LUISS

Dipartimento
di Impresa e Management

Cattedra Machine Learning & Object Driven Marketing

The power of personalization: Un confronto sperimentale sulle previsioni del dataset MovieLens

Prof. Luigi Laura
RELATORE

Prof. Marco Querini
CORRELATORE

Ilaria Ampola Matr.727501
CANDIDATO

Anno Accademico 2020/2021

INTRODUZIONE **3**

CAPITOLO 1. INTRODUZIONE AL MACHINE LEARNING **5**

1.1 OVERVIEW 5

1.2 CARATTERISTICHE 8

1.2.1 APPRENDIMENTO SUPERVISIONATO *9*

1.2.2 APPRENDIMENTO NON SUPERVISIONATO *11*

1.2.3 APPRENDIMENTO CON RINFORZO *13*

1.3 UTILIZZO DEL MACHINE LEARNING PER OBIETTIVI AZIENDALI 14

CAPITOLO 2. RECOMMENDATION SYSTEM **18**

2.1 OVERVIEW 18

2.1.1 CARATTERISTICHE *21*

2.1.2 VALUTAZIONI *23*

2.1.3 LIVELLI DI PERSONALIZZAZIONE *25*

2.2 CONTENT- BASED FILTERING 27

2.3 COLLABORATIVE FILTERING 30

2.4 SISTEMI IBRIDI DI RACCOMANDAZIONE 34

2.5 CASO NETFLIX 37

CAPITOLO 3. MOVIELENS: CONFRONTO SPERIMENTALE TRA LIBRERIE PYTHON **41**

3.1 LITERATURE REVIEW 41

3.2 LINGUAGGIO PYTHON: OVERVIEW 44

3.3 LIBRERIE PYTHON PER RECOMMENDER SYSTEM 46

3.3.1 LENSKIT *48*

3.3.2 CRAB *50*

3.3.3 SURPRISE *50*

3.3.4 REXY *51*

3.3.5 TENSORFLOW *51*

3.3.6 TENSORREC *51*

3.3.7 LIGHTFM *53*

3.3.8 CASE RECOMMENDER *53*

3.3.9 SPOTLIGHT *54*

3.4 IL DATASET MOVIELENS: ORIGINI E CARATTERISTICHE 54

3.5 IL DATASET MOVIELENS: IMPLICAZIONI MANAGERIALI 61

3.6 CONFRONTO SPERIMENTALE: METODOLOGIA 63

3.6.1 ANALISI E RISULTATI LIGHTFM *65*

3.6.2 ANALISI E RISULTATI SURPRISE *67*
3.6.3 ANALISI E RISULTATI LENSKIT *72*
3.6.4 ANALISI E RISULTATI TENSORFLOW *74*
3.7 CONFRONO SPERIMENTALE: DISCUSSIONE GENERALE 77
3.8 LIMITAZIONI E RICERCA FUTURA 79
CONCLUSIONE 82
ELENCO DELLE FIGURE 84
APPENDICE 85
BIBLIOGRAFIA 107
SITOGRAFIA 113

INTRODUZIONE

Nell'era dei big data, la quantità di informazioni è cresciuta esponenzialmente, la velocità di trasmissione è estremamente elevata e le risorse possono essere condivise (Lang, F. et al., 2021). I dati vengono sfruttati per fare scelte informate, prevedere le tendenze del mercato e i modelli nelle preferenze dei consumatori. In questo contesto, i negozi online, le librerie di musica, i video e le immagini online, i motori di ricerca e i sistemi di raccomandazione sono diventati i modi più convenienti per trovare informazioni rilevanti in breve tempo. L'idea è di utilizzare tecniche di filtraggio e clustering per suggerire elementi di interesse per gli utenti. Molte grandi aziende come Amazon, eBay e Netflix hanno adottato tecniche di raccomandazione per i loro sistemi per stimare le potenziali preferenze dei clienti, consigliare prodotti o articoli pertinenti all'utente e promuovere le vendite utilizzando differenti piattaforme. Le prestazioni di raccomandazione hanno un impatto enorme sul successo commerciale di queste aziende in termini di entrate e utenti soddisfacenti (Wei J. Et al., 2017). Proprio per questo, ad oggi, il ruolo di questi sistemi non può essere sottovalutato sia nel mondo accademico che nell'industria. Ad esempio, la maggior parte dei film guardati su YouTube e altri database di video online proviene dai sistemi di raccomandazione (Covington et al. 2016). Proprio alla luce di queste affermazioni e dell'importanza dei sistemi di raccomandazione, la presente tesi mira all'applicazione di un metodo di raccomandazione, rappresentato dal dataset MovieLens, per comprendere e valutare le performance dell'algoritmo, confrontando i risultati ottenuti da alcune librerie Python. L'obiettivo finale di questo studio è quello di comprendere quanto la previsione di una preferenza sia effettivamente accurata in relazione alle scelte reali di un utente. Per effettuare tale studio si utilizza il dataset MovieLens che si basa sulla valutazione di film, generando consigli, ampiamente utilizzato nell'ambito dell'istruzione, della ricerca e dell'industria. Per un prodotto multimediale come i film, i suggerimenti vengono forniti agli utenti trovando i profili di individui con gusti simili. Inizialmente, la preferenza dell'utente si ottiene consentendo loro di porre una propria valutazione ai film. Dopo l'utilizzo, il sistema di raccomandazione sarà in grado di comprendere meglio l'utente e suggerire film che hanno maggiori probabilità di essere valutati "più affini".[1] Per comprendere meglio il contesto, questo dataset e di conseguenza la tipologia di algoritmo, questo lavoro si suddivide in tre capitoli. Nel primo capitolo si introduce il concetto del Machine Learning, un sistema che effettua predizioni utilizzando una grande quantità di dati per stabilire una mappa tra input e obiettivi. Si spiega la sua origine e le sue caratteristiche, concentrandosi sulle sue funzionalità

[1]Subramaniyaswamy, V., Logesh, R., Chandrashekhar, M., Challa, A., & Vijayakumar, V. (2017). A personalised movie recommendation system based on collaborative filtering. International Journal Of High Performance Computing And Networking, 10(1/2), 54. DOI:https://doi.org/10.1504/ijhpcn.2017.083199

e metodologie usate, divise in: apprendimento supervisionato, apprendimento non supervisionato e apprendimento con rinforzo. Inoltre, si illustra il suo utilizzo per il raggiungimento di obiettivi aziendali e incremento del business di alcuni settori, ponendo un focus sull'ambito del *digital marketing.*

Nel secondo capitolo si sposta l'attenzione sul *Recommendation System*, una tecnica di Machine Learning supervisionata e molto utilizzata in diversi ambiti. Vengono spiegate le sue origini, le sue caratteristiche che aumentano i benefici degli algoritmi a favore di utenti e aziende e gli elementi che lo compongono. Si illustrano le diverse tipologie di valutazione usate per misurare l'efficacia del sistema: *user studies*, valutazione online e offline da parte di esperti. Si distinguono e analizzano i vari livelli di personalizzazione, composti da: raccomandazioni non personalizzate, *semi/segment personalized*, raccomandazioni personalizzate e raccomandazioni non personalizzate + *semi/segment.* Infine, vengono spiegate le varie tecniche di filtraggio (*collaborative filtering*, *content-based filtering* e *hybrid filtering*), ponendo un focus sul caso Netflix.

Il terzo e ultimo capitolo si divide in due sezioni. Nella prima ci si concentra sulle caratteristiche del linguaggio di programmazione *Python*, citando alcune principali librerie per i sistemi di raccomandazione. Successivamente viene introdotto il *dataset MovieLens* esponendo le sue origini, caratteristiche e implicazioni manageriali. Nella seconda sezione si determina la tipologia di studio che si vuole effettuare, evidenziando la metodologia usata e commentando i risultati ottenuti. Infine, alla luce di alcuni limiti appartenenti al dataset usato e allo studio effettuato, si propongono possibili ricerche future, consigliando tecniche e metodologie.

CAPITOLO 1. INTRODUZIONE AL MACHINE LEARNING

Il presente capitolo mira a fornire una panoramica generale del cosiddetto Machine Learning, indagando sulle sue origini e sul motivo della sua importanza, soprattutto per raggiungere obiettivi aziendali. Un particolare focus è posto sugli approcci dell'apprendimento automatico e le loro caratteristiche.

1.1 OVERVIEW

Negli anni '50 inizia a svilupparsi il campo della computazione e nel 1956 si parla per la prima volta di intelligenza artificiale (AI), quando un gruppo di scienziati americani provarono a realizzare una macchina con intelligenza umana. Uno tra questi scienziati, Arthur Samuel definì il Machine Learning come "La scienza che dà ai computer l'abilità di imparare senza essere programmati esplicitamente." Successivamente, nel 1980 venne realizzata "Eliza", uno dei primi "sistemi esperti" che possedeva il sapere di uno psicologo.[2]

L'obiettivo di tali scienziati era quello di scoprire se i computer potessero apprendere dai dati. Su tale pensiero si sviluppa la teoria che i computer possono imparare ad eseguire compiti specifici senza essere programmati, ma grazie all'uso di algoritmi (elenco delle istruzioni programmate utilizzate dai computer per trovare il risultato di un determinato problema) che riconoscono schemi di dati. Ad esempio, individuare informazioni sconosciute senza indicare chiaramente dove cercarle.[3]

Il primo grande nome legato al machine learning è quello di Alan Turing, in quanto pensò di realizzare macchine in grado di apprendere.

Gli investimenti in intelligenza artificiale divennero un ulteriore bolla economica nel 1987, quando iniziarono i primi segni di crisi nelle aziende specializzate in AI. Tuttavia, questo fallimento è stato utile per cambiare approccio e spostare il punto di vista da una ricerca basata su intuizioni ad una ricerca basata su teorie, risultati matematici e una grande quantità di esperimenti. Da quel momento si registrano successi significativi: "dal 1995 in poi si ebbe così una rinascita dell'intelligenza artificiale, rinascita che persiste fino ai giorni nostri. Nel 1996 Garry Kasparov, campione mondiale di scacchi, fu battuto dalla macchina Blue Deep. Nel 2005 un veicolo a guida autonoma vinse la DARPA Grand Challenge. Nel 2007, un veicolo a guida autonoma vinse la DARPA Urban Challenge".[4]

[2]Velocci, S. (2020). Machine Learning: Cos'è e perchè è Importante. start2impact. Retrieved from https://www.start2impact.it/blog/programmazione/cose-machine-learning/.

[3]Machine Learning: che cos'è e perché è importante. Sas.com. Retrieved from https://www.sas.com/it_it/insights/analytics/machine-learning.html.

[4]Stecher, M. (2018). La storia dell'intelligenza artificiale, da Turing ad oggi. CyberLaws. Retrieved from https://www.cyberlaws.it/2018/la-storia-dellintelligenza-artificiale-da-turing-ad-oggi/.

Quindi, si parla di una scienza in continuo progresso che sta trovando il suo pieno sviluppo solo negli ultimi anni. Questi risultati derivano soprattutto dalle nuove tecnologie di elaborazione, dal miglioramento delle capacità di calcolo e dall'uso di algoritmi più complessi e sofisticati. Inoltre, gli utenti hanno contribuito in maniera inconsapevole al miglioramento dell'AI, consentendo l'accesso a tantissimi dati personali.

Lo sviluppo dell'AI e del machine learning deriva proprio da questa grande quantità di dati disponibili, cresciuti considerevolmente in maniera esponenziale insieme alla capacità di elaborazione computazionale, rendendo possibile il verificarsi di complicati calcoli in tempi ragionevoli. Tali dati dell'utente derivano da qualsiasi attività compiuta in rete, da profili social, ricerche, elementi salvati e sono proprio questi la fonte principale dello sviluppo economico.

Di conseguenza, il Machine Learning e l'intelligenza artificiale diventano essenziali per effettuare predizioni che possono rivelarsi molto utili per tutta la comunità mondiale.[5]

Alcuni esempi possono riguardare la guida autonoma, suggerimenti di offerte online (Amazon) o musica (Spotify) o film (Netflix), conoscere il pensiero dei propri clienti attraverso i social network, fino a raggiungere livelli maggiori con intercettazione di una frode o problemi bancari.

Intelligenza artificiale ed apprendimento automatico vengono spesso utilizzati insieme in maniera assoluta, ma sono nettamente differenti. Infatti, il machine Learning è contenuto all'interno della macro-sfera dell'AI (figura1) e ne costituisce la sua maggiore area di interesse.[6]

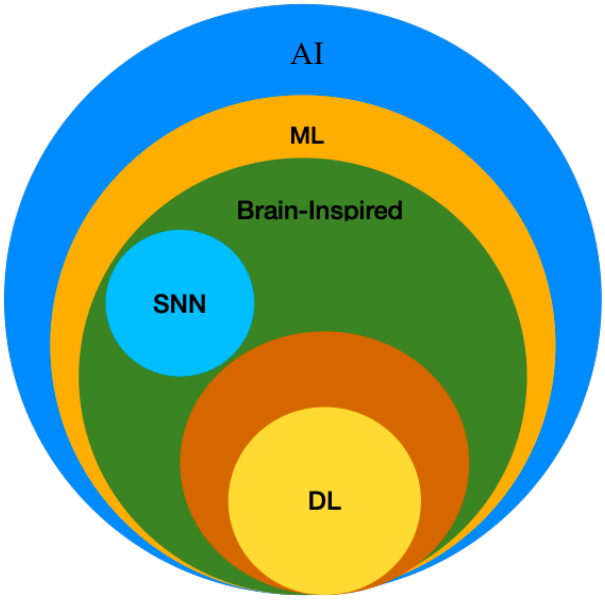


Figura 1. Correlazione tra intelligenza artificiale, machine learning e deep learning[7]

[5]Gosmar, D. (2021). Machine Learning: il sesto chakra dell'intelligenza artificiale (3rd ed.). Independently published.

[6]Clayton, R. (2021). Cos'è il machine learning?. Oracle.com. Retrieved from https://www.oracle.com/it/data-science/machine-learning/what-is-machine-learning/.

[7]Shinde P. P., Shah S., A review of machine learning and deep learning applications. Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) (2018)

L'AI riguarda l'intelligenza delle macchine e il loro modo di simulare funzione cognitive come l'apprendimento e la risoluzione dei problemi. Il Machine Learning, invece, riguarda quell'insieme di algoritmi che automatizzano la costruzione di modelli analitici, consentendo ai sistemi di apprendere e svilupparsi senza programmazione diretta. Consiste in un autoapprendimento basato su algoritmi, di conseguenza il sistema si sviluppa e si addestra sulla base della sua esperienza, di cluster per parallelizzare le operazioni e dello storico di dati. Tale algoritmo possiede anche un apprendimento statistico che lo sviluppa e migliora senza l'aiuto dell'intelligenza umana.[8] Il ML possiede una forte capacità predittiva e di apprendimento, riesce ad utilizzare una grande quantità di dati storici per stabilire una mappa tra input e obiettivi.[9]
Proprio per questo motivo, una delle funzionalità più importanti dell'apprendimento automatico riguarda l'intensità di elaborazione e la ripetitività, in quanto più un modello viene esposto ai dati e più è in grado di riproporsi adattandosi in maniera completamente autonoma, perché affinché l'apprendimento e il processo decisionale venga migliorato è necessario l'addestramento ripetitivo.[10] Per questo motivo, i computer apprendono da elaborazioni precedenti per produrre nuovi risultati e continuare a prendere decisioni del tutto affidabili e replicabili.[11]
Le tecnologie di apprendimento automatico svolgono un ruolo importante, soprattutto quando l'elaborazione di grandi quantità di dati apporta un valore aggiunto significativo al risparmio di tempo e alla massimizzazione delle risorse di elaborazione.[12]

Lo scrittore di fantascienza Arthur C. Clarke disse che "qualsiasi tecnologia sufficientemente avanzata è indistinguibile dalla magia". Da questo scaturiscono numerosi esempi di come la vita delle persone sia cambiata con l'utilizzo della tecnologia, grazie all'elaborazione di set di dati per migliorare l'esperienza del consumatore. La storia precedente e nello specifico la storia del marketing, hanno potuto vantare pochi momenti "magici", ma questo sta cambiando proprio

[8]Vanam, M., Amirali Jiwani, B., Swathi, A., & Madhavi, V. (2021). High performance machine learning and data science based implementation using Weka. Materials Today: Proceedings. https://www.sciencedirect.com/science/article/pii/S2214785321005617.
[9]Wu, P., Chang, X., Yuan, W., Sun, J., Zhang, W., Arcucci, R., & Guo, Y. (2021). Fast data assimilation (FDA): Data assimilation by machine learning for faster optimize model state. Journal Of Computational Science, 51. Retrieved from https://www.sciencedirect.com/science/article/pii/S187775032100020X.
[10]Ashok Mahant, M., & Pellakuri, V. (2021). Innovative supervised machine learning techniques for classification of data. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2020.11.092
[11]Machine Learning: che cos'è e perché è importante. Sas.com. Retrieved from https://www.sas.com/it_it/insights/analytics/machine-learning.html.
[12]Xu, Y., Zhou, Y., Sekula, P., & Ding, L. (2021). Machine learning in construction: From shallow to deep learning. Developments In The Built Environment, 6. https://www.sciencedirect.com/science/article/pii/S2666165921000041.

grazie all'utilizzo dell'intelligenza artificiale e dell'apprendimento automatico. Ogni interazione del cliente viene modellata sulla base del contesto e dell'intelligenza. Questa è quella che Rob Tarkoff (Oracle Data Cloud, Fusion CX Development) definisce come "economia dell'esperienza".[13]

1.2 CARATTERISTICHE

L'apprendimento, in generale, è un processo iterativo, che si accresce nel corso della vita e migliora le conoscenze sulla base delle informazioni acquisite. La stessa cosa vale per le macchine, infatti l'obiettivo principale del Machine Learning è quello di estrarre delle feature (valori che utilizzo come input per l'apprendimento) il più possibile distintive e rappresentative all'interno dei dati relativi ad un fenomeno che desideriamo analizzare attraverso modelli (o algoritmi di varia natura) per apprenderne i meccanismi (ovvero i datapoint che consistono in dati che permettono al computer di imparare) ed effettuare delle predizioni (label) massimizzandone l'accuratezza, ricavandone modelli di apprendimento.[14]
Con l'utilizzo di questi modelli è possibile ricavare algoritmi di apprendimento, utili per risolvere un problema specifico, in quanto indicano alla macchina le operazioni che si possono effettuare. Sono proprio gli algoritmi i "motori" che alimentano il ML.

L'esecuzione di una attività di machine learning avviene sulla base di cinque passaggi principali: raccolta dati, preparazione dei dati, addestramento del modello, valutazione del modello e miglioramento delle prestazioni. Durante la raccolta vengono raggruppati tutti i dati con migliore varietà, densità e volume, che possano fornire la base per l'apprendimento futuro. I dati di qualità vengono poi preparati in un lungo processo analitico, per trovare la coerenza dei dati e intraprendere azioni utili per affrontare diverse problematiche possibili, come la mancanza di dati e l'intervallo. Successivamente viene scelto l'algoritmo appropriato e la sua rappresentazione come modello. La parte successiva dei dati viene utilizzata per controllare l'accuratezza del modello attraverso le tracce degli output dei dati, utile per convalidare la precisione nella selezione dell'algoritmo. A questo punto vengono introdotte più variabili per verificarne l'efficienza e selezionare un ulteriore modello.[15]

---

[13]Tarkoff, R. How AI/ML are Driving Customer Experience and the Experience Economy - The Six Five Summit. The Six Five Summit. Retrieved from https://thesixfivesummit.com/session/how-ai-ml-are-driving-customer-experience-and-the-experience-economy/.
[14]Gosmar, D. (2021). Machine Learning: il sesto chakra dell'intelligenza artificiale (3rd ed.). Independently published.
[15]Ashok Mahant, M., & Pellakuri, V. (2021). Innovative supervised machine learning techniques for classification of data. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2020.11.092

A seconda del tipo di algoritmo utilizzato in funzione della metodologia per apprendere i dati e fare previsioni, si possono classifica differenti sistemi di apprendimento automatico in tre macro categorie: supervisionato, non supervisionato e per rinforzo (Figura 2)[16]. Tali modelli vengono utilizzati in maniera differente per garantire la massima performance e il migliore risultato possibile per la risposta agli stimoli esterni.

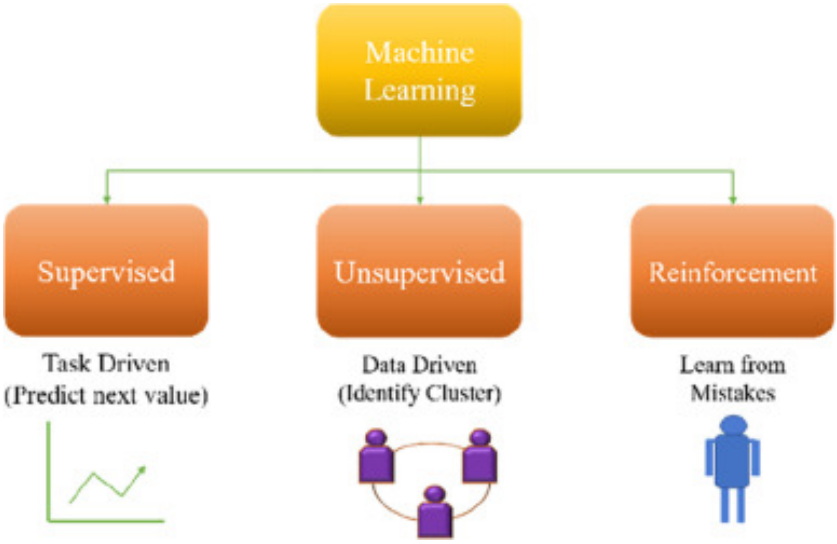


Figura 2. Approcci dell'apprendimento automatico

1.2.1 APPRENDIMENTO SUPERVISIONATO

L'apprendimento supervisionato è la metodologia più utilizzata nel machine learning e fornisce al sistema informatico una serie di nozioni specifiche e codificate (come modelli ed esempi), che permettono di costruire un database di informazioni e di esperienze, da attingere ad essi nel proprio sistema nei casi in cui la macchina deve affrontare un problema.[17]

Quindi questo approccio utilizza inizialmente la guida di un data scientist che insegna all'algoritmo i risultati da genere e successivamente cerca di imparare da esempi passati. L'algoritmo viene addestrato e testato da un set di dati che viene già etichettato e possiede un output predefinito.[18] Infatti, si ottiene l'apprendimento supervisionato quando l'algoritmo utilizza variabili di input (X) e prevede o classifica dal database di addestramento variabili di output (Y), per apprendere ed estrarre la funzione di mappatura che dall'input genera l'output: **Y=f(X)**

[16]Sharma, N., Sharma, R., & Jindal, N. (2021). Machine Learning and Deep Learning Applications-A Vision. Global Transitions Proceedings, 2, 24-28. Retrieved from https://www.sciencedirect.com/science/article/pii/S2666285X21000042#fig0001.

[17]Machine learning (apprendimento automatico) - Cos'è e come funziona?. Intelligenza Artificiale. Retrieved from https://www.intelligenzaartificiale.it/machine-learning/.

[18]Clayton, R. (2021). Cos'è il machine learning?. Oracle.com. Retrieved from https://www.oracle.com/it/data-science/machine-learning/what-is-machine-learning/.

L'obiettivo è quello di approssimare tale funzione in modo da riuscire a prevedere il valore dell'output per quel dato, anche quando si ha un nuovo dato di input. Più grande è il set di dati, maggiore sarà l'apprendimento che può raggiungere la macchina.[19]

Si possono sottoporre alla macchina alcune immagini, che identifica dei pattern, similitudini o correlazioni tra di esse (Figura3). Durante la fase zero avviene la preparazione dei dati come precedentemente spiegato, mentre la prima fase di ogni metodologia è il processo di Data Training, costituito da dati conosciuti che prendono il nome di "Labeled data" e formano un dataset di "Test e Validation". Tali dati vengono utilizzati con risultati di output noti, durante la fase di Training, per costruire la funzione di apprendimento. L'adattamento degli algoritmi utilizzati al dataset di training costituisce il processo di fitting del modello. In questa fase viene adattato il modello di machine learning.
La seconda fase si compone di un processo di Test, dove si utilizza un dataset con elemento da predire noto. Questo test è utile per valutare quanto il modello sia accurato e corretto. In questa fase, il modello di fitting viene applicato per ricavare dei risultati, che saranno confrontati con il dato noto per verificarne il corretto funzionamento del modello.
Uno degli obiettivi nella fase di training e validazione consiste nell'effettuare un "tuning" del modello per minimizzare l'errore medio, agendo sugli Hyperparameter dell'algoritmo. Viene denominata "Cross Validation" una delle tecniche utilizzate in questo modello e consiste nel separare i dati di training in due sottoinsiemi per iterare un processo che trasla sottoinsiemi di training e validation in modo da mescolare le percentuali del dataset di validazione e di training, per ottenere differenti possibilità. Il rischio è quello di produrre un modello incoerente, ottenendo dati di validation completamente differenti da quelli usati per il training. In altre parole, è un metodo statistico utilizzato per stimare l'abilità dei modelli di apprendimento automatico.
Dopo aver concluso tutte queste fasi, si passa alle fasi di predizione e inferenza, dove è possibile utilizzare il modello costruito per predire risultati "non noti" con l'utilizzo di nuovi dati input.
Nell'apprendimento supervisionato gli algoritmi indirizzano normalmente due diverse problematiche: la regressione e la classificazione. La differenza tra i due sta nella tipologia di dati usati. Per la regressione il dato si può predire attraverso un'analisi dei dati attuali e tale variabile può assumere valori numerici continui potenzialmente infiniti. Si tratta di input e output continui che variano nel tempo e si studia la relazione tra due o più variabili indipendenti l'uno dall'altra. Mentre, per il problema di classificazione si vuole predire l'appartenenza ad

[19]Govoni, L. Machine Learning e principio di funzionamento | Lorenzo Govoni. Lorenzo Govoni Business e Tecnologia. Retrieved from https://www.lorenzogovoni.com/machine-learning-e-funzionamento/

una classe o categoria, dove input e output sono discreti. È il processo che permette alla macchina di riconoscere e categorizzare oggetti e dimensioni da un set di dati e l'output generalmente consiste in una categoria o un gruppo (esempio: "assente" o "non assente" nella classificazione).

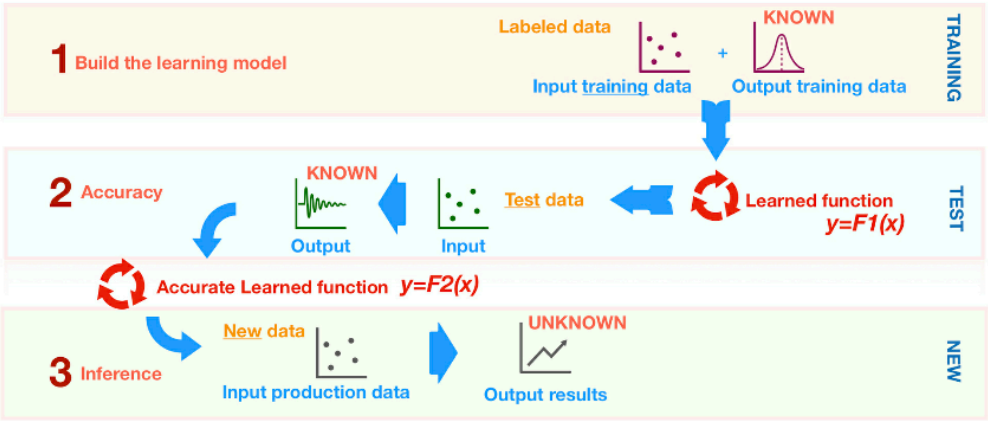


Figura 3. Apprendimento supervisionato[20]

1.2.2 APPRENDIMENTO NON SUPERVISIONATO

L'apprendimento non supervisionato utilizza un approccio più indipendente ed è molto utile nelle situazioni in cui non si hanno a disposizione Labeled data, ovvero dati storici con risultati noti. Prevede che le informazioni inserite nella macchina non siano codificate, ma si possa avere la possibilità di attingere a determinate informazioni senza avere alcun esempio di utilizzo e conoscenza dei risultati attesi. Si offre maggiore libertà alla macchina che organizzerà le informazioni in maniera intelligente e memorizzando i risultati migliori per differenti situazioni presentabili.[21] In questo caso si possiede una variabile di input (i dati), ma nessuna variabile di output corrispondente. L'obiettivo principale è quello di analizzare i dati a disposizione per ricavare informazioni utili e crearne schemi o relazioni, analizzando i dati senza utilizzare una categorizzazione, ma eseguendo predizioni attraverso metodi induttivi. L'apprendimento non supervisionato viene utilizzato anche per la preelaborazione dei dati come l'estrazione di feature, la selezione di feature e il ricampionamento.[22]

[20]Gosmar, D. (2021). Machine Learning: il sesto chakra dell'intelligenza artificiale (3rd ed.). Independently published.
[21]Machine learning (apprendimento automatico) - Cos'è e come funziona?. Intelligenza Artificiale. Retrieved from https://www.intelligenzaartificiale.it/machine-learning/.
[22]Jafari-Marandi, R. (2021). Supervised or unsupervised learning? Investigating the role of pattern recognition assumptions in the success of binary predictive prescriptions. Neurocomputing, 434, 165-193. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925231220319639#b0260.

