi e tasso di falsi positivi) ad ogni lunghezza della lista di raccomandazioni N per ogni utente, e quindi calcolare la *precision* media e il *recall* (o tasso positivo vero e tasso di falsi positivi) ad ogni N. Le curve risultanti sono particolarmente utili, pioché prescrivono un valore di N per ogni raggiungibile *precision* e *recall* (o tasso vero positivo e tasso di falsi positivi) e, al contrario, possono essere utilizzate per stimare le prestazioni ad una data N. Una curva ROC così ottenuta è chiamata curva ROC (CROC) del cliente [36].

2.5. Proprietà e valutazioni degli algoritmi di raccomandazione

Tra le proprietà degli algoritmi di raccomandazione da tenere in considerazione quando si parla di valutazione del sistema bisogna ricordare la cosiddetta copertura. Nello specifico, questo termine fa riferimento alla percentuale di elementi che il sistema di raccomandazione può suggerire all'utente. Questa misura può essere calcolata in molte circostanze direttamente in base all'algoritmo e al *set* di dati di input; in alcuni casi, può essere utile ponderare gli elementi, ad esempio, in base alla loro popolarità o utilità [12].

Questa misura assume importanza, in quanto l'accuratezza di previsione di un sistema, specialmente nei sistemi di filtraggio collaborativo, in molti casi cresce con la quantità di dati; per cui possono esserci algoritmi in grado di fornire raccomandazioni di alta qualità, ma solo per una piccola parte degli elementi per cui dispongono di enormi quantità di dati.

Per copertura può anche intendersi la percentuale di utenti o di interazioni con gli utenti per i quali il sistema può raccomandare degli articoli. In base alle applicazioni e al settore di riferimento, un algoritmo potrebbe non fornire raccomandazioni per alcuni utenti a causa, ad esempio, della scarsa fiducia nell'accuratezza delle previsioni per quell'individuo [13]. Nei casi in cui i sistemi d raccomandazione forniscono suggerimenti per una gamma più ampia di utenti, la loro valutazione sarà data dal *trade-off* tra copertura e *accuracy*. La copertura, in questi casi, può essere misurata dalla ricchezza del profilo utente richiesto per fare una raccomandazione: nell'esempio di *collaborative filtering*, tale "ricchezza" potrebbe essere calcolata come il numero di voci oggetto di valutazione per l'utente prima che quest'ultimo riceva raccomandazioni.

Non è, inoltre, da sottovalutare la fiducia dell'utente nei confronti delle raccomandazioni del sistema. A titolo di esempio, possono verificarsi dei casi in cui il sistema suggerisca prodotti che l'utente già conosce e gradisce. In questo caso, anche se l'utente non trae alcun valore da questo tipo di raccomandazioni, può comunque dedurre che il sistema fornisce suggerimenti ragionevoli, il che può indirettamente portare ad un aumento della sua fiducia anche per elementi a lui ancora sconosciuti [12].

In un test *online*, la fiducia nel sistema di raccomandazione può essere associata al numero di suggerimenti seguiti dall'individuo o, in alternativa, al numero di utenti che ripetutamente sia affida ad un algoritmo. Per quanto riguarda esperimenti *offline*, risulta arduo misurare la fiducia, poiché questa è costruita attraverso un'interazione tra il sistema e l'utente.

Se da un lato la fiducia in un algoritmo di raccomandazione assume un ruolo importante per l'utente e quest'ultimo è consapevole di rivelare le sue preferenze per ottenere raccomandazioni utili; tuttavia, è fondamentale per la maggior parte degli individui che i loro gusti rimangano

privati, ossia che nessun terzo possa utilizzare queste informazioni per ulteriori fini. Un esempio in questo senso può essere considerato il caso in cui un utente interessato in fotografia ha acquistato un libro che ha poco a che fare con questo suo interesse, intitolato "Organizzare e pianificare un divorzio". A questo punto, si potrebbe creare la situazione in cui il coniuge di quell'utente, navigando in Internet per cercare un regalo, alla ricerca del libro "Manuale di fotografia" può ottenere una raccomandazione del tipo "persone che hanno comprato questo libro anche comprato" per l'organizzatore del divorzio, rivelando così informazioni private sensibili sul primo individuo.

Poichè generalmente è considerato inappropriato per un sistema di raccomandazione rivelare informazioni private anche per un singolo utente, vi sono studi che confrontano gli algoritmi valutando la porzione di utenti le cui informazioni private sono state compromesse[29]. Il presupposto di tali studi risiede comunque nella consapevolezza che una totale privacy non è realistica e che quindi bisogna scendere a compromessi per ridurre al minimo le violazioni. Un'ulteriore alternativa è quella di definire diversi livelli di privacy e confrontare la sensibilità degli algoritmi al variare di tali livelli.

Sebbene la privacy sia un aspetto fondamentale da tutelare, questa può influenzare negativamente l'accuratezza delle raccomandazioni: per avere una visione realistica di ciò si dovrebbe calcolare l'accuratezza o qualsiasi altra proprietà dell'algoritmo prima e dopo aver apportato modifiche in merito alla privacy.

Un'ulteriore importante caratteristica degli algoritmi di raccomandazione si rintraccia nella loro capacità di operare in un ambiente in cui i prodotti cambiano rapidamente oppure muta l'andamento dell'interesse sugli stessi: è ciò che avviene, ad esempio nella raccomandazione di notizie o articoli nei giornali *online*.

È possibile valutare l'adattamento di un sistema partendo dall'analisi quantitativa di informazioni necessarie di un *item* prima di raccomandarlo. Una volta raccolte le notizie necessarie su una tipologia di prodotto e averlo classificato come "interessante" per l'utente, è probabile che l'algoritmo sia regolato per raccomandare quella tipologia di elementi più velocemente, sacrificando, però, una certa precisione di previsione.

Un altro tipo di adattabilità è la velocità con cui il sistema si adatta alle preferenze personali di un utente o ai cambiamenti del profilo dello stesso.

[29] Frankowski, D., Cosley, D., Sen, S., Terveen, L., Riedl, J. You are what you say: privacy risks of public mentions. In SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval, pages 565–572, New York, NY, USA, 2006.

Capitolo 3 - Applicazione del metodo collaborative filtering alla raccomandazione di libri

Nei precedenti capitoli si sono passati in rassegna i principali tipi di sistemi di raccomandazione (i.e. *collaborative filtering*, *content based filtering* e *hybrid filtering*) e i loro primari algoritmi e metodi computazionali.
Nella seconda parte di questa tesi si sono, invece, approfonditi alcuni dei metodi valutativi di tali algoritmi, con particolare focus sulle metriche di *accuracy*.
Quanto descritto nelle prime due parti di tale elaborato è finalizzato ad essere applicato ad un'analisi empirica di un dataset, in cui viene sfruttato il metodo del *collaborative filtering* e, in particolare le tecniche user-based e item-based. Si commentano infine i risultati.

3.1. Literature review

I sistemi di raccomandazione rappresentano una soluzione alla problematica ormai globale e condivisa di ricerca di notizie pertinenti al soddisfacimento di determinati bisogni informativi degli individui.
Si è ampiamente visto all'interno del primo capitolo che tali algoritmi vengono classificati sulla base di differenti modalità di raccomandazione dei prodotti: si distinguono, quindi, tecniche di filtraggio collaborativo o basate sul contenuto, oppure modelli ibridi che combinano le caratteristiche dei primi due tipi di raccomandazioni.
I sistemi di raccomandazione operano ormai in molti campi e settori tra cui l'*e-commerce*, l'*e-learning* e, in generale, aiutano un consumatore nella scelta di film, musica, notizie o libri.
In questa tesi particolare *focus* è posto sulla raccomandazione di libri; per questo motivo, di seguito si illustreranno differenti modelli, per comprendere lo stato dell'arte in questo campo.
I sistemi di raccomandazione di libri utilizzano diversi approcci di filtraggio delle informazioni a seconda del contesto e del dominio in cui questi sono sviluppati: Libra, ad esempio, è un sistema di raccomandazione *collaborative filtering*, che suggerisce libri sulla base di informazioni raccolte dalle pagine web di Amazon.com [15]. Tale sistema prende in

considerazione anche il profilo utente e le raccomandazioni sui titoli utilizzando le valutazioni fornite dagli utenti. Libra applica anche la categorizzazione del testo ai dati semi-strutturati ottenuti attraverso l'estrazione di informazioni dalle pagine web di Amazon.com.

Un ulteriore sistema che suggerisce libri è K3Rec: questo analizza il contenuto del libro, l'idoneità tematica e altre caratteristiche, al fine di raccomandare i libri ai bambini che meglio rispondono a delle scelte di lettura e livelli di leggibilità prefissati. K3Rec ha, però, lo svantaggio di essere limitato nel valutare la qualità degli articoli, se il contenuto manca di sufficienti informazioni in merito agli articoli raccomandati; di conseguenza, K3Rec non sarà in grado di distinguere tra ciò che rientra o meno nelle preferenze dell'utente.

In generale, nella progettazione di sistemi di raccomandazione, si adoperano spesso diverse tecniche di *data mining* per estrarre regole utili nelle raccomandazioni di grandi insiemi di dati. Esistono, infatti, sistemi che utilizzano tecniche di *clustering k-means* per creare gruppi di utenti in base alla cronologia delle transazioni di questi ultimi e fornire in seguito ad ogni utente presente in ciascun cluster raccomandazioni di elenchi di libri personalizzati. Tali liste di libri potrebbero, però, essere poco rappresentative delle preferenze degli utenti poiché non si considera il reale contenuto del libro [34].

Vi sono, inoltre, anche sistemi di raccomandazione di libri che operano in un contesto di prestito bibliotecario e applicano tecniche di associazione per consigliare i libri all'interno della biblioteca digitale. Questo approccio è limitato nel trovare regole che soddisfino l'interesse degli utenti che non visitano frequentemente le biblioteche e non eseguono transazioni con il sistema [37].

Sebbene oggi sistemi di raccomandazione mostrano risultati più efficienti quando implementano tecniche avanzate di *deep learning*, che non verranno trattate in questa tesi perché di difficile implementazione; tuttavia, non bisogna dimenticare che questi algoritmi sono addestrati su risultati che derivano dalle tecniche di filtraggio collaborativo e di contenuto.

Uno studio molto interessante sul primo modello di raccomandazione, i.e. *collaborative filtering*, confronta i due principali metodi di filtraggio collaborativo, ovvero *item-based* e *user-based collaborative filtering*, per capire quale dei due performa meglio e quindi registra un errore minore nel fornire raccomandazioni corrette, ossia suggerimenti che rispecchiano realmente le preferenze di un utente [4].

Questa ricerca è stata condotta da P. Boström e M. Filipsson dell'Istituto Reale di Tecnologia di Stoccolma, i quali hanno applicato la tecnica del *collaborative filtering* al celeberrimo *dataset* di Movielens 100k. Nello specifico, dopo aver impostato i modelli di raccomandazione, hanno calcolato i valori di RMSE connessi sia al modello di *user-based* che *item-based collaborative*

filtering. Tale metrica è stata computata tre volte per entrambe le tecniche di *collaborative filtering* su dei *dataset* che mostravano ogni volta differenti percentuali tra *training* e *test set*: nel dettaglio, si è visto, in primo luogo, una suddivisione di 75% *training* e 25% *test*, successivamente 90% *training* e 10% *test* e infine 50% *training* e 50% *test*.
Inoltre, ciascuno dei tre test è stato eseguito cento volte.
Scopo della ricerca è comprendere quali sono le principali differenze e uguaglianze tra questi modelli in termini di errori commessi nella raccomandazione.
In tutti e tre o casi si sono registrate performance migliori in termini di RMSE per l'algoritmo *user-based*.

3.2. Il Dataset

Il dataset oggetto di analisi proviene dal sito www.kaggle.com, all'interno del quale vengono pubblicati dei dati da studiosi di *data science* e *machine learning*. La community di Kaggle, attiva dal 2010 e presente in più di 194 Paesi, rende disponibile al pubblico una vasta mole di set di dati, destinati ad essere analizzati ed elaborati da chiunque sia registrato alla pagina web.
I dati da cui ho attinto per la mia analisi empirica fanno parte del *dataset* "goodsbook-10k" presente su Kaggle, il quale raccoglie le valutazioni fornite dagli utenti per circa diecimila libri e in media si registrano cento *rating*, che sfruttano una scala da 1 a 5, per ogni libro.
La scelta di utilizzare il presente *dataset* è dovuta al desiderio e curiosità di applicare una tecnica di raccomandazione (i.e. *collaborative filtering*) in un campo, ossia quello del suggerimento di libri, non ancora caratterizzato da *dataset* noti e su cui non sono stati effettuati numerosi studi.
Al contrario, le raccomandazioni in ambito musicale e nella scelta di film fanno riferimento a dati molto utilizzati nelle applicazioni empiriche e ormai corroborati e perfezionati nel tempo, quali rispettivamente i dataset Million Song e Movielens.
Gli elementi oggetto della mia analisi si riferiscono ad un *dataset* di 300000 osservazioni e tre variabili: rispettivamente il codice identificativo degli utenti (*user_id*), quello per i libri (*book_id*) e i *rating* che gli *user* hanno fornito come valutazione dei libri, che sono compresi in un range di valori da 1 a 5.

3.3. Domanda di ricerca e metodologia

Alla luce dello studio effettuato dall'università svedese e in considerazione dei dati a disposizione, le domande di ricerca a cui la presente tesi intende rispondere sono due.
La prima, di carattere più generico, consiste nel delineare le performance generali dei sistemi *item-based* e *user-based collaborative filtering* applicati al *dataset* oggetto di analisi, in termini di valutazione mediante le metriche di *accuracy* e considerando la cosiddetta matrice di confusione.
La seconda domanda di ricerca ha l'obiettivo di dimostrare che, alla stregua del caso della ricerca svedese, anche per il *dataset* in esame è valido il risultato per cui, in termini di RMSE, la tecnica *user- to- user* performa meglio rispetto alla tecnica *item-to-item*.
Nella presente tesi, l'elaborazione e l'analisi dei dati è stata effettuata mediante l'ausilio del linguaggio di programmazione statistica R sull'interfaccia R-studio. In particolare, fondamentale per la costruzione e l'implementazione degli algoritmi di raccomandazioni è stato il pacchetto di funzione *recommenderlab*. Molto utilizzato è stato altresì il pacchetto *ggplot2* per la costruzione di alcuni grafici riportati nei paragrafi successivi.

3.4. Fase esplorativa: analisi delle variabili

Prima di computare la matrice *user-item*, è necessario avviare una fase di esplorazione dei dati, in modo da avere già una prima panoramica della distribuzione degli stessi e in particolare delle variabili che compongono il *dataset*.
Nello specifico, si nota come gli utenti in questione siano identificati fino al codice 53424, con un valore medio di 24804 e mediano di 23576; il che vuol dire che la prima metà di utenti ha mediamente fornito più *rating* rispetto alla seconda metà di *user*. Mentre la distribuzione di media e mediana per i libri risulta abbastanza regolare, per quanto riguarda i rating, si osserva che, sebbene vi sia un range 1-5, la maggior parte degli individui ha espresso una valutazione di circa 3,8.
Per una visione più completa si osservi la tabella sottostante (Tabella 3.1).

Tabella 3.1 – Distribuzione variabili

	user_id	**book_id**	**rating**
Min.	1	1	1.000
1st Qu.	11837	751	3.000
Mediana	23576	1501	4.000
Media	24804	1501	3.831
3rd Qu.	37381	2252	5.000
Max.	53424	3003	5.000

Per quanto riguardala varianza di queste tre variabili, si nota dalla tabella 3.2 che esse registrano valori molto elevati, sintomo di una distribuzione molto variegata per cui difficilmente si riuscirà nelle fasi successive a creare sottoinsiemi di dati simili tra di loro.

Tabella 3.2 – Varianza e deviazione standard

	user_id	**book_id**	**rating**
Varianza	225459300	751221,8	1,00962
Dev. St.	15015,302	866,7305	1,00480

A questo punto è necessario andare ad analizzare ciascuna di queste tre variabili per avere una visione più completa e dettagliata dei dati oggetto di analisi.

Prendendo in considerazione la prima variabile, ossia gli utenti, è possibile verificare la distribuzione del numero di *user* che leggono libri. Nello specifico, come si osserva dal grafico 3.1, si contano circa duecento individui che leggono circa un centinaio di libri, mentre si osserva che all'incirca venti utenti leggono quasi diecimila libri.

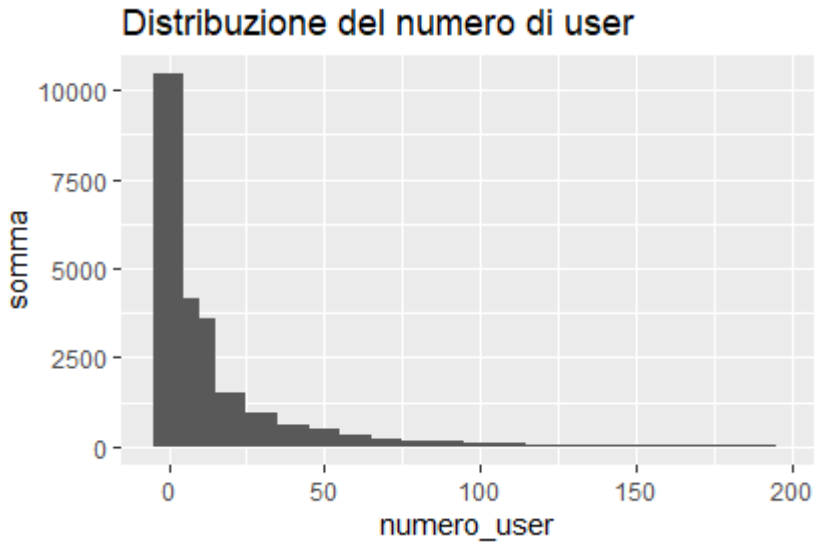


Figura 3.1 – Distribuzione del numero di user

In media vi sono anche circa 16 libri ad utente (15,84368) e quasi 100 utenti per libro (99,9001). Se si costruisce, partendo dal *dataframe* iniziale, una matrice di dati si avrà una matrice di 18935x3003 con 299432 *rating*, dove il primo elemento rappresenta gli utenti e 3003 sono, invece, i libri.

In merito agli utenti non è possibile asserire altro, in quanto, sulla base dei dati in mio possesso, non si è potuto indagare e avere informazioni circa la similarità tra i vari utenti.

Per quanto riguarda gli *item*, invece, è possibile ricavare ulteriori informazioni mediante il calcolo di una matrice di similarità tra gli stessi. In questo caso, la matrice computata utilizzando la distanza del coseno[30] e prendendo in considerazione per comodità i primi cinque *item* del dataset, rivela che questi sono molto simili tra di loro: a titolo di esempio, si può vedere come il prodotto numero 1, che corrisponde al libro "Hunger Games" registra una similarità di 0,91 e 0,95 rispettivamente con i libri tre e cinque, ossia rispettivamente "Twilight" e "The Great Gatsby"; mentre ha una similarità pressoché perfetta con i prodotti due e quattro, "Harry Potter and The Philosopher's Stone" e "To Kill a Mockingbird". È possibile confrontare quanto detto osservando la matrice sottostante (Tabella 3.3) e avere una visione ancora più rappresentativa attraverso la rappresentazione grafica della matrice, in cui più le celle tendono al rosso scuro, più questo è sintono di più elevata similarità (cfr. Figura 3.2).

[30] Questa tecnica misura la similitudine tra due vettori, in questo caso gli *item*, calcolando il coseno tra di loro.

Tabella 3.3 – Similarità tra gli item

	1	**2**	**3**	**4**	**5**
1	0,0000	0,9840	0,9149	0,9753	0,9492
2	0,9840	0,0000	0,9068	0,9645	0,9448
3	0,9149	0,9068	0,0000	0,9003	0,8713
4	0,9753	0,9645	0,9003	0,0000	0,9639
5	0,9492	0,9448	0,8713	0,9639	0,0000

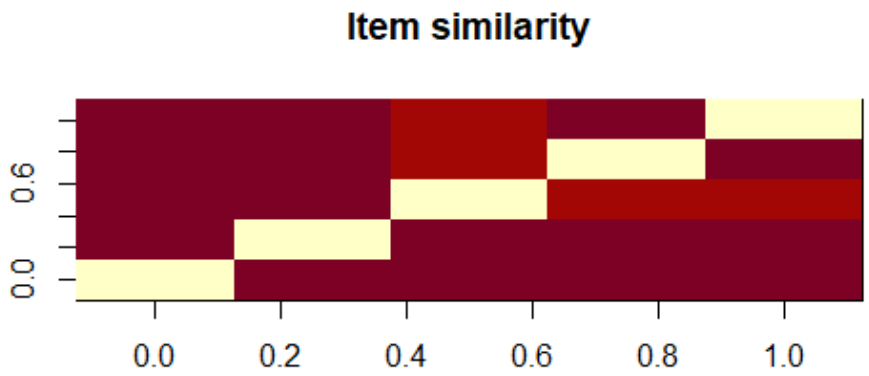


Figura 3.2 – Similarità degli item

Dato che si è visto come molti libri risultino tanto simili l'uno con l'altro, può essere utile comprendere quali e quanti sono i libri più letti: anche in questo caso, dopo aver ordinato in modo crescente il dataset in base agli *id* dei libri e aver tabulato i dati, è emerso che i primi 2000 libri sono stati letti da 100 utenti; per cui non è possibile indentificare dei trend che permetta di individuare o quantomeno intuire in modo netto le preferenze degli individui oggetto di analisi.

Dopo aver esaminato le prime due variabili, i.e. *user* e *book*, si prende ora in considerazione la variabile "rating".

Come già visto in precedenza (cfr. Tabella 3.1), la media di valori di *rating* che gli utenti hanno espresso nei confronti di un particolare libro si assesta intorno a 3,8. Tale valore viene confermato dal grafico sottostante della distribuzione degli stessi, dal quale si nota perfettamente come le valutazioni maggiormente assegnate ai libri siano, nell'ordine, 4, 5 e 3 (Figura 3.3).

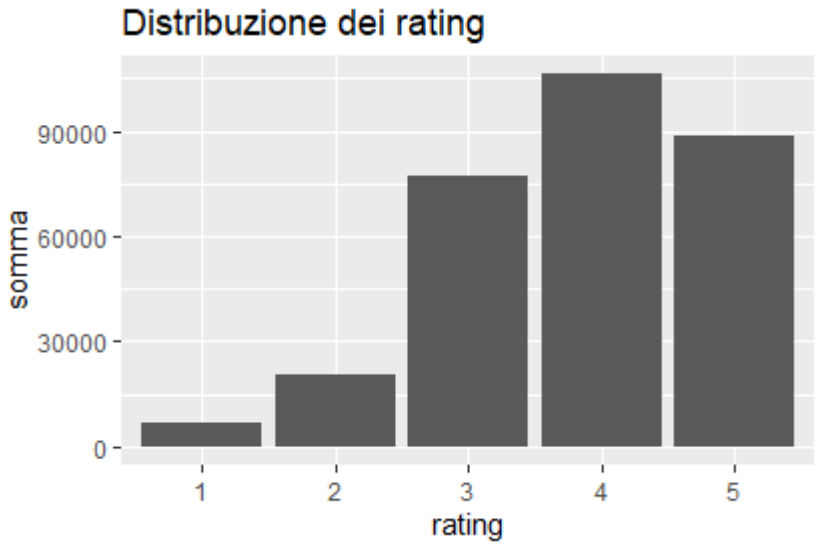


Figura 3.3 – Distribuzione dei rating

La constatazione che il *rating* più comune assegnato sia 4 emerge con chiarezza nel grafico 3.4, dove si sono calcolati i valori medi delle valutazioni degli utenti. I valori più elevati si hanno, quindi, dal punteggio di 3,5 a 4; al contrario, ci sono solo pochi libri a cui è stato assegnato un punteggio di 1 o 5, tanto che il valore 1 non appare nemmeno nel grafico.
Una spiegazione a tali risultati può essere ascritta al fatto che i libri con valori 1 o 5 possono aver ricevuto un *rating* solo da pochi individui, di conseguenza questo valore marginale non ha registrato un peso tale da essere visibile nei dati (Figura 3.4).

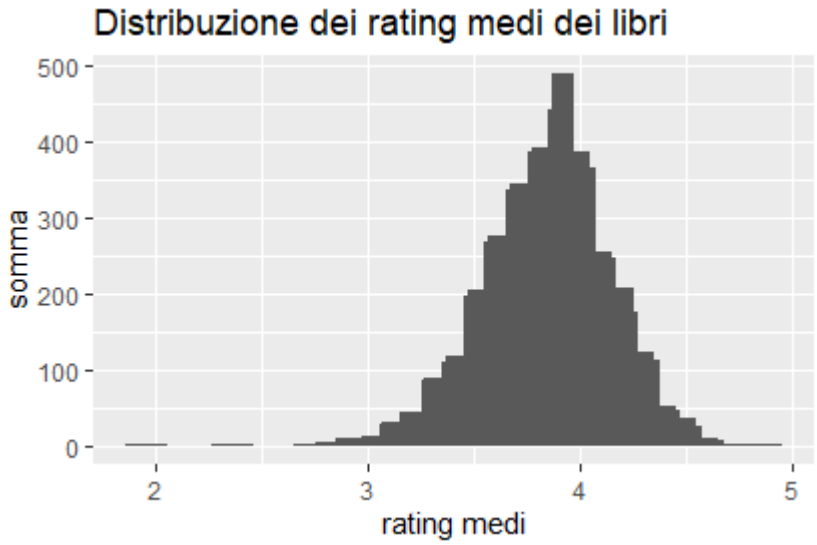


Figura 3.4 – Distribuzione dei rating medi dei libri

A livello grafico, si può avere una visione complessiva e riassuntiva del *dataset* sul quale si andranno ad applicare gli algoritmi di raccomandazione attraverso la costruzione di una *heatmap*, letteralmente una mappa di calore, la quale, in base a sfumature di uno stesso colore, permette di fornire una visione grafica dei singoli valori contenuti in una matrice.

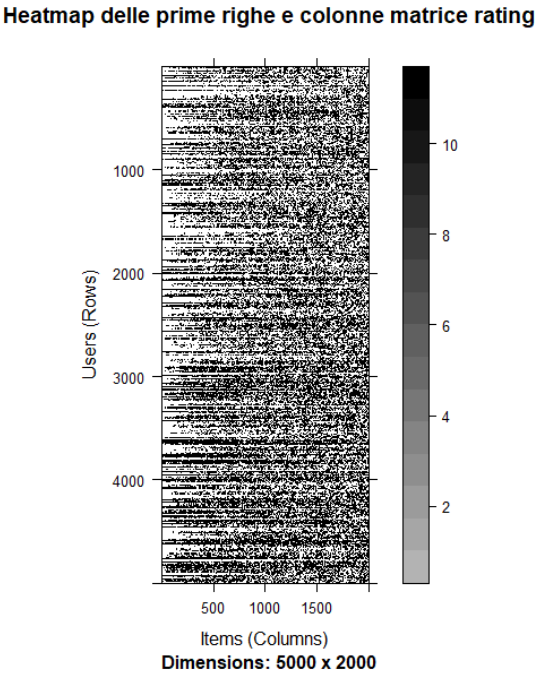


Figura 3.5 – Heatmap delle prime righe e colonne matrice rating

Dalla *heatmap* (Figura 3.5) si evince chiaramente che vi è una distribuzione abbastanza uniforme dei dati, poiché vi sono sempre in generale, a diversi livelli di grandezza, tanti utenti che leggono tanti libri. Nello specifico, si può notare come le zone di calore più intense si concentrino nei punti in cui più di 1000 utenti leggono e assegnano un rating a circa mille libri, pressoché 3000 leggono tra i 500 e i 1000 libri e più di 4000 persone sono interessante e meno di 500 libri. Questo grafico dimostra che più del 20% degli utenti totali ha letto quasi la metà dei libri presi in considerazione e, in linea generale, quasi tutti gli utenti hanno letto e votato almeno il 30% dei libri.

Sebbene tale *heatmap* riveli questo tipo di informazioni, bisogna tenere a mente che questa è la rappresentazione solo dei primi utenti e libri della matrice; per cui la mia analisi procede con la selezione degli utenti e *item* più rilevanti, ossia lo scopo è quello di visualizzare solo gli utenti che hanno letto tanti libri e libri che sono stati letti da tanti individui.

A tale fine, ho iniziato a determinare il numero minimo di libri per utente e il numero minimo di utenti per libro, per poi selezionare gli individui e i libri che rispondono ad entrambi i criteri.

Nel caso pratico, ho determinato il numero minimo di utenti e libri visualizzando i percentili per *user* e *item* ad un livello di confidenza del 99%. Anche in questo caso, ho computato l'*heatmap* (cfr. Figura 3.6) per visualizzare i risultati e ciò che emerge è che la maggior parte degli utenti ha letto e assegnato *rating* elevati ai primi mille libri più rilevanti, mentre più aumenta il numero di libri più si registrano meno utenti che hanno espresso un *rating* in merito.

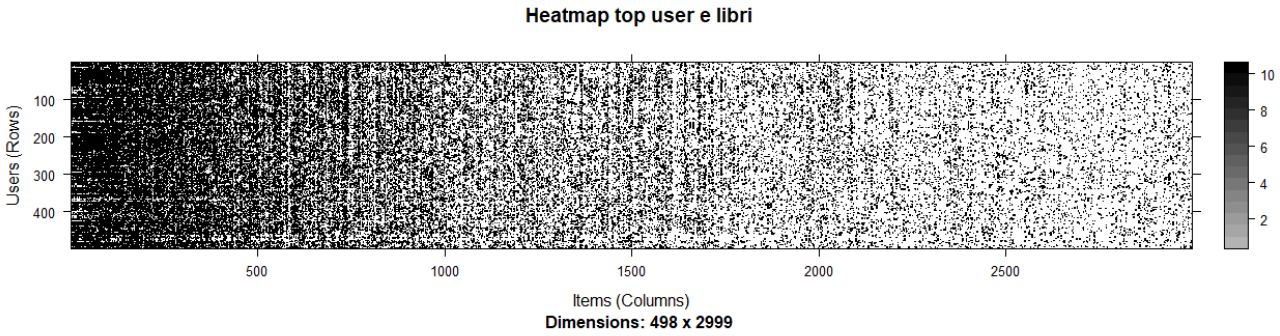


Figura 3.6 – Heatmap top user e libri

Le colonne con una colorazione più intensa rappresentano, invece, i libri con un *rating* alto; anche in questo caso, valutazioni più elevate sono concentrate nei primi 1000 libri.
Avere utenti che forniscono una valutazione alta o bassa per tutti i libri potrebbe portare a dei *bias* nei risultati finali. Per questo motivo, si è cercato di eliminare questo effetto normalizzando i dati in modo tale che il *rating* medio per ogni utente sia uguale a 0, anche se nel mio caso questi comportamenti degli utenti non sono molto frequenti e il risultato finale è molto simile a quello visibile nella Figura 3.6; per questo motivo, non viene riportato il grafico.

3.5. Item-based e user-based collaborative filtering

Prima di procedere con la descrizione del modello di raccomandazione e dei suoi risultati, si descrivono brevemente i due approcci della tecnica del *collaborative filtering*.
Il modello *item-based* si costruisce sulla similarità tra gli *item* per comprendere se un *user* sarà interessato a questi ultimi o meno, mentre un modello *user-based* computa gli utenti con abitudini di consumo simili e raccomanda ad un terzo utente simile ai primi un prodotto che questi trovano interessante.

Queste due tecniche si ascrivono all'approcio *memory based* (cfr. Capitolo 1).

La tecnica *item-based*, anche chiamata *item-to-item*, prende in considerazione un set di prodotti valutati da un individuo. Successivamente, si computa la similarità di questi ultimi con un determinato item "target", in modo tale che il *rating* di questo sarà dato dalla media pesata dei *rating* degli *user* rispetto a *item* simili a quello target [22].

Questo metodo fu introdotto nel 1998 da Amazon, che per primo ha sfruttato la misura di similarità tra i diversi prodotti per capire e registrare quanti "utenti che hanno acquistato un prodotto X hanno anche acquistato l'elemento Y". Se la correlazione è abbastanza alta, si presume che esista una similarità tra i due *item* [17].

A livello generale, un modello *item-based* tende ad avere migliori risultati quando si è in presenza di dati in cui il numero di utenti supera il numero di *item*, poiché in questa situazione ogni prodotto tenderà ad avere più *rating* rispetto ad ogni utente e, in questo modo, l'algoritmo di raccomandazione potrà avere a disposizione un numero maggiore di dati da poter processare [22].

Il metodo *user-based* utilizza lo stesso procedimento della tecnica *item-to-item*, anche se parte dal calcolo delle similarità dei *rating* di utenti simili. Tale tecnica, però, non performa molto bene quando si è nella situazione di molti *item* e pochi *rating* ed è, inoltre, soggetta al possibile cambiamento dei profili degli utenti, che possono portare a dover ricomputare tutto il modello.

3.6. Costruzione e applicazione del modello di raccomandazione

Prima di applicare la tecnica del *collaborative filtering* (sia *user-based* che *item-based*) ai dati analizzati, è opportuno anticipare che il *dataset* sarà diviso in *training* e *test set*, dove nel primo gruppo di dati l'algoritmo viene addestrato sugli utenti presenti e successivamente questo applica quanto appreso agli individui del *test set*. Gli utenti dei due gruppi vengono selezionati in modo random e secondo le percentuali dell'80% nel *training* e del 20% nel *test set*, ampiamente utilizzate in letteratura[31]. Questa suddivisione sarà utile per la valutazione del sistema.