I ricercatori hanno dimostrato che spesso le prestazioni degli algoritmi di classificazione sui cluster sono migliori delle prestazioni degli stessi algoritmi sull'intero dato.[23]

Esistono due problemi di apprendimento non supervisionato e si possono dividere in: raggruppamento e associazione. Il raggruppamento o "Clustering" (Figura 4) consiste in varie tecniche di analisi statistiche multivariate capaci di partire da una grande quantità di dati per creare gruppi di elementi simili e correlati. Gli algoritmi elaborano i dati a disposizione per individuare logiche e correlazioni, al fine di suddividere i dati.[24] In questo modo si estrae una regola utile per raggruppare i casi secondo caratteristiche intrinseche ricavabili dai dati stessi. L'associazione, invece, si pone l'obiettivo di trovare schemi frequenti, associazioni, correlazioni o strutture casuali tra un insieme di item od oggetti in un database relazionale, che descrivano grandi porzioni di dati (esempio: persone che acquistano un prodotto A tendono ad acquistare anche un prodotto B). Questo è utile per predire l'evento di un item in base agli eventi di altri item nella transazione.[25]

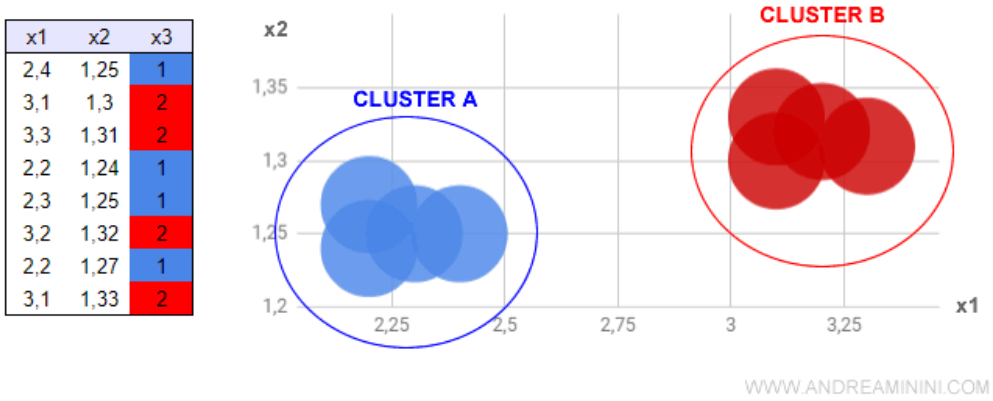


Figura 4. Esempio di Clustering[26]

Gli algoritmi di clustering possono essere approssimativamente suddivisi in quattro categorie: metodi gerarchici, partizionali, basati sulla densità e basati su griglia. I metodi gerarchici mirano a unire o dividere gli oggetti di dati a ogni livello fino a quando non vengono soddisfatti i criteri di interruzione e si pone come obiettivo la creazione di una gerarchia di cluster multilivello.

[23]Liu, F., Du, P., Weng, F., Qu, J., 2007. Use clustering to improve neural network in financial time series prediction, Third International Conference on Natural Computation (ICNC 2007). IEEE, pp. 89-93.
[24]Gosmar, D. (2021). Machine Learning: il sesto chakra dell'intelligenza artificiale (3rd ed.). Independently published.
[25]Govoni, L. Machine Learning e principio di funzionamento | Lorenzo Govoni. Lorenzo Govoni Business e Tecnologia. Retrieved from https://www.lorenzogovoni.com/machine-learning-e-funzionamento/
[26]Minini, A. Apprendimento senza supervisione. Andreaminini.com. Retrieved from http://www.andreaminini.com/ai/machine-learning/apprendimento-senza-supervisione.

Partendo da questa definizione, vengono proposti alcuni metodi essenziali, ad esempio il clustering agglomerativo e divisivo. [27] Nel primo caso, ogni dato è un cluster e ogni osservazione inizia in un cluster e attraverso analisi di vari algoritmi, si giunge alla creazione di gruppi che vengono uniti man mano che si sale in gerarchia. La seconda tipologia lavora in senso contrario, dove le osservazioni iniziano sempre in ogni cluster, ma non si operano in raggruppamenti bensì in suddivisioni, spostandosi verso il basso nella gerarchia.[28]

Uno degli algoritmi di clustering più conosciuto è il K-mean, che consiste in un'ulteriore metodologia utilizzata da tempo per la comprensione del segnale, per minimizzare l'errore e la perdita di informazioni. Ci si riferisce alla dimensione dell'esclusività, dove si raggruppano i dati in K cluster e le osservazioni (dataset) e ordinarli in modo che un singolo dato possa appartenere ad un solo cluster.

Esistono anche clustering non esclusivi che consentono ad un elemento di far parte di più cluster, soprattutto nei contesti in cui le differenze tra dati non sono molto evidenti. Infine, il Fuzzy clustering consiste in un metodo che consente ad ogni elemento di appartenere a tutti i cluster. Il Clustering in generale è una tecnica di descriptive analtyics che meglio aiuta a comprendere i propri dati per compiere data driven decisions. Molto utile per conoscere meglio la propria audience e differenziare i buyer personas, in modo da realizzare strategie di digital marketing fondamentali per ogni azienda.[29]

Un'altra dimensione riguarda quella della completezza, dove nel "complete clustering" ogni elemento appartiene ad almeno un cluster e nel "partial clustering" è possibile che alcuni elementi non siano raggruppati.

1.2.3 APPRENDIMENTO CON RINFORZO

L'apprendimento con rinforzo è uno dei settori più innovativi e anche il sistema più complesso del machine learning. Rappresenta la versione computerizzata dell'apprendimento umano per tentativi ed errori e prevede che la macchina possieda strumenti che migliorino il proprio apprendimento e comprendano le caratteristiche dell'ambiente circostante. Infatti, tale apprendimento si basa su due condizioni: esito ritardato e ricerca per tentativi ed errori.

Tale modello utilizza distinte tipologie di logiche. La prima riguarda le predizioni, quindi ricevere dati e paramenti in input per fornire output adeguati.

---

[27]Hu, W., Chen, C., Ye, F., Zheng, Z., & Du, Y. (2021). Learning deep discriminative representations with pseudo supervision for image clustering. Information Sciences, 568, 199-215. https://doi.org/10.1016/j.ins.2021.03.066

[28]Gosmar, D. (2021). Machine Learning: il sesto chakra dell'intelligenza artificiale (3rd ed.). Independently published.

[29]Provino, A. (2020). Clustering: tutto quello che devi sapere | Edizione 2021. Machine Learning & Data Science Blog. Retrieved from https://andreaprovino.it/clustering/.

La seconda riguarda l'algoritmo usato che apprende e si adatta ai cambiamenti ambientali attraverso un sistema di valutazione, agendo sulla base di ricompense (valore reale positivo) se l'azione compiuta è corretta e l'obiettivo viene raggiunto o è vicino; oppure penalità (valore reale negativo) se ciò non accade. Il fine ultimo è quello di massimizzare la ricompensa ricevuta.[30] Sulla base di tali feedback (ricompense o punizioni) ricevuti si stabilisce la terza logica, dove si andrà riprogrammare il modello per migliorarlo. La raccolta di questi feedback arricchisce l'esperienza della macchina e accresce la sua Knowledge Base (Figura 5). L'idea di base è che l'algoritmo modifichi i suoi comportamenti per ricevere più premi e ridurre al minimo le punizioni.

La logica di questi algoritmi di rinforzo è analoga, da un punto di vista matematico, ai processi decisionali di Markov, dove l'obiettivo consiste nell'individuare l'azione da effettuare in un determinato stato per massimizzare la ricompensa.

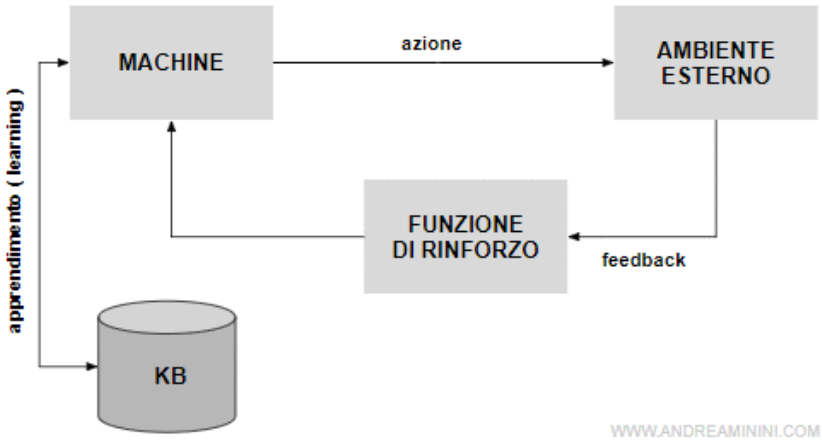


Figura 5. Schema funzione di rinforzo[31]

1.3 UTILIZZO DEL MACHINE LEARNING PER OBIETTIVI AZIENDALI

L'intelligenza artificiale oggi è sempre più utilizzata e sviluppata con un ampio spettro di applicazioni per sperimentare nuovi processi all'interno delle articolazioni di business e portare valore, soprattutto in relazione alle strategie di marketing. Un nuovo studio di Mckinsey "The state of Al" sottolinea un maggiore ottimismo rispetto al passato e scrive: «Le organizzazioni utilizzano l'IA come strumento per generare valore. Sempre più spesso, questo valore si presenta sotto forma di entrate. Un piccolo gruppo di intervistati provenienti da diversi settori industriali assegna all'IA il 20% o più dei guadagni delle loro organizzazioni al lordo di interessi

[30]Govoni, L. Machine Learning e principio di funzionamento | Lorenzo Govoni. Lorenzo Govoni Business e Tecnologia. Retrieved from https://www.lorenzogovoni.com/machine-learning-e-funzionamento/

[31]Minini, A. L'apprendimento con rinforzo. Andreaminini.com. Retrieved from http://www.andreaminini.com/ai/machine-learning/apprendimento-con-rinforzo.

e tasse (EBIT). Queste aziende prevedono di investire ancora di più nell'IA in risposta alla pandemia Covid-19». Questo strumento non è utilizzato solo per ottimizzare i tempi o analizzare il mercato, ma le sue potenzialità sono più ampie, infatti sempre più aziende fanno un uso meno conservativo di algoritmi predittivi e «la maggior parte degli intervistati presso aziende top performer ha aumentato gli investimenti in ciascuna delle principali funzioni aziendali in risposta alla pandemia, mentre meno del 30% degli altri intervistati afferma altrettanto».[32]

Il machine learning sta rivoluzionando il marketing, rendendolo più accurato e in grado di auto apprendere informazioni in tempo reale, lavorando su un grande quantità di dati in maniera evoluta. Questo è fondamentale per l'estrapolazione delle informazioni di business, utili a migliorare offerte di prodotti e servizi e ottimizzare i processi di vendita, semplificando il lavoro delle risorse umane. Grazie ai big data proveniente da più fonti, è possibile profilare una customer base automatizzata, identificando contenuti più adatti per ogni individuo. Con l'utilizzo del machine learning è possibile anche individuare situazioni di insoddisfazione, attraverso la sentiment analysis e individuare modelli di comportamento predefiniti, per segnare la possibilità che aumenti il tasso di abbandono e intervenire tempestivamente sul churn rate.[33] Forbes stima che nel 2018, il 75% delle aziende che ha introdotto l'intelligenza artificiale e il machine learning, ha incrementato la sua customer satisfaction di oltre il 10%.

Con maggiore riferimento all'ambito del digital marketing, le tecniche di apprendimento automatico permettono di trasformare in valore monetario tutte le informazioni provenienti dai contatti e dalle comunicazioni multicanale. Grazie a queste tecniche, si possono estrarre intuizione ed evidenze utili dai big data provenienti da più fonti per profilare il cliente e conoscerlo in profondità, così da costruire una customer journey mirata, attraverso la personalizzazione di contenuti che permettono di aumentare l'engagement di clienti acquisiti e prospect, la fidelizzazione, strumenti di marketing, promozioni mirate e il tasso di risposta alla call-to-action, cercando di ottimizzare tutte le campagne promozionali e politiche di pricing.[34]

L'uso dell'apprendimento automatico trova molte applicazioni a livello quotidiano, si intende in quelle situazioni in cui si utilizza la tecnologia (esempio: riconoscimento vocale). Alcuni settori che hanno incrementato il proprio business utilizzando tali tecnologie possono essere:

- Servizi finanziari: per identificare e prevenire le frodi finanziarie.

[32]Tre., L. (2021). Con l'intelligenza artificiale i profitti aziendali possono crescere anche del 20%. Ilsole24ore.com. Retrieved from https://www.ilsole24ore.com/art/con-l-intelligenza-artificiale-profitti-aziendali-possono-crescere-anche-20percento-ADetU0GB.

[33]Machine Learning applicato al Marketing - CRM Team. CRM Team. (2021). Retrieved from https://www.crmteam.it/machine-learning-applicato-al-marketing/.

[34]Leonardi, A. (2019). Il valore del Machine Learning nel Marketing. AI4Business. Retrieved https://www.ai4business.it/intelligenza-artificiale/machine-learning/machine-learning-nel-marketing/#Come_sfruttare_le_potenzialita_del_machine_learning_nel_Digital_Marketing

- Marketing e vendite: per analizzare e fornire consigli personalizzati sui prodotti, basandosi sullo storico degli acquisti dei clienti e prevedendo la prossima "aggiunta al carrello". Questa è una grande opportunità per il futuro delle vendite e del marketing, perché si può sfruttare questa capacità di raccogliere, analizzare e utilizzare i dati per offrire migliori esperienze di acquisto.
- Pubblica amministrazione: per ridurre fonti di dati multiple, in modo da identificare modelli e informazioni utili all'interno delle agenzie governative per la riservatezza dei dati. con l'apprendimento automatico è possibile trovare soluzioni e misure preventive per mantenere la sicurezza.
- Settore sanitario: sensori e dispositivi vengono utilizzati per accedere ai dati di un paziente in tempo reale. Avere tempestivamente parametri e informazioni importanti dei pazienti per delinearne la sua storia clinica e semplificare al singolo medico l'analisi delle condizioni di salute, per prevenire eventuali errori futuri.
- Servizio dei trasporti: per prevedere ostacoli futuri su alcune strade. A seconda dell'esperienza di viaggio e del layout, vengono consigliati ai clienti percorsi differenti. Anche nell'ambito delle consegne vengono sfruttate le tecnologie di apprendimento per migliorare l'organizzazione del lavoro, tenere traccia dei prodotti e offrire un servizio migliore al cliente finale.[35]

Per le organizzazioni aziendali, creare un modello che calcoli il valore del ciclo di vita di un cliente diventa fondamentale. Si utilizzano algoritmi di machine learning per identificare, conoscere e fidelizzare i clienti più importanti e prevedere le entrate future che si possono ottenere con un singolo cliente in un determinato periodo. In questo modo è possibile concentrarsi sulle strategie di marketing più adeguate ed incoraggiare i clienti ad alto valore ad interagire sempre più con il brand e attrarre nuovi clienti. Questo ultimo passaggio non è semplice, richiede tempo e risorse economiche ed umane più elevate rispetto al mantenimento dei clienti esistenti e della loro soddisfazione e fedeltà.[36]

In un'era sempre più connessa e "comoda", i clienti possono trovare online tutto ciò che desiderano con un'ampia possibilità di scelta, confrontando caratteristiche e prezzi in tempo reale. Questa determinazione dei prezzi dinamici (o determinazione dei prezzi basati sulla domanda) è importante anche per le aziende, le quali hanno la possibilità di rimanere aggiornate sulla tipologia di domanda e le dinamiche di mercato. Consente loro di assegnare prezzi flessibili sulla base dell'evoluzione del mercato e quindi del livello di interesse del cliente

[35]Ashok Mahant, M., & Pellakuri, V. (2021). Innovative supervised machine learning techniques for classification of data. Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2020.11.092

[36]Clayton, R. (2021). Cos'è il machine learning?. Oracle.com. Retrieved from https://www.oracle.com/it/data-science/machine-learning/what-is-machine-learning/.

target. Per giungere a questo livello di agilità aziendale, è richiesta un strategie di machine learning affidabile, che contenga una grande quantità di dati utili per comprendere meglio l'evoluzione del cliente, della sua disponibilità a pagare in diverse situazioni.

Lo studioso John Caldwell disse che *"per fornire il messaggio giusto alla persona giusta al momento giusto, prima devi raccogliere i dati giusti nel database giusto al momento giusto"*. Poco tempo fa, i marketer si basavano sul loro intuito per eseguire le segmentazioni dei clienti affinché si creassero campagne mirate. Oggi, grazie al grande sostegno del machine learning si utilizzano gli algoritmi di clustering e classificazione per suddividere i vari gruppi i clienti, basandosi su caratteristiche demografiche, comportamentali e di affinità. L'aumento dei dati disponibili fa diventare gli algoritmi più sofisticati e accurati, in grado di garantire ai clienti esperienze personalizzate e alle aziende un segmento di clienti ideali.[37]

Nello specifico ambito del digital marketing è possibile comprendere alcune applicazioni concrete del machine learning riconducibili a:

- Marketing Automation: per programmare l'invio di messaggi e newsletter a clienti specifici, tenendo traccia delle loro preferenze per affinare in maniera dinamica la segmentazione degli utenti destinatari.
- Social Media Marketing: per programmare automaticamente post e campagne, intercettazione e profilazione di contatti in target, mostrando loro contenuti di interesse e aumentando le possibilità di interazioni.
- Pagine web: per tracciare comportamenti e interessi durante la navigazione degli utenti, utilizzando gli algoritmi per ottimizzare i contenuti delle successive navigazioni.[38]

A questo punto è possibile affermare che l'utilizzo del machine learning offre alle aziende un vantaggio competitivo, in quanto è capace di automatizzare e velocizzare i processi decisionali, ma anche accelerare il time to value, aumentando la visibilità aziendale e la collaborazione.

[37]Ibidem
[38]Machine Learning applicato al Marketing - CRM Team. CRM Team. (2021). Retrieved from https://www.crmteam.it/machine-learning-applicato-al-marketing/.

CAPITOLO 2. RECOMMENDATION SYSTEM

Il presente capitolo fornisce una spiegazione sui sistemi di raccomandazione, le loro caratteristiche e l'influenza che hanno sugli utenti e le loro esperienze di acquisto. Si indaga sulla loro importanza e sulle tecniche principali utilizzate. Un particolare focus è posto sul Caso Netflix e su come tale azienda sia riuscita a creare il suo vantaggio competitivo grazie all'utilizzo di questi sistemi.

2.1 OVERVIEW

La piattaforma Cisco nel suo Annual Internet Report valuta la trasformazione digitale in vari segmenti di business ed ha previsto che, quasi due terzi della popolazione mondiale avrà accesso a Internet entro il 2023. Ci saranno 5,3 miliardi di utenti Internet totali (66% della popolazione mondiale) entro il 2023, rispetto ai 3,9 miliardi (51% della popolazione globale) nel 2018. Inoltre, Il segmento consumer avrà quasi i tre quarti della quota dei dispositivi e delle connessioni totali entro il 2023. A livello globale, la quota del segmento consumer dei dispositivi e delle connessioni totali sarà del 74%, mentre il segmento business rivendicherà il restante 26%.[39] A questa previsione di crescita si aggiunge il nuovo report Global Statshot Digital 2020 di ottobre (Figura 6), prodotto in collaborazione con Hootsuite e We Are Social, che mostra come la tecnologia connessa continua a svolgere un ruolo sempre più importante in vari aspetti della vita quotidiana delle persone.

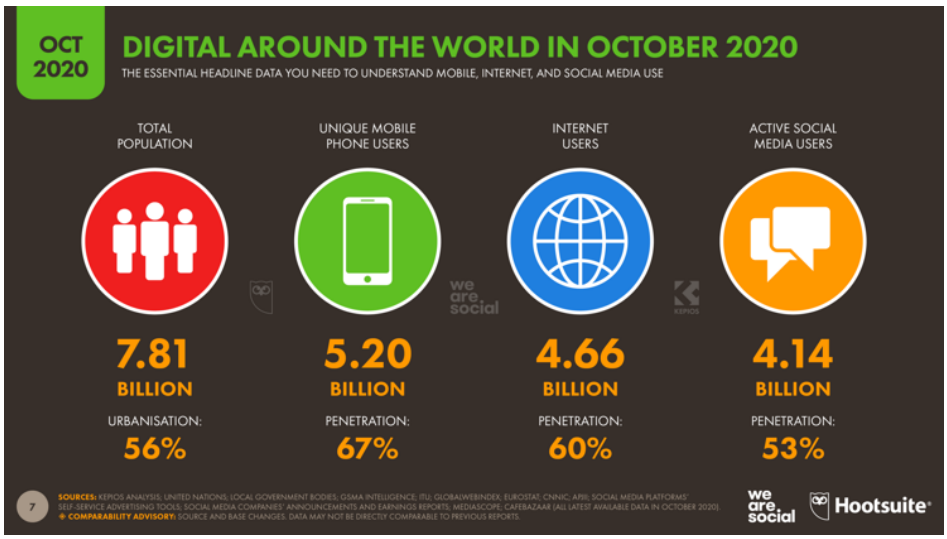


Figura 6. Global Digital Overview ottobre 2020

[39]Report, C. (2020). Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper. Cisco. Retrieved from https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

Gran parte della crescita di questo trimestre nell'uso dei social media è il risultato diretto delle nuove abitudini che le persone hanno adottato durante i blocchi di COVID-19 (come social media, piattaforme streaming, e-commerce).[40]

Ogni bene e servizio diviene a portata di clic per qualsiasi utenti ed ogni sua ricerca, transazione o condivisione può essere analizzata e tracciata. Il singolo utente viene quotidianamente a contatto con input diversi e numerosi, esposto a sovrabbondanti informazioni, con la conseguenza di riscontrare difficoltà nelle sue scelte e non soddisfare pienamente le sue esigenze o bisogni. H.A. Simon afferma che: "quello che l'informazione consuma è piuttosto ovvio: consuma l'attenzione dei suoi destinatari. Dunque, un'abbondanza di informazione crea povertà d'attenzione, e il bisogno di scegliere come distribuire in maniera efficiente questa attenzione tra la sovrabbondanza di informazioni che potrebbero consumarla".

In questo scenario attuale, i cosiddetti sistemi di raccomandazione trovano il loro campo fertile per svilupparsi e migliorarsi, già cresciuti negli anni grazie alle tecniche di intelligenza artificiale, come machine learning e data mining. Un recommender system, dunque, è un programma di filtraggio dei contenuti, sulla base di preferenze, interessi o atteggiamenti osservati e registrati, che crea delle raccomandazioni personalizzate per l'utente per supportare il suo processo decisionale e consigliare prodotti, informazioni e servizi adeguati, facendogli ottenere solo le risorse più pertinenti alle sue esigenze. Gli effetti conseguenti a tale sistema non diventano vantaggiosi solo per l'utente o il cliente di un prodotto/servizio al quale viene facilitata la scelta, ma anche per l'azienda che offre quel determinato servizio o prodotto.

I sistemi di raccomandazione utilizzano un'analisi di un tipo specifico di dati per prevedere le valutazioni degli utenti per i singoli articoli. Successivamente, (sulla base di questa analisi), creano raccomandazioni e modificano il contenuto della pagina visualizzata in modo che corrisponda il più possibile alle preferenze dell'utente. Questo è uno dei motivi per cui una vasta gamma di aziende e applicazioni web ha recentemente implementato sistemi di analisi del comportamento degli utenti rispetto alla raccomandazione dei prodotti, servizi o informazioni più adatti. L'obiettivo è, ovviamente, aumentare le vendite e i profitti di queste società.[41]

Utilizzando le tecniche di raccomandazione e ponendo il beneficio dal punto di vista dell'utente, si può dedurre la seguente definizione formale: Dati C l'insieme degli m ∈ N, ovvero tutti gli utenti per i quali il sistema provvederà ad una raccomandazione, I l'insieme degli n ∈ N, ovvero

[40]Kemp, S. (2020). *DIGITAL 2020: OCTOBER GLOBAL STATSHOT*. data reportal. Retrieved from https://datareportal.com/reports/digital-2020-october-global-statshot.

[41]Walek, B., & Fojtik, V. (2020). A hybrid recommender system for recommending relevant movies using an expert system. *Expert Systems With Applications*, (158). Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417420302761#sec0011.

tutti i possibili prodotti (item), che possono essere raccomandati ed i valori m e n possono corrispondere alle centinaia di migliaia di unità. Si definisce u la funzione di utilità:

$$u: Cx\mathrm{I} \Rightarrow R$$

dove R è l'insieme totale raccomandato e ordinato. Tale funzione misura quanto un oggetto I sia effettivamente utile ad un utente C.

Per esaminare un'ampia quantità di dati e riuscire a stabilire qual è la corretta probabilità con la quale un cliente acquisterà un articolo o usufruirà di un servizio e di conseguenza fornire suggerimenti personalizzati ad esso, è necessario che i sistemi di raccomandazione utilizzino gli algoritmi di machine learning.

L'interesse in questo settore aumenta con la crescita delle richieste degli utenti. Di conseguenza, per fornire raccomandazione ottimizzate è necessario costruire sistemi di raccomandazione di alta qualità, utili per migliorare anche l'ampia gamma di applicazioni anche nella vita quotidiana.

Un recommendation system è costituito da un insieme di elementi essenziali:

- *Items*, gli oggetti che devono essere raccomandati e che creano l'intero sistema;
- *Users*, gli utenti del sistema ai quali devono essere inviate le raccomandazioni, ma anche coloro che alimentano il dataset con le loro caratteristiche;
- *Transactions*, l'insieme delle interazioni avvenute tra gli utenti e il sistema stesso, utile per prevedere le preferenze dell'individuo rispetto agli items;
- *Rating*, l'insieme delle valutazioni fornite dagli utenti.

La figura 7 mostra un modello generale di un tipico processo di raccomandazione, evidenziando il ruolo fondamentale degli utenti e degli elementi nel processo di raccomandazione. [42]

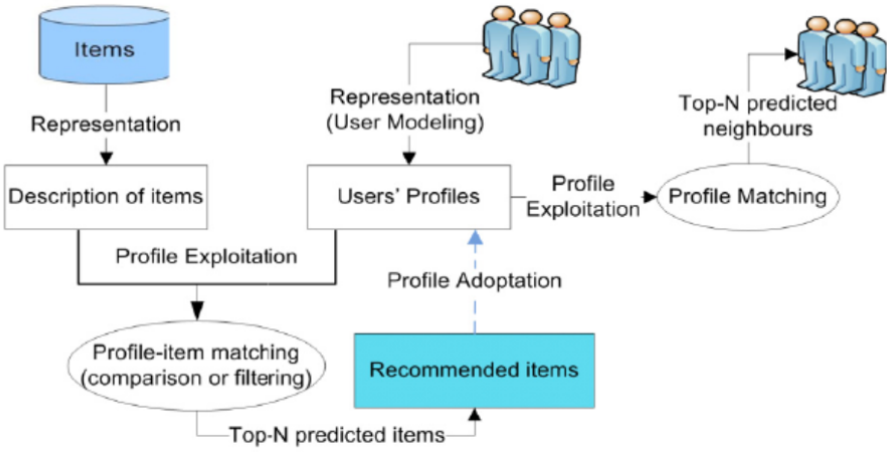


Figura 7. Modello generale del processo di raccomandazione

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

L'utilizzo dei motori di raccomandazioni è utile per tutte le aziende che vogliono attuare strategie di cross-selling e up-selling ai consumatori, al fine di favorire una migliore esperienza. È principalmente l'area delle piattaforme di streaming e dei noleggi di film online (generalmente tutti i servizi che distribuiscono opere audiovisive) in cui la raccomandazione di prodotti pertinenti a un utente gioca un ruolo significativo. L'obiettivo principale di tutti questi servizi è che ogni visualizzazione di un prodotto da parte di un utente si traduca in conversioni, ovvero azioni dell'utente quali acquisto dell'accesso per guardare un film, registrazione dell'abbonamento, ecc. Inoltre, questi servizi cercano anche di consigliare la migliore esperienza possibile per l'acquirente: i clienti soddisfatti con i prodotti visualizzati corrispondenti ai loro gusti potrebbero potenzialmente tornare e acquistare qualcosa di nuovo. I giocatori più importanti tra i pionieri dei sistemi di raccomandazione sono Amazon e Netflix.[43] Ad esempio, Amazon afferma che "il suo sistema consente di aumentare le vendite dal 20 al 30 percento all'anno".[44]

È quindi evidente che questi sistemi giocano un ruolo essenziale nel mondo della vendita online; e il loro potenziale (con il crescente numero di persone che acquistano online) è ancora in aumento. Un'esperienza di acquisto più personalizzata e coinvolgente può ridurre il tasso di abbandono.

2.1.1 CARATTERISTICHE

È necessario che il sistema di raccomandazione possieda delle caratteristiche specifiche affinché sia efficace per gli utenti (Aggarwal, 2016)[45]:

- *Rilevanza*: gli item raccomandati devono essere rilevanti, ovvero devono contenere elementi che corrispondano effettivamente con gli interessi degli utenti. Da questo si deduce l'accuratezza percepita, ovvero una stima del grado di "correttezza" da parte degli utenti nel ricevere raccomandazioni inclini ai loro gusti.
- *Diversità*: è una caratteristica importante quando si possiede una lista di elementi, in quanto gli item consigliati non devono essere simili tra loro per dare la possibilità all'utente di aumentare le probabilità di trovare almeno un elemento coerente con le sue preferenze. È consigliabile che gli elementi debbano anche variare nel tempo, per adattarsi alle diverse situazioni e momenti della vita dell'utente.

[43]Walek, B., & Fojtik, V. (2020). A hybrid recommender system for recommending relevant movies using an expert system. *Expert Systems With Applications*, (158). Retrieved from https://www.sciencedirect.com/science/article/pii/S0957417420302761#sec0011.

[44]Clayton, R. (2021). Cos'è il machine learning?. Oracle.com. Retrieved from https://www.oracle.com/it/data-science/machine-learning/what-is-machine-learning/.

[45]Tesi Luiss Guido Carli. (2019). Recommendation System: Implementazione Di Algoritmi Collaborativi Per Servizi Video-Streaming On Demand (VOD). Retrieved from http://tesi.luiss.it/25923/1/692341_VITALE_FABRIZIO.pdf

- *Novità*: elementi sempre nuovi ed originali devono essere al centro delle raccomandazioni effettuate dal sistema. Si devono evitare contenuti ripetitivi, già incontrati dall'utente. Tale parametro potrà aumentare la percezione che l'utente ha del funzionamento del sistema.
- *Serendipity*: come per la novità, anche in questo caso è importante che le raccomandazioni diventino anche "sorprendenti scoperte", riuscendo a suggerire all'individuo anche elementi imprevisti e non cercati (H. Walpole). Il valore aggiunto consiste nel trovare un prodotto o servizio nuovo e inaspettato che risulti comunque tra gli interessi latenti dell'individuo.

I sistemi di raccomandazione cercano di bilanciare queste caratteristiche per aumentare i benefici degli algoritmi a favore di utenti e aziende. Alcuni benefici riguardano: riduzione dei costi di transazione per la ricerca e la selezione di articoli in un ambiente di shopping online, miglioramento del processo decisionale e della qualità delle decisioni, aumento dei ricavi per gli e-commerce, approfondimento delle ricerche scientifiche grazie alla disponibilità di biblioteche online. Di conseguenza, la necessità di utilizzare tecniche di raccomandazione efficienti e accurate all'interno di un sistema che fornirà raccomandazioni pertinenti e affidabili per gli utenti non può essere sottovalutata.

Ogni tipologia di raccomandazione ha bisogno di seguire tre fasi specifiche, rappresentate nella Figura 8[46], per fornire i propri suggerimenti personalizzati.

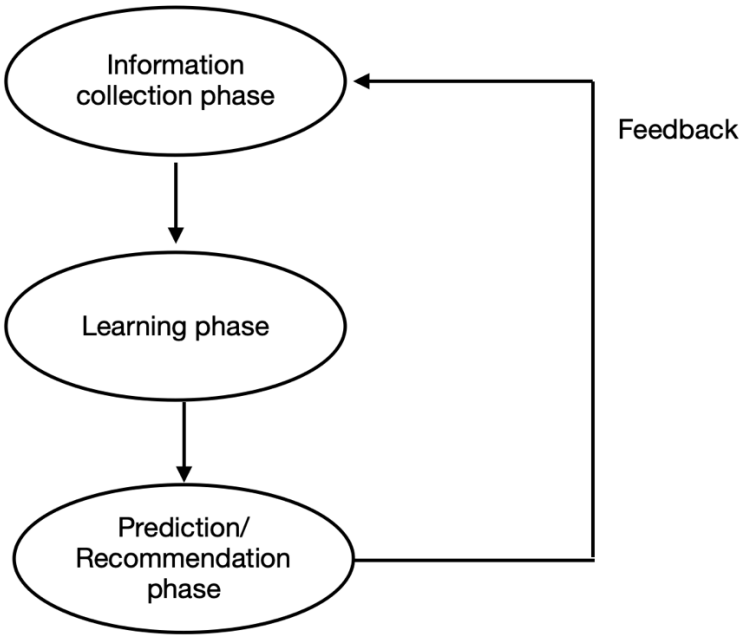


Figura 8. Fasi di raccomandazione

[46]Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261-273. https://doi.org/https://doi.org/10.1016/j.eij.2015.06.005

- *Information collecting phase*. Durante questa fase si raccolgono tutte le informazioni rilevanti per realizzare un corretto profilo dell'utente o un modello per le attività di previsione, inclusi gli attributi dell'utente, i comportamenti o il contenuto delle risorse a cui l'utente accede. Solo quando si costruirà un profilo o modello, sarà possibile avere una raccomandazione accurata sulla base della grande quantità di informazioni e dati indispensabili relativi all'individuo. I sistemi di raccomandazione si basano su diversi tipi di input come il feedback esplicito caratterizzato da alta qualità e convenienza, che include l'input esplicito degli utenti riguardo al loro interesse per l'elemento o il feedback implicito inferendo le preferenze dell'utente indirettamente attraverso l'osservazione del comportamento dell'utente. Dalla combinazione del feedback esplicito ed implicito si ottiene il feedback ibrido. Il profilo dell'utente diventa una raccolta di informazioni personali come abilità cognitive, capacità intellettuali, stili di apprendimento, interesse, preferenze e interazioni con il sistema. Il successo di qualsiasi sistema di raccomandazione dipende in gran parte dalla sua capacità di rappresentare tali interessi attuali dell'utente personalizzando ogni ricerca.
- *Learning phase*. Applica un algoritmo di apprendimento per filtrare e sfruttare le funzionalità dell'utente dal feedback ricevuto durante la prima fase di raccolta delle informazioni. La tipologia di filtraggio può avere varie tecniche, a seconda della tipologia di dominio di applicazione.
- *Prediction/Recommendation phase*. Raccomanda o prevede il tipo di articoli che l'utente potrebbe preferire. Ciò può essere effettuato direttamente sulla base del set di dati raccolto nella prima fase che potrebbe essere basata sulla memoria o su un modello o attraverso le attività osservate dal sistema dell'utente.[47]

2.1.2 VALUTAZIONI

Per riuscire a misurare l'efficacia di un sistema di raccomandazione, si possono effettuare tre diverse tipologie di valutazione: user studies, valutazione online e valutazione offline da parte di esperti.

- Il metodo di *valutazione user studies* viene anche definito esplicito, in quanto utilizza approcci valutativi per verificare affidabilità ed efficienza degli algoritmi sviluppati per creare previsioni e stabilisce l'algoritmo migliore sulla base della media dei voti ricevuti per ogni algoritmo.

[47] Ibidem

- Per quanto riguarda la *valutazione online*, parliamo invece di metodo implicito, perché la valutazione del sistema avviene attraverso il momento di navigazione su un sito per analizzare il comportamento dell'utente e osservare le sue decisioni e il tempo che intercorre tra una raccomandazione fornita dal sistema stesso e la sua scelta di accettarla. Lo scopo ultimo è quello di influenzare il comportamento dell'utente, portandolo anche a cambiare le sue scelte sulla base delle raccomandazioni. Tale comportamento viene misurato con l'utilizzo del CTR (click-trough rate), ovvero l'indice che indica la percentuale di click ricevuti dagli elementi. Inoltre, si possono raccogliere feedback mediante l'osservazione dei loro comportamenti oppure in attraverso esplicita richiesta.[48] Tale modello di valutazione è molto utile per misurare direttamente gli obiettivi generali del sistema quali profitto a lungo termine e fidelizzazione dell'utente, perché riesce a misurare l'effettiva soddisfazione dell'utente. I due grandi svantaggi di questa valutazione riguardano l'elevato costo economico e di tempo.
- La *valutazione offline* utilizza dataset esistenti che si dividono in training e test set, ai quali vengono eliminate alcune informazioni. Lo scopo è quello di riuscire a prevedere le informazioni mancanti e sviluppare suggerimenti sulla base di queste previsioni e dei dati disponibili. In questo modello vengono fatte delle previsioni da parte dei sistemi e gli utenti le correggono oppure le utilizzano. Tali valutazioni sono veloci e semplici da condurre anche su una quantità elevata di dati. Non viene richiesta un'interazione con un utente reale e ciò permette di comparare un ampio intervallo di algoritmi a basso costo[49]. Lo svantaggio principale di questo modello consiste nella limitata funziona dell'algoritmo di riuscire a rispondere solo ad un numero di richieste. Vengono identificati tre tipologie di valutazioni offline: *true offline datasets*, *user offline datatset* ed *expert online dataset*. La prima tipologia di basa sull'ipotesi che il sistema dovrebbe riuscire a prevedere anche informazioni sconosciute o comunque mancanti. Di conseguenza vengono rimosse delle valutazioni dal database e il sistema deve riuscire a creare alcuni suggerimenti facendo riferimento solo alla lista di utenti e ai dati disponibili. La raccomandazione diventa sempre più accurata, man mano che le predizioni dei dati mancanti diventano corrette. La seconda tipologia si riferisce al caso in cui l'utente non assegna una valutazione agli elementi che utilizza. In questo caso si possono creare previsioni sulla base di azioni

[48]Sistema di raccomandazione - Wikipedia. It.wikipedia.org. Retrieved 4 May 2021, from https://it.wikipedia.org/wiki/Sistema_di_raccomandazione#:~:text=basato%20sul%20contenuto.-,Valutazione,user%20studies%20viene%20definito%20esplicito.

[49]Delmedico, A. (2019). Sistemi di raccomandazione e comportamento del consumatore: confronto tra user-based e item- based collaborative filtering (Laurea Magistrale). Luiss Guido Carli.

compiuti dall'utente, come download e condivisioni. L'ultima tipologia riguarda la creazione dei dataset da parte di esperti.

I sistemi di raccomandazione mirano ad essere molto accurati e precisi per ogni utente, così da offrire una migliore esperienza di navigazione. Riuscire a dare suggerimenti appropriati e personalizzati è possibile solo grazie all'utilizzo dello storico dell'individuo, attraverso sue ricerche, commenti e acquisti. Nei casi in cui i dati degli utenti dovessero risultare pochi o non sufficienti, il sistema mirerà a raggruppare tali utenti con profili simili, provando a stabilire raccomandazioni "di gruppo", per poi affinare la personalizzazione con l'aiuto delle interazioni di ogni utente.[50]

2.1.3 LIVELLI DI PERSONALIZZAZIONE

Uno studio condotto da eMarketer dimostra che l'80% delle aziende ritiene fondamentale l'utilizzo di tecniche di personalizzazione per aumentare i propri ricavi e raggiungere i propri obiettivi, mentre il 91% di essi conferma che la propria azienda ha bisogno di migliorare la tipologia di tecniche utilizzate e la qualità dei dati input. La necessità di utilizzare tecniche sempre più avanzate deriva soprattutto dalla tipologia di utente che oggi si presenta online. I clienti diventano sempre più critici ed esigenti, richiedono esperienze contestualizzate che riescano a adattarsi perfettamente alle proprie esigenze. Essi sanno di poter pretendere soluzioni tempestive e personalizzate, in quanto si ritrovano una vasta possibilità di scelta a portata di click. Di conseguenza, l'azienda è consapevole che il tasso di abbandono potrebbe crescere se non si è pronti a rispondere ad ogni richiesta specifica.

Akim Demora direttore marketing e-commerce Auchan, afferma "Se moltiplichiamo l'audience per l'offerta, abbiamo customer journey molto diversi e potenzialmente molto complessi. Abbiamo anche alti volumi di traffico, il che vuol dire che la segmentazione dei nostri visitatori e il perfezionamento del customer journey richiede molto lavoro."[51] Questa frase evidenzia la vasta differenza di clienti che ogni azienda può avere e con la quale dovrebbe interagire, ed è proprio per questo motivo che possedere sistemi di raccomandazione sempre accurati diventa necessario per generare lead, convertire, fidelizzare e riattivare un profilo.

L'algoritmo del sistema riceve dati caldi e freddi (Figura 9), che deve analizzare insieme per restituire output che riescano a collocarsi su diversi livelli di personalizzazione sulla base di statistiche generiche o dati personali dell'individuo.

---

[50]Santucci, U. Sistema di raccomandazione. Retrieved from http://www.umbertosantucci.it/atlante/sistema-di-raccomandazione/.

[51]Bellec, A. (2020). Le 4 fasi di una personalizzazione di successo. Kameleoon. Retrieved from https://www.kameleoon.com/it/blog/4-fasi-personalizzazione-successo.

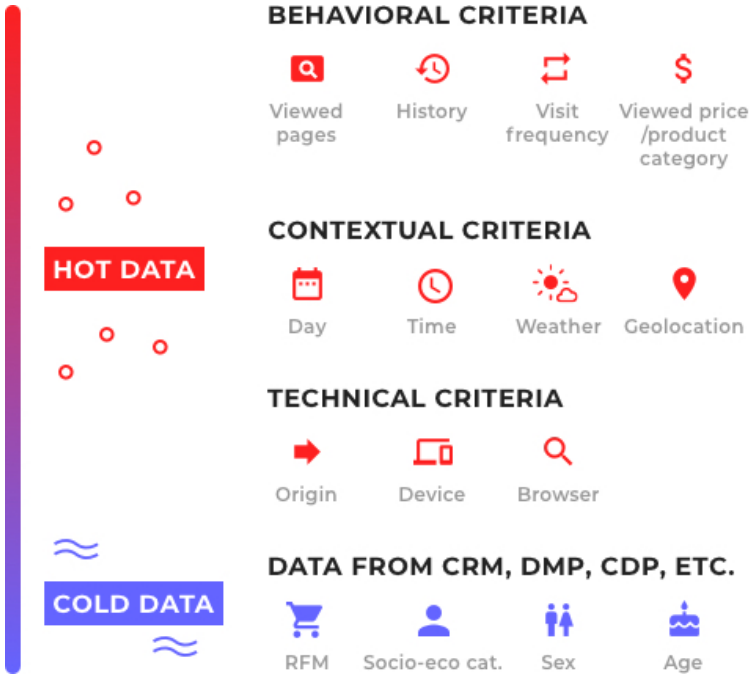


Figura 9. Mostra la tipologia di dati raccolti dall'algoritmo[52]

I diversi livelli possono essere distinti come segue:

- *Raccomandazioni non personalizzate*. Il sistema mostra una serie di item che possono interessare a molti utenti (esempio: notizie più recenti), e che vengono presentati a tutti coloro che interagiscono con questa tipologia di sistema.
- *Semi/segment personalized.* Il sistema si basa su un criterio per raggruppare gli utenti (esempio: età, provenienza geografica, professione). Si tratta di tutti i dati che non si riferiscono alla sfera personale dell'individuo, ma alle sue caratteristiche di natura generale.
- *Raccomandazione personalizzata*. Il sistema si basa su tutti i dati personali dell'utente, che mostrano le sue interazioni, preferenze, acquisti, condivisioni. Elaborare queste informazioni è la chiave per la creazione di una personalizzazione ad hoc.
- *Raccomandazioni non personalizzate + semi/segment*. La fusione dei primi due livelli può comunque generare raccomandazioni personalizzate. Amazon utilizza frequentemente questa tipologia di personalizzazione, mostrando gli articoli più venduti (primo livello) e articolo acquistati da altri utenti che hanno effettuato un acquisto simile in precedenza (secondo livello).

Un algoritmo che possiede una grande quantità di dati, derivati da un traffico elevato su un sito, sarà sicuramente molto più preciso ed elaborato. È possibile raccogliere informazioni sia in maniera diretta, come il log in ad un sito web, sia in maniera indiretta, come l'accettazione di

[52] Kameleoon. Cos'è la personalizzazione?. Retrieved from https://www.kameleoon.com/it/personalizzazione#personalization%20types.

cookie.[53] È utile ricordare che non è sempre possibile assegnare un rating a tutti gli items. Per questo motivo i sistemi di raccomandazione devono utilizzare adeguati metodi di filtraggio delle informazioni, per includere tutti i prodotti valutati sia in maniera esplicita che implicita e suggeriti all'utente.[54]

Le tecniche di filtraggio (Figura 10) si possono suddividere in: collaborative filtering, content-based filtering e hybrid filtering.

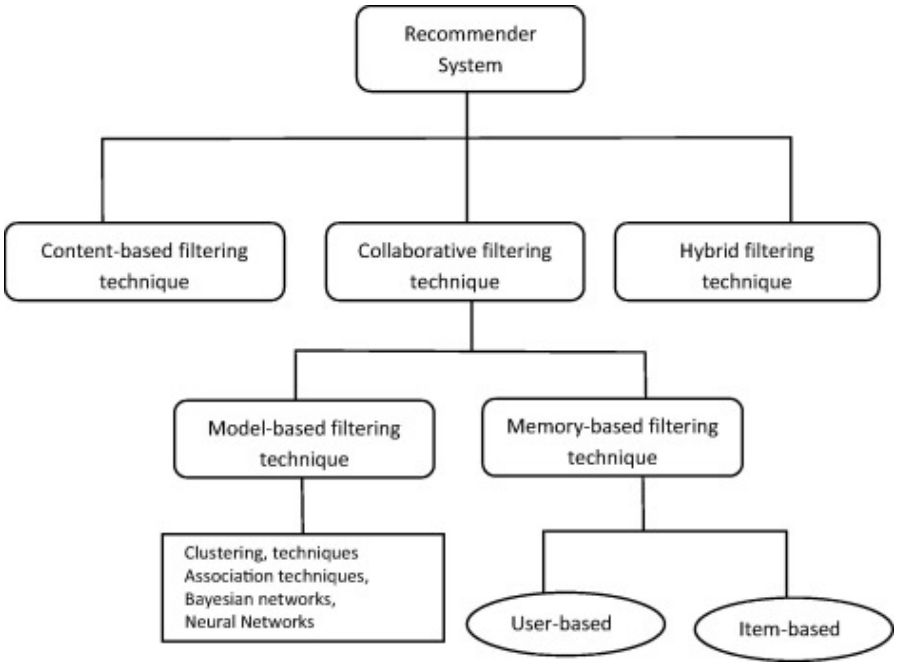


Figura 10. Tecniche di raccomandazione[55]

2.2 CONTENT- BASED FILTERING

Il filtraggio basato sui contenuti rappresenta una delle tecniche di raccomandazione di maggior successo. Per contenuti si intendono gli attributi degli elementi che vengono analizzati per generare nuove previsioni su caratteristiche, comportamenti e preferenze dell'utente. La tecnica utilizzata consiste in un algoritmo dipendente dal dominio, dove si incrocia il contenuto di un item (come parole chiave e tags) con l'user-profile (incluse le preferenze aggiornate in real time) [56], come mostrato in figura 11.

[53]I *cookie* dei siti web raccolgono le informazioni dell'utente (lasciate in maniere consensuale, seppur a volte non consapevole), che vengono trasferite al suo profilo una volta effettuata una registrazione sul sito.

[54]Delmedico, A. (2019). Sistemi di raccomandazione e comportamento del consumatore: confronto tra user-based e item- based collaborative filtering (Laurea Magistrale). Luiss Guido Carli.

[55]Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261-273. https://doi.org/https://doi.org/10.1016/j.eij.2015.06.005

[56]Leonardi, A. (2019). Il valore del Machine Learning nel Marketing. AI4Business. Retrieved from https://bit.ly/34uWo0N

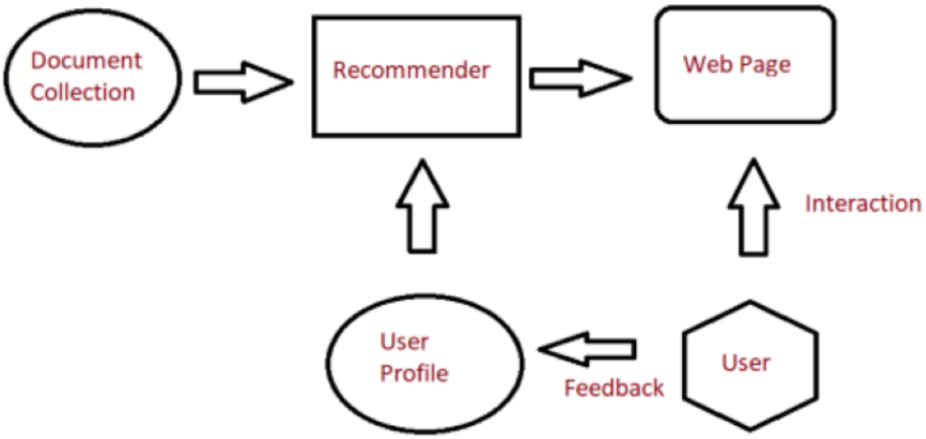


Figura 11. Modello del sistema di raccomandazione basato sui contenuti[57]

Il contenuto in questione si basa su item differenti che variano a seconda del sistema, del dataset e dell'ambito di applicazione; mentre il profilo utente racchiude tutti quei comportamenti impliciti ed espliciti nei confronti degli item che gli vengono proposti. Gli item possono essere:

- Strutturati: ogni item è caratterizzato dallo stesso set di attributi, variabili o campi. E tutti i valori possibili da far assumere alle variabili sono validi.
- Non strutturati: ad ogni attributo non viene associata un'etichetta né un valore definito (come nel caso degli articoli dove si ha una complessità di linguaggio per l'uso di vocaboli polisemantici o sinonimi).
- Semi-strutturati: sono presenti alcuni attributi associati ad un set ristretto di valori. Si prendono in considerazione campi di testo difficili per gli algoritmi da analizzare, per trasformarli in dati strutturati con l'ausilio di tecniche specifiche.[58]

Quindi, la raccomandazione che ne segue sfrutta la funzionalità del contenuto degli elementi già valutati positivamente dall'utente in passato. "Il sistema cerca di raccomandare articoli dal contenuto simile agli utenti che hanno già espresso interesse per quella tipologia di contenuti" (Balabonovic et al., 1997). Pertanto, questi sistemi richiedono tecniche appropriate per rappresentare gli elementi e determinare le preferenze dell'utente, nonché strategie per confrontare le preferenze dell'utente con le rappresentazioni degli utenti. Alla fine, verrà consigliato il prodotto con le caratteristiche più sovrapposte agli interessi degli utenti.

Algoritmi utilizzati per questo sistema di raccomandazione sono i classification learning, basati proprio sulle preferenze dell'utente e le sue scelte passate, prevedendo l'interesse di un utente

[57]Shekhar, Y. (2020). ML - Content Based Recommender System. GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/ml-content-based-recommender-system/.

[58]Delmedico, A. (2019). Sistemi di raccomandazione e comportamento del consumatore: confronto tra user-based e item- based collaborative filtering (Laurea Magistrale). Luiss Guido Carli.

verso una nuova tipologia di item. In questo modo si riesce a stabilire con quale probabilità un individuo possa interessarsi ad un prodotto che non ha ancora visualizzato oppure quale sia l'indice di interesse. Esistono diversi algoritmi di classification learning: modello dello spazio vettoriale; modelli probabilistici, come Decision Tree e Naïve Bayes; metodi del Nearest Neighbor; feedback di rilevanza; classificatori lineari.

Le tecniche di filtraggio content-based riescono a superare molte sfide ed è per questo motivo che vengono spesso utilizzate dalle aziende. Di seguito vengono spiegati alcuni vantaggi[59]:

- Il modello è facilmente *scalabile* per la bassa quantità di dati.
- Vengono analizzati gli elementi e il profilo di un *singolo utente* e ciò rende il sistema più semplice.
- Il modello vanta una forte capacità di *adattamento*, in quanto riesce a consigliare anche articoli che non hanno valutazione da parte degli utenti. Inoltre, se le preferenze dell'utente dovessero cambiare, riuscirebbe a modificare i consigli in breve tempo.
- *Facile da interpretare.*
- Protezione della *privacy*, in quanto gli utenti possono ricevere suggerimenti senza essere costretti a condividere il proprio profilo.
- È una tecnica *trasparente*, in quanto gli elementi vengono consigliati a livello di funzionalità, prendendo in considerazione anche altri utenti simili sconosciuti.
- È possibile *suggerire nuovi elementi* prima di essere valutati da un numero considerevole di utenti.

Tuttavia, queste tecniche soffrono anche vari problemi e limitazioni:

- *Analisi del contenuto limitato*, ovvero si ha bisogno di una ricca descrizione degli articoli e un profilo utente molto ben organizzato prima che la raccomandazione possa essere realizzata.
- La *specializzazione eccessiva del contenuto* diventa un limite da parte degli utenti che riceveranno solo consigli simili agli elementi già definiti nei loro profili.
- Può essere molto difficile trovare le *giuste feature.*
- È *limitato*, in quanto dipende dagli interessi degli utenti precedentemente noti. Di conseguenza, non è in grado di espandere gli interessi degli utenti noti.[60]

[59]Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261-273. https://doi.org/https://doi.org/10.1016/j.eij.2015.06.005
[60]Ibidem

2.3 COLLABORATIVE FILTERING

Il filtraggio collaborativo è una strategia di raccomandazione personalizzata che identifica e analizza le somiglianze tra vari utenti per fornire successivamente consigli pertinenti su prodotti. È una tecnica di previsione indipendente dal dominio per i contenuti che non possono essere descritti facilmente e adeguatamente da metadati, quali film e musica. Ogni elemento e ogni utente viene descritto da un vettore di funzionalità oppure da un incorporamento. Di conseguenza, il sistema costruisce un database di matrice elemento-utente, basandosi sulla raccolta di un grande quantità di dati riguardanti comportamenti, attività e preferenze di ogni utente per calcolare la somiglianza tra i vari profili, abbinarli tra di loro e formulare sia previsioni che raccomandazioni.[61]

I dati utilizzati si basano esclusivamente su interazioni passate (dati storici) tra gli utenti e gli elementi target. Infatti, generalmente viene utilizzata una matrice dove le righe rappresentano gli utenti e le colonne gli elementi, che dovranno essere sufficienti per fare una previsione corretta.[62] Il presupposto è che gli elementi concordati in passato dagli utenti possano essere da loro scelti di nuovo in futuro. Ed è in questo ambito che si raccolgono i due diverse tipologie di feedback degli utenti per ricreare nuovi consigli:

- *Feedback esplicito*. Dati che un utente fornisce attivamente tramite risposte ad un questionario o sondaggio oppure valutazione e reazioni ad un oggetto. Tali feedback possono essere sia positivi che negativi
- *Feedback implicito*. Dati che il sistema deduce in base al comportamento dell'utente, come visualizzazioni di una pagina, click, acquisti. Sono azioni continue dell'utente che però esprimono principalmente solo feedback positivi.

Amazon utilizza questa tipologia di filtraggio abbinando i prodotti agli utenti, basandosi sugli acquisti passati. Inoltre, riesce ad identificare altre categorie di prodotti che potrebbero piacere osservando le somiglianze tra i prodotti e i comportamenti di utenti simili.[63]

La tecnica di filtraggio collaborativo può essere suddivisa in due categorie, come mostra la figura 12.

[61]Ibidem

[62]I sistemi di raccomandazione - Recommender System. Rocket To Ride. (2019). from http://www.rockettoride.com/r-tutorial-archive/i-sistemi-di-raccomandazione-recommender-system.

[63]Retta, L. The power of collaborative filtering. Dynamic Yield. Retrieved from https://www.dynamicyield.com/lesson/collaborative-filtering/.