Per quanto riguarda l'implementazione del modello *item-based collaborative filtering*, si parte dall'idea che, dato un nuovo utente, l'algoritmo considererà le scelte dello stesso e gli

[31] Per ulteriori informazioni si veda il capitolo 2 paragrafo 2.2. In letteratura si è fatto principalmente riferimento a Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Analysis of recommendation algorithms for e-commerce. Proceedings of the 2nd ACM conference on Electronic commerce pages 158-167, 2000.

raccomanderà prodotti simili. Nello specifico, l'algoritmo, considerando di volta in volta coppie di *item*, misurerà la loro reciproca similarità in base al fatto che questi prodotti abbiano ricevuto *rating* simili da simili utenti [11]. In seguito, si identificheranno i *k item* più simili per ogni prodotto e, infine, per ogni utente si prenderanno gli *item* che rispecchiamo maggiormente le sue scelte.

Dopo aver computato la similarità tra ciascuna coppia di *item* (nello specifico si avrà una matrice quadrata, nel mio caso di 3003x3003 *item*[32]), si è impostato il modello di raccomandazione mediante l'utilizzo del pacchetto *recommenderlab* di R e da ciò risulta che questo modello di *item-based collaborative filtering* apprende da una base di 3934 utenti.

In seguito, è possibile applicare quanto studiato nel *training test* al *test set*. In particolare, per ogni individuo l'algoritmo estrarrà i libri valutati e per ogni libro mostrerà gli elementi simili partendo dalla matrice di similarità. Gli step che l'algoritmo utilizza per ordinare ogni *item* simile sono i seguenti: in primo luogo, si estraggono i *rating* dell'utente per ogni libro letto. Tali *rating* sono utilizzati come dei pesi, che verranno successivamente moltiplicati per la misura di similarità tra quell'*item* e ogni acquisto associato a quell'*item*.

Nel mio caso, l'algoritmo computerà le prime sei raccomandazioni all'interno del *test set*, che saranno le seguenti (per comodità si riportano i primi sei suggerimenti solo del primo utente):

- 296[33]: "The Very Hungry Caterpillar" di E. Carle
- 390: "The Strange Case of Dr. Jekyll and Mr. Hyde" di R.L.Stevenson
- 492: "Speaker for the Dead" di O. Scott Card
- 659: "Dreams from My Father" di B.Obama
- 936: "Wolves of the Calla" di S. King e B. Wrightson
- 953: "The Dark Tower" di S.King

Per un'analisi più completa ho applicato al *dataset* anche la tecnica *dell'user-based collaborative filtering*.

Mentre la tecnica di filtraggio collaborativo con il focus sugli *item* prevede di identificare i prodotti simili sulla base del fatto che sono stati acquistati dalle stesse persone e, infine, suggerire a un nuovo utente prodotti che sono simili a quelli acquistati; l'approccio *user-based* segue uno schema differente.

[32] Si suppone che questa matrice non sarà simmetrica, in quanto il numero di elementi non nulli per ogni colonna dipende dal numero di volte in cui il corrispondente libro è incluso nella lista top k di un altro libro.

[33] I numeri 296, 390, 492, 659, 936, 953 sono gli *id* a cui corrisponde il nome del libro, suggeriti per il primo utente. Gli *id* dei libri verranno riportati anche alla pagina successiva, quando si citano le raccomandazioni con il metodo *user-based collaborative filtering*.

In primis, dato un nuovo utente, si identificano individui simili a lui e successivamente si raccomandano gli item con rating più elevati acquistati da utenti simili all'*user* target.
Nel dettaglio, per ciascun nuovo utente, si misura quanto ogni individuo è simile al nuovo user attraverso, come nel caso di *item based collaborative filtering*, della misura di similarità del coseno. In seguito, si identificano gli individui più simili, mediante l'utilizzo della tecnica del *k-nearest- neighbors*, che prende in considerazione solo k-utenti più vicini al nuovo utente; oppure si prendono semplicemente in considerazione gli user che registrano una similarità sopra una definita soglia: in questa analisi viene utilizzato il primo approccio.
Si fornisce una valutazione agli *item* acquistati dagli utenti più simili. Il *rating* non è altro che la media pesata, secondo la misura della similarità, dei *rating* computati tra gli utenti simili. Come ultimo passaggio, si prendono in considerazione i prodotti con una valutazione più elevata.
Dopo aver applicato tali passaggi ai dati, le raccomandazioni, anche in questo caso, per semplicità, solo su un utente, sono le seguenti:

- 842: "The Court of Thorns and Roses" di S.J.Maas;
- 374: "A Short History of Nearly Everything" di B. Bryson;
- 766: "Prelude & Nocturnes" di N. Gaiman;
- 403: "The Art of War" di S. Tzu e T. Clearly;
- 307: "The Wise Man's Fear" di P.Rothfuss;
- 490: "Maus: A Supervisor's Tale: My Father Bleeds History" di A. Spiegelman.

Per avere una visione più dettagliata, è possibile calcolare il numero di volte in cui un libro viene raccomandato e costruire un istogramma di frequenza, come si può osservare nella figura 3.7.

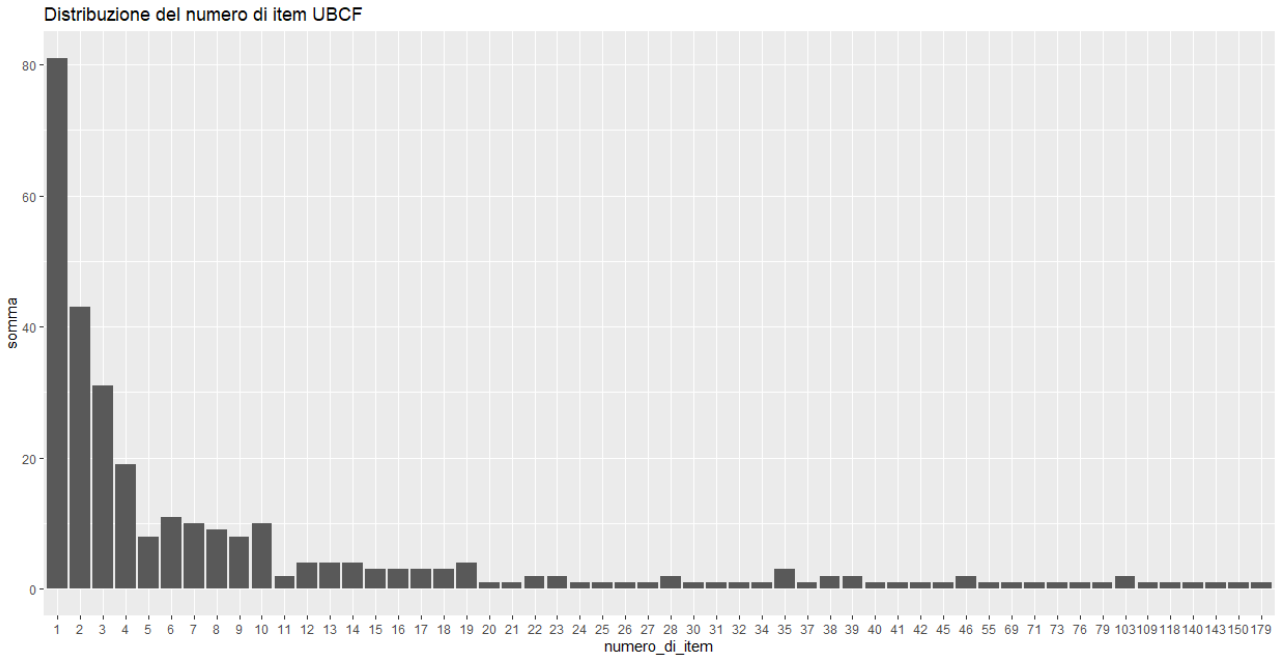


Figura 3.7 – Distribuzione del numero di item (UBCF)

Dal grafico si può notare come ci siano alcuni libri che sono stati suggeriti molte più volte rispetto ad altri.

3.7. Valutazione dei modelli

Sebbene un sistema di raccomandazione possa fornire suggerimenti in grado di aiutare un utente a scegliere tra una grande quantità di informazioni, è necessario, al fine di implementarli maggiormente e apportare anche modifiche migliorative, comprendere in che modo e in quale misura un algoritmo possa essere efficiente, nel senso di fornire informazioni che realmente rispecchiano i gusti e le preferenze di un utente. È, dunque, importante indagare se l'algoritmo sbaglia con le sue raccomandazioni e soprattutto, in che misura commette errori.

A livello operativo, si opera uno split il *dataset* in due parti, *training* e *test*, i quali hanno le stesse caratteristiche del modello di raccomandazione iniziale.

Inoltre, per ogni utente presente nel *test set*, si definiscono quanti prodotti utilizzare per le raccomandazioni, mentre la parte rimanente sarà usata per testare l'accuratezza del modello. Inoltre, il valore di questo parametro (i.e. *item* per costruire raccomandazioni) risulta minore di circa il 20% del numero minimo di *item* acquistati da un utente, in modo tale da non trovarci nella situazione di utenti senza *item* per testare i modelli.

Come affermato in linee generali nel capitolo 2, valutare un modello consiste nel comparare le raccomandazioni costruite nella parte di *test set* nota con degli acquisti "non conosciuti".
Poiché questo approccio testa l'algoritmo solo su una parte di utenti, per tenere in considerazione tutti gli utenti e avere delle prestazioni più accurate, ho scelto di utilizzare il metodo del *k-fold* (cfr. capitolo 2 paragrafo 2.2), il quale divide il *dataset* in insiemi di dati di uguale numerosità; si esclude un gruppo di questi insiemi che viene considerato come *test set* e calcola, così, l'accuratezza dell'algoritmo di raccomandazione.
È, inoltre, necessario, definire una soglia di *rating* che identifichi le valutazioni ritenute positive da quelle ritenute negative: nel caso specifico, avendo a riferimento una scala di *rating* 1-5, la soglia di *rating* che definisce una valutazione come almeno buona è 3; di conseguenza libri che hanno ricevuto *rating* 1 o 2 non sono stati valutati positivamente.
In merito all'analisi valutativa dei *rating*, si prenderanno in considerazione le misure di accuratezza previsionale: RMSE, MSE e MAE.
La prima metrica (RMSE), ossia la radice quadrata del MSE, misura la deviazione standard della differenza tra *rating* reali e *rating* predetti dal sistema; l'errore medio assoluto è, invece, la media della differenza assoluta tra *rating* reali e *rating* calcolati dall'algoritmo.
Di seguito verranno analizzati prima i risultati del metodo *item-based* e successivamente quelli computati con la tecnica *user-based*.
Dall'osservazione degli errori nelle valutazioni in un contesto di applicazione della tecnica di filtraggio collaborativo *item-based*, si nota come l'errore assoluto oscilli da circa 0.4 a 2 (Tabella 3.4). L'RMSE, che smorza l'effetto di enfasi di errori molto elevati, propria dell'errore quadratico medio (MSE), è ritenuta una metrica molto significativa dell'errore di previsione e in questo caso assume valori tra 0,6 e 2, in linea con l'errore medio assoluto.

Tabella 3.4 – RMSE, MSE, MAE (tecnica item-based)

	RMSE	**MSE**	**MAE**
35	0,6492	0,4215	0,4310
368	1,0863	1,1800	0,8000
588	2,2928	5,2569	2,0694
1088	1,4194	2,0147	1,0294
1185	1,0946	1,1981	0,8556
1350	2,0682	4,2773	1,3453

Una visione globale dell'andamento del RMSE è possibile averla mediante il grafico sottostante (Figura 3.8), che mostra la distribuzione di tale metrica per ogni utente. Emerge, come aveva

suggerito la tabella precedente, che per la maggior parte degli individui il valore di RMSE si trova in un range compreso tra 0,5 e 1,7.

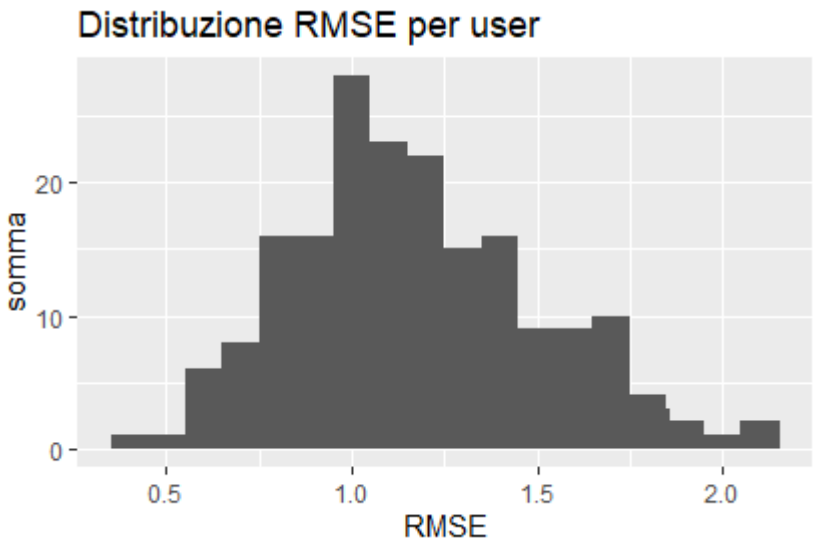


Figura 3.8 – Distribuzione RMSE per user

In modo analogo, per quanto concerne il metodo *user- based*, si nota che i valori delle medesime metriche risultano molto più simili tra gli utenti. Nello specifico, si osserva che la media della differenza assoluta tra valori predetti al sistema di raccomandazione e *rating* effettivamente assegnati dagli utenti varia tra 0,61 e 0,94; l'RMSE ricopre, invece, un range tra 0.78 e 1.32. Anche l'MSE assume valori simili al RMSE, il che lascia intendere che non sono presenti errori molto grandi (cfr. Tabella 3.5).

Tabella 3.5 – RMSE, MSE, MAE (tecnica user-based)

	RMSE	**MSE**	**MAE**
35	0,7815	0,6107	0,6181
368	0,9993	0,9987	0,7283
588	0,7819	0,6113	0,6306
1088	0,9658	0,9328	0,7958
1185	0,7119	0,5068	0,6316
1350	1,3253	1,7565	0,9421

Questo risultato emerge con chiarezza anche visualizzando il grafico della distribuzione del RMSE per ogni utente: i valori sono compresi tra 0,5 e 1,3 circa (cfr. Figura 3.9).

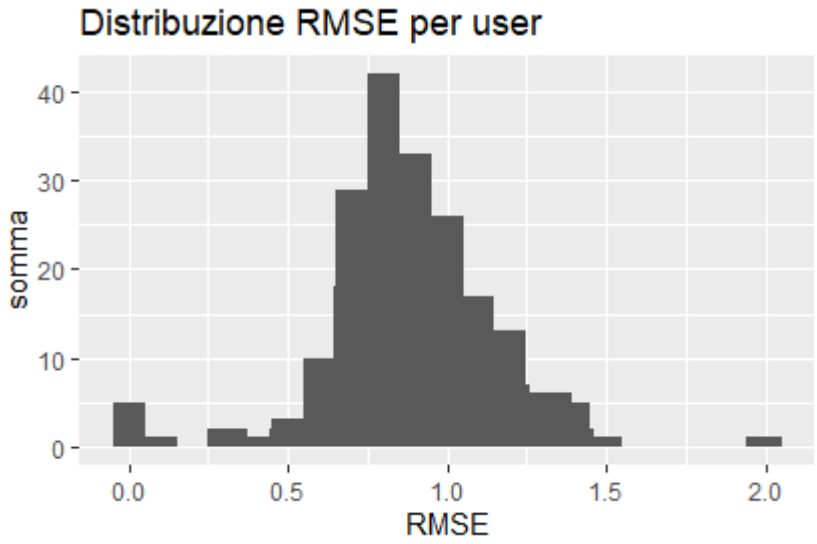


Figura 3.9 – Distribuzione RMSE per user

Un ulteriore modalità per misurare l'accuratezza di un algoritmo consiste nel comparare le raccomandazioni con gli acquisti reali che hanno ricevuto un *rating positivo*.

In questo caso, si fa riferimento alla cosiddetta matrice di confusione che confronta le previsioni delle raccomandazioni con lo stato effettivo della scelta dell'utente. In particolare, si calcolano i seguenti valori:

- Veri positivi: sono *item* che sono stati raccomandati e poi realmente acquistati;
- Veri negativi: sono, in questo caso, libri che non sono stati suggeriti dall'algoritmo e che non stati acquistati dagli utenti;
- Falsi positivi: sono prodotti raccomandati dal sistema, ma che non sono stati acquistati;
- Falsi negativi: *item* che non sono stati suggeriti dall'algoritmo di raccomandazione, ma acquistati realmente dagli utenti.