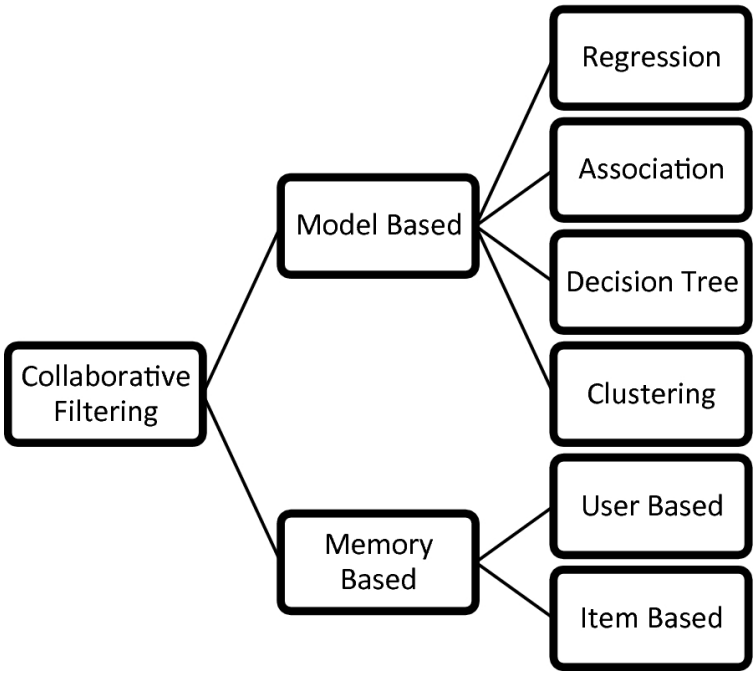


Figura 12. Classificazione degli algoritmi di Filtraggio Collaborativo[64]

- *Tecniche basate sulla memoria*: si utilizzano come input gli articoli valutati in precedenza per ricercare l'utente che condivide lo stesso apprezzamento. Una volta individuato, è possibile utilizzare diversi algoritmi per combinare le preferenze degli utenti simili all'interno di uno stesso dataset, usando metriche di similarità, e generare nuove raccomandazioni. Tale filtraggio ha riportato grandi benefici alle aziende che l'hanno utilizzato.[65] La tecnica basata sulla memoria può essere ottenuta in due modi differenti:
    a. Tecniche basate sull'*utente.* Calcolano la somiglianza tra gli utenti confrontando le loro valutazioni sullo stesso articolo. La logica utilizzata consiste nella seguente definizione: "se le caratteristiche di A sono simili a quelle di B, allora i prodotti scelti da B potranno piacere ad A". La valutazione prevista dell'articolo da parte dell'utente viene calcolata come la media ponderata delle valutazioni dell'elemento da parte degli utenti simili, dove i pesi sono le somiglianze degli utenti con gli elementi di destinazione.
    b. Tecniche basate sugli *elementi.* Calcolano le previsioni utilizzando solo la somiglianza tra gli elementi. La previsione viene effettuata prendendo una media ponderata della valutazione degli utenti per calcolare la somiglianza tra elemento-

[64]Aggarwal S., Goswami D., Hooda M., Chakravarty A., Kar A., Vasudha (2020) Recommendation Systems for Interactive Multimedia Entertainment. In: Hemanth J., Bhatia M., Geman O. (eds) Data Visualization and Knowledge Engineering. Lecture Notes on Data Engineering and Communications Technologies, vol 32. Springer, Cham. https://doi.org/10.1007/978-3-030-25797-2_2

[65]Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261-273. https://doi.org/https://doi.org/10.1016/j.eij.2015.06.005

utente e le misure di somiglianza più popolari vengono basate sul calcolo delle correlazioni e del coseno.

- *Tecniche basate sui modelli*: si utilizzano le valutazioni precedenti per apprendere un modello al fine di migliorare le prestazioni della tecnica del filtraggio collaborativo. Il processo di creazione del modello, parte da un training-set che viene validato attraverso l'utilizzo di vari metodi su dati di test. In questo modo, riescono a consigliare rapidamente una serie di elementi sulla base di un modello calcolato in precedenza. Si analizza la matrice utente-elemento per identificare le relazioni tra gli elementi, usandole per confrontare l'elenco delle prime N raccomandazioni. A differenza dei metodi memory-based, si considera solamente un sottoinsieme di utenti per la costruzione del modello.[66] Con l'utilizzo degli algoritmi di apprendimento si è modificato drasticamente il modo di consigliare, passando da "cosa consumare" a "quando consumarla". E tra le tecniche di raccomandazione basate sui modelli, gli algoritmi più utilizzati sono:
    a. *Regressione*: analisi utilizzata quando si ritiene che due o più variabili siano sistematicamente collegate da una relazione lineare. Rappresenta un processo potente e diversificato, dove sono contenuti adattamenti della curva, previsioni e verifica di ipotesi sistematiche sulle relazioni tra le variabili. La tipologia di curva è utile per identificare una tendenza all'interno del set di dati.
    b. *Associazione*: gli algoritmi di mining delle regole di associazione estraggono le regole che prevedono l'occorrenza di un elemento in base alla presenza di altri elementi in una transazione. In questa regola si applica la formula A→ B, dove A e B rappresentano due insiemi di elementi. Si possono formare rappresentazioni molto compatte dei dati sulle preferenze, che possono migliorare l'efficienza dell'archiviazione e delle prestazioni.
    c. *Albero decisionale*: si basa sulla metodologia dei grafici ad albero, costruiti analizzando una serie di esempi di formazione per i quali sono note le etichette delle classi. Se vengono addestrati su dati di qualità molto elevata, hanno la capacità di fare previsioni molto accurate. Sono flessibili nella gestione degli elementi con un misto di caratteristiche, nonché elementi che presentano alcune caratteristiche specifiche.
    d. *Clustering*: questo algoritmo cerca di partizionare un insieme di dati in un insieme di sotto-cluster al fine di scoprire gruppi significativi che esistono al loro interno. Una volta che i cluster sono stati formati, le opinioni degli altri utenti in un cluster possono essere mediate e utilizzate per formulare raccomandazioni per i singoli

[66]Ibidem

utenti. Un metodo buono di raggruppamento produrrà cluster di alta qualità in cui la somiglianza intra-cluster è alta, mentre la similarità tra cluster è bassa.

Esistono anche ulteriori algoritmi utilizzati come rete neurale artificiale, link analysis e classificatori bayesiani.[67]

Il filtraggio collaborativo presenta una serie di vantaggi e svantaggi differenti[68]. È evidente che esistono vantaggi rispetto al filtraggio content-based, ma anche rispetto alle due tecniche utilizzate[69]. Alcuni vantaggi sono:

- *Accuratezza:* nei metodi neighborhood la qualità dell'analisi dipende dal rapporto tra numero di utenti e item. Infatti, l'accuratezza del metodo basato sugli elementi è più elevata nei casi in cui il numero di utenti risulta essere molto più grande di quello degli item. Mentre, nel caso inverso, risulta essere migliore il metodo basato su utenti.
- *Efficienza:* anche questo vantaggio dipende dal rapporto tra numero di utenti e item. E si utilizza il metodo basato sugli elementi quando il numero di utenti è superiore a quello degli item, in quanto è necessaria meno memoria.
- *Stabilità:* il metodo basato sugli elementi è più stabile nei casi in cui l'elenco degli item sia più statico rispetto all'elenco degli users. Al contrario, risulta stabile l'apporccio basato sugli utenti.
- *Giustificazione:* rappresenta un vantaggio specifico dei metodi item-based, dove gli utenti hanno anche la possibilità di partecipare alla raccomandazione modificando la propria lista di item.
- *Serendipity*: l'approccio user-based genera originalità nelle raccomandazioni e può sorprendere l'individuo facendo notare articoli differenti. Al contrario, l'altro approccio potrebbe generare solo liste di elementi già apprezzati in precedenza.
- *Scalabilità:* negli algoritmi basati sul modello si utilizzano dataset più piccoli che riescono ad essere molto efficienti. Di conseguenza. Si otterrà un impatto sulla scalabilità del sistema.
- *Velocità nella predizione:* un vantaggio riferito ai model-based, dove la predizione su un modello avviene in tempi inferiori rispetto a quelli che analizzano un intero dataset.
- *Evita overfitting:* nei casi in cui il modello costruito rappresenta il mondo reale.
- *Funziona anche in domini con pochi contenuti associati agli elementi:* oppure anche in circostanze dove il contenuto è difficile da analizzare, come per opinioni e gli ideali.

[67]Ibidem
[68]Ricci, F., Rokach, L., & Shapira, B. (2011). Recommender Systems Handbook.
[69]Vitale, F. (2019). Recommendation system: implementazione di algoritmi collaborativi per servizi video-streaming on demand (VOD) (Laurea Magistrale). Luiss Guido Carli.

- *Raccomandazioni fortuite:* può consigliare elementi rilevanti anche senza che il contenuto sia nel profilo dell'utente.

Nonostante il successo delle tecniche di filtraggio collaborativo, si sono rivelati alcuni potenziali problemi come i seguenti[70]:

- *Qualità della predizione:* per gli algoritmi model-based, dove non si usa l'intero dataset, è possibile riscontare problematiche relative alla mancata accuratezza. Questo dipende principalmente da come il modello è stato realizzato.
- *Avviamento a freddo:* si riferisce alle situazioni in cui non si dispone di informazioni adeguate sull'utente o sull'elemento per fare previsioni pertinenti. È uno dei problemi che riducono le prestazioni del sistema di raccomandazione. Di conseguenza i profili nuovi di utenti saranno vuoti, in quanto non si avrà ancora in possesso nessun elemento.
- *Scarsità dei dati:* si verifica quando si è in assenza di informazioni sufficienti, ovvero solo pochi elementi disponibili sono valutati dagli utenti. Questo comporta una matrice di elementi utente sparsa, incapacità di individuare i soggetti simili e la conseguente realizzazione di raccomandazioni deboli.
- *Scalabilità:* il calcolo degli algoritmi normalmente cresce in modo lineare con il numero di utenti e di elementi. Quindi, una tecnica efficiente di raccomandazione quando i dati sono limitati, potrebbe non essere in grado di generare un numero soddisfacente di raccomandazioni, quando il volume del set di dati aumenta. Sarebbe opportuno applicare tecniche che sono in grado di scalare con l'aumento del numero di set di dati.
- *Sinonimo:* riguarda la tendenza di elementi molto simili ad avere nomi o voci differenti. I sistemi di filtraggio collaborativo di solito non trovano corrispondenza tra i due termini per poter calcolare la loro somiglianza.

2.4 SISTEMI IBRIDI DI RACCOMANDAZIONE

La tecnica di filtraggio ibrido combina diverse tecniche di raccomandazione al fine di ottenere una migliore ottimizzazione del sistema per evitare alcune limitazioni e problemi del sistema di raccomandazione puri. L'idea di base è che una combinazione di algoritmi può fornire raccomandazioni più accurate ed efficaci rispetto a un singolo algoritmo, poiché in questo modo è possibile superare gli svantaggi o i limiti di un algoritmo utilizzandone un altro (figura 13).

[70]Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261-273. https://doi.org/https://doi.org/10.1016/j.eij.2015.06.005

L'utilizzo di più tecniche può sopperire i punti deboli di una singola tecnica in un modello combinato (come, ad esempio, l'avvio a freddo e la scarsità).

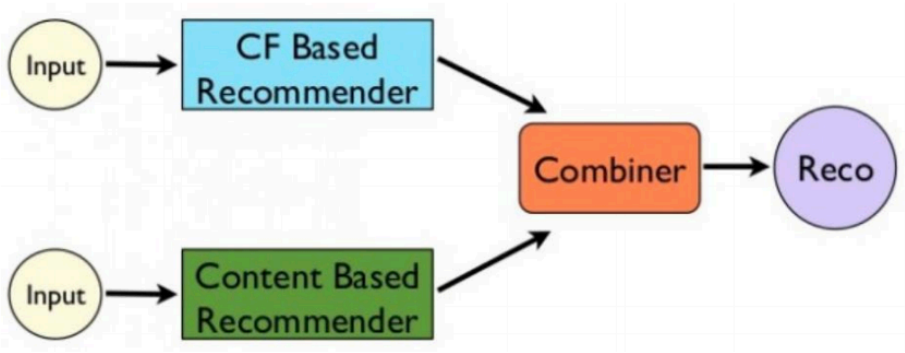


Figura 13. Sistema di raccomandazione ibrido[71]

La combinazione degli approcci può avvenire in vari modi: inserire il content-based approach all'interno di un sistema collaborativo o viceversa, inserire un approccio collaborativo ad un sistema che si basa sul contenuto delle caratteristiche, applicare entrambi gli approcci in maniera separata e confrontarli successivamente oppure costruire un modello che utilizzi entrambi gli approcci.[72] Nello specifico, è possibile effettuare una classificazione dei metodi ibridi più diffusi[73]:

- *Ibridazione ponderata:* combina i risultati di diverse raccomandazioni per generarne un elenco o una previsione integrando i punteggi di ciascuna tecnica utilizzata attraverso una formula lineare. Il vantaggio è che tutti i punti di forza del sistema di raccomandazione vengono utilizzati durante il processo di raccomandazione in modo semplice.
- *Switching hybridization:* il sistema passa a una delle tecniche di raccomandazione secondo un'euristica che riflette la capacità del suggeritore di produrre una buona valutazione. Il vantaggio è che il sistema è sensibile ai punti di forza e di debolezza dei suoi raccomandatori costitutivi. Il principale svantaggio del cambio di ibridi è che di solito introduce una maggiore complessità al processo di raccomandazione perché deve essere determinato il criterio di commutazione, che normalmente aumenta il numero di parametri del sistema di raccomandazione.
- *Ibridazione a cascata:* applica un processo di raffinamento iterativo nella costruzione di un ordine di preferenza tra i diversi elementi. Le raccomandazioni di una tecnica vengono

[71]Vitale, F. (2019). Recommendation system: implementazione di algoritmi collaborativi per servizi video-streaming on demand (VOD) (Laurea Magistrale). Luiss Guido Carli.
[72]Sistema di raccomandazione - software libero Che cos'è. ww.it.freejournal.info. (2020). Retrieved from https://amp.ww.it.freejournal.info/6966909/1/sistema-di-raccomandazione.html.
[73]Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261-273.

perfezionate da un'altra tecnica di raccomandazione. La prima tecnica di raccomandazione produce un elenco grossolano di raccomandazioni che a sua volta viene perfezionato dalla successiva tecnica.

- *Ibridazione mista:* combina i risultati dei suggerimenti da parte di differenti tecniche di raccomandazione contemporaneamente, invece di utilizzare un solo consiglio per articolo. Le prestazioni individuali non sempre influenzano le prestazioni generali.
- *Combinazione di funzionalità:* Le caratteristiche prodotte da una specifica tecnica di raccomandazione vengono inserite in un'altra tecnica di raccomandazione. Ad esempio, la valutazione di utenti simili, che è una caratteristica del filtraggio collaborativo, viene utilizzata in una tecnica di raccomandazione di ragionamento basata sui casi come una delle caratteristiche per determinare la somiglianza tra gli elementi. Il vantaggio di questa tecnica è che non si basa sempre esclusivamente sui dati di collaborazione.
- *Miglioramento delle funzionalità:* La tecnica fa uso delle valutazioni e di altre informazioni prodotte dal precedente suggeritore e richiede anche funzionalità aggiuntive dai sistemi di raccomandazione. Gli ibridi di potenziamento delle funzionalità sono superiori ai metodi di combinazione di funzionalità in quanto aggiungono un numero limitato di funzionalità al suggeritore principale.
- *Meta Level:* Il modello interno generato da una tecnica di raccomandazione viene utilizzato come input per un'altra. Il modello generato è sempre più ricco di informazioni rispetto a un singolo rating. Gli ibridi di meta-livello sono in grado di risolvere il problema della scarsità delle tecniche di filtraggio collaborativo utilizzando l'intero modello appreso dalla prima tecnica come input per la seconda tecnica.

Tali metodi appena introdotti, possono essere a loro volta suddivisi in due categorie, basandosi sul criterio che tengano in considerazione l'ordine con il quale gli algoritmi vengono combinati. I metodi di ibridazione ponderata, *switching*, mista e combinazione di funzionalità non tengono in considerazione l'ordine delle combinazioni, in quanto non influenza il risultato finale. Mentre, per il miglioramento delle funzionalità, l'ibridazione a cascata e il meta-level è importante.

2.5 CASO NETFLIX

Netflix è un esempio di applicazione di un sistema di raccomandazione ibrido, che suggerisce agli utenti film o serie tv per migliorare la *customer experiences*, abbassare il tasso di abbandono e mantenere gli utenti attivi sulla piattaforma per molto tempo sulla base delle preferenze dell'utente stesso e degli altri utenti simili (Figura 14).

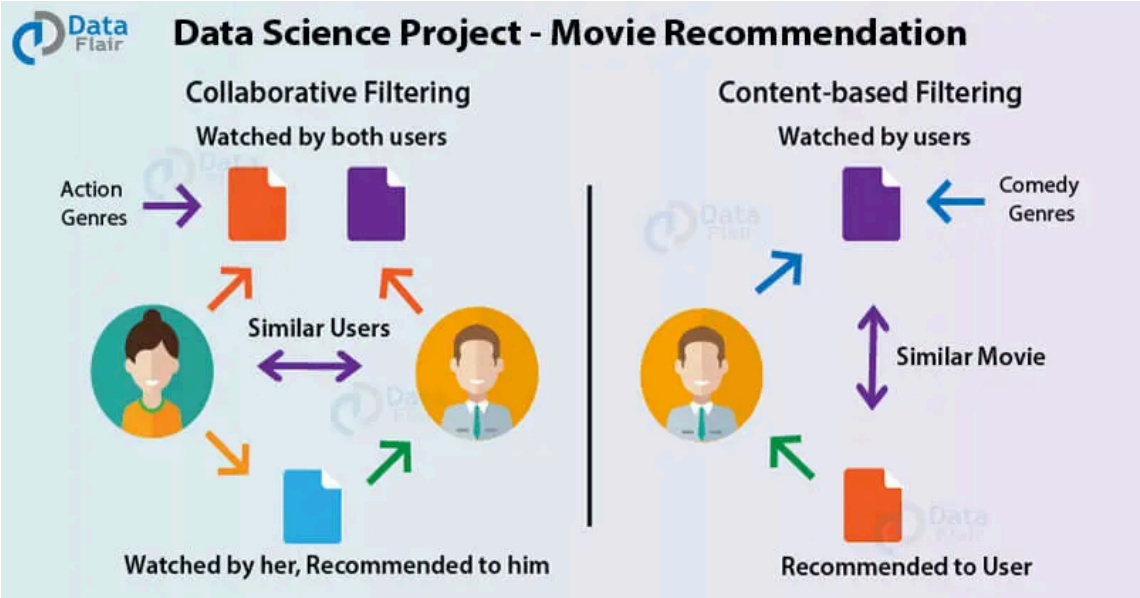


Figura 14. Sistema di raccomandazione di Netflix[74]

Negli anni '90 Netflix si basava su un sistema di abbonamenti, inviando direttamente a casa dei clienti il DVD richiesto. Successivamente intorno agli anni 2000 inizia ad aggiungere consigli personalizzati sui film, utilizzando come sistema di raccomandazione "Cinematch" con un errore quadratico medio pari a 0,9525. Nel 2006 viene lanciata la competizione "Netflix Prize", un mix di machine learning e data mining con un conseguente premio di 1 milione di dollari, dove si chiedeva ai partecipanti di battere l'errore quadratico medio. Un anno dopo il vincitore ottenne un RMSE pari a 0,88 utilizzando una combinazione lineare di Matrix Factorisation e Restricted Boltzmann Machined e dopo alcuni adattamenti da parte di Netflix si mise in produzione tale algoritmo. L'aspetto interessante riguarda proprio la scelta del vincitore, in quanto successivamente altri partecipanti ottennero un RMSE pari a 0,8567, che non fu mai preso in considerazione per l'enorme richiesta di sforzo ingegneristico richiesto per aumentare la precisione dell'algoritmo. Inoltre, in quegli anni Netflix stava già utilizzando il servizio streaming, di conseguenza i dati degli utenti in suo possesso erano aumentati in maniera notevole. Si è passati così da "un servizio di posta che pubblica DVD negli Stati Uniti a un

[74]Bernardini, F. (2021). Che cosa è il recommender system: il caso Netflix. Neuragate. Retrieved from https://www.neuragate.it/intelligenza-artificiale/che-cosa-e-il-recommender-system-il-caso-netflix/.

servizio di streaming globale con 182,8 milioni di abbonati". Il problema della regressione adesso lascia spazio a problemi di classificazione, generazione di pagine, massimizzazione dell'esperienza come mostrato in figura 15.[75]

Figura 15. Evoluzione dell'approccio di personalizzazione di Netflix

Il suo sistema di raccomandazione raccoglie vari dati dell'utente prima di suggerire un titolo: le interazioni con il servizio della piattaforma come visione e valutazione; legame con abbonati che hanno simili preferenze; informazioni sui film come titolo, anno di uscita del film, attori, genere; orario di visione; dispositivi utilizzati per la visione, tempo trascorso sulla piattaforma. Tutti questi elementi costituiscono dati input fondamentali per l'algoritmo che li elabora per la realizzazione dell'home page di ogni utente, luogo dove si crea proprio l'esperienza di raccomandazione con copertine di film in linea con i suoi gusti. Circa l'80% delle ore di visione sono influenzate dall'implementazione degli algoritmi di raccomandazione".[76]

Tale personalizzazione viene poi completata con l'aggiunta dei video più popolari in maniera simile per ogni utente. La pagina principale è formata da diverse sezioni, suddivise tra righe (Figura 16) di video a tre livelli di personalizzazione (es. continua a guardare, i titoli del momento, film consigliati in base a ciò che hai visto, commedie premiate, ecc.) e ciascuna di essa utilizza algoritmi differenti.

- Spostandosi sulla riga si trovano le raccomandazioni più forti partendo da sinistra e proseguendo a destra in quelle gradualmente meno affini.
- Spostandosi su più righe si trovano i consigli più efficaci in alto.[77]

[75]ICHI.PRO. Approfondisci il sistema di raccomandazione di Netflix. Retrieved from https://ichi.pro/it/approfondisci-il-sistema-di-raccomandazione-di-netflix-57277795941192.
[76]Casagranda, P., & Metta, S. (2016). *Leggi questo articolo, una tua amica lo ha trovato interessante | Un'introduzione alle opportunità e criticità dei recommender system per la personalizzazione dei contenuti audiovisivi.* Crit.rai.it. Retrieved from http://www.crit.rai.it/eletel/2016-2/162-4.pdf.
[77]Netflix. Funzionamento del sistema di consigli di Netflix. Retrieved from https://help.netflix.com/it/node/100639.

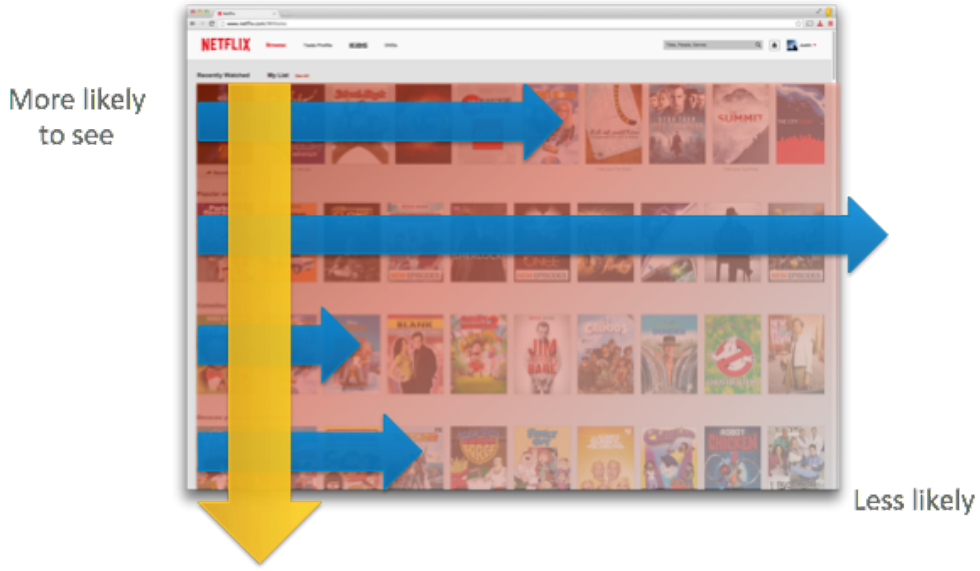


Figura 16. In che modo Netflix classifica i titoli[78]

Ovviamente Netflix utilizza un ulteriore sistema di raccomandazione per decidere il tipo di riga da mostrare, dove ad esempio la categoria "consigliati perché hai guardato questo film" (Figura 17) rappresenta un content-based recommender system, mentre la categoria "migliori film storici" rappresenta un community-based recommender system.

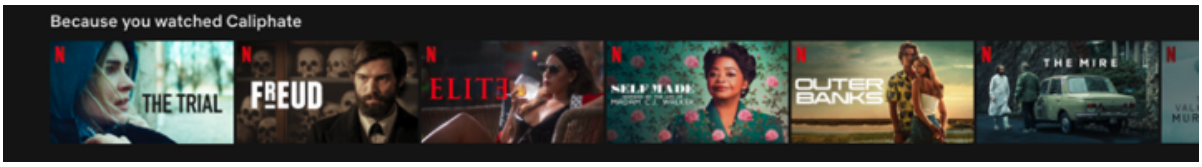


Figura 17. Esempio di titoli generati dal ranker per similarità video-video[79]

Il sistema di Netflix vorrebbe prevedere ciò che gli utenti hanno intenzione di guardare durante ogni sessione e questo include anche serie tv iniziate o film non conclusi. Nonostante ciò, si vuole sempre mettere in evidenza la vasta scelta di titoli in catalogo, sfruttando novità o popolarità. Di conseguenza l'algoritmo generato si basa sulla precisione e sulla vasta fornitura di diversità, accessibilità e stabilità, come mostrato in figura 18.

[78]ICHI.PRO. Approfondisci il sistema di raccomandazione di Netflix. Retrieved from https://ichi.pro/it/approfondisci-il-sistema-di-raccomandazione-di-netflix-57277795941192.
[79]Ibidem

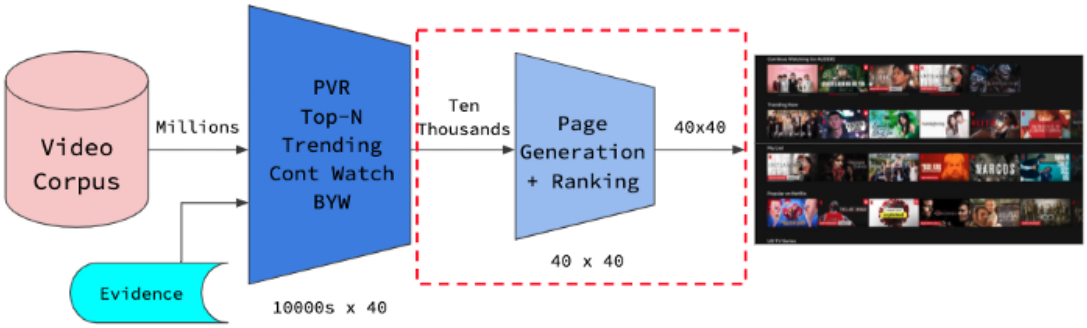


Figura 18. Flusso di lavoro del modello di Netflix[80]

Il sistema utilizzato si migliora raccogliendo feedback ogni volta che si utilizza il servizio e sfruttando le risposte degli utenti è possibile riaddestrare l'algoritmo con nuove informazioni, così da creare previsione più accurate. I dati, gli algoritmi e i sistemi di calcolo si sviluppano gli uni con gli altri, producendo nuovi consigli e migliorando l'esperienza.

In questo modo si può trarre un duplice vantaggio:

- Da parte dell'utente, il quale si trova davanti ad una scelta più facilitata
- Da parte dell'azienda, la quale raccoglie feedback immediati.

[80]Ibidem

CAPITOLO 3. MOVIELENS: CONFRONTO SPERIMENTALE TRA LIBRERIE PYTHON

Nei precedenti capitoli sono state illustrate le tipologie di machine learning e le loro applicazioni. In modo particolare è stato posto un focus sulle tipologie di sistemi di raccomandazione, i loro algoritmi e le loro applicazioni nelle scelte quotidiane.
In questo capitolo, invece, vengono introdotte le principali librerie *Python* per i *recommender system*, soffermandosi sull'utilizzo del database "*MovieLens*". Su questo viene effettuata un'analisi e un confronto sperimentale tra alcune librerie che lo utilizzano, per testarne l'accuratezza ed evidenziarne i limiti. Infine, si commentano i risultati suggerendo lo studio migliore.

3.1 LITERATURE REVIEW

L'informazione tecnologia si è ampiamente diffusa negli ultimi anni e grandi quantitià di dati sono state rese accessibili a tutti gli utenti. Questo ha reso difficile la selezione dei dati che fossero in accordo con le esigenze degli utenti stessi. Per risolvere tale problema sono emersi sistemi di raccomandazione, in grado di seguire e aiutare gli utenti nel processo decisionale e nella selezione dei dati rilevanti.
Studi recenti hanno dimostrato che i clienti apprezzano i contenuti personalizzati come annunci, offerte o consigli, ma si aspettano che i retailer usino i dati accumulati di ogni consumatore, solo ed esclusivamente per personalizzare i loro contenuti. Inoltre, la personalizzazione risulta positivamente correlata all'esperienza del consumatore, in quanto semplifica le ricerche, riduce i costi di valutazione dei prodotti e conseguentemente aumenta la fedeltà dei clienti stessi (Tyrväinen et al., 2020). La personalizzazione nel contesto online implica il monitoraggio costante delle precedenti abitudini di acquisto dei consumatori attraverso l'uso dei sistemi di raccomandazione, per modificare cosa visualizzare e come visualizzarlo (Zhang et al., 2011). In questo modo è possibile offrire i prodotti più adatti per un determinato consumatore. L'integrazione dell'intelligenza artificiale provoca un cambiamento nel modo in cui i consumatori percepiscono le loro esperienze di acquisto e i fattori che considerano importanti per raggiungere il successo di queste esperienze (Ameen et al., 2021).
Una maggiore qualità del servizio e migliori vantaggi della personalizzazione aumentano l'intenzione di acquisto (Pappas et al., 2016), in quanto migliorano l'esperienza totale del consumatore. L'esperienza di acquisto online passata, l'utilità percepita e la soddisfazione del cliente sono fattori in grado di influenzare l'intenzione di riacquisto di un cliente (Tsai e Huang, 2007; Chen et al., 2009). L'intenzione di riacquisto viene considerata dagli studiosi come un

riflesso della lealtà di un consumatore e come un costrutto di stima del successo del commercio online anziché usare l'intenzione di utilizzo.

L'esperienza del cliente viene intesa negli studi come un costrutto multidimensionale che caratterizza le risposte cognitive, emotive, comportamentali, sensoriali e sociali di un cliente ai processi di erogazione del servizio. Nonostante una misurazione totale dell'esperienza del consumatore sia fondamentale per analizzare il processo decisionale, gli studiosi hanno iniziato a condurre queste ricerche solo di recente.

Uno dei fattori fondamentali per il successo di un qualsiasi negozio online è la soddisfazione del consumatore, la cui comprensione è di estrema importanza al fine di massimizzare le vendite all'interno di un e-commerce. La soddisfazione è stata definita come una sensazione personale di appagamento o piacere, come risultato del confronto tra i risultati effettivi dello shopping online e le loro aspettative (Ives et al., 1983). La soddisfazione dei clienti online avrà un impatto positivo sulla loro intenzione futura di acquisto, direttamente o indirettamente (Tyrväinen et al., 2020).

Uno studio condotto da Kumar et al., (2021) ha dimostrato che i retailer online devono concentrarsi sulla qualità delle informazioni, sul sistema di sicurezza e sulla qualità del sistema dei siti e-commerce per poter migliorare la soddisfazione dei consumatori, così da condurre il cliente all'acquisto.

Al giorno d'oggi, l'esistenza di tecnologie di comunicazione come Internet e tecnologie dell'informazione come servizi elettronici e strumenti digitali hanno portato all'aumento della velocità del contenuto generato e del trasferimento delle informazioni. D'altra parte, con l'aumento del numero di utenti di Internet, c'è stato un aumento esponenziale della quantità di dati e informazioni su queste reti. Pertanto, con lo sviluppo dell'e-commerce, delle biblioteche digitali e simili, gli utenti sono circondati da un enorme volume di contenuti e dati generati che rendono difficile il processo di acquisto e scelta dei prodotti, con conseguente insoddisfazione degli utenti. Cioè, senza la guida corretta, l'utente potrebbe fare scelte imprecise o non ottimali tra di loro. Perciò, l'uso di sistemi di raccomandazione basati su computer è diventato una necessità ed è una soluzione al problema di cui sopra. I sistemi di raccomandazione (RS) sono ampiamente utilizzati in diverse aree, compreso il commercio elettronico [1], biblioteche digitali [2], cinema [3], medicina [4], gestione delle relazioni con i clienti [5], salute [6], marketing [7] e turismo [8]. Molti siti Web utilizzano attualmente RS per fornire ai propri clienti servizi appropriati. Nella seguente figura vengono presentati alcuni motori di raccomandazione con il prodotto suggerito[81]:

[81]Rashidi, R., Khamforoosh, K., & Sheikhahmadi, A. (2020). An analytic approach to separate users by introducing new combinations of initial centers of clustering. Physica A: Statistical Mechanics And Its Applications, 551. https://doi.org/https://doi.org/10.1016/j.physa.2020.124185

Site link	Recommended items
Amazon.com	Books, Other items
YouTube.com	Videos
Netflix.com	Films
MovieLens.org	Films
IMDb.com	Films
Facebook.com	Friends
Twitter.com	Friends
LinkedIn.com	Friends
Foursquare.com	Locations
google.com/maps	Routes
Yelp.com	Restaurants, Foods, ...
eBay.com	Many Items
Last.fm	Music
Barnes & Noble.com	Books
ScienceDirect.com	Articles

Figura 19.[82] Alcuni motori di raccomandazione

Dal 1990, gli algoritmi dei sistemi di raccomandazione sono stati ampiamente utilizzati dalle aziende più differenti per aumentare i propri ricavi, sfruttando sia dati impliciti che espliciti forniti dai propri utenti al fine di migliorare l'interazione tra essi e il sistema. Tecnologie di alto livello come Google, Amazon, Facebook e Spotify utilizzano appunto sistemi di raccomandazioni e hanno sviluppato i propri algoritmi, strumenti necessari per la creazione del loro valore e vantaggio competitivo. È possibile trovare un'ampia varietà di algoritmi nella letteratura del sistema di raccomandazione per il calcolo delle previsioni e l'ottenimento di elenchi di raccomandazioni. Le prestazioni di questi algoritmi possono essere valutate mediante numerose metriche a seconda dell'obiettivo aziendale che si intende ottimizzare. Tuttavia, le prestazioni possono variare drasticamente a seconda dell'algoritmo selezionato e della metrica selezionata. Inoltre, le prestazioni algoritmiche possono essere fortemente influenzate dalla

[82]Ibidem

configurazione sperimentale come: set di dati, strategia di convalida incrociata, framework di raccomandazione e iperparametri del modello scelti.[83]

Lo scopo di questo studio è quello di eseguire un confronto sperimentale di alcune librerie Python per i sistemi di raccomandazione che utilizzano il dataset MovieLens, per rispondere alle seguenti domanda di ricerca: “Come varia l'accuratezza delle previsioni nei sistemi di raccomandazione?”

3.2 LINGUAGGIO PYTHON: OVERVIEW

La scelta del linguaggio di programmazione è una decisione fondamentale nella progettazione del software scientifico. Linguaggi di scripting di alto livello, di cui *Python* e *Octave* sono due esempi, forniscono un livello superiore di astrazione dall'architettura della macchina, consentendo allo sviluppatore di concentrarsi sull'algoritmo, riducendo i tempi di sviluppo e facilitando la prototipazione rapida di nuove idee. Forniscono inoltre un set più ricco di funzionalità, integrate nella lingua oppure disponibili tramite pacchetti facili da installare scaricati da un *repository* centrale. Tuttavia, in genere non vengono compilati in base al codice macchina e pertanto le prestazioni potrebbero non corrispondere a quelle fornite dal codice macchina di basso livello. Spesso, un nuovo pacchetto software vorrà utilizzare le librerie esistenti che forniscono routine particolarmente efficienti, ben testate e affidabili dalla più ampia comunità scientifica e / o difficili da reimplementare. In tali casi, lo sviluppatore può essere costretto a utilizzare una particolare lingua - la stessa lingua della libreria esistente - e quindi essere costretto ad accettare i costi e i benefici di quella particolare lingua. Una soluzione a questo problema è scrivere un *wrapper software* attorno alla libreria esistente, che poi espone le sue routine in modo che possa essere utilizzato dal linguaggio di programmazione prescelto. Ad esempio, i *wrapper* software possono consentire allo sviluppatore di utilizzare le librerie scritte in C, beneficiando anche della facilità di sviluppo e del ricco set di funzionalità fornite da linguaggi di alto livello come Python.[84]

Python è un linguaggio di programmazione di alto livello molto utilizzato in ambito machine learning, scritto da Guido van Rossum 1991, che si caratterizza per la sua programmazione multi-paradigma. È un linguaggio interpretato che mira a enfatizzare la leggibilità del codice

[83]A. Paullier e R. Sotelo, "Valutazione dell'algoritmo dei sistemi di raccomandazione utilizzando la libreria Lenskit e i database MovieLens," 2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), 2020, pp. 1-7, doi: 10.1109 / BMSB49480.2020.9379914.

[84]Wette, K. (2020). SWIGLAL: Python and Octave interfaces to the LALSuite gravitational-wave data analysis libraries. Softwarex, 12. https://doi.org/https://doi.org/10.1016/j.softx.2020.100634

utilizzando una sintassi semplice e ad evitare casi speciali ed eccezioni. Utilizza un sistema di tipo dinamico e supporta più paradigmi di programmazione. Gli interpreti Python sono disponibili per molti sistemi operativi e supportano l'integrazione con altri linguaggi e programmi.[85]

Python si interfaccia bene con altri linguaggi di programmazione come C, Fortran, R e Java; perciò; è un linguaggio ideale per un'interfaccia software scientifica, poiché questi linguaggi sono usati frequentemente nelle applicazioni scientifiche.

Tale linguaggio avanzato viene utilizzato da colossi del web, come Google e Youtube, ma anche da organizzazioni mondiali come la Nasa che lo sfrutta per lo sviluppo dei sistemi di controllo. Python è un linguaggio versatile, potente e di facile apprendimento con un'ampia comunità di utenti. È disponibile gratuitamente con una licenza open source e funziona su tutte le principali piattaforme informatiche. A differenza dei linguaggi di codifica tradizionali come *Fortran* o *C / C ++,* gli script Python non hanno bisogno di essere compilati, il che semplifica il loro sviluppo. Python offre una sintassi pulita e strutture dati moderne e flessibili orientate agli oggetti. Le librerie di estensione sono disponibili per un'ampia gamma di attività, incluso il calcolo scientifico.[86] La filosofia di progettazione di Python enfatizza la leggibilità del codice con il suo notevole uso di rientranze significative. I suoi costrutti linguistici e il suo approccio orientato agli oggetti mirano ad aiutare i programmatori a scrivere codice chiaro e logico per progetti su piccola e grande scala.[87]

Il linguaggio di programmazione Python sta guadagnando un'enorme popolarità tra i data scientist e gli sviluppatori di software (Robinson, 2017). Diverso dal linguaggio di programmazione R che è principalmente destinato all'analisi statistica dei dati, Python si presenta in una gamma molto più ampia di applicazioni come Internet e sviluppo di siti Web, accesso a database, GUI desktop, calcolo scientifico e sviluppo di software e giochi. Python non è un linguaggio compilato, il che significa che non precompila il codice in binario. Invece, un ambiente software, interprete Python, traduce lo script in binario durante l'esecuzione del codice in tempo reale. Con la sua distribuzione, Python viene fornito con alcune funzionalità di base ma si basa su pacchetti esterni per eseguire quasi tutti i calcoli numerici.[88]

---

[85]Guzzi, P. (2019). Computing Languages for Bioinformatics: Python. Encyclopedia Of Bioinformatics And Computational Biology, 1(195-198). https://doi.org/https://doi.org/10.1016/B978-0-12-809633-8.20366-X

[86]Wellmann, J., Croucher, A., & Regenauer-Lieb, K. (2012). Python scripting libraries for subsurface fluid and heat flow simulations with TOUGH2 and SHEMAT. Computers & Geosciences, 43, 197-206. https://doi.org/https://doi.org/10.1016/j.cageo.2011.10.011

[87]Python (programming language). The Reader Wiki, Reader View of Wikipedia. Retrieved from https://thereaderwiki.com/en/Python_(programming_language).

[88]Hao, J., & Ho, T. (2019). Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. Journal of Educational and Behavioral Statistics. Retrieved 18 May 2021, from https://doi.org/10.3102/1076998619832248.

Di conseguenza, Python è il linguaggio più utilizzato per i suoi numerosi vantaggi:

- *Linguaggio gratuito.* Non ci sono restrizioni di copyright e contiene una comunità attiva di utenti
- *Multi-paradigma.* È in grado di supportare sia programmazioni procedurali che ad oggetti.
- *Versatilità.* Riesce ad eseguire un medesimo codice su varie piattaforme, consentendo la libertà di utilizzo. Inoltre, viene utilizzato sui principali sistemi operativi.
- *Elevate performance.* Anche grazie alla possibilità di poter integrare facilmente altri linguaggi di programmazione.
- *Semplice.* Sia la sintassi che la struttura del linguaggio risultano molto semplici da imparare e confrontare con altri linguaggi, anche per coloro che lo utilizzano per la prima volta.
- *Tools ricchi.* Contiene librerie molto ricche che ne consentono un ampio utilizzo.
- *Facilmente integrabile.*
- *Memoria automatica.* La memoria viene gestita in maniera automatica, sfruttando un meccanismo di *garbage collection*, grazie all'utilizzo delle librerie standard. Così si consente di allocare correttamente le risorse, calibrando la memoria da usare.
- *Riduzione tempo di consumo.*
- *Usa codici frequenti.*
- *Sviluppo servizi web.* Consente di sviluppare vari servizi web (applicazioni software, pagine all'interno di browser web, sistemi operativi) in modo semplice.

3.3 LIBRERIE PYTHON PER RECOMMENDER SYSTEM

Storicamente, una vasta gamma di diversi linguaggi di programmazione e ambienti è stata utilizzata per consentire la ricerca sull'apprendimento automatico e lo sviluppo di applicazioni. Tuttavia, poiché negli ultimi dieci anni il linguaggio Python generico ha registrato un'enorme crescita di popolarità all'interno della comunità informatica scientifica, le più recenti librerie di machine learning e deep learning sono ora basate su Python. A parte i vantaggi del linguaggio stesso, la comunità attorno agli strumenti e alle librerie disponibili rende Python particolarmente attraente per i carichi di lavoro nella scienza dei dati, nell'apprendimento automatico e nell'elaborazione scientifica. Secondo un recente sondaggio di KDnuggets che ha intervistato più di 1800 partecipanti per le preferenze in analisi, scienza dei dati e

apprendimento automatico, Python ha mantenuto la sua posizione al vertice del linguaggio più utilizzato nel 2019.[89]

Piuttosto che avere tutte le sue funzionalità integrate nel suo nucleo, Python è stato progettato per essere altamente estensibile attraverso l'uso di moduli contenenti funzioni utili per la stesura del programma (Figura 20). Questa modularità compatta lo ha reso particolarmente popolare come mezzo per aggiungere interfacce programmabili alle applicazioni esistenti. La visione di Van Rossum di un piccolo linguaggio di base con una grande libreria standard e un interprete facilmente estensibile derivava dalle sue frustrazioni con ABC, che adottò l'approccio opposto.[90] Le librerie di Python vengono definite come "collezioni di metodi e funzioni che permettono di svolgere delle azioni senza scrivere il codice di ogni passaggio."[91] In questo modo si riesce a semplificare il codice, riducendo il numero di operazioni. Inoltre, le librerie sono utili per costruire un modello automatico di successo, grazie all'elevata quantità di elementi che contengono.

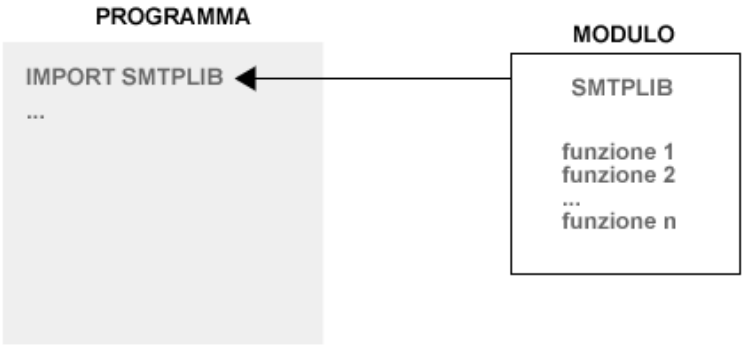


Figura 20. Esempio modulo di Python[92]

La vasta libreria standard di Python, comunemente citata come uno dei suoi maggiori punti di forza, fornisce strumenti adatti a molte attività. Per le applicazioni con connessione a Internet, sono supportati molti formati e protocolli standard come MIME e HTTP. Comprende moduli per la creazione di interfacce utente grafiche, connessione a database relazionali, generazione di numeri pseudocasuali, aritmetica con decimali di precisione arbitraria, manipolazione di

[89]Raschka, S., Patterson, J., & Nolet, C. (2020). Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. Information, 11(4), 193. MDPI AG. Retrieved from http://dx.doi.org/10.3390/info11040193

[90]Hao, J., & Ho, T. (2019). Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. Journal of Educational and Behavioral Statistics. Retrieved from https://doi.org/10.3102/1076998619832248.

[91]Nardini, D. (2019). Le 5 migliori librerie di Python per il Machine Learning. Pulp Learning - Tutto sul Machine Learning. Retrieved from https://bit.ly/2QtzPpP

[92]Minini, A. Le librerie e i moduli del linguaggio Python. Retrieved from https://bit.ly/3bChU7D.

espressioni regolari e test unitari. A partire da marzo 2021, il Python Package Index (PyPI), il repository ufficiale per il software Python di terze parti, contiene oltre 290.000 pacchetti con un'ampia gamma di funzionalità.[93] Di seguito verranno presentate alcune principali librerie di Python per lo sviluppo dei sistemi di raccomandazione tradizionali, ovvero basati sulla seguente figura:

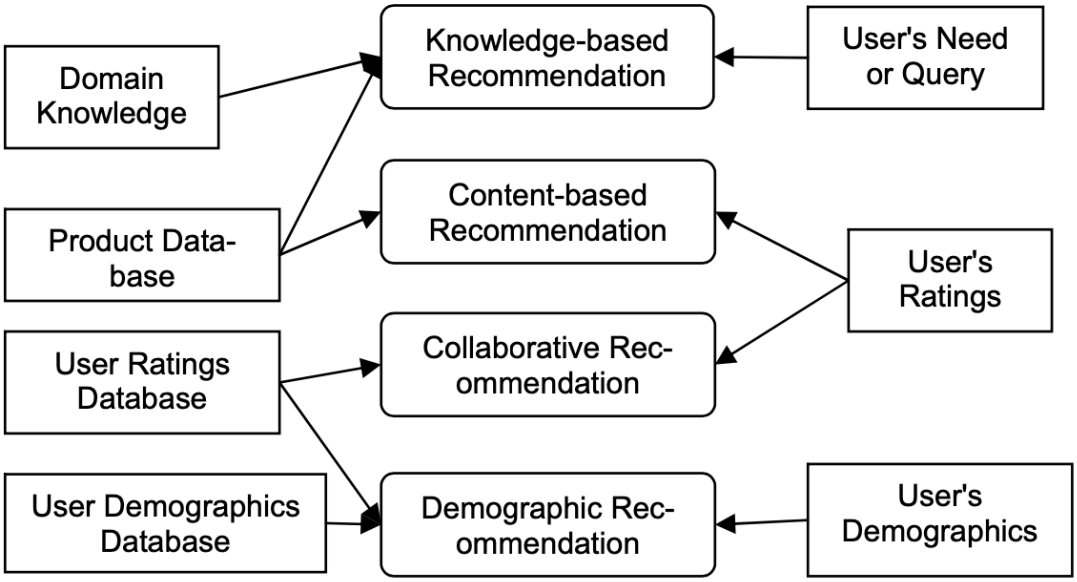


Figura 21. Tipologie di Recommender System e fonti informative[94]

3.3.1 LENSKIT

LensKit è un toolkit open source, realizzato dal gruppo di ricerca GroupLens, per la creazione, la ricerca e l'apprendimento dei sistemi di raccomandazione. Questa libreria è composta da una serie di strumenti specifici da utilizzare in algoritmi di raccomandazione con caratteristiche particolari per implementazioni modulari di diversi algoritmi, in particolare gli algoritmi di raccomandazione implementati con filtri collaborativi. Fornisce strumenti e supporto infrastrutturale per la gestione di dati e configurazioni di algoritmi, implementazioni di diversi algoritmi di filtraggio collaborativo e una suite di valutazione per condurre esperimenti offline (Ekstrand, 2019). La prima versione sviluppata è stata implementata in Java nel 2010 e successivamente è stata migrata per l'utilizzo in Python.[95] Ha supportato diverse pubblicazioni di ricerche, implementazioni di produzione su piccola scala e istruzione sia in MOOC che in

[93]Hao, J., & Ho, T. (2019). Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. Journal of Educational and Behavioral Statistics. Retrieved from https://doi.org/10.3102/1076998619832248.

[94]Recommender systems: librerie python per recommender systems tradizionali

[95]Noguera Torres, A. D. P. 2021. Sistema de recomendación de matrícula de cursos electivos para estudiantes de Ingeniería Electrónica e Ingeniería de Telecomunicaciones de la UNAD. Universidad Nacional Abierta y a Distancia UNAD, Yopal, Retrieved from https://repository.unad.edu.co/bitstream/handle/10596/40465/adnoguerat.pdf?sequence=1

ambienti di classe tradizionali. “LensKit for Python (LKPY) consente a ricercatori e studenti di costruire esperimenti robusti, flessibili e riproducibili che fanno uso dell'ampio e in crescita ecosistema PyData e Scientific Python, tra cui scikit-learn e TensorFlow.” [96] Il vantaggio di LensKit è che il suo framework contiene molti algoritmi (come funksvd e KNN) e gli utenti possono chiamare utilizzarli in base alle loro esigenze. Tuttavia, il framework LensKit può elaborare solo dati di utenti, articoli e valutazioni. Se l'utente dispone di nuovi dati, come le informazioni dell'utente o la classificazione degli elementi, il framework LensKit non può gestirli.[97]

A tal fine, fornisce implementazioni classiche di filtraggio collaborativo, metriche di valutazione del sistema di raccomandazione, routine di preparazione dei dati e strumenti per algoritmi di raccomandazione in esecuzione in batch in modo efficiente, tutti utilizzabili in qualsiasi combinazione tra loro o con altri software Python.

LensKit for Python consente la valutazione offline dell'efficacia di algoritmi nuovi e migliorati attraverso diverse funzionalità:

- Supporto per raccomandazioni e previsioni di esecuzione in blocco per il confronto con il test di dati.
- Codice di valutazione per calcolare l'accuratezza delle previsioni comuni e le metriche di massima accuratezza.
- Implementazioni di alta qualità e ben collaudate dei classici filtri collaborativi da utilizzare come linee di base.
- Interfacce facili da implementare per consentire confronti diretti tra nuovi approcci e tecniche di base.
- Facilmente utilizzabile in e con altri strumenti, come Jupyter per la creazione di report sui risultati degli esperimenti e *scikit-optimisation* per la ricerca negli iperparametri.

Queste caratteristiche sono liberamente accoppiate, quindi possono essere ricombinate, adattate e sostituite individualmente per soddisfare le esigenze di un'ampia gamma di progetti di ricerca.[98]

---

[96]Michael D. Ekstrand. 2020. LensKit for Python: Next-Generation Software for Recommender Systems Experiments. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM '20). Association for Computing Machinery, New York, NY, USA, 2999–3006. DOI: https://doi.org/10.1145/3340531.3412778

[97]Mao, Y., Mokhov, S. A., & Mudur, S. P. (2021). Application of Knowledge Graphs to Provide Side Information for Improved Recommendation Accuracy. Retrievede from https://arxiv.org/pdf/2101.03054.pdf.

[98]Michael D. Ekstrand. 2020. LensKit for Python: Next-Generation Software for Recommender Systems Experiments. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM '20). Association for Computing Machinery, New York, NY, USA, 2999–3006. DOI:https://doi.org/10.1145/3340531.3412778

3.3.2 CRAB

Crab – in precedenza conosciuto come scikits.recommender - è un motore di raccomandazione per Python, flessibile e veloce. Crab è un framework Python per la creazione di motori di raccomandazione integrati con i pacchetti scientifici Python (numpy, scipy, matplotlib). Ha il supporto per algoritmi di raccomandazione come il filtraggio collaborativo basato sugli utenti e sugli elementi. Le interfacce di raccomandazione possono essere facilmente combinate con più di dieci metriche diverse, come coseno, tanimoto, pearson, euclidean using, Scipy e Numpy.[99]

3.3.3 SURPRISE

Surprise (Simple Python RecommendatIon System Engine) è uno scikit Python per la creazione e l'analisi di sistemi di raccomandazione che gestiscono dati di valutazione espliciti. È stato progettato per:

- Offrire agli utenti un controllo accurato sugli esperimenti.
- Fornire una raccolta di stimatori (o algoritmi di previsione) per la previsione della valutazione (come algoritmi di base).
- Semplificare il problema della gestione dei dati in un set, in quanto si possono utilizzare sia dati di set incorporati che di set personalizzati.
- Implementare algoritmi classici come i principali algoritmi basati sulla somiglianza, nonché algoritmi basati sulla fattorizzazione matriciale come SVD o NMF.
- Grazie a semplici primitive e un'API leggera, gli utenti possono anche implementare la propria tecnica di raccomandazione con una quantità minima di codice.
- Fornire strumenti utili per l'analisi e il confronto delle prestazioni di algoritmi.[100]

Surprise è stato progettato per essere utile ai ricercatori che desiderano esplorare rapidamente nuove idee di raccomandazione supportando la creazione di algoritmi di previsione personalizzati, ma può anche servire come risorsa di apprendimento per studenti e utenti meno esperti grazie alla sua documentazione dettagliata.[101]

[99] Dwivedi, S., & Roshni, V. K. (2017, August). Recommender system for big data in education. In 2017 5th National Conference on E-Learning & E-Learning Technologies (ELELTECH) (pp. 1-4). IEEE. Retrieved from https://ieeexplore.ieee.org/abstract/document/8074993

[100] Approfondimento su: http://surpriselib.com/

[101] Hug, N., (2020). Surprise: A Python library for recommender systems. Journal of Open Source Software, 5(52), 2174. https://doi.org/10.21105/joss.02174

3.3.4 REXY

Rexy è un vero e proprio sistema di raccomandazione open source, basato su un concetto generale di tag-prodotto-utente e una struttura flessibile, progettata per essere adattabile con vari schemi di dati. Esistono molti metodi e modi utilizzati da Rexy per consigliare i prodotti agli utenti. Ciò include consigli generali come i migliori prodotti, consigli basati su eventi e nuovi prodotti a cui l'utente potrebbe essere interessato. Ci sono altri consigli che sono direttamente correlati alle attività dell'utente o altri utenti che hanno un comportamento simile a un determinato utente.[102]

3.3.5 TENSORFLOW

TensorFlow, originariamente sviluppato da Google, è un sistema di machine learning e uno strumento open source che consente di creare, ottimizzare e distribuire sistemi di apprendimento automatico di grandi dimensioni e arbitrari. In TensorFlow, un processo di apprendimento automatico è espresso come un "grafico" che mostra come i dati fluiscono attraverso il sistema, per rappresentare il calcolo, lo stato condiviso e le operazioni che modificano quello stato. Mappa i nodi di un grafico del flusso di dati su molte macchine in un cluster e all'interno di una macchina su più dispositivi computazionali, comprese CPU multicore, GPU generiche e ASIC progettati su misura noti come TPU (Tensor Processing Unit). Questa architettura offre flessibilità allo sviluppatore dell'applicazione: mentre nei precedenti progetti di "server dei parametri" la gestione dello stato condiviso è incorporata nel sistema, TensorFlow consente agli sviluppatori di sperimentare nuove ottimizzazioni e algoritmi di addestramento. Supporta una varietà di applicazioni, con particolare attenzione all'addestramento e all'inferenza su reti neurali profonde.[103] I programmi TensorFlow sono suscettibili a diversi errori sistematici, specialmente nell'impostazione di digitazione dinamica di Python.

3.3.6 TENSORREC

TensorRec è un sistema di raccomandazione Python che consente di sviluppare rapidamente algoritmi di raccomandazione e personalizzarli utilizzando TensorFlow.

È possibile personalizzare le funzioni di rappresentazione (o anche incorporamento) del proprio sistema di suggerimenti e di perdita, mentre TensorRec gestisce la manipolazione dei dati, il punteggio e la classificazione per generare consigli. È un algoritmo di raccomandazione che utilizza una semplice API esterna per l'addestramento e la previsione che assomiglia ai comuni

[102] Approfondimenti su: https://github.com/kasravnd/Rexy

[103] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., & Dean, J. et al. (2016). TensorFlow: A System for Large-Scale Machine Learning. Usenix.org. Retrieved from https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi.

strumenti di machine learning in Python. Ti offre anche la flessibilità di sperimentare le tue funzioni di rappresentazione e perdita, permettendoti di creare un sistema di consigli su misura per comprendere i tuoi particolari utenti e oggetti.

Un sistema come questo consuma tre pezzi di dati per imparare a formulare e classificare i consigli: user_features, item_features, e interactions. Le interazioni vengono utilizzate durante l'adattamento del modello: le previsioni vengono confrontate con le interazioni e viene calcolata una perdita / penalità, che il sistema impara a diminuire.