In linea generale, un modello perfetto dovrebbe registrare solo valori veri positivi e veri negativi: in questo caso, però, bisogna accertarsi che non ci sia *overfitting*.

Di seguito la matrice di confusione (cfr. Tabella 3.6).

Tabella 3.6 – Matrice di confusione

		Realtà	
		Positivi	Negativi
Previsioni	Positivi	Veri Positivi	Falsi Positivi
	Negativi	Falsi Negativi	Veri Negativi

A livello empirico e tornando a focalizzarci sull'analisi del dataset Goodbooks, si otterranno i seguenti valori utilizzando la tecnica *item-based* (cfr. Tabella 3.7). I valori ottenuti devono essere letti considerando che è stato impostato a tre valore soglia per i rating da ritenersi positivi e inoltre i risultati fanno riferimento ai primi 2510 item raccomandati per ogni user[34].

Tabella 3.7 – Matrice di confusione (item-based collaborative filtering)

	TP	FP	FN	TN	Precision	Recall	TPR	FPR	Accuracy
10	0,4718	9,1179	52,3795	2871,0310	0,0492	0,0064	0,0064	0,0032	0,98
510	11,3026	477,2974	41,5487	2402,8510	0,0231	0,2380	0,2380	0,1656	0,82
1010	17,6308	830,5744	35,2205	2049,5740	0,0210	0,3936	0,3936	0,2882	0,70
1510	18,3179	895,4718	34,5333	1984,6770	0,0206	0,4126	0,4126	0,3106	0,68
2010	18,3282	896,1538	34,5231	1983,9950	0,0206	0,4131	0,4131	0,3109	0,68
2510	18,3282	896,1538	34,5231	1983,9950	0,0206	0,4131	0,4131	0,3109	0,68

Da tale tabella (Tabella 3.7) emerge come i risultati siano abbastanza positivi, con la metrica del *recall* che arriva fino a 41% e delle misure di *accuracy* molto vicine al valore 1, che indica a massima accuratezza.

L'accuratezza, infatti, è una metrica importante quando si parla della valutazione di un algoritmo di raccomandazione e, in generale, di un classificatore, in quanto valuta la percentuale di raccomandazioni correttamente classificate.

Nel caso specifico, si nota chiaramente che i livelli di *accuracy* diminuiscono all'aumentare dei prodotti raccomandati agli utenti, proprio in ragione del fatto che l'algoritmo tende a non essere più tanto preciso nei suggerimenti quando è portato a raccomandare maggiori quantità di libri, per cui inevitabilmente tenderà a consigliare anche elementi che non risultano poi tanto vicini ai gusti dei consumatori. Tale considerazione è avvalorata anche dal fatto che valori di *precision* decrescono al crescere della quantità di prodotti suggeriti dall'algoritmo.

A livello grafico, le curve che meglio descrivono i *trade-off* tra il tasso di valori vero positivi e il tasso di valori falso positivi, e le misure di *precision* e *recall* sono rispettivamente la curva ROC e la curva PR.

La curva ROC rappresenta una tecnica statistica che misura l'accuratezza di un algoritmo di *machine learning* confrontando due misure: il tasso di veri positivi e il tasso di falsi positivi. La prima misura indica la percentuale di prodotti raccomandati sulla totalità dei prodotti acquistati realmente dagli *user*. Al contrario, il tasso di falsi positivi fornisce una misura degli *item* consigliati dal sistema di raccomandazione, ma che non sono stati acquistati. Ogni punto che compone questa curva indica in che misura sono presenti queste componenti e, in generale, più

[34] I valori della tabella, come quelli illustrati nella successiva Tabella 8, sono valori in termini di TP, FP, FN, TN, precision, recall, TRP, FRP e accuracy degli item suggeriti ad ogni utente. Tali valori sono indicativi dell'andamento generale delle raccomandazioni per ogni utente.

questa curva mostra una grande pendenza più significa che vi è un buon incremento del *true positive rate* o sensibilità, con una piccola perdita del tasso di valori falso positivi. Di conseguenza, dato un valore soglia di rating da ritenere positivi, che in questo caso è tre, più la curva è spostata a nord-ovest migliore sarà la performance dell'algoritmo, nel senso che si massimizzeranno le raccomandazioni "corrette" e minimizzati gli errori.

Nel caso pratico, la curva sottostante (Figura 3.10) non restituisce una performance molto significativa, in quanto i valori registrati si trovano all'incirca sulla retta inclinata di 45°, la quale identifica valori che hanno la medesima percentuale di corrette ed errate raccomandazioni.

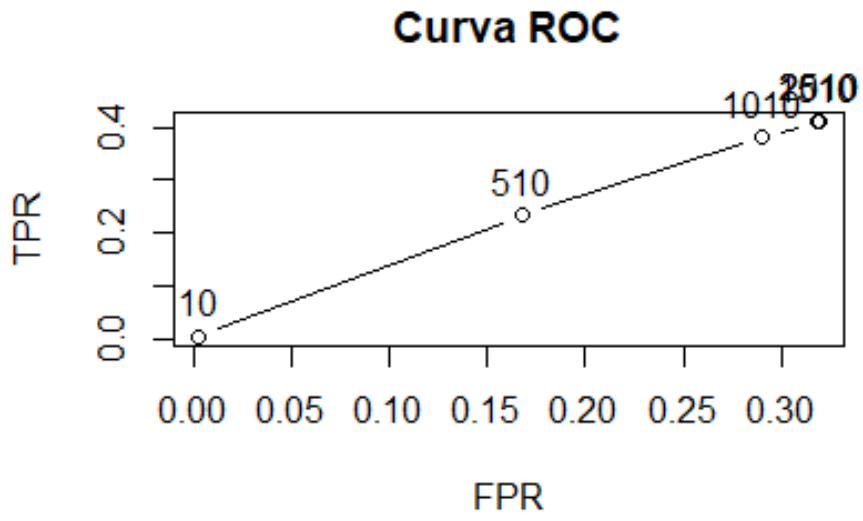


Figura 3.10 – Curva ROC

La curva di *precision- recall* concentra l'analisi sui valori positivi. Nello specifico, questa curva mostra il *trade-off* tra *precision*, che fornisce il numero di *item* raccomandati scelti successivamente dagli utenti su tutti i prodotti raccomandati, e *recall*, ossia la percentuale di prodotti acquistati che sono stati raccomandati.

Dopo aver applicato il metodo *item-based collaborative filtering*, la curva PR segue il trend atteso, anche se si nota come più elevati livelli di *recall* implichino un grande sacrificio in termini di *precision* (cfr. Figura 3.11).

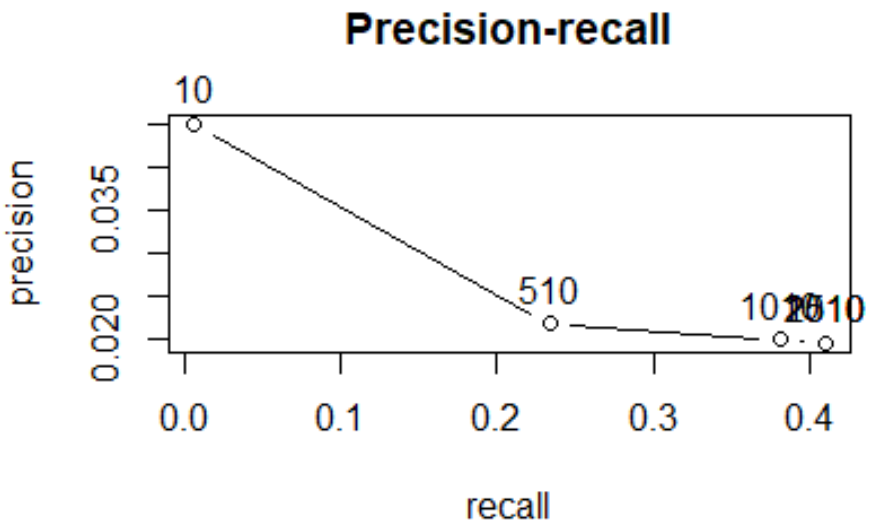


Figura 3.11 – Curva precision-recall

Performance più o meno in linea con i risultati precedenti si registrano anche per l'algoritmo *user-based*, il quale, sebbene mostri livelli molto più elevati in termini di *true positive rate* e *recall*, i quali arrivano quasi al 90%, registra valori dell'*accuracy* molto meno convincenti rispetto al modello precedente[35] (Tabella 3.8): questo risulta particolarmente evidente quando l'algoritmo raccomanda maggiori quantità di libri. Di conseguenza, si può affermare che il modello *item-based* risulta migliore di quello *user-based* in termini di *accuracy* e soprattutto per grandi quantità di *item* raccomandati.

Tabella 3.8 – Matrice di confusione (user-based collaborative filtering)

	TP	FP	FN	TN	Precision	Recall	TPR	FPR	Accuracy
10	0,1538	9,8462	52,6974	2870,3026	0,0154	0,0023	0,0023	0,0034	0,98
510	10,0308	499,9692	42,8205	2380,1795	0,0197	0,1663	0,1663	0,1736	0,81
1010	31,5436	978,4564	21,3077	1901,6923	0,0312	0,5381	0,5381	0,3397	0,66
1510	38,6923	1471,3077	14,1590	1408,8410	0,0256	0,6998	0,6998	0,5108	0,49
2010	41,9692	1968,0308	10,8821	912,1179	0,0209	0,7814	0,7814	0,6833	0,33
2510	46,9641	2463,0359	5,8872	417,1128	0,0187	0,8853	0,8853	0,8552	0,16

Anche la curva ROC mostra il suo tipico andamento crescente e in questo caso si nota maggiormente che, sempre considerando tre come valore soglia, i punti della curva, soprattutto

[35] Anche per quanto riguarda l'algortmo user-based il valore soglia nella cui ottica leggere ed interpretare i risultati della tabella 3.8 è tre.

dal numero di item 510 al numero 1010, tendono in maniera maggiore al punto di massima utilità situato a nord-ovest del grafico (Figura 3.12).

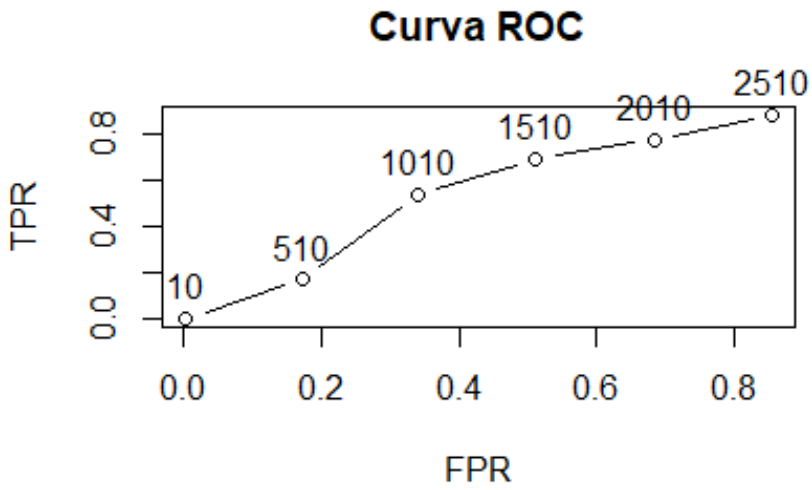


Figura 3.12 – Curva ROC

Per la curva *precision-recall*, si registrano due andamenti: in particolare, si vede come per i primi 1000 item ad aumenti del valore della *precision* seguono aumenti del *recall*; tuttavia, valori ottimali si hanno nella zona in alto a destra del grafico, in cui entrambe le metriche assumono valori elevati. Questo è il trend seguito dalla curva dall'*item* 1010 all'*item* 2500 (Figura 3.13).

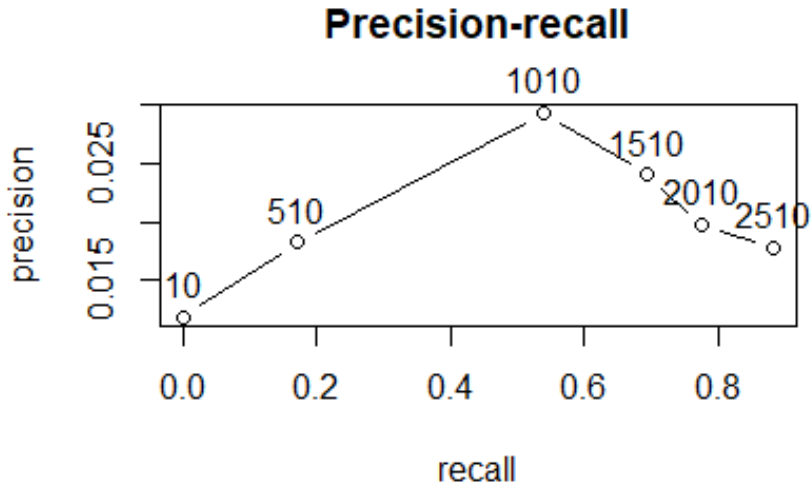


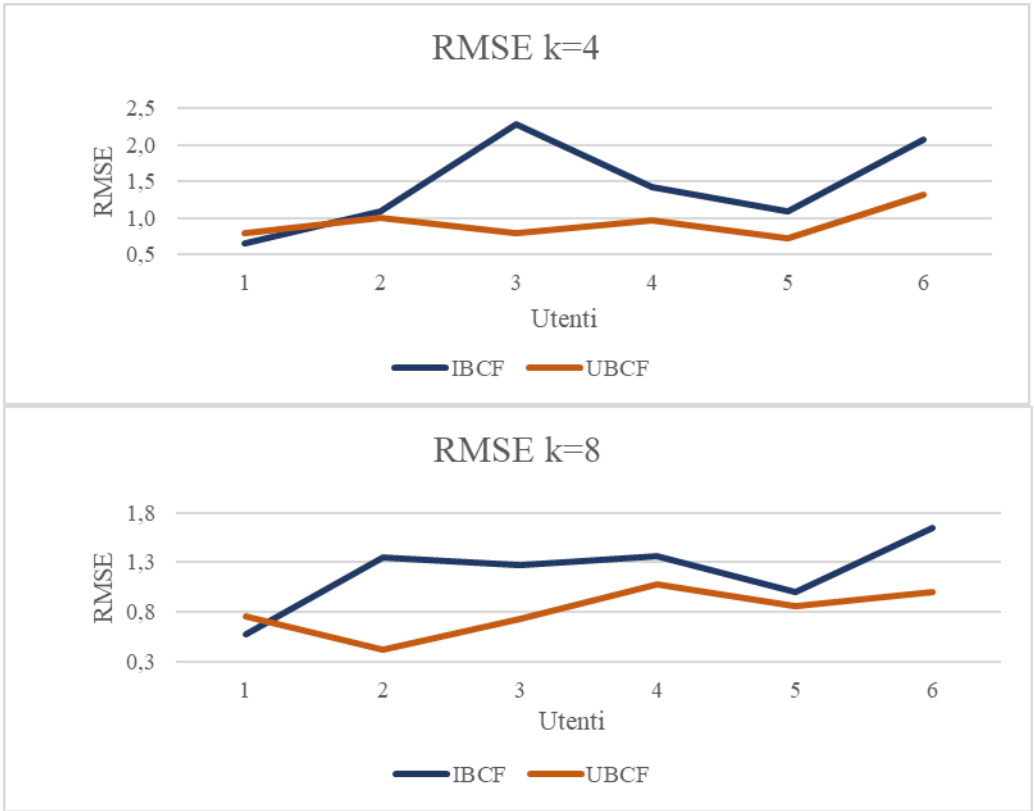
Figura 3.13 – Curva precision-recall

Se quanto precedentemente affermato definisce in generale i modelli e ne valuta le performance, e quindi si potrebbe affermare che risponde alla prima domanda di ricerca, per il secondo punto della presente tesi è necessario operare dei confronti tra i valori dell'errore RMSE, computati per diverse composizioni del *training* e *test set*.

Se già in una prima analisi dei valori della radice quadrata dell'errore quadratico medio riportata nelle pagine precedenti si osserva che la tecnica *user-based* riporta valori decisamente migliori, per confermare quanto studiato nella ricerca svedese, è stato necessario confrontare l'andamento di questa metrica al variare della composizione e delle metodologie di suddivisione del *set* di *training* e di *test*.

Poiché nel corso dell'analisi generale del *dataset* si è utilizzata la tecnica del *k-fold* per la suddivisione dei dati, ho deciso di confrontare i valori di RMSE al variare del numero di parti *k*. Nel dettaglio, ho confrontato l'andamento per valori di *k*=4 e *k*=8 (Figura 3.14).