L'algoritmo assegna un punteggio ai consigli sfruttando le caratteristiche dell'utente e dell'oggetto (come, ad esempio, ID e tag) e costruisce due vettori a bassa dimensione, uno riguarda la "rappresentazione dell'utente" e l'altra la "rappresentazione dell'oggetto" (come mostrato in Figura 22). Calcolando il prodotto scalare tra questi due vettori, risulta la relazione tra l'utente e l'elemento. A punteggi più elevati si associano consigli migliori. È possibile personalizzare tale algoritmo: può essere applicato qualsiasi cosa, da una trasformazione lineare diretta a una rete neurale profonda, a seconda dei dettagli del problema che TensorRec deve risolvere e dei dati della caratteristica hai a disposizione. Inoltre, le funzioni di rappresentazione dell'utente e dell'oggetto possono essere personalizzate in modo indipendente.[104]

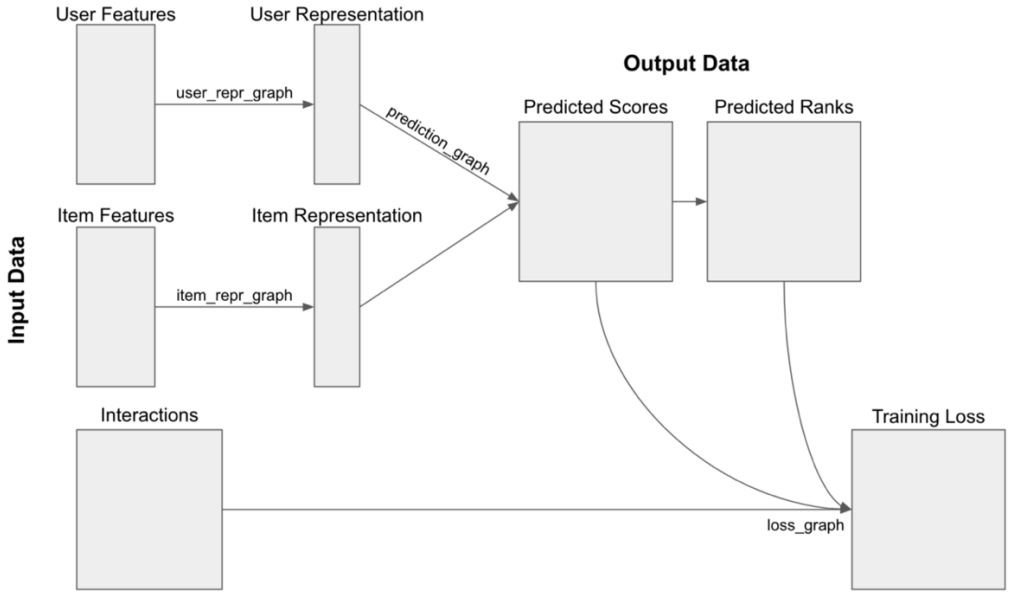


Figura 22. Diagramma di sistema[105]

[104]Kirk J., 2017. TensorRec: A Recommendation Engine Framework in TensorFlow. Medium. Retrieved from https://medium.com/hackernoon/tensorrec-a-recommendation-engine-framework-in-tensorflow-d85e4f0874e8

[105]Ibidem

3.3.7 LIGHTFM

LightFM è un'implementazione Python di una serie di algoritmi di raccomandazione popolari per feedback sia implicito che esplicito, inclusa l'implementazione efficiente delle perdite di ranking BPR e WARP. È facile da usare, veloce (tramite stima del modello multithread) e produce risultati di alta qualità. È un modello ibrido che consente inoltre di incorporare metadati sia degli elementi che degli utenti nei tradizionali algoritmi di fattorizzazione a matrice. Rappresenta ogni utente e oggetto come la somma delle rappresentazioni latenti delle loro caratteristiche, consentendo così la generalizzazione dei consigli a nuovi elementi (tramite le caratteristiche dell'oggetto) e ai nuovi utenti (tramite le caratteristiche dell'utente).[106]

LightFM genera rappresentazioni di utenti e oggetti funzionando come una macchina di fattorizzazione e apprendendo incorporamenti lineari per ciascuna funzionalità. Prendendo il prodotto scalare di questi due vettori di rappresentazione, ottieni un punteggio senza unità che, se classificato, ti dice quanto sarebbe buona (o cattiva) una data corrispondenza utente-elemento.[107]

3.3.8 CASE RECOMMENDER

Case Recommender implementa diversi algoritmi di raccomandazione popolari per feedback, sia implicito che esplicito, che mirano a fornire un ricco set di componenti utili per costruire un RS personalizzato da un insieme di algoritmi.[108] Case Raccomender è stato sviluppato per fornire flessibilità ed estensibilità negli ambienti di ricerca, pur mantenendo alte prestazioni, fornendo una varietà di raccomandazioni e algoritmi di clustering, nonché funzioni per la manipolazione dei dati. Contiene un'interfaccia generica per implementazioni di sistemi di raccomandazione, tra cui gli approcci di filtraggio collaborativo e basato sui contenuti come i modelli basati sul vicinato (NB) e sulla fattorizzazione di matrice (MF), che sono già disponibili per l'uso. Inoltre, sono disponibili due algoritmi di clustering: PaCo e KMedoids, per supportare i motori di raccomandazione. Il vantaggio principale di Case Recommend è la possibilità di integrare algoritmi di clustering e ensemble con motori di raccomandazione, facilitando lo

[106]Choudhury, A. (2020). Top Open Source Recommender Systems In Python For Your ML Project. Analytics India Magazine. Retrieved from https://analyticsindiamag.com/top-open-source-recommender-systems-in-python-for-your-ml-project/.

[107]Kirk, J. (2017). TensorRec: A Recommendation Engine Framework in TensorFlow. Medium. Retrieved from https://medium.com/hackernoon/tensorrec-a-recommendation-engine-framework-in-tensorflow-d85e4f0874e8.

[108]Ortega, F., Mayor, J., López-Fernández, D., & Lara-Cabrera, R. (2021). CF4J 2.0: Adapting Collaborative Filtering for Java to new challenges of collaborative filtering based recommender systems. Knowledge-Based Systems, 215. https://doi.org/https://doi.org/10.1016/j.knosys.2020.106629

sviluppo di approcci più accurati ed efficienti. Inoltre, consente una facile integrazione con diversi domini di input come database, file di testo o librerie Python.[109]

3.3.9 SPOTLIGHT

Spotlight è un sistema di raccomandazione Python che utilizza PyTorch per creare modelli di raccomandazioni sia profondi che superficiali. Fornendo una serie di elementi costitutivi per le rappresentazioni delle funzioni di perdita e utilità per il recupero o la generazione di set di dati di raccomandazione, Spotlight mira a essere uno strumento per l'esplorazione rapida e la prototipazione di nuovi modelli di raccomandazione. Spotlight offre una serie di set di dati popolari, tra cui Movielens 100K, 1M, 10M e 20M. Incorpora anche utilità per la creazione di set di dati sintetici.[110]

3.4 IL DATASET MOVIELENS: ORIGINI E CARATTERISTICHE

Sono disponibili numerosi set di dati per la ricerca di raccomandazioni. Tra questi, il set di dati *MovieLens* è probabilmente uno dei più popolari. MovieLens è un sistema di raccomandazione di film basato sul web non commerciale. È stato creato nel 1997 e gestito da GroupLens, un laboratorio di ricerca presso l'Università del Minnesota, al fine di raccogliere dati sulla classificazione dei film a scopo di ricerca. I set di dati MovieLens sono ampiamente utilizzati nell'istruzione, nella ricerca e nell'industria. Vengono scaricati centinaia di migliaia di volte all'anno, riflettendo il loro utilizzo nei libri di programmazione della stampa popolare, nei corsi tradizionali e online e nel software.[111]

I set di dati MovieLens, rilasciati per la prima volta nel 1998, descrivono le preferenze espresse dalle persone per i film. Queste preferenze assumono la forma di tuples[112] *<utente, elemento, valutazione, data e ora>*, ciascuna del risultato di una persona che esprime una preferenza (una valutazione da 0 a 5 stelle) per un film in un determinato momento. Queste preferenze sono state inserite tramite il sito web MovieLens1, un sistema di consigli che chiede ai suoi utenti di fornire valutazioni sui film per ricevere consigli sui film personalizzati. I set di dati di MovieLens sono ampiamente scaricati (oltre 140.000 download[2] nel 2014) e citati nella

[109]Campello, R., da Costa, A., Fressato, E., Neto, F., & Manzato, M. (2018). Case recommender | Proceedings of the 12th ACM Conference on Recommender Systems. Dl.acm.org. Retrieved 23 May 2021, from https://dl.acm.org/doi/pdf/10.1145/3240323.3241611.

[110]Choudhury, A. (2020). Top Open Source Recommender Systems In Python For Your ML Project. Analytics India Magazine. Retrieved from https://analyticsindiamag.com/top-open-source-recommender-systems-in-python-for-your-ml-project/.

[111]Colab. The MovieLens Dataset. Colab.research.google.com. Retrieved from https://bit.ly/3vaJ1ON.

[112]Per *tuple* si intende una lista immutabile di valori separati da virgole. Approfondimenti su https://www.python.it/doc/Howtothink/Howtothink-html-it/chap09.htm

letteratura di ricerca (oltre 7.500 riferimenti a "movielens" in GoogleScholar). Questa popolarità è, in una certa misura, un riflesso dell'incredibile tasso di crescita della personalizzazione e della ricerca sulle raccomandazioni, in cui set di dati come questi hanno un valore sostanziale nell'esplorare e convalidare le idee. La popolarità potrebbe anche essere attribuita alla flessibilità dei dati sulle valutazioni, che è naturalmente adatta non solo alla tecnologia di raccomandazione, ma anche a una famiglia più generale di tecniche scientifiche dei dati che si occupano di riepilogo, identificazione dei modelli e visualizzazione. Inoltre, perché le preferenze dei film sono altamente soggette a gusti personali, il dominio del film è adatto per testare la tecnologia di personalizzazione. Infine, la popolarità può riflettere l'accessibilità percepita dei film come un dominio di contenuto: i film sono un interesse comune, rendendo l'output algoritmico facile da discutere.[113]

I set di dati MovieLens sono il risultato dell'interazione degli utenti con il sistema online recommender di MovieLens nel corso degli anni. Come con la maggior parte dei sistemi online dinamici e longevi, MovieLens ha subito molti cambiamenti, sia nel design che nella funzionalità. Questi cambiamenti hanno necessariamente un impatto sulla generazione delle valutazioni: gli utenti valutano solo i film che appaiono sui loro schermi; le valutazioni degli utenti sono anche influenzate dalle valutazioni stesse previste (Cosley et al. 2003).

La prima versione di MovieLens 1997 è stata sviluppata per guardare come l'interfaccia EachMovie che stava sostituendo. Quest'ultimo utilizzava un algoritmo di raccomandazione proprietario; Movie-Lens ha invece incorporato il filtraggio collaborativo utente-utente (CF) implementato nel sistema di raccomandazione di notizie usenet di GroupLens (Konstan et al. 1997). L'uso di MovieLens è aumentato in modo significativo alla fine del 1999, quando ha ricevuto l'attenzione dei mass media. La crescita del sistema è stata notevolmente stabile, soprattutto considerando la quasi totale assenza di sforzi di marketing. MovieLens ha una media di 20-30 iscrizioni di nuovi utenti ogni giorno per molto tempo, che sono in gran parte il risultato del passaparola o della stampa non richiesta. L'analisi condatta da Harper et al., 2015 mostrata in figura 23, evidenzia la crescita del dataset dal suo lancio fino al 2015.[114]

[113] F. Maxwell Harper e Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Trans. Interagire. Intell. Syst. VOl. 5, n. 4, articolo 19 (gennaio 2016), 19 pagine. DOI: https: //doi.org/10.1145/2827872

[114] Ibidem

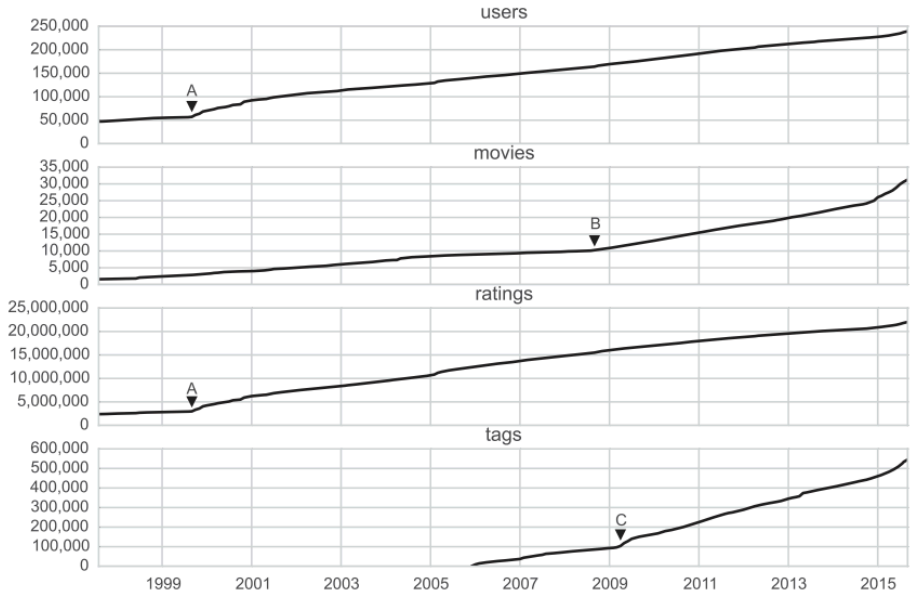


Figura 23. Crescita di Movielens dal 1999 al 2015[115]

Sono stati rilasciati quattro set di dati MovieLens, noti come 100k, 1m, 10m e 20m, che riflettono il numero approssimativo di valutazioni in ciascun set di dati (Figura 24). Questi set di dati sono aumentati di dimensioni insieme al sistema. Con il rilascio del set di dati da 20 milioni, GroupLens ha iniziato a ospitare set di dati aggiuntivi non archiviati (una versione integrale per la completezza con una versione ridotta per la velocità) che vengono aggiornati periodicamente per includere film aggiornati: ultimi e ultimi-piccoli.

Name	Date Range	Rating Scale	Users	Movies	Ratings	Tag Apps	Density
ML 100K	9/1997–4/1998	1–5, stars	943	1,682	100,000	0	6.30%
ML 1M	4/2000–2/2003	1–5, stars	6,040	3,706	1,000,209	0	4.47%
ML 10M	1/1995–1/2009	0.5–5, half-stars[a]	69,878	10,681	10,000,054	95,580	1.34%
ML 20M	1/1995–3/2015	0.5–5, half-stars[a]	138,493	27,278	20,000,263	465,564	0.54%

Figura 24. Riassunto quantitativo del “MovieLens Ratings Datasets”

Le valutazioni e altri dati vengono attribuiti a ID utente anonimi (questi ID non vengono mappati tra i set di dati). I film sono elencati insieme ai loro titoli in MovieLens, insieme a zero o più generi (gli ID dei film vengono mappati tra i set di dati). Sono inclusi solo gli utenti con almeno 20 valutazioni. Questi quattro set di dati differiscono nei metodi di campionamento:

- 100k: richiesti anche i dati demografici completi ((età, sesso, occupazione, codice postale);

[115] Una panoramica della crescita di movielens.org in 17 anni, annotata con gli eventi A, B e C. La registrazione degli utenti e l'attività di valutazione mostrano una crescita stabile in questo periodo, con un'accelerazione dovuta alla copertura mediatica (A). Il tasso di film aggiunti a MovieLens è cresciuto (B) quando il processo è stato aperto alla comunità. È evidente un'accelerazione nelle applicazioni di tag con il rilascio della funzione "espressione di tag" (C).

- 1m: utenti uniti al sistema nel 2000, partendo da una serie limitata di limitazioni dal 2001 al 2002. Campione raccolto nel 2003, dove vengono richiesti i dati demografici completi degli utenti (età, sesso, occupazione, codice postale);
- 10m e 20m: utenti casuali selezionati. Non richiedono nessuna informazione demografica, ma includono applicazioni di tag (i set di dati precedenti sono stati rilasciati prima che esistesse la funzione di tagging). MovieLens 20m include anche una tabella che associa gli ID dei film MovieLens agli ID dei film in due siti esterni, per consentire agli utenti dei set di dati di creare rappresentazioni più complete basate sul contenuto. Il risultato ottenuto consiste in un set di dati di persone reali che interagiscono in un sistema in continua evoluzione.

Questi metodi di raccolta hanno creato set di dati che riflettono le raffiche di attività di classificazione avvenute in FilmLens.

Alcune delle modifiche al design, ad esempio, il passaggio a valutazioni a mezza stella, hanno un effetto direttamente misurabile sui contenuti dei set di dati. Altre modifiche, ad esempio, l'aggiunta di un minimo di 15 valutazioni al processo di registrazione, possono avere un effetto maggiore un'influenza sottile. Modifiche come queste potrebbero aver portato diversi utenti a raggiungere il minimo di 20 valutazioni o a valutare diversi tipi di film. Queste sottili differenze sono un artefatto necessario per la generazione di set di dati da un sistema di lunga durata e in continua evoluzione.[116] Di conseguenza, MovieLens riesce a trovare i film che possono piacere all'utente e li valuta per realizzare un profilo personalizzato, così da riuscire a suggerire ulteriori film da guardare (Figura 25).

[116]F. Maxwell Harper e Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Trans. Interagire. Intell. Syst. VOl. 5, n. 4, articolo 19 (gennaio 2016), 19 pagine. DOI: https: //doi.org/10.1145/2827872

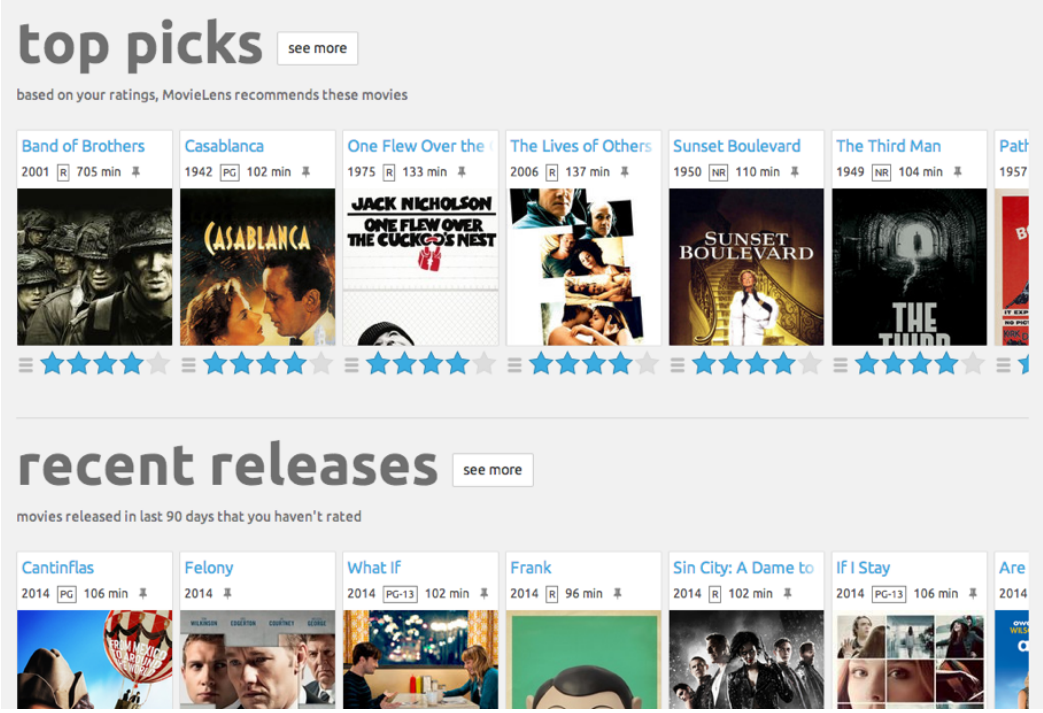


Figura 25. Esempio di homepage MovieLens[117]

Dopo aver selezionato il film che si vuole guardare, il sistema fornisce all'utente ulteriori informazioni sui film con dati, immagini e trailer. Si sfogliano i film in base ai tag applicati dalla comunità oppure il sistema applica i tag dell'utente, esplorando il database con strumenti espressivi di ricerca (Figura 26). È possibile anche regolare l'algoritmo di corrispondenza (esempio: più romantico, meno violento, più realistico), in modo che i risultati siano ancora più in linea con le proprie preferenze.

[117] Approfondimento su: https://movielens.org/

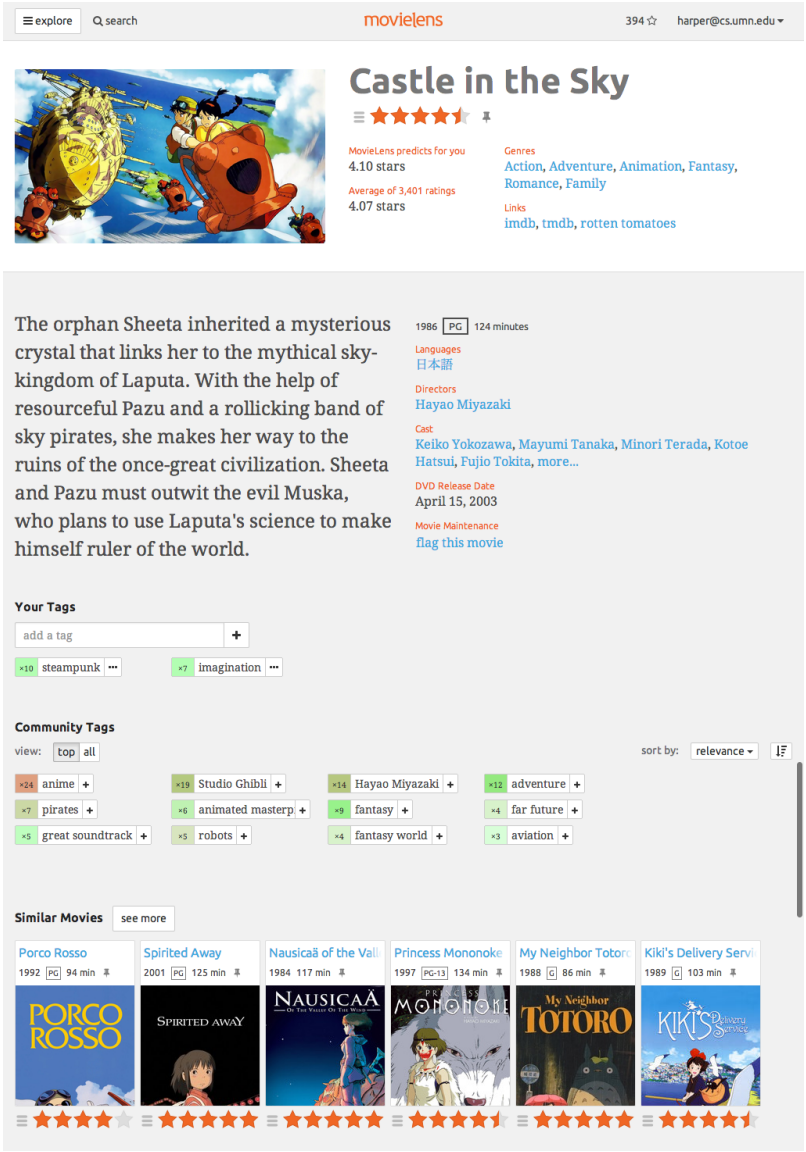


Figura 26. Esempio di schermata MovieLens[118]

I meccanismi di ricerca, filtraggio e ordinamento influenzano i film che un utente può valutare, creando una conseguente modifica sulla forma del set di dati. L'esperienza principale dell'utente in MovieLens ruota attorno a un ciclo di feedback tra la valutazione dei film e la visualizzazione dei consigli sui film: l'utente visualizza gli elementi in un elenco di consigli e valuta tali elementi, il che a sua volta altera gli elementi mostrati nelle successive visualizzazioni di pagina. Questo ciclo di valutazione / raccomandazione è abilitato dagli algoritmi CF (Resnick et al. 1994). MovieLens, fino al rilascio della versione 4, ordinava elenchi di film rigorosamente basati su valori di valutazione previsti personalizzati dall'utente, l'output dei suoi algoritmi di filtraggio collaborativo sottostanti. In altre parole, i film che appaiono per primi nei risultati di ricerca di un utente sono quelli che l'algoritmo prevede che l'utente prenderà maggiormente in considerazione. La quarta versione combina un fattore di popolarità con la valutazione prevista, per ordinare gli elenchi di raccomandazioni (Ekstrand et al. 2015; Harper et al. 2015).

[118] Ibidem

L'algoritmo stesso è cambiato nel tempo, il che ha avuto effetti per lo più non studiati sulla soddisfazione degli utenti e sui tassi di contribuzione.

Dal lancio di MovieLens, le valutazioni sono state espresse come un valore "stella", che è un modello standard dell'interfaccia utente per consentire agli utenti di inserire le preferenze. Il cambiamento più grande all'interfaccia delle valutazioni è avvenuto con il lancio della versione 3 (febbraio 2003), quando l'interfaccia è passata da valutazioni a "stella intera" a "mezza stella", la funzione più richiesta in un sondaggio tra gli utenti, raddoppiando la gamma dei valori di preferenza da cinque (1–5) a dieci (0,5–5,0). Le versioni dalla v0 alla v3 utilizzavano due elementi visivi separati per il widget delle valutazioni. L'elemento di visualizzazione era un'immagine che mostrava un certo numero di stelle, il colore che rappresentava la previsione o la valutazione effettiva. Con il rilascio della v4, l'interfaccia utente ha combinato questi due elementi in un'unica rappresentazione a cinque stelle che accetta eventi di tocco / clic. A un livello di analisi approssimativo non ci sono stati cambiamenti globali visibili nelle distribuzioni dei rating, risultanti da tali modifiche dell'interfaccia.[119]

È possibile scaricare il dataset dal seguente sito: *http://grouplens.org/datasets/*.

A prima vista il set di dati, si presenta suddiviso in:

- *movies.csv*: questa è la tabella che contiene tutte le informazioni sui film, inclusi titolo, slogan, descrizione, ecc. Ci sono 21 caratteristiche/colonne in totale, quindi i candidati possono concentrarsi solo su alcune di esse o provare a utilizzarle tutte. Rispetta la struttura: *movieId,title,genres.*
- *rating_small .csv*: una tabella che registra tutti i comportamenti di valutazione degli utenti, coprendo le loro tariffe e il timestamp (momento in cui l'utente esprime la valutazione) quando hanno pubblicato le tariffe. Ogni riga consiste in un rating di un utente per un determinato film e si presenta in questo modo: *userId, movieId, rating, timestamp.*
- *links.csv*: una tabella che registra l'ID univoco di ogni film su due rispettivi database: IMDB e TMDB.
- *tags.csv*: Ogni riga rappresenta un tag, ovvero una parola o una frase breve, che viene assegnato da un utente per un film per descriverlo. La struttura si presenta come: *userID, movieID, tag, timestamp.*[120]

[119] F. Maxwell Harper e Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Trans. Interagire. Intell. Syst. VOl. 5, n. 4, articolo 19 (gennaio 2016), 19 pagine. DOI: https: //doi.org/10.1145/2827872

[120] Meng, M. (2020). Movie Recommendation System based on MovieLens. Towards data science. Retrieved from https://towardsdatascience.com/movie-recommendation-system-based-on-movielens-ef0df580cd0e.

3.5 IL DATASET MOVIELENS: IMPLICAZIONI MANAGERIALI

Già dalla sua creazione, MovieLens è stata una piattaforma di ricerca che ha riscontrato elevata visibilità ed è sempre più riconosciuta sia dai ricercatori che dai finanziatori della ricerca. La possibilità di avere tali set di dati condivisi crea un impatto positivo sostanziale sulla ricerca, l'istruzione e la pratica. Pubblicando set di dati che coprono più di 20 anni di utilizzo di un unico sistema in evoluzione, i set di dati MovieLens sono stati in grado di dare un contributo distintivo nel campo dei sistemi di raccomandazione e nei campi correlati. Si caratterizza per essere senza scopi di lucro, ma con finalità di ricerca. Ogni utente può contribuire agli studi futuri entrando a far parte della community.

Per quanto riguarda la ricerca, il sistema vanta moltissimi contributi di vario genere. Tra questi ricordiamo:

- Nella primavera 2015, una ricerca di "movielens" ha prodotto 2.750 risultati in Google Libri e 7.580 risultati in Google Scholar.
- I set di dati MovieLens sono stati utilizzati per sviluppare e testare molti dei principali progressi algoritmici nei sistemi di raccomandazione, incluso il filtraggio collaborativo item-item (Sarwar et al. 2001; Karypis 2001; Deshpande e Karypis 2004), filtraggio collaborativo per la riduzione della dimensionalità (Sarwar et al. 2000), raccomandazioni basati sulla fiducia (O'Donovan e Smyth 2005; Massa e Avesani 2007), che consigliano gruppi di articoli in rapida evoluzione (Das et al. 2007) e algoritmi di avviamento a freddo.
- Alcune università come quella del Michigan, Minnesota, Pitsburg e Carnegie Mellon, hanno realizzato uno studio collaborativo utilizzando i dati di MovieLen, dal titolo "Using Social Psychology to Motivate Contributions to Online Communities"
- I set di dati continuano a essere utilizzati attivamente oggi (20.500 risultati di ricerca in Google Scholar risalgono al maggio 2021) e si prevede che possa aiutare ancora molti altri ricercatori.

L'ambito dell'istruzione e della pratica è strettamente correlato a quello della ricerca e tra le scoperte più significative si ricordano[121]:

- Nel 2014, i set di dati sono stati scaricati più di 140.000 volte dal sito Web di GroupLens (http://grouplens.org/datasets/movielens). Sebbene alcuni di questi download riflettano l'uso della ricerca, si ritiene che la stragrande maggioranza di essi sia per scopi educativi. Esempio da docenti che utilizzano i set di dati nei loro corsi, da studenti che utilizzano i

[121] F. Maxwell Harper e Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Trans. Interagire. Intell. Syst. VOl. 5, n. 4, articolo 19 (gennaio 2016), 19 pagine. DOI: https: //doi.org/10.1145/2827872

set di dati in compiti o studio autonomo, e da autori di libri di testo e altri che utilizzano il set di dati nei loro materiali.

- I set di dati hanno trovato una nicchia come dati di prova per il codice di esempio nella programmazione della stampa popolare e nei libri di scienza dei dati. Ad esempio, i set di dati sono stati utilizzati per dimostrare la costruzione di un algoritmo di filtraggio collaborativo item-item in Python (Segaran 2007), nonché per dimostrare il caricamento, l'esplorazione e la visualizzazione dei dati in una piattaforma di elaborazione distribuita ad alte prestazioni (Pentreath 2015).
- Molte aziende utilizzano i set di dati MovieLens per testare i propri algoritmi e sistemi di raccomandazione, per confrontare sistemi esterni e per scopi di formazione e dimostrazione.
- Grazie alle valutazioni dei film da parte degli utenti, si crea un grande valore per gli enti commerciali interessati al marketing dei film, che possono sfruttare l'ampia scala delle valutazioni come un "set di dati di base". Ciò consente, di conseguenza, di aiutare i propri clienti in maniera più efficace nelle applicazioni di personalizzazione.
- Chen, Harper, Konstan e Li nel 2007 hanno pubblicato i risultati di un esperimento sul campo su MovieLens. L'indagine su MovieLens concentra il suo interesse sull'applicazione della teoria del confronto sociale. Il concetto alla base di questa teoria è che le persone valutano sé stesse conducendo un confronto con altre persone. In particolare, tendiamo a paragonarci ad altri che sono migliori di noi come guida e ad altri che stanno peggio per migliorare la nostra autostima (Festinger 1954). Esiste un'ampia letteratura sull'esistenza che prova l'influenza dei confronti sociali sul comportamento umano, specialmente quando le persone hanno bisogno di trovare il "comportamento giusto" in situazioni ambigue (Buunk, Mussweiler 2001; Suls, Martin, Wheeler 2002). Per riassumere, hanno dimostrato che se sono disponibili informazioni sociali, aumentano i contributi al forum online. Lo hanno fatto inviando agli utenti un'e-mail che fornisce o il numero mediano di valutazioni o il punteggio di vantaggio netto di un utente medio.[122]

[122] Chen Y., Harper F.M., Konstan J., Xin Li S. (2010), "Social comparisons and contributions to online communities: a field experiment on MovieLens", The American Economic Review, Vol. 100, n.4. Retrieved from https://yanchen.people.si.umich.edu/papers/soc_pref_movielens_20090619_aer.pdf

3.6 CONFRONTO SPERIMENTALE: METODOLOGIA

In questo paragrafo viene illustrata la metodologia eseguita per effettuare un confronto sperimentale su alcune librerie Python, partendo da un medesimo dataset di riferimento: MovieLens (https://grouplens.org/datasets/movielens/). Nello specifico, per realizzare i sistemi di raccomandazione è stato utilizzato per sviluppo e test il dataset MovieLens100K contenente 100 mila righe, composto da: 100.000 valutazioni (da 1 a 5) effettuate da 943 utenti su 1682 film; e il dataset MovieLens1M, contenente 1.000.209 valutazioni anonime di circa 3.900 film realizzato da 6.040 utenti che si sono uniti a MovieLens nel 2000.
Le librerie Python prese in considerazioni ed analizzate per questo studio sono: LightFM, Surprise, LensKit e TensorFlow.
Il codice sorgente di tutte le analisi seguono i seguenti passi:

1. Installazione e/o import di librerie necessarie;
2. Caricamento del dataset;
3. Suddivisione del dataset:
   - 80% per il *training* dell'algoritmo/modello
   - 20% per il *test* del modello sui nuovi dati, per capire come lo stesso si comporta sui dati nuovi e valutarne la sua capacità di generalizzazione dei pattern appresi durante la fase di training;
4. Check della dimensionalità dei dati dopo lo splitting: *TRAINING* + *TEST* = dimensione *DATASET* iniziale;
5. Addestramento del modello sui dati di training;
6. Calcolo di performance metrics sui dati di test.

La suddivisione del dataset iniziale MovieLens segue la seguente impostazione:

- *Shuffle*, ovvero mescolamento del dataset iniziale in modo tale da avere un set di dati quanto più randomico possibile in termini di ordinamento (ad esempio per mescolare l'identificativo utente).
- *Training e test.* Il primo 80% del dataset viene utilizzato per il training, il rimanente 20% per il test dell'algoritmo. Successivamente il sistema "acquista" conoscenza e apprende gli *"hidden patterns"* tra i dati di training. Tale metodologia e la suddivisione in percentuale è abbastanza diffusa in letteratura, anche in maniera standard. In alcuni casi è anche possibile trovare 70% training e 30 test. La percentuale più alta viene utilizzata come training set, perché il modello/algoritmo ha bisogno di più esempi dai quali imparare e un numero maggiore di dati porta ad una maggiore precisione del modello. Inoltre, con una percentuale più elevata di dati è possibile evitare il fenomeno del "overfitting", in quanto si riduce la possibilità di un eccesso di adattamento (Figura 27).

Lo scopo è quello di fare in modo che il modello generalizzi al meglio quanto appreso su nuovi dati.

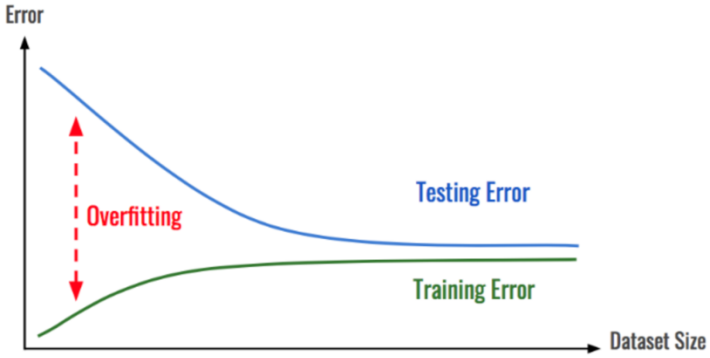


Figura 27. Esempio di Overfitting[123]

La fase successiva per la creazione del modello riguarda l'utilizzo della funzione *predict*. Tale passaggio riguarda il momento in cui il modello mette in prova la sua conoscenza sui dati di test, applicando il sistema di raccomandazione appena implementato sul nuovo set. Le predizioni ottenute, in questo caso, riguardano *ratings* dei vari film per tutti gli utenti (overall del sistema). Per quantificare la bontà di questi outputs e valutarne l'accuratezza delle predizioni, sono state utilizzate due differenti metriche, in base alla libreria in esame:

- *RMSE (Root Mean Square Error)*: Si utilizza per misurare l'errore di un modello nella previsione dei dati quantitativi. Viene definito e calcolato con la seguente formula:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

  Dove $\hat{y}_1, \hat{y}_2, \ldots \hat{y}_n$ sono i valori predetti; $y_1, y_2, \ldots y_n$ sono i valori osservati e n corrisponde al numero di osservazioni.

  Nel *data science*, la metrica RMSE non serve solo per valutare le prestazioni, ma è utile anche per comprendere se il modello di riferimento può effettivamente essere utile per risolvere un problema. Valutare se RMSE è sufficientemente piccolo o meno dipenderà da quanto è necessario che il modello sia accurato per lo studio. In questo caso, verrà valutato positivamente il modello con accuratezza più elevata.[124]

- *AUC (Area Under The Curve)*: Rappresenta il grado o la misura di separabilità della curva di probabilità. L'AUC misura l'intera area bidimensionale sotto l'intera curva ROC (Figura 28), ovvero il grafico che mostra le prestazioni di un modello di classificazione a

[123]https://www.lorenzogovoni.com/overfitting-e-underfitting-machine-learning/
[124]Approfondimenti su https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e

tutte le soglie di classificazione. L'idea di base è il ragionamento del calcolo integrale da (0,0) a (1,1). Più alto è il suo valore, migliore sarà il modello nel prevedere gli 0 come 0 e gli 1 come 1. Di conseguenza maggiore è l'AUC, migliore è il modello.[125] L'AUC ha un valore compreso tra 0 e 1. Un modello le cui previsioni sono errate al 100% ha un AUC di 0,0; uno le cui previsioni sono corrette al 100% ha un AUC di 1.0.[126]

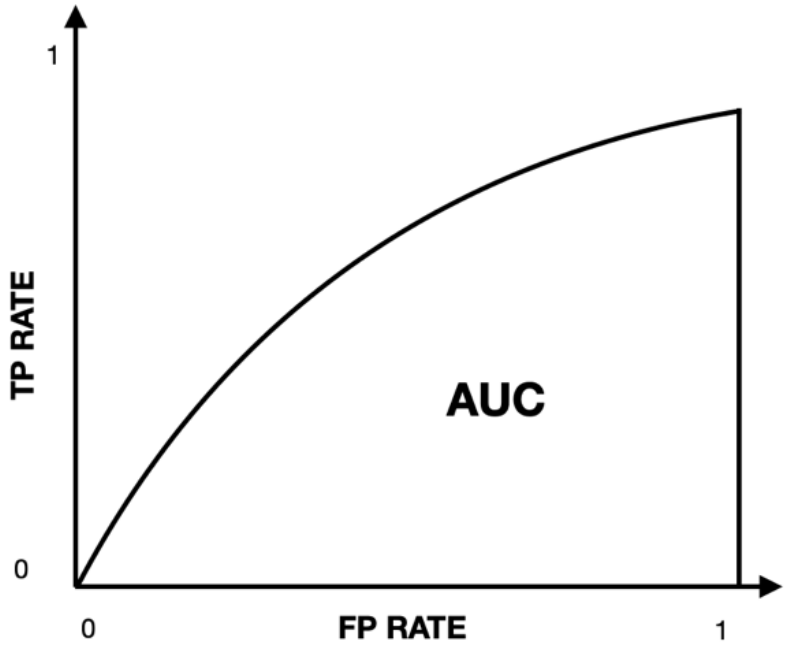


Figura 28. AUC (Area sotto la curva ROC)

3.6.1 ANALISI E RISULTATI LIGHTFM

I passaggi seguiti per analizzare questo modello sono quelli già spiegati precedentemente.

Per questa libreria ottenere i dati di MovieLens è molto semplice, in quanto il dataset consiste in una delle funzioni fornite da LightFM stesso:

```
import numpy as np
from lightfm import LightFM
import pandas as pd
from lightfm.evaluation import precision_at_k, auc_score
from lightfm.datasets import fetch_movielens
```

[125] Approfondimenti su https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[126] Approfondimenti su https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Vengono effettuati gli split dei dati in modo che ci sia un 80% di training e un 20% di test. Successivamente viene addestrato il modello e se ne osserva l'accuratezza.

```
def split_data(data):
  _train, _test = train = data['train'], data['test']
  return _train, _test
```

```
def train_model(train, test, model, epochs, k):
  model.fit_partial(train, epochs=epochs)
  _train_precision = precision_at_k(model, train, k=k).mean()
  _test_precision = precision_at_k(model, test, k=k, train_interactions=train).mean()
  return _train_precision, _test_precision
```

```
def compute_auc(model, train, test):
  _train_auc = auc_score(model, train).mean()
  _test_auc = auc_score(model, test).mean()
  return _train_auc, _test_auc
```

Basandosi sulle caratteristiche di questa libreria e sulla sua originaria costruzione, non è stato possibile calcolare la RMSE bensì la AUC. Si utilizza il modello WARP per ottimizzare le prestazioni in modo che si ottenga un input migliore in termini di precisione.

```
# train the model and get precisions
model = LightFM(learning_rate=0.05, loss='warp')
train_precision, test_precision = train_model(train, test, model, 10, 10)
```

```
# compute the area under curve (auc)
train_auc, test_auc = compute_auc(model, train, test)
```

```
print('Precision: train %.2f, test %.2f.' % (train_precision, test_precision))
print('AUC: train %.2f, test %.2f.' % (train_auc, test_auc))
```

```
Precision: train 0.60, test 0.22.
AUC: train 0.94, test 0.90.
```

L'AUC di training rappresenta la capacità dell'algoritmo di apprendere le relazioni nascoste tra i dati di training durante l'addestramento dell'algoritmo e corrisponde al 94%. Mentre la AUC di test evidenzia quanto l'algoritmo ha predetto bene i nuovi dati, quelli di test, e corrisponde al 90% di accuratezza rispetto ai valori reali.

Per quanto riguarda l'analisi del dataset da 1 milione di righe si seguono i medesimi passaggi.

```
# get input dataset
ratings = get_data(2) # 1M records
ratings
```

Anche in questo caso viene utilizzato il modello *"warp"* per ottimizzare le prestazioni e si calcola la AUC.

```
# train the model and get precisions
model = LightFM(learning_rate=0.05, loss='warp')
train_precision, test_precision = train_model(train, test, model, 20, 20)
```

```
# compute the area under curve (auc)
train_auc, test_auc = compute_auc(model, train, test)
```

```
print('Precision: train %.2f, test %.2f.' % (train_precision, test_precision))
print('AUC: train %.2f, test %.2f.' % (train_auc, test_auc))
```

```
Precision: train 0.59, test 0.18.
AUC: train 0.94, test 0.91.
```

Si ottiene un AUC per il training pari al 94% e un AUC per il test pari al 91% di accuratezza rispetto ai valori reali. Risultati molto buoni, quasi uguali allo studio del dataset da 100k

3.6.2 ANALISI E RISULTATI SURPRISE

Per questa libreria, la quale ha al suo interno una serie di algoritmi e set di dati integrati, ho deciso di usare e analizzare gli algoritmi più comuni, come[127]:

- *SVD:* è stato reso popolare da Simon Funk durante il Premio Netflix. Quando le linee di base non vengono utilizzate, questo è equivalente alla fattorizzazione della matrice probabilistica [salakhutdinov2008a].
  classe **surprise.prediction_algorithms.matrix_factorization.SVD**
- *NMF:* si tratta di un algoritmo di filtraggio collaborativo basato sulla fattorizzazione di matrici non negative. È molto simile al SVD
  classe **surprise.prediction_algorithms.matrix_factorization.NMF**
- *SlopeOne:* si tratta di un filtraggio collaborativo semplice, ma accurato. Si basa su una semplice implementazione dell'algoritmo SlopeOne (limire 2007a).
  classe **surprise.prediction_algorithms.slope_one.SlopeOne**
- *KNNBasic:* si tratta di un algoritmo di filtraggio collaborativo di base.
  classe **surprise.prediction_algorithms.knns.KNNBasic**
- *CoClustering:* si tratta di un filtraggio collaborativo basato sul *co-clustering.* Si basa su una semplice implementazione di George, 2005. Agli utenti e agli elementi vengono assegnati alcuni *cluster* C_u, C_i e alcuni *co-cluster* C_{ui}.
  classe **surprise.prediction_algorithms.co_clustering.CoClustering**

[127] Approfondimento su https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html

Per il dataset da 100mila righe, viene utilizzato il `train_test_split()` per campionare un *trainset* e un *testset* con determinate dimensioni e utilizzare l'opzione desiderata. Si utilizza il metodo che addestrerà l'algoritmo sul *trainset* e il metodo che restituirà le previsioni fatte dal *testset*:

```
def split_data(data, test_size):
  _train, _test = train_test_split(data, test_size=test_size)
  return _train, _test
```

```
def format_data(data, columns):
  _reader = Reader(rating_scale=(1, 5))
  _data = Dataset.load_from_df(data[columns], _reader)
  return _data
```

```
def train_model(train, test, model):
  model.fit(train)
  _prediction = model.test(test)
  return _prediction
```

Rating delle prime 5 righe:

	user	item	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

Successivamente si addestrano i diversi modelli appartenenti al pacchetto Surprise:

```
models = [SVD(), NMF(), SlopeOne(), KNNBasic(), CoClustering()]
models_name = ['SVD', 'NMF', 'SlopeOne', 'KNNBasic', 'CoClustering']
model_rmse = pd.DataFrame()
predictions_dict = dict()
index = 0
```

Dopo aver addestrato il modello e aver ottenuto l'RMSE si prevedono i nuovi valori:

```
for model in models:
  index += 1
  model_name = models_name[index-1]
  prediction = train_model(train, test, model)
  rmse = compute_rmse(prediction)
  predictions_dict[model_name] = rmse

model_rmse = model_rmse.append(predictions_dict, ignore_index=True)
model_rmse
```

Per la misurazione delle prestazioni, utilizzando questo pacchetto, non è possibile calcolare altre metriche oltre al RMSE, in quanto la sua struttura non lo permette (avvolge altre librerie comunemente usate in Machine Learning).

```
RMSE: 0.8738
RMSE: 0.9247
RMSE: 0.9044
Computing the msd similarity matrix...
Done computing similarity matrix.
RMSE: 0.9518
RMSE: 0.9448
```

Più precisamente, i risultati ottenuti da ciascun algoritmo sono i seguenti:

	CoClustering	KNNBasic	NMF	SVD	SlopeOne
0	0.944849	0.951764	0.924705	0.873799	0.904411

Come si può notare dalla figura 29, l'algoritmo *KNNBasic* mostra il miglior valore RMSE per il set di test di input, il suo comportamento approssima al meglio i valori previsti rispetto a quelli reali. I suoi valori predetti sono affidabili al 95% rispetto a quelli reali. Tuttavia, anche gli altri algoritmi hanno ottenuto buoni risultati in termini di affidabilità, superiori al 90%. Solamente SVD potrebbe essere migliorato, in quanto i suoi valori predetti risultano affidabili all'87% rispetto a quelli reali.

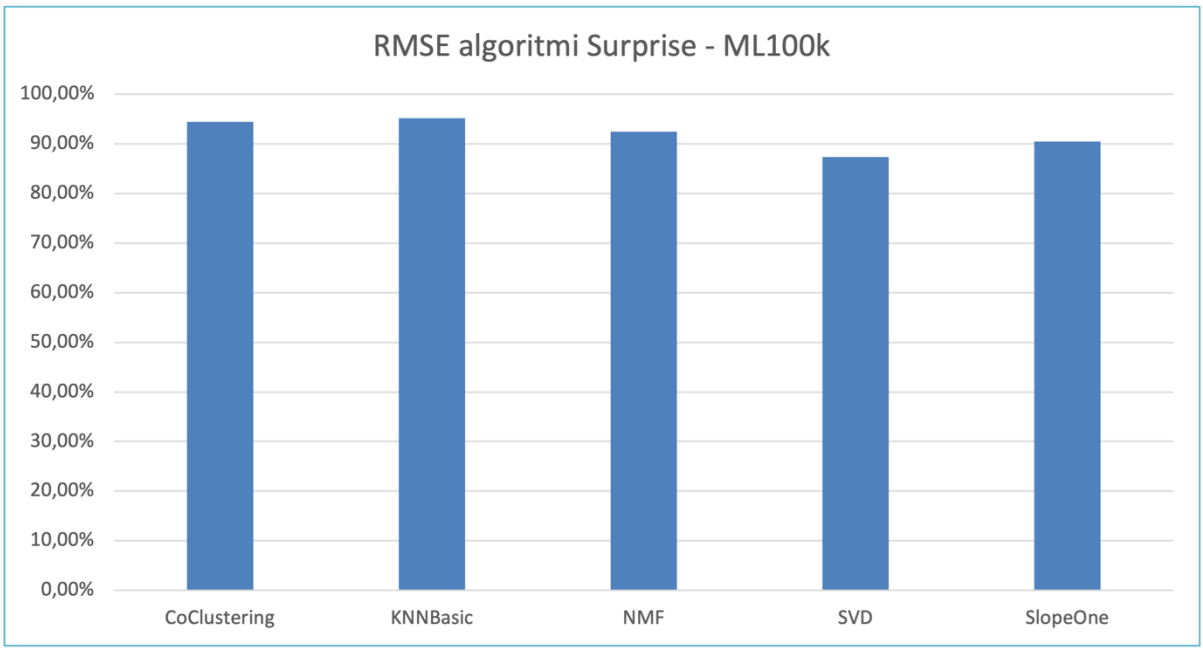


Figura 29. Risultati RMSE degli algoritmi Surprise per MovieLens 100k

Per quanto riguarda le analisi condotte sul dataset da 1 milione di righe, vengono seguiti i medesimi messaggi precedentemente descritti. Viene caricato il "rating dataset" prevedendo che i rating contengano le seguenti colonne (*ratings.columns*): *user*, *item*, *rating*, *timestamp*.

```
# load ratings dataset
ratings = get_data(2) # 1M records
ratings.columns = ['user', 'item', 'rating', 'timestamp']
# print the first 5 rows
ratings.head()
```

	user	item	rating	timestamp
0	316.0	1892.0	3.0	1.111494e+09
1	357.0	2291.0	4.0	1.348610e+09
2	428.0	5445.0	3.5	1.111488e+09
3	219.0	45499.0	3.0	1.194932e+09
4	380.0	4744.0	4.0	1.494529e+09

Successivamente i dati vengo formattati per essere leggibili dalla libreria ('*user*', '*item*', '*rating*') e divisi in 80% *train* e 20% *test* delle osservazioni totali. L'addestramento viene effettuato per i differenti algoritmi.

```
models = [SVD(), NMF(), SlopeOne(), KNNBasic(), CoClustering()]
models_name = ['SVD', 'NMF', 'SlopeOne', 'KNNBasic', 'CoClustering']
model_rmse = pd.DataFrame()
predictions_dict = dict()
index = 0
# batch size
k = 0
```

Dopo aver addestrato il modello, si prevedono i nuovi valori e si ottiene l'RMSE.

```
for model in models:
  index += 1
  model_name = models_name[index-1]
  prediction = train_model(train, test, model)
  print(prediction)
  rmse = compute_rmse(prediction)
  predictions_dict[model_name] = rmse.mean()

model_rmse = model_rmse.append(predictions_dict, ignore_index=True)
model_rmse
```

	CoClustering	KNNBasic	NMF	SVD	SlopeOne
0	0.855808	0.725167	0.693429	0.520548	0.661144

Come si può notare dalla figura 30, per questo dataset il migliore algoritmo è stato CoClustering con l'85% di RMSE. Tuttavia, nessun algoritmo ha raggiunto il 90% e i risultati ottenuti sono stati discreti, partendo dal 52% con SVD che rimane l'algoritmo con percentuale inferiore. Si può chiaramente affermare che la libreria Surprise è più accurata con il dataset più piccolo da 100mila riga.

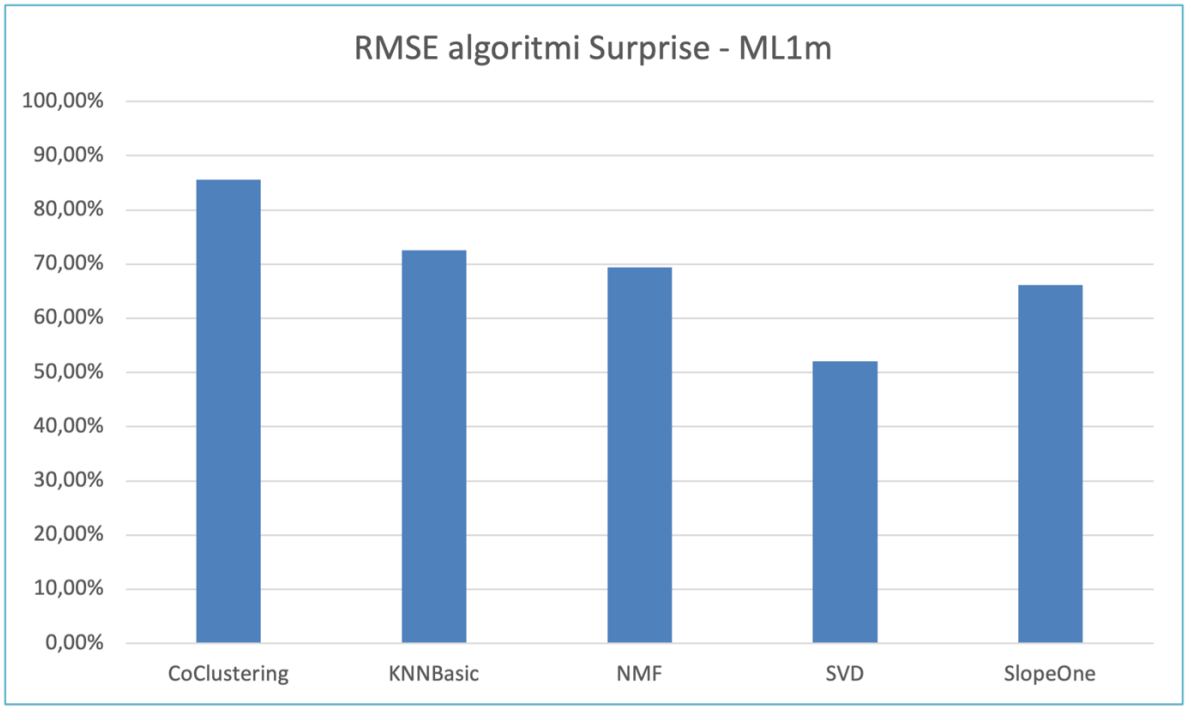


Figura 30. Risultati RMSE degli algoritmi Surprise per MovieLens 1m

3.6.3 ANALISI E RISULTATI LENSKIT

Per spiegare i passaggi effettuati, valgono le stesse considerazioni sopra.

Gli algoritmi di LensKit prevedono che un *ratings* rame contenga le seguenti colonne (in qualsiasi ordine)[128]:

- `user`, contenente gli identificatori utente. Nessun requisito è posto sugli ID utente.
- `item`, contenente gli identificatori degli elementi.
- `rating`, contenente le valutazioni degli utenti (se disponibili). Il codice di feedback implicito non richiede valutazioni.

```
# get input dataset
ratings = get_data()
ratings.columns = ['user', 'item', 'rating', 'timestamp']
ratings.head()
```

Questo è il risultato delle prime 5 predizioni:

	user	item	rating	timestamp	prediction
25146	177	2080	4.0	1435534751	3.255186
537	5	344	3.0	847434802	2.998113
25292	177	4022	3.0	1435524241	3.612992
6934	47	724	2.0	1496205806	2.432235
25062	177	1252	4.0	1435536871	4.105738

Le performance dell'algoritmo sono buone e pari quasi a 89%.

```
# compute the rmse between predicted and actual values
rmse = compute_rmse(prediction)
rmse
```

```
0.8925411538190678
```

[128] Approfondimenti su https://lkpy.readthedocs.io/en/stable/datasets.html#movielens-data-sets

Lo studio del dataset da 1 milione segue la stessa analisi e prevede che il *ratings* rame contenga le seguenti colonne: *user*, *item*, *rating*, *timestamp*.

```
# get input dataset
ratings = get_data(2) # 1M records
ratings.columns = ['user', 'item', 'rating', 'timestamp']
ratings.head()
```

	user	item	rating	timestamp
0	474.0	7414.0	2.5	1.080569e+09
1	232.0	32298.0	3.0	1.218170e+09
2	294.0	518.0	2.0	9.665967e+08
3	297.0	70.0	2.0	9.008720e+08
4	91.0	4105.0	4.0	1.112712e+09

Vengono suddivisi i dati in 80% training e 20% test e avviato il *sanity check.*

```
len(train) + len(test) == len(ratings)
```

Si ottengono i seguenti risultati:

```
# print the first 5 prediction results
prediction.head()
```

	user	item	rating	timestamp	prediction
533369	291.0	1.0	4.0	1.453052e+09	4.414183
533369	291.0	1.0	4.0	1.453052e+09	4.414183
533369	291.0	1.0	4.0	1.453052e+09	4.414183
533369	291.0	1.0	4.0	1.453052e+09	4.414183
533369	291.0	1.0	4.0	1.453052e+09	4.414183

```
# compute the rmse between predicted and actual values
rmse = compute_rmse(prediction)
rmse
```

```
0.802027068669124
```

Le performance dell'algoritmo sono pari a 80%, quindi risultano inferiori a quelle calcolate dal dataset di 100k.