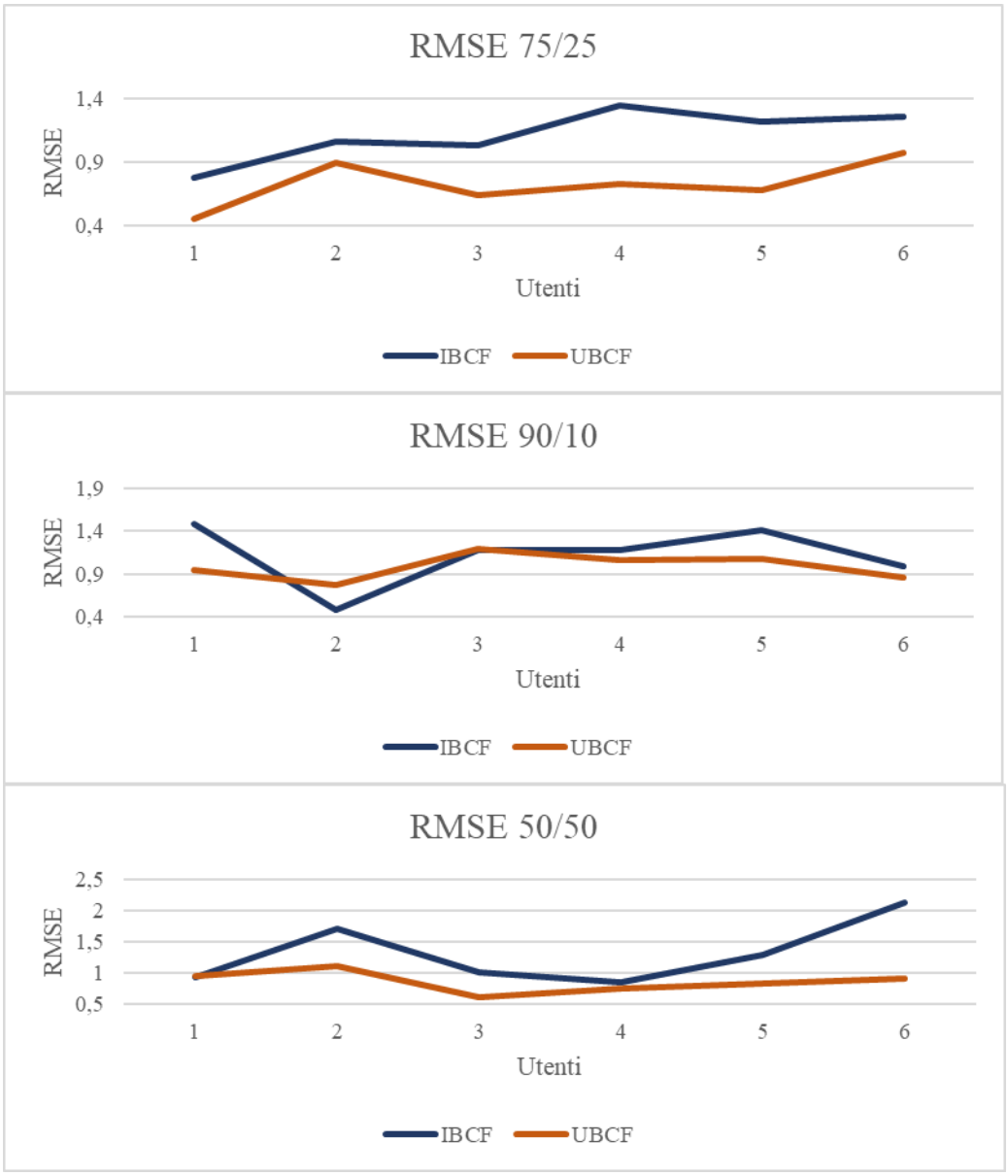
Figura 3.14 – Confronto dei valori di RMSE per k=4 e k=8 per i metodi UBCF e IBCF



Da quello che emerge da questi grafici, la tecnica *user-based* (UBCF) registra un andamento dell'errore quadratico medio migliore rispetto al modello *item-based*. Si nota, inoltre, come all'aumentare del numero di *k*, i valori di RMSE in media diminuiscono per ogni utente, anche se il risultato a livello globale non cambia.

Sebbene quanto osservato confermi in linea generale i risultati della ricerca di Boström e Filipsson [4], bisogna considerare che i grafici appena mostrati applicano la tecnica della convalida incrociata per la divisione del *dataset*, al contrario dello studio originario che prevedeva la divisione dei dati attraverso l'assegnazione di diverse percentuali alle due parti. [36] Per questa ragione, si è provveduto a suddividere i dati secondo le percentuali proposte dalla ricerca svedese: i.e. 75% *training test* e 25% *test set*, successivamente 90% *training* e 10% *test* e infine 50% *training* e 50% *test.* La figura 3.15 riassume graficamente i risultati.

Figura 3.15 – Confronto dei valori RMSE per differenti composizioni del training e test set



[36] È opportuno, però, notare che alla luce del procedimento di suddivisione dei dati con il metodo *k-fold*, un valore di *k*=4 significa che nel *set* di *training* è presente il 75% dei dati mentre il 25% è nel *test set*. Analogamente, *k*=8 implica una suddivisione tra *training/test* di 12,5% - 87%: percentuali, che, mentre nella prima circostanza rispecchiano una casistica trattata dagli studiosi svedesi, nel secondo punto si avvicina molto alla suddivisione 90% *training* e 10% *test*.

I dati presentano una situazione in cui si nota chiaramente, soprattutto nei casi 75/25 e 50/50, che il modello *user-based collaborative filtering* commette meno errori in termini di RMSE.
In particolare, si osserva come una netta miglior performance dell'algoritmo *user-based* sia più evidente nella situazione in cui il *training test* contiene il 75% dei dati: in questo caso, infatti, i valori del RMSE per il modello *user-based* oscillano tra lo 0,4 e 0,9, a fronte di un *range* che va da 0,8 a 1,4 del metodo *item-based*.
Negli altri casi, questa differenza nel comportamento dei due algoritmi non risulta così netta anche se ampiamente osservabile. Dall'osservazione dei dati si può anche notare che entrambi i modelli registrano performance migliori nel caso della divisione 75/25 e che l'algoritmo *item-based* assume valori dell'errore quadratico medio molto più variabili rispetto al modello *user-based*.
Inoltre, per entrambi i modelli, si osservano le peggior performance in termini di errore nel caso di suddivisione del *dataset* in 50% training e 50% test: infatti, l'RMSE calcolato con la tecnica *user-based collaborative filtering* assume valori intorno all'1, mentre l'algoritmo *item-based* collaborative filtering raggiunge indici di errore quadratico medio che superano addirittura 2.
Tali risultati sono stati ottenuti iterando la procedura di valutazione dieci volte per ciascun modello e ciascuno split di *dataset*.

3.8. Esiti della ricerca e commenti

L'applicazione delle tecniche di *collaborative filtering* ha mostrato dei risultati abbastanza soddisfacenti in considerazione di entrambe le domande di ricerca.
Ad un'analisi più generale che fornisce una risposta alla prima domanda di ricerca e in considerazione delle tecniche applicate, si nota come il metodo dell'*item-based collaborative filtering* mostri valori di *accuracy* leggermente superiori rispetto alla tecnica *user-based,* anche se per entrambi i sistemi all'aumentare della sensitività si registra una diminuzione dell'accuratezza generale; inoltre, la metrica della *precision* assume valori bassi per entrambi i metodi utilizzati.
Questo trend potrebbe essere ascritto al fatto che l'accuratezza e la precisione delle raccomandazioni diminuiscono all'aumentare del numero di prodotti raccomandati per ogni utente.
Un'ulteriore motivazione che potrebbe spiegare e giustificare la situazione di alta *recall* e bassa *precision* potrebbe anche trovarsi nella composizione della matrice di confusione: nello

specifico, quando il set di dati è relativamente bilanciato e il numero delle istanze positive (TP e TN) è maggiore rispetto a quello dei valori negativi, si crea questa situazione di più alta *recall* e bassa *precision,* poiché molti valori negativi vengono classificati in modo errato portando come risultato un aumento di falsi positivi.

È, in linea generale, difficile comprendere le motivazioni che hanno portato a tali risultati, poiché sicuramente vi è l'influenza di più fattori, che contribuiscono in proporzione diversa alle performance finali. Sicuramente il *dataset* usato per la ricerca non è tra i più rinomati per l'applicazione di tali algoritmi, in quanto molto recente. È possibile, però, trovare una spiegazione di quanto riscontrato nella distribuzione dei dati: infatti, come già affermato nel paragrafo 3.4 dell'analisi esplorativa, le tre variabili registrano una varianza molto elevata, il che potrebbe aver potuto portare a delle difficoltà nel raggruppamento di dati più o meno simili: uno step necessario per poter addestrare e far funzionare correttamente l'algoritmo.

Anche per quanto concerne la similarità tra gli utenti, si è notato che molti di questi leggono molti libri; di conseguenza, l'algoritmo potrebbe aver sbagliato nell'operare suddivisioni efficaci (e.g. applicazione del metodo *k-nearest neighbors*), proprio perché non in grado di discriminare tra i prodotti o tra gli utenti e, dunque, in difficoltà nel dividere gruppi di utenti simili al loro interno ma meno simili ad insiemi diversi da quello di appartenenza.

Per ricapitolare, dunque, il d*ataset* oggetto di analisi si compone di elementi molto variabili, in cui sono presenti più utenti che *item.*

Inoltre, questi *item* risultano per la maggior parte molto simili e una considerazione simile può essere fatta per gli utenti che avranno a loro volta preferenze simili, dato che per la maggior parte questi leggono una mole considerevole di libri, e questi *item* registrano tra di loro forti somiglianze.

Queste possono essere considerate alcune motivazioni che potrebbero spiegare i risultati ottenuti. Non, è, d'altronde, da sottovalutare il fatto che il metodo del *collaborative filtering* è soggetto di per sè a delle limitazioni che possono portare a potenziali problemi dell'algoritmo (cfr. cap. 1, par.1.7).

In particolare, questo tipo di raccomandazione riscontra dei problemi con nuovi utenti e nuovi *item*. Se, ad esempio, un nuovo utente non ha ancora letto un libro, nessun tipo di algoritmo di raccomandazione (né *user-based* né *item-based*) sarà in grado di suggerirgli alcun *item*.

Inoltre, un sistema *item-based* non può operare se non conosce gli *item* acquistati dal nuovo utente e, allo stesso modo, un algoritmo *user-based* necessita di conoscere quali utenti registrano simili preferenze rispetto al nuovo *user,* anche se non si hanno informazioni rispetto ai *rating* di quest'ultimo. Nel caso in esame, si è visto come ci siano *item* molto simili, di conseguenza, si potrebbe incorrere nel caso semplicistico in cui molti prodotti risultino simili

al nuovo *item*, creando potenzialmente problemi di *overfitting* e allo stesso tempo l'algoritmo potrebbe classificare erroneamente gruppi di *item* simili [4].

Se questo vale per il caso di algoritmo *item-based*, anche per la tecnica *user-based*, e in presenza di informazioni meno dettagliate sugli utenti, si potrebbe presentare una situazione in cui può essere arduo trovare gli utenti più simili ad un altro di cui non si sa nulla e, perciò, questo potrebbe compromettere il corretto funzionamento dell'algoritmo.

In maniera analoga, se nuovi libri non sono stati acquistati da nessuno, non potranno mai essere raccomandati. L'algoritmo *item-based collaborative filtering* opera un match tra gli *item* che sono stati acquistati dagli stessi utenti e, così facendo, il nuovo *item* non sarà mai preso in considerazione [27].

Stesso destino avrà un nuovo *item* anche nel caso del metodo *user-based*, in cui le raccomandazioni a ciascun utente sono fatte sulla base di preferenze di simili individui; per cui se nessuno ha acquistato o espresso un *rating* per questo nuovo libro, esso non verrà mai raccomandato.

È possibile ovviare a questo problema ricercando ulteriori informazioni in merito a nuovi *user* o *item*, quali descrizioni dei prodotti o profilo di un utente, per includerli nell'analisi: sfortunatamente nel caso concreto non si hanno notizie aggiuntive sulle variabili e questo potrebbe anche essere un ulteriore fattore che ha contribuito a fare registrare valori bassi di alcuni indicatori.

Connessa a queste considerazioni vi è un altro importante limite del metodo *collaborative filtering*: nella maggior parte dei casi, i *rating* rappresentano l'unico strumento in grado di delineare le preferenze di un utente e molto spesso tali valutazioni risultano anche incomplete poiché non tutti gli utenti valutano tutti i prodotti, per cui risulta impegnativo non solo migliorare e ottimizzare l'algoritmo, ma soprattutto, cercare *in primis* che questo operi correttamente.

In merito alla seconda domanda di ricerca, i grafici presentati nel paragrafo precedente hanno ampiamente dimostrato che quanto postulato dalla ricerca condotta dagli studiosi dell'Istituto Reale di Tecnologia di Stoccolma sia vero, anche utilizzando il dataset da me scelto per l'analisi empirica [4].

Infatti, in un'ottica di valutazione della radice quadrata dell'errore quadratico medio, che fornisce una misura della differenza tra le valutazioni espresse sottoforma di *rating* che gli utenti attribuiscono ad un prodotto e le stesse valutazioni previste dall'algoritmo di raccomandazione, si è osservato che tra le due tecniche di *collaborative filtering*, il modello *user-based* ha registrato risultati migliori della tecnica *item-based*.

Inoltre, e in aggiunta a quanto dimostrato dalla ricerca svedese, si è notato che performance più accurate dell'algoritmo *user-based* in termini di errore si registrano sia in caso di split dei dati secondo le percentuali 75% *training* – 25% *test*, 90% *training* – 10% *test* e 50% *training* – 50% *test*, sia mediante l'utilizzo del metodo di convalida incrociata *k-fold*.

Conclusioni

La presente tesi si pone il principale obiettivo di illustrare le tecniche di raccomandazione più comunemente usate e applicarne una ad un *set* di dati, al fine di comprendere a livello empirico i meccanismi che regolano tali algoritmi e analizzare le performance secondo i vari indicatori.
Nell'ultimo capitolo è stata applicata la tecnica del filtraggio collaborativo ad un *dataset* di libri, per avere suggerimenti in merito alle preferenze di individui simili all'utente per il quale il sistema fornirà raccomandazioni.
I risultati dell'analisi da me svolta hanno dimostrato che, tra i due tipi di algoritmi di *collaborative filtering*, ossia *item-based* e *user-based*, quest'ultimo registra performance migliori in termini di RMSE: in altre parole, l'algoritmo *user-based* sbaglia meno, poiché l'errore dovuto alla differenza tra *rating* che assegna l'utente e *rating* predetto dal sistema risulta minore rispetto all'altra tecnica di raccomandazione. Come si è già affermato, tale risultato conferma le conclusioni a cui era precedentemente giunta una ricerca dell'Istituto Reale di Tecnologia di Stoccolma nel 2017 [4].
A livello generale, l'analisi sperimentale svolta sul *dataset* Goodbooks riporta dei risultati che in base agli indicatori sono da considerarsi positivi o meno positivi: in primo luogo si osserva che il modello *item-based collaborative filtering* registra valori migliori di *accuracy* rispetto alla tecnica *user-based*. Inoltre, entrambi i metodi di *collaborative filtering* implementati hanno registrato alti livelli di *recall*, che dà informazioni circa la percentuale di libri che rientrano nelle preferenze degli utenti effettivamente raccomandati, e bassa *precision*, i.e. la percentuale di prodotti raccomandati che sono di interesse per gli utenti: ciò dovuto molto probabilmente alla circostanza in cui suggerire diversi libri potrebbe far aumentare il margine di errore dell'algoritmo.
Delle possibili giustificazioni a tali risultati possono trovarsi nella distribuzione dei dati, i quali risultano in media tanto variabili e, di conseguenza, rendono difficile per l'algoritmo implementato operare una suddivisione corretta e individuare gli elementi più simili. Un'ulteriore possibile spiegazione potrebbe essere connessa ai limiti ontologicamente insiti nel tipo di raccomandazione del *collaborative filtering*: problematiche quali la sparsità delle matrici *user-item* o il cosiddetto problema del *cold start* (i.e. quando un nuovo utente entra nel sistema o viene inserito un nuovo prodotto in un catalogo) possono influire negativamente sui risultati ottenuti.

È, in ogni caso, da tenere in considerazione la natura del *dataset* analizzato: si parla, infatti, di dati raccolti *online* e pubblicati su un sito accessibile a tutti, per cui questi potrebbero non essere abbastanza strutturati per garantire risultati perfetti.
Nonostante queste osservazioni, l'analisi globale effettuata rileva comunque dei buoni *output*. D'altronde, correggere e perfezionare un algoritmo di raccomandazione risulta alquanto ostico, poiché richiede strumenti ed abilità non indifferenti.
A tal proposito, non si può non ricordare il celeberrimo Netflix Prize. Nell'ottobre 2006 Netflix, colosso nella distribuzione via internet di film, serie tv e documentari a pagamento lancia, infatti, una competizione internazionale aperta a chiunque per implementare il suo algoritmo di raccomandazione, il famoso Cinematch, del 10%. La sfida consisteva nell'implementazione di un algoritmo di filtraggio collaborativo che prevedesse i *rating* degli utenti per i film sulla base di valutazioni precedenti e senza avere alcuna informazione aggiuntiva circa gli utenti o i film, se non i loro codici identificativi. La sfida fu accettata, ma solo tre anni dopo un team, i BellKor's Pragmatic Chaos, riuscì a portare a termine il compito: per loro una ricompensa da un milione di dollari.

R Commands

https://www.kaggle.com/zygmunt/goodbooks-10k#ratings.csv

```
ratings <- read.csv("C:/Users/Alessia/Downloads/ratings.csv")
View(ratings)

#preparazione del dataset e analisi delle variabili
#caricamento dei pacchetti utili per l'analisi e l'implementazione del modello
library("data.table")
library("ggplot2")
library("recommenderlab")
library("countrycode")
dim(ratings)
#[1] 981756      3
class(ratings)
str(book_final)
book_final<-ratings[-c(300001:981756),]
View(book_final)
book_final<-book_final[,c(2,1,3)]

#analisi esplorativa
summary(book_final)
var(book_final$user_id)
var(book_final$book_id)
var(book_final$rating)
Book_matrix<-as(book_final,"realRatingMatrix")
Book_matrix
#18935 x 3003 rating matrix of class 'realRatingMatrix' with 299432 ratings.
similarity_users<-similarity(Book_matrix[1:5,],method = "cosine", which="users")
```

```
class(similarity_users)

similarity_items<-similarity(Book_matrix[,1:5],method =
"cosine",which="items")

as.matrix(similarity_items)

image(as.matrix(similarity_items),main="Item similarity")
```

#costruzione modello di raccomandazione

```
recommender_models<-recommenderRegistry$get_entries
(dataType="realRatingMatrix")

names(recommender_models)

recommender_models$IBCF_realRatingMatrix$parameters

dim(Book_matrix)

#[1] 18935  3003

slotNames(Book_matrix)

dim(Book_matrix@data)

numero_user<-rowCounts(Book_matrix)

qplot(numero_user)+stat_bin(binwidth=10)+ggtitle("Distribuzione del
numero di user)+ labs(x="numero user", y="somma")

vector_ratings<-as.vector(Book_matrix@data)

unique(vector_ratings)

table_ratings<-table(vector_ratings)

table_ratings

vector_ratings<-vector_ratings[vector_ratings!=0]

vector_ratings<-factor(vector_ratings)

qplot(vector_ratings)+ggtitle("Distribuzione dei rating")+labs(x=
"rating", y="somma")

average_ratings<-colMeans(Book_matrix)

qplot(average_ratings)+stat_bin(binwidth = 0.1)+ggtitle
("Distribuzione dei rating medi die libri")+labs(x="rating medi",
y="somma")

image(Book_matrix,main="Heatmap della matrice rating")

image(Book_matrix[1:5000,1:2000], main="Heatmap delle prime righe e
colonne")
```

```
min_n_books<-quantile(rowCounts(Book_matrix),0.99)

min_n_users<-quantile(colCounts(Book_matrix),0.99)

min_n_books

min_n_users

image(Book_matrix[rowCounts(Book_matrix)>min_n_books,colCounts(Book_
matrix)>min_n_users],main="Heatmap top user e libri")
```

```
#normalizzazione dei dati
```

```
ratings_books_norm<-normalize(Book_matrix)

ratings_books_norm_mean<-rowMeans(ratings_books_norm)

sum(ratings_books_norm_mean[ratings_books_norm_mean>0.00001],na.rm=T
RUE)

image(ratings_books_norm[rowCounts(ratings_books_norm)>min_books,col
Counts(ratings_books_norm)>min_users],main="Heatmap dei top user e
libri")
```

```
#definizione training e test set e costruzione dei modelli di
raccomandazione IBCF & UBCF
```

```
which_train<-sample(x=c(TRUE,FALSE),size = nrow(Book_matrix),replace
= TRUE,prob = c(0.8,0.2))

head(which_train)

recc_data_train<-Book_matrix[which_train,]

recc_data_test<-Book_matrix[!which_train,]

recommender_models<-
recommenderRegistry$get_entries(dataType="realRatingMatrix")

recommender_models$IBCF_realRatingMatrix$parameters

recc_model<-Recommender(data = recc_data_train, method = "IBCF"[37],
parameter=list(k=30))

recc_model

model_details<-getModel(recc_model)

model_details$description
```

[37] Sostituendo la parola "IBCF" con "UBCF" si costruirà il modello *user-based collaborative filtering*. Nello script mostrato si fa riferimento all'algoritmo *item-based*.

```
model_details$k

dim(model_details$sim)

row_sums<-rowSums(model_details$sim >0)

table(row_sums)

which_max<-order(col_sums,decreasing = TRUE)[1:6]

rownames(model_details$sim)[which_max]
```

#applicazione dei modelli di raccomandazione al test set

```
n_recommended<-6

recc_predicted<-predict(object = recc_model, newdata =recc_data_test,
n=n_recommended)

recc_predicted

class(recc_predicted)

slotNames(recc_predicted)

recc_predicted@items[[1]]

books_user_1<-recc_predicted@itemLabels[recc_user_1]

books_user_1

recc_matrix<-
sapply(recc_predicted@items,function(x){colnames(rating_books)[x]})

recc_matrix[,1:4]

dim(recc_matrix)

numero_di_item<-factor(table(recc_matrix))

chart_title<-"Distribuzione del numero di item UBCF"

qplot(numero_di_item)+ggtitle(chart_title)

number_of_items_sorted<-sort(number_of_items,decreasing=TRUE)

number_of_items_top<-head(number_of_items_sorted,n=4)

table_top<-
data.frame(names(number_of_items_top),number_of_items_top)

table_top
```

#valutazione degli algoritmi

#metodo k-fold

```
n_fold<-4[38]

eval_sets<-evaluationScheme(data=Book_matrix,method="cross-
validation",k=n_fold,given=items_to_keep,goodRating=rating_threshold
)

size_sets<-sapply(eval_sets@runsTrain,length)

size_sets
```

#valutazione delle tecniche di raccomandazione utilizzando il metodo k-fold -IBCF[39]

```
#valutazione dei rating utilizzando metodo k-fold con item_to_keep=68
e rating_threshold=3

model_to_evaluate<-"IBCF"

model_parameters<-NULL

eval_recommender<-Recommender(data=getData(eval_sets,"train"),
method= model_to_evaluate,parameter=model_parameters)

items_to_recommend<-10

eval_prediction<-predict(object  =  eval_recommender,newdata=getData
(eval_sets,"known"),n=items_to_recommend, type="ratings")

class(eval_prediction)

eval_accuracy<-calcPredictionAccuracy(x=eval_prediction,data=getData
(eval_sets,"unknown"),byUser=TRUE)

head(eval_accuracy)

qplot(eval_accuracy[,"RMSE"])+geom_histogram(binwidth                =
0.1)+ggtitle("Distribuzione RMSE per user")

eval_accuracy<-calcPredictionAccuracy(x=eval_prediction,
data=getData (eval_sets,"unknown"),byUser=FALSE)

eval_accuracy

results<-evaluate(x=eval_sets,method=model_to_evaluate,
n=seq(10,3000,500))

class(results)

head(getConfusionMatrix(results)[[1]])

columns_to_sum<-c("TP","FP","FN","TN")

indices_summed<-
Reduce("+",getConfusionMatrix(results))[,columns_to_sum]

head(indices_summed)

plot(results,annotate=TRUE,main="ROC curve")
```

[38] Per rispondere alla seconda domanda di ricerca, è stato eseguito lo stesso procedimento sostituendo k=8 a k=4.
[39] I comandi eseguiti per la tecnica dell'*item-based collaborative filtering* sono gli stessi utilizzati per il metodo *user-based collaborative filtering*, solo che la dicitura "IBCF" viene sostituita con "UBCF".

```
plot(results,"prec/rec",annotate=TRUE,main="Precision-recall")
```

#split dei dati

```
percentage_training<-0.75[40]

min(rowCounts(rating_books))   #81

items_to_keep<-68

rating_threshold<-3

n_eval<-10

eval_sets<-evaluationScheme(data=rating_books,method="split",  train=
percentage_training,given=items_to_keep,goodRating=rating_threshold,
k=n_eval)

eval_sets

getData(eval_sets,"train")

nrow(getData(eval_sets,"train"))/nrow(rating_books)

getData(eval_sets,"known")

getData(eval_sets,"unknown")

nrow(getData(eval_sets,"known"))/nrow(rating_books)

unique(rowCounts(getData(eval_sets,"known")))

model_to_evaluate<-"IBCF"

model_parameters<-NULL

eval_recommender<-
Recommender(data=getData(eval_sets,"train"),method=
model_to_evaluate,parameter=model_parameters)

items_to_recommend<-8

eval_prediction<-predict(object      =      eval_recommender,newdata=
getData(eval_sets,"known"), n=items_to_recommend, type="ratings")

class(eval_prediction)


eval_accuracy<-calcPredictionAccuracy(x=eval_prediction,data=
getData(eval_sets,"unknown"),byUser=TRUE)

head(eval_accuracy)
```

[40] Per soddisfare le richieste poste dalla seconda domanda di ricerca sono stati utilizzati i seguenti valori percentuali del training set: 0.75, 0.90 e 0.50.

BIBLIOGRAFIA

[1] Adomavicius, G., Kwon, Y., (2008). “Overcoming Accuracy-Diversity Tradeoff in Recommender Systems:A Variance-Based Approach”, WITS’08.

[2] Anwar, T., Uma, V., (2019). “CD-SPM: Cross-domain book recommendation using sequential pattern mining and rule mining”, *Journal of King Saud University – Computer and Information Sciences.* Accessibile da: https://doi.org/10.1016/j.jksuci.2019.01.012

[3] Ashenfelter, A., Donaldson, J., Hangartner, R., Martin, F.J., Torrens, M., (2011). “The Big Promise of Recommender Systems”, *AI Magazine*, 32 (3), 19-27.

[4] Boström, P., Filipsson, M., (2017). “Comparison of User Based and Item Based Collaborative Filtering Recommendation Services”. Accessibile da: https://kth.diva-portal.org/smash/get/diva2:1111865/FULLTEXT01.pdf.

[5] Burke, R., Felferning, A., Göker, M. H., (2011). “Recommender Systems: An Overview”, *AI Magazine*, 32 (3), 13-18.

[6] Cremonesi, P., Lentini, E., Matteucci, M., Turrin, R., (2008). “An Evaluation Methodology for Collaborative Recommender System”, AXMEDIS '08. International Conference.

[7] Falk, K., (2019). “Practical Recommender System”, Manning Shelter Island.

[8] Ferrara, G., Muggeo, V. M.R., (2005), “Il linguaggio R: concetti introduttivi ed esempi”, II edizione. Accessibile da: https://cran.r-project.org/doc/contrib/nozioniR.pdf.

[9] Gomez-Uribe, C.A., Hunt, N., (2015). “The Netflix recommender system: Algorithms, business value and innovation”, *ACM Trans. Manage. Inf. Syst.6*, 4, Article 13.

[10] Gorakala, S. K., (2015), “Item Based Collaborative Filtering Recommender Systems in R”. Accessibile da: https://www.r-bloggers.com/item-based-collaborative-filtering-recommender-systems-in-r/.

[11] Gorakala, S. K., Usuelli, M., (2015). “Building a Recommendation System with R”, (2015), Packt Publishing.

[12] Gunawardana, A., Shani, G., (2011). “Evaluating Recommendation System” in Recommender System Handbook, 257-297.

[13] Herlocker, J., Konstan, J., Terveen, L., and Riedl, J., (2004). “Evaluating collaborative filtering recommender systems”, *ACM Transactions on Information Systems*, 22(1):5–53.

[14] Hofmann, T., (2004). “Latent Semantic Models For Collaborative Filtering”. *ACM Transactions on Information Systems*, 22(1): 89-115.

[15] Jomsri, P., (2018). “FUCL mining tecnique for book recommender system in library service”, *Procedia Manufacturing* (22), 550-557.

[16] Kaggle, (2017), “Goodbooks-10k”.
Accessibile da: https://www.kaggle.com/zygmunt/goodbooks-10k#books.csv.

[17] Linden, G., Smith, B., York, J., (2003). “Amazon.com recommendations: item-to-item collaborative filtering”, *Internet Computing, IEEE Internet Computing*,76-80.

[18] Liu, X., Haihong, E., Song, J., Song, M., Tong, J., (2013). “Collaborative Book Recommendation Based on Readers' Borrowing Records”, International Conference on Advanced Cloud and Big Data, Nanjing, Jiangsu, China, 159-163.

[19] Ma, Y. G., (2014). “Research of Library Book Recommendation System Based on Cloud Computing”, In Proceedings of the 9th International Symposium on Linear Drives for Industry Applications, 549-555.

[20] Maneewongvatana, S., (2010). “A recommendation model for personalized book lists”, *International Symposium on Communications and Information Technologies (ISCIT)*, 389-394.

[21] Miller, B. N., (1995). “ GroupLens: An Open Architecture for Collaborative Filtering“.
Accessibile da:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.3784&rep=rep1&type=pdf.

[22] Medium (2018), “Comparison of User-Based and Item-Based Collaborative Filtering”. Accessibile da: https://medium.com/@wwwbbb8510/comparison-of-user-based-and-item-based-collaborative-filtering-f58a1c8a3f1d.

[23] Mobasher, B., Dai, H., Luo, T., Nakagawa, M., (2001). “Effective personalization based on association rule discovery from web usage data”, *Web Information And Data Management*.

[24] Okpedia (2016), “TF-IDF”. Accessibile da: https://www.okpedia.it/tf-idf.

[25] Oikonomou, M., Vafopoulos, M., (2011). “Recommendation systems: a joint analysis of technical aspects with marketing implications”. *ArXiv e.print*: 1112.2251, 4-11.

[26] Pera, M.S., (2015). “Analyzing Book-Related Features to Recommend Books for Emergent Readers“, The 26th ACM Conference on Hypertext & Social Media ACM, 221-230.

[27] Pinela, C., (2017). “Recommender Systems - User-Based and Item-Based Collaborative Filtering“. Accessibile da: https://medium.com/@cfpinela/recommender-systems-user-based-and-item-based-collaborative-filtering-5d5f375a127f.

[28] Powers Ward D.M., (2011). “Evaluation: From precision, recall and F-measure to ROC, informedness, markedness & correlation”, J. Mach. Learn. Technol, vol2.

[29] Renuka, J., (2016), “Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures”. Accessibile da: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/.

[30] Ricci, F., Rokach, L., Shapira, B., Kantor, P.B., (2011). “Recommender Systems Handbook”, Springer-Science Business Media, LLC, 367-386.

[31] Rocca, B., (2019), “Introduction to recommender systems. Overview of some major recommendation algorithms”. Accessibile da: https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada.

[32] Sarwar, J. B., Karypis, G., Konstan, J., and Riedl, J., (2000). “Analysis of recommendation algorithms for e-commerce”, 2nd *ACM conference on Electronic commerce,* 158-167.

[33] Schafer, J. B., Frankowski, D., Herlocker, J., Sen, S., (2007). “Collaborative filtering recommender systems“, *Springer Computer Science*, 291-324.

[34] Sohail, S. S., Siddiqui, J., Ali, R., (2015). “OWA based Book Recommendation Technique”, *Procedia Computer Science*, 126 – 133.

[35] Tewari, A. S., Singh, J. P., Barman, A. G., (2018). “Generating Top-N Items Recommendation Set Using Collaborative, Content Based Filtering and Rating Variance”, *Procedia Computer Science* (132), 1678–1684.

[36] Tharwat, A., (2018). “Applied Computing and Informatics”. Accessibile su: https://doi.org/10.1016/j.aci.2018.08.003.

[37] Tsiji, K., Takizawa, N., Sato, S., Ikeuchi, U., Yoshikane, F., Itsumura, H., (2014). “Book Recommendation Based on Library Loan Records and Bibliographic Information”, *Procedia Social and Behavioral Sciences* (147), 478 – 486.

[38] Topor, J., (2017), “User-Based and Item-Based Collaborative Filtering”, RPubs. Accessibile da: https://rpubs.com/jt_rpubs/285729.

[39] Tzu-Chan, C., Zhi-Hong, C., Tak-Wai, C., (2017). “Exploring Long-term Behavior Patterns in a Book Recommendation System for Reading”, *Journal of Educational Technology & Society*,20 (2), 27-36.

[40] Wang, J., Vries, A. P., Reinders, M. J. T., (2006). “Unifying User-based and Item-based CollaborativE Filtering Approaches by Similarity Fusion”, 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA.

[41] Wit, J.D. (2008). “Evaluating Recommender Systems - An evaluation framework to predict user satisfaction for recommender systems in an electronic programme guide context“, *TNO Information and Communication Technology*.

[42] Zhou, W., Wen, J., Qu, Q., Zeng, J., Cheng, T., (2018). “Shilling attack detection for recommender system based on credibility of group users and rating time series”. PLoS ONE 13(5).

LUISS

Dipartimento di Impresa e Management

Corso di Laurea Magistrale in Marketing

Cattedra di Machine Learning & Object Driven Marketing

RIASSUNTO: SISTEMI DI RACCOMANDAZIONE E COMPORTAMENTO DEL CONSUMATORE: CONFRONTO TRA USER-BASED E ITEM-BASED COLLABORATIVE FILTERING

RELATORE

Chiar.mo Prof. Luigi Laura

CANDIDATO

Alessia Delmedico

Matr. 701231

Correlatore

Chiar.mo Prof. Massimo Bernaschi

ANNO ACCADEMICO 2018/2019

Introduzione

È sempre più complicato trovare la giusta informazione al momento più opportuno, riuscire a comprare l'abito adatto al nostro evento nel più breve lasso di tempo, avere la possibilità di scegliere, tra una miriade di film a disposizione, quello più adatto al nostro *mood*. Queste sono solo alcune delle ragioni che determinano l'importanza dei cosiddetti sistemi di raccomandazione, modelli in grado di suggerire notizie personalizzate ad un generico consumatore o utente mediante tecniche di filtraggio di informazioni in grado di soddisfare le preferenze di quel consumatore. Questi sistemi, proprio perché suggeriscono dei prodotti o servizi a potenziali consumatori o, in generale, ad utenti, hanno un grande potere di condizionare le scelte di questi ultimi e di modificare, in alcuni casi, il ciclo di vita dei prodotti. Proprio alla luce di questo fondamentale ruolo svolto da tali modelli, la presente tesi mira all'analisi dei dati e all'applicazione di uno dei principali metodi di raccomandazione, i.e. il criterio del *collaborative filtering*, ai fini di comprendere e valutare le performance di questo algoritmo e confrontare i risultati ottenuti.

Poiché l'argomento che si intende trattare in questo elaborato risulta molto articolato e non di immediata comprensione, ho deciso di suddividere il mio lavoro di tesi in tre macro-sezioni, in cui le prime due sono finalizzate ad introdurre e spiegare le molteplici caratteristiche e proprietà dei sistemi di raccomandazione, anche in considerazione di metodi valutativi in grado di misurare la bontà dei suggerimenti forniti dagli algoritmi; il terzo capitolo mette in pratica quanto asserito nelle prime due parti e focalizza l'attenzione sull'analisi sperimentale di un dataset per la raccomandazione di libri.

I motivi alla base della mia scelta di tesi risiedono principalmente nel desiderio di approfondire una tematica del *machine learning* a me prima poco conosciuta e, in generale, nel particolare interesse per gli algoritmi di raccomandazione e le loro potenziali applicazioni ad un numero sempre maggiore di settori.

Se, infatti, in principio queste tecniche sono state maggiormente utilizzate in ambito informatico e si sono prestate fin da subito ad essere sfruttate nel settore dell'*e-commerce*, oggi si assiste ad un'espansione più capillare di questi algoritmi, che hanno trovato applicazione anche in settori più "delicati" e rischiosi, come può essere l'ambito medico.

Capitolo 1 – I sistemi di raccomandazione

In un contesto, dove qualsiasi tipo di bene o servizio è a portata di clic per ogni individuo e dove tutto può essere analizzato e tracciato, il singolo utente è esposto a sovrabbondanti informazioni, con la conseguenza di riscontrare difficoltà e di ritrovarsi a prendere decisioni "povere", come afferma Carlos Gomez Uribe, Vice Presidente per l'Innovazione di Prodotto in Netflix; e non perfettamente adatte a soddisfare i propri bisogni oppure di non riuscire affatto a scegliere tra alternative simili [21].

È in questo ambito che trovano campo fertile i cosiddetti sistemi di raccomandazione sui quali, implementati dall'ultimo decennio del secolo scorso, numerose ricerche e miglioramenti sono stati apportati grazie anche a tecniche di intelligenza artificiale che includono, tra le altre, *machine learning* e *data mining*. Tali sistemi di raccomandazione sono, dunque, strumenti che filtrano informazioni rilevanti per il singolo individuo in base alle sue preferenze, interessi o atteggiamento osservato nei confronti del prodotto, oggetto di scelta.

Un sistema di raccomandazione efficiente è diretto a fornire suggerimenti personalizzati che derivano dall'unione delle notizie sull'*user* e le caratteristiche del prodotto, le quali includono anche *feedback* di altri utenti e comparazione con altri articoli [3].

Uno dei loro scopi principali consiste nel predire il comportamento dell'individuo grazie alle tecniche di *artificial intelligence* utilizzate: da questa prospettiva, essi vengono sfruttati dalle imprese per adattare le caratteristiche del prodotto da loro commercializzato alle esigenze del potenziale consumatore. Indubbiamente queste ultime vedranno aumentato il loro vantaggio competitivo e assisteranno anche ad una crescita delle vendite e, di conseguenza, dei ricavi.[1]

Alla luce di quanto detto si possono, quindi, delineare due *value proposition* dei sistemi di raccomandazione (o *recommender system*s, nell'espressione inglese): da un lato, i *recommender systems* vengono utilizzati per aiutare gli utenti a fronteggiare la mole smisurata di informazioni sul web e per indirizzarli, in seguito, verso gli articoli e le informazioni con più alta utilità; dall'altro si configurano come strumenti in grado di operare efficacemente e garantire alle imprese che ne fanno uso un incremento nelle vendite e, in generale, un livello più alto di *engagement* nei loro sevizi da parte degli stessi consumatori [32].

[1] Per approfondimenti: I Big Data: alla scoperta del big fenomeno, disponibile al sito https://www.teamsystem.com/store/blog/digitalizzazione/big-data-alla-scoperta-del-big-fenomeno/

Sono proprio queste due macro-esigenze a rappresentare i punti cardine da cui è possibile tracciare, raggruppare e scandire differenti fasi, che segnano la nascita, gli sviluppi e i miglioramenti in questo ambito.

In una prospettiva che pone l'enfasi sul beneficio ottenuto dall'utente mediante l'utilizzo delle tecniche di raccomandazione, si può dedurre la seguente definizione.

Dati *C* l'insieme di tutti gli utenti per i quali il sistema provvederà ad una raccomandazione, *S* l'insieme di tutti i possibili prodotti (*item*), che possono essere raccomandati e *u* la funzione di utilità, che misura l'adeguatezza del prodotto *s* per il singolo user *c*, per cui si avrà che:

$$u: C\ x\ S \Rightarrow R \quad [1.1]$$

dove *R* è l'insieme totale raccomandato. Dunque, per ogni user $c \in C$ si vuole scegliere il prodotto $s' \in S$ che massimizza l'utilità dell'utente, che è generalmente indicata da una valutazione dell'utente stesso[2].

Gli *output* che restituisce il sistema grazie all'utilizzo di un algoritmo, ossia le raccomandazioni, possono collocarsi su differenti livelli di personalizzazione, in base all'utilizzo di più generiche statistiche o di dati strettamente associati all'individuo. Nello specifico, possono distinguersi raccomandazioni non personalizzate, semi-personalizzate e personalizzate.

Per quanto riguarda le informazioni sugli utenti, queste si basano su dati relativi all'*user* in target e sulla sua *community* di riferimento. Nello specifico, si possono delineare tre categorie di conoscenza che sono alla base del corretto funzionamento dei diversi tipi di algoritmi di raccomandazione [5]. Esse sono:

- Sociale: conoscenza di una vasta comunità di utenti oltre all'individuo target;
- Individuale: conoscenza del target *user*;
- Contenuto: conoscenza delle caratteristiche dei prodotti che vengono raccomandati e, più in generale, dei loro usi.

Vi è un collegamento tra le fonti di conoscenza e le differenti tipologie di sistemi di raccomandazione. Il metodo del *collaborative filtering* connette la fonte di conoscenza individuale con quella sociale e fornisce suggerimenti per il target di utente sulla base di preferenze di individui considerati simili a lui.

[2] Generalmente per la valutazione degli *item* si fa riferimento ad una scala ad intervalli 1-5 o 1-7, dove 1 rappresenta il totale disinteresse dell'*user* nei confronti di quel particolare *item*, viceversa il 5 o il 7. Oltre ad un *rating* espresso in forma scalare, ci possono essere *rating* binari, in cui l'utente esprime valutazioni come Accordo/Disaccordo, Bene/Male; e *rating* unari, che permettono agli utenti di assegnare gli *item* ad una singola classe, in genere positiva (e.g. bottone "Like" su Facebook).

Raccomandazioni basate sul contenuto si implementano, invece, sulle fonti di conoscenza individuale e utilizzano le caratteristiche di un *item* e le opinioni di un individuo per impostare classificazioni, in modo tale da poter predire le preferenze degli utenti o nuovi prodotti.
Sistemi *knowledge-based* raccolgono informazioni più strettamente connesse all'individuo, come, ad esempio, i dati demografici, e le connettono conoscenze del settore e contesto. Per quanto riguarda sistemi di raccomandazioni ibridi, dalla prospettiva delle fonti di conoscenza, non è possibile delineare in modo univoco le fonti a cui queste sono connesse.
Il primo tipo di sistema di raccomandazione, i.e. *collaborative filtering* si basa sulla considerazione che se due utenti hanno manifestato lo stesso interesse nel passato, come, ad esempio, amare lo stesso film, essi mostreranno simili preferenze anche in futuro.
È possibile suddividere gli algoritmi di *collaborative filtering* in due sezioni: gli algoritmi *memory-based* e quelli *model-based*. Per quanto riguarda l'organizzazione degli algoritmi di *collaborative filtering*, si possono individuare tecniche probabilistiche e non probabilistiche.
Tra i modelli non probabilistici più noti si annovera l'algoritmo *nearest neighbor*. L'obiettivo del *k-nearest neighbors* consiste nel prevedere il *rating* dell'utente attivo per i prodotti non ancora valutati. In particolare, si individua un sottoinsieme di *k* individui più vicini e simili all'utente in questione e si raccomandano a quest'ultimo i prodotti per i quali i suoi vicini hanno espresso un *rating* positivo. I *rating* dell'individuo target non saranno altro che combinazioni pesate delle valutazioni fornite per gli stessi elementi dai *k* individui più simili. Da un punto di vista operativo, per misurare una somiglianza si costruisce una matrice *user-item*, in cui si riportano le valutazioni dei prodotti e degli articoli per i vari *n* utenti. La similarità è calcolata principalmente con la distanza euclidea tra due *rating* degli utenti, ma vi sono anche altre misure come il calcolo del coseno o la distanza di Manhattan. La tecnica del *nearest neighbor* può essere sua volta suddivisa in *user- based* o *item-based.* L'ultimo tipo di algoritmo non probabilistico è il modello della riduzione della dimensionalità (*dimensionality reduction*).
Gli algoritmi probabilistici riprendono esplicitamente le distribuzioni di probabilità nel calcolo delle previsioni dei *rating* o delle liste di raccomandazioni classificate.
I modelli più utilizzati in questo ambito sono le reti bayesiane (*Bayesian networks*), che calcolano la dipendenza probabilistica per gli utenti o gli *item*. In generale, una rete bayesiana è rappresentata da grafi diretti aciclici (DAG), che danno informazioni riguardo alle dipendenze condizionali di variabili stocastiche [30]. Accanto allo studio e uso delle reti bayesiane, sono state condotte anche numerose ricerche per sviluppare tecniche probabilistiche di clustering o riduzione della dimensionalità.

Accanto al *collaborative filtering*, algoritmi *content-based* costituiscono un ulteriore tipo di sistema di raccomandazione. Tale sistema suggerisce dei prodotti ad un utente, simili rispetto a quelli per cui quest'ultimo ha dimostrato interesse in passato. In particolare, sistemi di raccomandazione *content-based* suggeriscono ad un utente un *item* sulla base della descrizione e caratteristiche del prodotto e al profilo di interessi dell'individuo.

Molti sistemi di personalizzazione che analizzano testi usano una tecnica per creare una rappresentazione strutturata proveniente da sistemi di ricerca testuale. L'algoritmo *tf-idf*, acronimo di frequenza del termine-frequenza inversa nei documenti, permette di estrarre informazioni di un testo da una base dati. Questo algoritmo è composto da due parti: la prima (*tf*) determina il rapporto tra il numero di volte in cui il termine *t* è presente nel documento *d* e il numero totale delle parole presenti nello stesso documento; la seconda parte (*idf*) è data dal logaritmo del numero totale di documenti presenti nel *database* sul numero di documenti in cui è presente la parola *t* (Formula 1.2).

$$TF - IDF(t,d) = \frac{tf_{t,d}}{max_z f_{z,d}} \cdot log \frac{N}{df_t} \qquad [1.2]$$

Per quanto riguarda il profilo di interessi dell'utente, è necessario focalizzarsi principalmente su due tipi di informazione: *in primis* bisogna considerare le preferenze dell'individuo, poi assume molta importanza avere uno storico delle informazioni delle interazioni dell'*user* con il sistema di raccomandazione. Tutto ciò è molto utile, perché permette al sistema di rendere visibile all'utente i prodotti appena visualizzati oppure che l'individuo ha acquistato: l'algoritmo di *machine learning* sfrutterà queste informazioni per creare un modello per quell'utente (*user model*), che sarà basato sull'approccio di customizzazione dell'utente (*user customization*) oppure su un criterio *rule-based*.

La creazione di un modello di preferenze dell'utente basate sulle scelte e interazioni passate di quest'ultimo è una forma di apprendimento di classificazione (*classification learning*). Gli algoritmi di *classification learning* sono i seguenti: Decision Trees, metodi del Nearest Neighbor, i feedback di rilevanza e l'algoritmo di Rocchio, Classificatori Lineari (*linear classifiers*) e Naïve Bayes.

Gli algoritmi utilizzati per fornire raccomandazioni basate sulla collaborazione e sul contenuto possono presentare delle problematiche.

Tra i principali limiti di questi sistemi vi è il cosiddetto problema del *cold start*: tale problematica si verifica quando nuovi utenti entrano nel sistema o nuovi *item* sono aggiunti al catalogo di quelli disponibili. Anche il problema della sparsità (*sparsity* in inglese) rappresenta

un limite al corretto ed efficace funzionamento degli algoritmi di raccomandazione. Tale limite consiste nell'avere una grande disponibilità di *item*, ma ogni utente è riluttante ad assegnare un *rating* agli oggetti presi in considerazione.

La scalabilità (*scalability*) si ritrova nei sistemi che utilizzano un approccio di *collaborative filtering*, quando è presente un dataset molto ampio, in cui il numero degli utenti e degli *item* cresce e il sistema richiede, quindi, più risorse per dare raccomandazioni più accurate all'utente. Sistemi di tipo *collaborative* soffrono anche dei cosiddetti *shilling attacks*: si è in una situazione in cui un utente o un competitor entrano in un sistema e iniziano ad assegnare falsi *rating* ad *item*, al fine di aumentare (*push attack*) o ridurre (*nuke attack*) la popolarità di questi ultimi. Anche il *grey sheep* si verifica in sistemi *collaborative filtering*, quando le opinioni di un utente non incontrano quelle di nessun gruppo e quindi quell'utente non potrà usufruire dei benefici degli algoritmi di raccomandazione. Sistemi che adottano metodi di filtraggio collaborativo presentano anche il problema dell'inattività (*latency problem*): quando nuovi prodotti sono aggiunti più frequentemente ad un *dataset* e l'algoritmo utilizzato continua a suggerire solo gli *item* già valutati e tratta i prodotti appena inseriti come ancora non valutati. Tipico, invece, di sistemi basati sul contenuto è la problematica dell'*overspecialization*: in molti settori sono, infatti, presenti alcuni rilevanti *item* che non possono essere suggeriti, finchè il contenuto analizzato non contiene informazioni sufficienti a distinguere le preferenze di un utente, oppure vi sono prodotti caratterizzati da uno scarso contenuto e quindi il sistema, in mancanza di adeguate informazioni, non riesce ad associarli ad altri contenuti simili e, dunque, a raccomandarli. Un ulteriore problema che caratterizza i sistemi di raccomandazione è legato alla *privacy.* Al fine di arginare alcune problematiche ricorrenti negli algoritmi dei sistemi *collaborative filtering* e basati su contenuto, sono stati implementati sistemi ibridi di raccomandazione, i quali uniscono tecniche proprie dei precedenti sistemi per ottenere performance più soddisfacenti. In linee generali, i criteri adottati dai sistemi ibridi possono prevedere l'applicazione dei criteri di *collaborative filtering* e *content based* in maniera separata, incrociando solo successivamente i risultati; oppure all'approccio collaborativo vengono incorporate caratteristiche degli algoritmi che forniscono raccomandazioni basate su contenuto e viceversa [42]. Un ulteriore metodo ibrido di raccomandazione consiste nella costruzione di un modello più complesso che cerca di unire l'approccio collaborativo con quello *content-based.* Tra i tipi di sistemi ibridi di raccomandazione se ne identificano sette: pesato, *switching*, misto, *feature combination*, *feature augmentation*, *cascade* e *meta-level*.

Capitolo 2 - Valutazione dei sistemi di raccomandazione

In merito agli approcci valutativi utilizzati per verificare l'affidabilità e l'efficienza di un algoritmo di raccomandazione, si può, in primo luogo, affermare che quest' ultimo è valutato e classificato sulla base del suo potere predittivo, ossia sulla sua abilità di predire in modo accurato le scelte e preferenze degli utenti.

La valutazione *offline* avviene su un insieme di dati strutturati che registrano le scelte degli utenti, nella maggior parte dei casi sottoforma di valutazioni espresse in *rating* per dei prodotti. Il *dataset* in questione è, dunque, già esistente, diviso in *training* e *test set* e sono coinvolti utenti che hanno fornito la loro opinione; inoltre, diversi approcci di raccomandazione sono comparati senza interazione dell'utente [6]. I *rating* presenti nel *training set* vengono utilizzati dagli algoritmi di raccomandazione per predire i *rating* reali nel *test set*.

Esistono diversi metodi di partizionamento dei dati tra *training* e *test set*: tra i più importanti vi sono il metodo *holdout*, *bootstrap* e la validazione dei dati *k-fold*.

Le valutazioni *offline* sono computazionalmente veloci e semplici da condurre su un gran numero di dati, non richiedono un'interazione con un utente reale e ciò permette di comparare un ampio intervallo di algoritmi a basso costo [6].

Si parla, invece, di valutazioni *online* quando gli utenti interagiscono con un sistema di raccomandazione in esecuzione e ricevono effettivamente delle raccomandazioni. Vengono successivamente raccolti *feedback* degli utenti mediante l'osservazione dei loro comportamenti o attraverso una esplicita richiesta. Le valutazioni *online* sono, inoltre, molto utili, in quanto consentono di misurare direttamente gli obiettivi generali del sistema, come il profitto a lungo termine o la fidelizzazione dell'utente.

A prescindere dal tipo di valutazione, esistono delle metriche che misurano empiricamente la precisione (*accuracy*) degli algoritmi di raccomandazione, misurano la bontà di riproduzione delle valutazioni e liste di classificazioni di un individuo da parte del sistema di raccomandazione. Lo studioso Herlocker suddivide tali metriche in due classi principali: metriche di accuratezza previsionale (*predictive accuracy metrics*) e metriche di accuratezza nella classificazione (*classification accuracy metrics*) [13].

Le metriche di accuratezza previsionale riferiscono in che misura un sistema di raccomandazione è in grado di prevedere i vari *rating* degli utenti.

Fanno parte di questa classe l'errore medio assoluto (*mean absolute error*-MAE), l'errore quadratico medio (*mean squared error*-MSE) e la radice dell'errore quadratico medio (*root*

mean squared error-RMSE). L'errore medio assoluto è calcolato come la somma della differenza tra il *rating* espresso da un utente e il *rating* previsto dal sistema, il tutto diviso per il numero di *item* considerati. Tra le varianti dell'errore medio assoluto si annoverano l'errore quadratico medio (MSE) e la radice dell'errore quadratico medio (RMSE). L'errore quadratico medio calcola la differenza quadratica media tra le valutazioni sottoforma di *rating* degli utenti e i valori dei *rating* stimati. In questo modo, si enfatizzano errori molto grandi elevando al quadrato ogni singolo errore.

Una visione più precisa, che tenga conto degli algoritmi che operano anche con *rating* impliciti è garantita da ulteriori metriche: *precision, recall, f-measure* e curva ROC, appartenenti al gruppo delle cosiddette *classification accuracy metrics*. Questo secondo gruppo di metriche rivela in che misura un sistema di raccomandazione è in grado di classificare correttamente prodotti come interessanti o meno per un utente. La *precision*, anche nota come valore predittivo positivo, è espressa dal rapporto tra gli *item* rilevanti selezionati e il numero di *item* presi in considerazione. La misura *recall*, anche conosciuta come sensitività, è, invece, calcolata come il numero degli *item* rilevanti selezionati sul numero totale di prodotti rilevanti presenti nel set di tutti gli *item*. L'*accuracy* è, invece, il rapporto tra le osservazioni correttamente predette sul totale delle osservazioni.

Una misura che combina le metriche *precision* e *recall* è la cosiddetta *f-measure*, calcolata come segue [2.8]

$$F - measure = \frac{2 \times recall \times precision}{recall + precision} = \frac{2 \cdot TP}{2 \cdot TP + FB + FP} \qquad [2.1]$$

Per visualizzare graficamente il *trade-off* tra le metriche di *precision* e *recall*, quest'ultima anche chiamata sensitività, è possibile computare una curva *precision-recall (PR)*. Esiste, però, un'ulteriore curva, chiamata ROC (*Receiver Operating Characteristic*), che compara tassi di valori vero-positivi con tassi di valori falso-positivi. Mentre entrambe le curve di *precision-recall* e ROC misurano la proporzione di item preferiti, che sono realmente consigliati dal sistema, la curva *precison-recall* pone l'accento sulla proporzione di *item* raccomandati che entrano nelle preferenze dell'utente; la curva ROC, al contrario, enfatizza la proporzione di prodotti non preferiti che finiscono tra quelli consigliati ai consumatori.

La scelta di utilizzare la curva *precision-recall* o la curva ROC avviene in base alle proprietà del dominio e allo scopo dell'applicazione.

Capitolo 3 - Applicazione del metodo collaborative filtering alla raccomandazione di libri

I sistemi di raccomandazione rappresentano una soluzione alla problematica ormai globale e condivisa di ricerca di notizie pertinenti al soddisfacimento di determinati bisogni informativi degli individui.

I dati da cui ho attinto per la mia analisi empirica fanno parte del *dataset* "goodsbook-10k" presente su Kaggle, il quale raccoglie le valutazioni fornite dagli utenti per circa diecimila libri e in media si registrano cento *rating*, che sfruttano una scala da 1 a 5, per ogni libro.

Gli elementi oggetto della mia analisi si riferiscono ad un *dataset* di 300000 osservazioni e tre variabili: rispettivamente il codice identificativo degli utenti (*user_id*), quello per i libri (*book_id*) e i *rating* che gli *user* hanno fornito come valutazione dei libri, che sono compresi in un range di valori da 1 a 5.

Inoltre, per la mia ricerca mi sono basata su uno studio molto interessante in merito all'applicazione della tecnica del *collaborative filtering*, il quale confronta i due principali metodi di filtraggio collaborativo, ovvero *item-based* e *user-based collaborative filtering*, per capire quale dei due performa meglio e quindi registra un errore minore nel fornire raccomandazioni corrette, ossia suggerimenti che rispecchiano realmente le preferenze di un utente [4].

Alla luce dello studio effettuato dall'università svedese e in considerazione dei dati a disposizione, le domande di ricerca a cui la presente tesi intende rispondere sono due.

La prima, di carattere più generico, consiste nel delineare le performance generali dei sistemi *item-based* e *user-based collaborative filtering* applicati al *dataset* oggetto di analisi, in termini di valutazione mediante le metriche di *accuracy* e considerando la cosiddetta matrice di confusione.

La seconda domanda di ricerca ha l'obiettivo di dimostrare che, alla stregua del caso della ricerca svedese, anche per il *dataset* in esame è valido il risultato per cui, in termini di RMSE, la tecnica *user- to- user* performa meglio rispetto alla tecnica *item-to-item*.

L'applicazione delle tecniche di *collaborative filtering* ha mostrato dei risultati abbastanza soddisfacenti in considerazione di entrambe le domande di ricerca.

Ad un'analisi più generale che fornisce una risposta alla prima domanda di ricerca e in considerazione delle tecniche applicate, si nota come il metodo dell'*item-based collaborative filtering* mostri valori di *accuracy* leggermente superiori rispetto alla tecnica *user-based*, anche

se per entrambi i sistemi all'aumentare della sensitività si registra una diminuzione dell'accuratezza generale; inoltre, la metrica della *precision* assume valori bassi per entrambi i metodi utilizzati.

Questo trend potrebbe essere ascritto al fatto che l'accuratezza e la precisione delle raccomandazioni diminuiscono all'aumentare del numero di prodotti raccomandati per ogni utente. Un'ulteriore motivazione che potrebbe spiegare e giustificare la situazione di alta *recall* e bassa *precision* potrebbe anche trovarsi nella composizione della matrice di confusione: nello specifico, quando il set di dati è relativamente bilanciato e il numero delle istanze positive (TP e TN) è maggiore rispetto a quello dei valori negativi, si crea questa situazione di più alta *recall* e bassa *precision*, poiché molti valori negativi vengono classificati in modo errato portando come risultato un aumento di falsi positivi.

È, in linea generale, difficile comprendere le motivazioni che hanno portato a tali risultati, poiché sicuramente vi è l'influenza di più fattori, che contribuiscono in proporzione diversa alle performance finali.

Sicuramente il *dataset* usato per la ricerca non è tra i più rinomati per l'applicazione di tali algoritmi, in quanto molto recente. È possibile, però, trovare una spiegazione di quanto riscontrato nella distribuzione dei dati: infatti, come già affermato nel paragrafo 3.4 dell'analisi esplorativa, le tre variabili registrano una varianza molto elevata, il che potrebbe aver potuto portare a delle difficoltà nel raggruppamento di dati più o meno simili: uno step necessario per poter addestrare e far funzionare correttamente l'algoritmo.

Anche per quanto concerne la similarità tra gli utenti, si è notato che molti di questi leggono molti libri; di conseguenza, l'algoritmo potrebbe aver sbagliato nell'operare suddivisioni efficaci (e.g. applicazione del metodo *k-nearest neighbors*), proprio perché non in grado di discriminare tra i prodotti o tra gli utenti e, dunque, in difficoltà nel dividere gruppi di utenti simili al loro interno ma meno simili ad insiemi diversi da quello di appartenenza.

Per ricapitolare, dunque, il d*ataset* oggetto di analisi si compone di elementi molto variabili, in cui sono presenti più utenti che *item*.

Inoltre, questi *item* risultano per la maggior parte molto simili e una considerazione simile può essere fatta per gli utenti che avranno a loro volta preferenze simili, dato che per la maggior parte questi leggono una mole considerevole di libri, e questi *item* registrano tra di loro forti somiglianze.

Queste possono essere considerate alcune motivazioni che potrebbero spiegare i risultati ottenuti. Non, è, d'altronde, da sottovalutare il fatto che il metodo del *collaborative filtering* è

soggetto di per sè a delle limitazioni che possono portare a potenziali problemi dell'algoritmo (cfr. cap. 1, par.1.7).

In particolare, questo tipo di raccomandazione riscontra dei problemi con nuovi utenti e nuovi *item*. Se, ad esempio, un nuovo utente non ha ancora letto un libro, nessun tipo di algoritmo di raccomandazione (né *user-based* né *item-based*) sarà in grado di suggerirgli alcun *item*.

Inoltre, un sistema *item-based* non può operare se non conosce gli *item* acquistati dal nuovo utente e, allo stesso modo, un algoritmo *user-based* necessita di conoscere quali utenti registrano simili preferenze rispetto al nuovo *user*, anche se non si hanno informazioni rispetto ai *rating* di quest'ultimo. Nel caso in esame, si è visto come ci siano *item* molto simili, di conseguenza, si potrebbe incorrere nel caso semplicistico in cui molti prodotti risultino simili al nuovo *item*, creando potenzialmente problemi di *overfitting* e allo stesso tempo l'algoritmo potrebbe classificare erroneamente gruppi di *item* simili [4].

Se questo vale per il caso di algoritmo *item-based*, anche per la tecnica *user-based*, e in presenza di informazioni meno dettagliate sugli utenti, si potrebbe presentare una situazione in cui può essere arduo trovare gli utenti più simili ad un altro di cui non si sa nulla e, perciò, questo potrebbe compromettere il corretto funzionamento dell'algoritmo.

In maniera analoga, se nuovi libri non sono stati acquistati da nessuno, non potranno mai essere raccomandati. L'algoritmo *item-based collaborative filtering* opera un match tra gli *item* che sono stati acquistati dagli stessi utenti e, così facendo, il nuovo *item* non sarà mai preso in considerazione [27].

Stesso destino avrà un nuovo *item* anche nel caso del metodo *user-based*, in cui le raccomandazioni a ciascun utente sono fatte sulla base di preferenze di simili individui; per cui se nessuno ha acquistato o espresso un *rating* per questo nuovo libro, esso non verrà mai raccomandato.

È possibile ovviare a questo problema ricercando ulteriori informazioni in merito a nuovi *user* o *item*, quali descrizioni dei prodotti o profilo di un utente, per includerli nell'analisi: sfortunatamente nel caso concreto non si hanno notizie aggiuntive sulle variabili e questo potrebbe anche essere un ulteriore fattore che ha contribuito a fare registrare valori bassi di alcuni indicatori.

Connessa a queste considerazioni vi è un altro importante limite del metodo *collaborative filtering*: nella maggior parte dei casi, i *rating* rappresentano l'unico strumento in grado di delineare le preferenze di un utente e molto spesso tali valutazioni risultano anche incomplete poiché non tutti gli utenti valutano tutti i prodotti, per cui risulta impegnativo non solo

migliorare e ottimizzare l'algoritmo, ma soprattutto, cercare *in primis* che questo operi correttamente.

In merito alla seconda domanda di ricerca, i grafici presentati nel paragrafo precedente hanno ampiamente dimostrato che quanto postulato dalla ricerca condotta dagli studiosi dell'Istituto Reale di Tecnologia di Stoccolma sia vero, anche utilizzando il *dataset* da me scelto per l'analisi empirica [4].

Infatti, in un'ottica di valutazione della radice quadrata dell'errore quadratico medio, che fornisce una misura della differenza tra le valutazioni espresse sottoforma di *rating* che gli utenti attribuiscono ad un prodotto e le stesse valutazioni previste dall'algoritmo di raccomandazione, si è osservato che tra le due tecniche di *collaborative filtering*, il modello *user-based* ha registrato risultati migliori della tecnica *item-based*.

Inoltre, e in aggiunta a quanto dimostrato dalla ricerca svedese, si è notato che performance più accurate dell'algoritmo *user-based* in termini di errore si registrano sia in caso di split dei dati secondo le percentuali 75% *training* – 25% *test*, 90% *training* – 10% *test* e 50% *training* – 50% *test*, sia mediante l'utilizzo del metodo di convalida incrociata *k-fold*.

Conclusioni

La presente tesi si pone il principale obiettivo di illustrare le tecniche di raccomandazione più comunemente usate e applicarne una ad un *set* di dati, al fine di comprendere a livello empirico i meccanismi che regolano tali algoritmi e analizzare le performance secondo i vari indicatori.
Nell'ultimo capitolo è stata applicata la tecnica del filtraggio collaborativo ad un *dataset* di libri, per avere suggerimenti in merito alle preferenze di individui simili all'utente per il quale il sistema fornirà raccomandazioni.
I risultati dell'analisi da me svolta hanno dimostrato che, tra i due tipi di algoritmi di *collaborative filtering*, ossia *item-based* e *user-based*, quest'ultimo registra performance migliori in termini di RMSE.
A livello generale, l'analisi sperimentale svolta sul *dataset* Goodbooks riporta dei risultati che in base agli indicatori sono da considerarsi positivi o meno positivi: in primo luogo si osserva che il modello *item-based collaborative filtering* registra valori migliori di *accuracy* rispetto alla tecnica *user-based*. Inoltre, entrambi i metodi di *collaborative filtering* implementati hanno registrato alti livelli di *recall*, che dà informazioni circa la percentuale di libri che rientrano nelle preferenze degli utenti effettivamente raccomandati, e bassa *precision*, i.e. la percentuale di prodotti raccomandati che sono di interesse per gli utenti: ciò dovuto molto probabilmente alla circostanza in cui suggerire diversi libri potrebbe far aumentare il margine di errore dell'algoritmo. Delle possibili giustificazioni a tali risultati possono trovarsi nella distribuzione dei dati, i quali risultano in media tanto variabili e, di conseguenza, rendono difficile per l'algoritmo implementato operare una suddivisione corretta e individuare gli elementi più simili. Un'ulteriore possibile spiegazione potrebbe essere connessa ai limiti ontologicamente insiti nel tipo di raccomandazione del *collaborative filtering*.
È, in ogni caso, da tenere in considerazione la natura del *dataset* analizzato: si parla, infatti, di dati raccolti *online* e pubblicati su un sito accessibile a tutti, per cui questi potrebbero non essere abbastanza strutturati per garantire risultati perfetti.
Nonostante queste osservazioni, l'analisi globale effettuata rileva comunque dei buoni *output*.
D'altronde, correggere e perfezionare un algoritmo di raccomandazione risulta alquanto ostico, poiché richiede strumenti ed abilità non indifferenti.