3.6.4 ANALISI E RISULTATI TENSORFLOW

Per analizzare questa potente libreria sono stati seguiti i passaggi precedentemente descritti, uguali per le altre librerie, ma sono state attuate le seguenti differenze tecniche:

- Il dataset è stato scaricato direttamente da TensorFlow, quindi non è stato fornito dall'esterno.

```
# install tensorflow-recommenders library and update the datasets
!pip install -q tensorflow-recommenders
!pip install -q --upgrade tensorflow-datasets
```

```
# useful imports
import pandas as pd
from typing import Dict, Text
import numpy as np
import os
import pprint
import tensorflow as tf
import tensorflow_datasets as tfds
import tensorflow_recommenders as tfrs
from tensorflow.keras.layers.experimental.preprocessing import StringLookup
from tensorflow.keras.layers import Embedding
from tensorflow_recommenders.metrics import FactorizedTopK
from tensorflow.keras.optimizers import Adagrad, Adam
```

- Sono state selezionate solo le colonne di interesse ai fini dell'analisi, alleggerendone l'esecuzione del modello.

```
def get_data():
  _ratings = tfds.load('movielens/100k-ratings', split="train")
  _movies = tfds.load('movielens/100k-movies', split="train")
  return _ratings, _movies
```

- Per la stessa motivazione del punto precedente, sono stati ripartiti i dati in batch.
- Sono stati *tokenizzati* alcuni dati per rendere idoneo il formato del modello.
- È stata eseguita una creazione e *overriding* di classi e funzioni usate dal modello.

```python
class MovieLensModel(tfrs.Model):

  def __init__(
      self,
      user_model: tf.keras.Model,
      movie_model: tf.keras.Model,
      task: tfrs.tasks.Retrieval):
    super().__init__()

    self.user_model = user_model
    self.movie_model = movie_model
    self.task = task

  def compute_loss(self, features: Dict[Text, tf.Tensor], training=False) -> tf.Tensor:
    user_embeddings = self.user_model(features["user_id"])
    movie_embeddings = self.movie_model(features["movie_title"])

    return self.task(user_embeddings, movie_embeddings)
```

```python
class OverridedMovielensModel(tf.keras.Model):

  def __init__(
      self,
      user_model,
      movie_model):
    super().__init__()
    self.movie_model: tf.keras.Model = movie_model
    self.user_model: tf.keras.Model = user_model
    self.task: tf.keras.layers.Layer = task

  def train_step(self, features: Dict[Text, tf.Tensor]) -> tf.Tensor:

    # set up a gradient tape to record gradients.
    with tf.GradientTape() as tape:

      # loss computation.
      user_embeddings = self.user_model(features["user_id"])
      positive_movie_embeddings = self.movie_model(features["movie_title"])
      loss = self.task(user_embeddings, positive_movie_embeddings)

      # handle regularization losses as well.
      regularization_loss = sum(self.losses)

      total_loss = loss + regularization_loss

    gradients = tape.gradient(total_loss, self.trainable_variables)
    self.optimizer.apply_gradients(zip(gradients, self.trainable_variables))
```

```
    metrics = {metric.name: metric.result() for metric in self.metrics}
    metrics["loss"] = loss
    metrics["regularization_loss"] = regularization_loss
    metrics["total_loss"] = total_loss

    return metrics

  def test_step(self, features: Dict[Text, tf.Tensor]) -> tf.Tensor:

    # loss computation.
    user_embeddings = self.user_model(features["user_id"])
    positive_movie_embeddings = self.movie_model(features["movie_title"])
    loss = self.task(user_embeddings, positive_movie_embeddings)

    # handle regularization losses as well.
    regularization_loss = sum(self.losses)

    total_loss = loss + regularization_loss

    metrics = {metric.name: metric.result() for metric in self.metrics}
    metrics["loss"] = loss
    metrics["regularization_loss"] = regularization_loss
    metrics["total_loss"] = total_loss

    return metrics
```

- I modelli costruiti sono i seguenti: utente, film, genere (*retrieval*, che usa i precedenti).

```
# select basic features: user_id, movie_title
ratings, movies = select_features(ratings_data, movies_data)
```

Nel complesso il modello di TensorFlow Recommender non è andato benissimo e non è riuscito a generalizzare i nuovi dati ai fini della *prediction*.

La performance del test mostra un risultato peggiore della fase di allenamento.

```
{'factorized_top_k/top_100_categorical_accuracy': 0.2113499939441681,
 'factorized_top_k/top_10_categorical_accuracy': 0.013050000183284283,
 'factorized_top_k/top_1_categorical_accuracy': 0.0003000000142492354,
 'factorized_top_k/top_50_categorical_accuracy': 0.1014999970793724,
 'factorized_top_k/top_5_categorical_accuracy': 0.0044999998062849045,
 'loss': 28389.564453125,
 'regularization_loss': 0,
 'total_loss': 28389.564453125}
```

Questo è causato da:

1) *Overfitting*: Si comporta meglio sui dati che ha visto, semplicemente perché può memorizzarli. Questo fenomeno di *"overfitting"* è particolarmente forte quando i modelli hanno molti parametri. Può essere mediato dalla regolarizzazione del modello e dall'uso di funzionalità utente e film che aiutano il modello a generalizzare meglio i dati invisibili.

2) *Re-recommending*: Il modello consiglia nuovamente alcuni dei film già visti dagli utenti. Questi può causare un allontanamento dalla lista migliore dei film raccomandati.
Infine, si può stabilire che questo approccio non è la scelta migliore per i motivi spiegati sopra ed è, inoltre, troppo complesso per raccomandare il set di dati MovieLens.
Per esempio, una metrica di accuratezza categoriale tra i primi 100 di 0,2113 ci direbbe che, in media, il vero positivo è tra i primi 5 elementi recuperati il 21,13% delle volte. Dovremmo migliorarlo con alcune tecniche speciali come regolarizzazione e livellamento.

Il file da 1 milione di righe ha seguito la medesima analisi e non ha avuto sostanziali modifiche, registrando un'accuratezza pari al 22%. Percentuale molto bassa che non si discosta dall'accuratezza del file da 100 mila righe. Di conseguenza è possibile affermare che anche in questo caso TensorFlow non è riuscito a generalizzare i nuovi dati ai fini della *prediction*.

3.7 CONFRONO SPERIMENTALE: DISCUSSIONE GENERALE

Per effettuare l'analisi di entrambi i dataset in ogni libreria, l'algoritmo è stato suddiviso in tre fasi principali: preparazione dei dati, creazione del modello previsionale (training e test) e verifica del modello. Ogni sistema ha restituito in output un valore differente di accuratezza, come mostrato nella seguente figura 31.

Surprise RMSE di KNN Basic	0.95
Surprise RMSE di CoClustering	0.94
LightFM AUC test	0.90
Surprise RMSE di SlopeOne	0.90
Surprise RMSE di NMF	0.92
LensKit RMSE	0.89
Surprise RMSE di SVD	0.87
TensorFlow	0.21

Figura 31. Riassunto accuratezza delle librerie in ML100k

Dall'analisi di ogni singolo algoritmo, KNN Basic risulta il migliore in termini di accuratezza prevedendo i risultati reali nel 95% dei casi. La differenza con l'algoritmo CoClustering risulta di pochissimi decimali e si può considerare anche molto buono in termini di accuratezza dei risultati reali. Per questi algoritmi è stato possibile raggiungere tali livelli di performance grazie alle loro caratteristiche. Il *k-nearest neighbors* (k-NN) è tra gli algoritmi più semplici utilizzati in machine learning e si basa sulla similarità dei dati, sulle caratteristiche degli oggetti vicini a quello considerato. Riesce ad individuare k cluster e identificare i dati all'interno dei *ratings*. Di conseguenza, è possibile trovare i valori più simili a quelli reali. Inoltre, il KNN Basic è stato realizzato proprio per funzionare bene con dataset più piccoli. Per gli stessi motivi è possibile giustificare l'accuratezza dell'algoritmo CoClustering, in quanto raggruppa gli elementi sulla base della loro distanza reciproca. In generale, la maggior parte degli algoritmi ha registrato risultati molto buoni, superiori al 90% che suggeriscano quanto il dataset sia già stato ottimizzato per essere usato a scopo sperimentale. I vantaggi nell'utilizzo di un dataset più piccolo stanno proprio nelle dimensioni della quantità di dati, in quanto l'algoritmo performa meglio con dati più piccoli. MovieLens 100k avrà al suo interno un processo di *data cleaning* in grado di rilevare gli errori e imprecisioni per correggerli. Di conseguenza, la correzione dei dati può essere già avvenuta attraverso l'utilizzo di alcuni sistemi come l'approccio a *batch.*

Dall'analisi di ogni singolo algoritmo del dataset MovieLens 1milione il migliore è stato LightFM, come mostra la figura 32, con un valore di AUC pari al 91%, il quale ha approssimato con un'accuratezza del 91% i valori reali e presenti nel test set.

LightFM AUC test	0.91
Surprise RMSE di CoClustering	0.85
LensKit RMSE	0.80
Surprise RMSE di KNN Basic	0.72
Surprise RMSE di NMF	0.69
Surprise RMSE di SlopeOne	0.66
Surprise RMSE di SVD	0.52
TensorFlow	0.22

Figura 32. Riassunto accuratezza delle librerie in ML1m

In generale, questo studio non è stato complessivamente soddisfacente come quello del dataset da 100mila righe, in quanto quasi tutte le librerie hanno registrato una percentuale di accuratezza inferiore al 90%. La motivazione alla base di queste performance può essere collegata alla grande quantità di dati presenti. I dataset da 1 milione non sono del tutto ottimizzati, quindi è possibile che ci siano errori ed imprecisioni che causino problemi di *overfitting*. In questo modo si avranno performance migliori in fase di training e peggiori in fase di test. Sarebbe opportuno risolvere questo problema utilizzando un approccio a *batch* per tutti gli algoritmi, così da dividere i dati in porzioni più piccole. Anche in questo caso l'algoritmo CoClustering si trova tra quelli più performanti e le motivazioni sono le stesse citate precedente. Inoltre, per costruzione questo algoritmo funziona meglio con dataset più grandi, al contrario del KNN Basic

Come si può notare dalle figure, TensorFlow Recommender riporta percentuali bassissime di accuratezza in entrambi i dataset, al contrario delle altre librerie che performano meglio in quanto sono algoritmi nati per le raccomandazioni. TensorFlow è stato concepito per il Deep Learning e non riguarda solamente la branca del Machine Learning, ma serve anche per il riconoscimento di immagini o attività di Computer Vision. Di conseguenza, è una libreria troppo potente per essere utilizzata in un sistema così semplice.

3.8 LIMITAZIONI E RICERCA FUTURA

Il dataset utilizzato come riferimento per il confronto sperimentale è molto utilizzato per analisi simili in ambito di sistemi di raccomandazione. Per questo studio specifico sono stati utilizzati solo due tipologie di dataset, ML100k e ML1m. Il set con 100k è la versione più vecchia dei set di dati MovieLens, contenete un piccolo set di dati con dati demografici, ma queste sue dimensioni più ridotte lo rendono veloce da scaricare, analizzare ed elaborare. Di conseguenza, ulteriori studi potrebbero affrontare lo stesso confronto utilizzando versione del dataset differenti e più elevate, come il ML25m. Tale set non è stato eseguito in questo studio a causa della sua grandezza e potenza. Infatti, dataset così grande contengono troppi dati e sono eccessivamente pesanti per essere supportati da un normale computer o da un sistema come Colab. Si suggerisce agli studi futuri di utilizzare un "*cloud*" per essere più veloci negli studi ed evitare di sovraccaricare il sistema. Alcune soluzioni possono essere: Amazon Web Services, Azure di Microsoft e Google Cloud Platform. Gli studiosi, che sono interessati a combinare valutazioni o *tag* di dati con dati di contenuti cinematografici ricchi, dovrebbero prendere in considerazione l'utilizzo del set di dati di 20 (o 25) milioni in combinazione con risorse esterne. Questo set include una mappa degli ID MovieLens sugli ID IMDb4 ("Internet Movie Database"). Poiché gli ID dei film sono stabili tra i set di dati, il file di collegamento di

20m può essere utilizzato in combinazione con uno qualsiasi dei set di dati rilasciati in precedenza.

Le ampie modifiche all'esperienza utente di MovieLens hanno inevitabilmente portato a dati di valutazione meno "puliti" rispetto alla situazione in cui avesse mantenuto l'interfaccia costante. Le modifiche agli strumenti di ricerca, agli algoritmi di raccomandazione e alle funzionalità disponibili modificano il contenuto mostrato agli utenti, il che a sua volta influisce sui dati di valutazione risultanti. Tuttavia, MovieLens non è l'unico con questo problema, i siti di raccomandazione come Amazon e Netflix hanno anche introdotto sostanziali modifiche e perturbazioni dell'interfaccia attraverso i test A/B.

L'approccio usato in questo studio è quello di partire da un dataset pronto e strutturato, quindi ulteriori studi potrebbero migliorare questo metodo usando un sistema di *real time*. Sarebbe interessanti utilizzare come codice sorgente non dati pronti, ma *Data Lake.* In questo modo sarà possibile recuperare e organizzare i dati secondo il tipo di analisi che si intende effettuare.

I set di dati MovieLens includono dati solo da utenti con almeno 20 valutazioni, quindi sono intrinsecamente sbilanciati verso utenti "di successo". Cioè, gli utenti che sono meno interessati a valutare i film, non sono stati in grado di trovare abbastanza contenuti valutabili o non hanno apprezzato la loro esperienza iniziale nel sistema, non sono inclusi nei set di dati. È possibile che questi utenti siano fondamentalmente diversi dagli utenti nei set di dati. I dataset associano i *timestamp* a ciascuna valutazione, ma questi *timestamp* non rappresentano la data di consumo. Le valutazioni in MovieLens possono verificarsi in qualsiasi momento, anche molti anni dopo aver visto un film. Spesso, gli utenti inseriscono un gran numero di valutazioni in una singola sessione, riempiendo la cronologia delle valutazioni per soddisfazione personale o nella speranza di ottenere consigli più personalizzati.

Quindi i prossimi studi possono analizzare ulteriori dataset che superino questo limite o che abbiano impostazioni differenti, per valutarne anche approcci e algoritmi differenti. I ricercatori potrebbero prendere in considerazione set di dati alternativi che corrispondono meglio al sistema online che desiderano simulare oppure potrebbero valutare un approccio su più set di dati. Alcuni dei set più citati e utilizzati sono: "Book-Crossing" per la valutazione di libri online, "Yahoo Music" come set di dati musicali oppure "Amazon". [129] (Figura 33)

[129]F. Maxwell Harper e Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Trans. Interagire. Intell. Syst. VOl. 5, n. 4, articolo 19 (gennaio 2016), 19 pagine. DOI: https: //doi.org/10.1145/2827872

Name	Date Range	Domain	Rating Scale	Ratings	Density
Book-Crossing	2001–2004	books	0–10, 11 discrete values	1.1m	0.003%
EachMovie[a]	1995–1997	movies	0–14, 26 discrete values[b]	2.7m	2.872%
Jester (dataset1)	1999–2003	jokes	−10–10, continuous	4.1m	57.463%
Amazon	1996–2014	many	1–5, 5 discrete values	82.8m	<0.001%
Netflix Prize[a]	1998–2005	movies	1–5, 5 discrete values	100.5m	1.178%
Yahoo Music (C15)	1999–2009	music[c]	0–100, 101 discrete values[d]	262.8m	0.042%

Figura 33. Esempi di dataset alternativi utilizzati in letteratura

Inoltre, future ricerche potrebbero prendere in considerazioni librerie differente per la creazione di sistemi di raccomandazioni. Sarebbe interessante analizzare anche lo stesso dataset su ulteriori sistemi per confrontarne i risultati e capire se esiste un livello maggiore di accuratezza delle previsioni.

CONCLUSIONE

Il presente lavoro di tesi ha contribuito a fornire una spiegazione teorica del machine learning, soffermandosi maggiormente sulla rilevanza e la classificazione dei sistemi di raccomandazione, approfondendo i differenti algoritmi utilizzati e le loro funzionalità. Viene illustrato l'esempio del "caso Netflix" per confermare le spiegazioni precedenti e dare rilevanza alle implicazioni manageriali per l'utilizzo dei *recommendation system.* L'ultimo capitolo si pone l'obiettivo di evidenziare l'importanza delle tecniche di raccomandazione a livello manageriale per utenti e aziende, oltre che analizzare un *dataset* attraverso delle librerie Python per comprenderne i meccanismi basilari e valutare le migliori performance delle librerie sulla base delle previsioni. Gli effetti conseguenti all'utilizzo di tale sistema non diventano vantaggiosi solo per l'utente o il cliente di un prodotto/servizio al quale viene facilitata la scelta migliorando la sua esperienza di acquisto o utilizzo, ma anche per l'azienda che offre quel determinato servizio o prodotto, che ne aumenta i ricavi. Il *dataset* di riferimento dello studio riguarda "MovieLens", uno dei più popolari sistemi di raccomandazione di film basato sul web non commerciale, uno dei settori in cui l'utente riscontra una grande difficoltà nella scelta finale e ha bisogno di suggerimenti in conformità con le proprie preferenze. Il cuore dell'analisi si focalizza sull'analisi di MoviLens 100k (composto dal 100mila valutazioni) e MovieLens 1m (composto da 1 milione di valutazioni) attraverso l'utilizzo delle seguenti librerie: LensKit, LightFM, Surprise e TensorFlow Recommender. Per effettuare tale analisi l'algoritmo è stato suddiviso in tre fasi principali: preparazione dei dati, shuffle (per rendere i dati più randomici), creazione del modello previsionale (training e test) e verifica del modello. Le predizioni ottenute, in questo caso, riguardano *ratings* dei vari film per tutti gli utenti (overall del sistema). Per quantificare la bontà degli *outputs* ottenuti e valutarne l'accuratezza delle predizioni, sono state utilizzate due differenti metriche, in base alla libreria in esame: RMSE e AUC.

A livello generale, l'analisi da me condotta ha dimostrato che ogni libreria ha restituito in output un valore differente di accuratezza. Nello specifico, per il dataset da 100 mila righe, le performance migliori sono state raggiunti da due algoritmi della libreria Surprise: KNN Basic con il 95% di accuratezza e CoClustering con il 94% di accuratezza. Entrambi gli algoritmi si basano sulla similarità dei dati e questo giustifica tale correttezza della previsione in un dataset più piccolo. Mentre, per il dataset da 1 milione di righe, la performance migliore è stata raggiunta dalla libreria LightFm, dove si ottiene un AUC per il training pari al 94% e un AUC per il test pari al 91% di accuratezza rispetto ai valori reali. Tuttavia, si sottolinea che le numerose esperienze dell'utente di MovieLens portano cambiamenti agli strumenti di ricerca, agli algoritmi di raccomandazione e alle funzionalità disponibili modificando il contenuto mostrato agli utenti e influenzando conseguentemente i dati di valutazione risultati, rispetto al caso in cui si fosse mantenuta l'interfaccia costante. I risultati di TensorFlow Recommender

sono stati quelli meno performanti in termini di accuratezza, in quanto la libreria è stata sviluppata per la macro-area del Deep Learning e non si sofferma solamente sui sistemi di raccomandazione, come le altre librerie. Tuttavia, l'analisi globale effettuata rileva dei buoni output, soprattutto per il dataset composto da 100 mila righe che sicuramente sarà già stato ottimizzato precedentemente per usi sperimentali, ottenendo dati già corretti. Al contrario, il dataset da 1 milione di righe ha ottenuto percentuali di performance inferiori, causate dalla grande quantità di dati presenti e quindi dalla possibile presenza di errori e imprecisioni. Correggere e perfezionare un algoritmo può essere molto complicato, richiedendo strumenti e abilità particolari.

ELENCO DELLE FIGURE

1. Correlazione tra intelligenza artificiale, machine learning e deep learning
2. Approcci dell'apprendimento automatico
3. Apprendimento supervisionato
4. Esempio di Clustering
5. Schema funzione di rinforzo
6. Global Digital Overview ottobre 2020
7. Modello generale del processo di raccomandazione
8. Fasi di raccomandazione
9. Mostra la tipologia di dati raccolti dall'algoritmo
10. Tecniche di raccomandazione
11. Modello del sistema di raccomandazione basato sui contenuti
12. Classificazione degli algoritmi di Filtraggio Collaborativo
13. Sistema di raccomandazione ibrido
14. Sistema di raccomandazione di Netflix
15. Evoluzione dell'approccio di personalizzazione di Netflix
16. In che modo Netflix classifica i titoli
17. Esempio di titoli generati dal ranker per similarità video-video
18. Flusso di lavoro del modello di Netflix
19. Alcuni motori di raccomandazione
20. Esempio modulo di Python
21. Tipologie di Recommender System e fonti informative
22. Diagramma di sistema
23. Crescita di Movielens dal 1999 al 2015
24. Riassunto quantitativo del "MovieLens Ratings Datasets"
25. Esempio di homepage MovieLens
26. Esempio di schermata MovieLens
27. Esempio di Overfitting
28. AUC (Area sotto la curva ROC)
29. Risultati RMSE degli algoritmi Surprise per MovieLens 100k
30. Risultati RMSE degli algoritmi Surprise per MovieLens 1m
31. Riassunto accuratezza delle librerie in ML100k
32. Riassunto accuratezza delle librerie in ML1m
33. Esempi di dataset alternativi utilizzati in letteratura

APPENDICE

MovieLens 100K

LensKit

```
# install lenskit package
%pip install lenskit
```

```
Collecting lenskit
  Downloading
https://files.pythonhosted.org/packages/a5/35/2d052536f880cc5504713d5af69562acea8afa8e1734fd5ef174b950887b/lenskit-0.12.3-py3-none-any.whl (82kB)
     |████████████████████████████████| 92kB 10.5MB/s
Requirement already satisfied: pyarrow>=0.15 in /usr/local/lib/python3.7/dist-packages (from lenskit) (3.0.0)
Collecting binpickle>=0.3.2
  Downloading
https://files.pythonhosted.org/packages/4d/e3/7aa8964f808f4e25ef0ff46b4fca7861079aacbbb572743557d6d686c3c6/binpickle-0.3.3-py3-none-any.whl
Requirement already satisfied: numba<0.53,>=0.51 in /usr/local/lib/python3.7/dist-packages (from lenskit) (0.51.2)
Requirement already satisfied: cffi>=1.12.2 in /usr/local/lib/python3.7/dist-packages (from lenskit) (1.14.5)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.7/dist-packages (from lenskit) (1.19.5)
Requirement already satisfied: scipy>=1.2 in /usr/local/lib/python3.7/dist-packages (from lenskit) (1.4.1)
Collecting csr<0.3,>=0.2
  Downloading
https://files.pythonhosted.org/packages/7b/cf/6f3e70794536754544b2fdbe04166e307e3876094b220354b98b1077508b/csr-0.2.0-py3-none-any.whl
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.7/dist-packages (from lenskit) (1.1.5)
Collecting pickle5; python_version < "3.8"
  Downloading
https://files.pythonhosted.org/packages/f7/4c/5c4dd0462c8d3a6bc4af500a6af240763c2ebd1efdc736fc2c946d44b70a/pickle5-0.0.11.tar.gz (132kB)
     |████████████████████████████████| 133kB 37.3MB/s
Requirement already satisfied: msgpack>=1.0 in /usr/local/lib/python3.7/dist-packages (from binpickle>=0.3.2->lenskit) (1.0.2)
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/local/lib/python3.7/dist-packages (from numba<0.53,>=0.51->lenskit) (0.34.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from numba<0.53,>=0.51->lenskit) (56.1.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-packages (from cffi>=1.12.2->lenskit) (2.20)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24->lenskit) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24->lenskit) (2018.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->pandas>=0.24->lenskit) (1.15.0)
Building wheels for collected packages: pickle5
  Building wheel for pickle5 (setup.py) ... done
  Created wheel for pickle5: filename=pickle5-0.0.11-cp37-cp37m-linux_x86_64.whl size=219269
sha256=f7e85e186d74b5aeff093b8a96a64f22a17fad34c62860849a3b6595651ed490
  Stored in directory:
/root/.cache/pip/wheels/a6/90/95/f889ca4aa8b0e0c7f21c8470b6f5d6032f0390a3a141a9a3bd
Successfully built pickle5
Installing collected packages: pickle5, binpickle, csr, lenskit
Successfully installed binpickle-0.3.3 csr-0.2.0 lenskit-0.12.3 pickle5-0.0.11
```

[ ]

```
# useful libraries
from lenskit.datasets import MovieLens
from lenskit.algorithms.bias import Bias
from lenskit.batch import predict
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd
```

[]

```
def get_data():
  url='https://drive.google.com/file/d/1lTzGIc0uonkBiDM-qjueTfFyb8YRH4L1/view?usp=sharing'
  file_id=url.split('/')[-2]
  dwn_url='https://drive.google.com/uc?id=' + file_id
  data = pd.read_csv(dwn_url)
  return data
```

[]

```
def split_data(data, test_size):
  _test, _train = train_test_split(data, test_size=test_size)
  return _test, _train
```

[]

```
def train_model(train, test, model):
  model.fit(train)
  _prediction = predict(model, test)
  return _prediction
```

[]

```
def compute_rmse(data):
  _rmse = mean_squared_error(data['prediction'], data['rating'], squared=False)
  return _rmse
```

[]

```
# get input dataset
ratings = get_data()
ratings.columns = ['user', 'item', 'rating', 'timestamp']
ratings.head()
```

---

	user	item	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

[]

```
# split data into train and test sets: 80% training, 20% test
train, test = split_data(ratings, 0.2)
```

[]

```
# shape of train set
train.shape
```
```
(80668, 4)
```

[]
```
# shape of train set
test.shape
```
```
(20168, 4)
```

[]
```
# sanity check
len(train) + len(test) == len(ratings)
```
```
True
```

[]
```
# train the model
prediction = train_model(train, test, Bias())
```

[]
```
# print the first 5 prediction results
prediction.head()
```

---

	user	item	rating	timestamp	prediction
88337	570	1784	4.0	1181477774	3.507490
88539	572	539	5.0	945893475	3.817754
81010	512	165	5.0	841449233	3.794505
40588	275	190	3.0	1049078712	2.971490
47586	307	44397	1.5	1186258628	3.051295

[]
```
# compute the rmse between predicted and actual values
rmse = compute_rmse(prediction)
rmse
```
```
0.8925411538190678
```

LightFM

```
# install LightFM library
!pip install lightfm
```

```
Collecting lightfm
  Downloading
https://files.pythonhosted.org/packages/5e/fe/8864d723daa8e5afc74080ce510c30f7ad52facf6a157d4b42dec83dfab4/lightfm-1.16.tar.gz (310kB)
     |████████████████████████████████| 317kB 6.9MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from lightfm) (1.19.5)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.7/dist-packages (from lightfm) (1.4.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from lightfm) (2.23.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from lightfm) (0.22.2.post1)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages
(from requests->lightfm) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->lightfm)
(3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->lightfm)
(2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->lightfm)
(2020.12.5)
```

```
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->lightfm)
(1.0.1)
Building wheels for collected packages: lightfm
  Building wheel for lightfm (setup.py) ... done
  Created wheel for lightfm: filename=lightfm-1.16-cp37-cp37m-linux_x86_64.whl size=706128
sha256=bdebaaf2bbd79ed91de2c5306efad083c269e0bd21df2758cb3636b86c91897a
  Stored in directory: /root/.cache/pip/wheels/c6/64/d4/673c7277f71ac4c5ad4835b94708c01b653ef2d3aa78ef20aa
Successfully built lightfm
Installing collected packages: lightfm
Successfully installed lightfm-1.16
```

[]

```
# useful import
import numpy as np
from lightfm import LightFM
import pandas as pd
from lightfm.evaluation import precision_at_k, auc_score
from lightfm.datasets import fetch_movielens
```

[]

```
def split_data(data):
  _train, _test = train = data['train'], data['test']
  return _train, _test
```

[]

```
def train_model(train, test, model, epochs, k):
  model.fit_partial(train, epochs=epochs)
  _train_precision = precision_at_k(model, train, k=k).mean()
  _test_precision = precision_at_k(model, test, k=k, train_interactions
=train).mean()
  return _train_precision, _test_precision
```

[]

```
def compute_auc(model, train, test):
  _train_auc = auc_score(model, train).mean()
  _test_auc = auc_score(model, test).mean()
  return _train_auc, _test_auc
```

[]

```
# load movielens dataset
data = fetch_movielens()
```

[]

```
# split data into train and test sets
train, test = split_data(data)
```

[]

```
# lenght of train set
train.shape[0]
```

```
943
```

[]
```
# lenght of test set
test.shape[0]
```
```
943
```

[]
```
# train the model and get precisions
model = LightFM(learning_rate=0.05, loss='warp')
train_precision, test_precision = train_model(train, test, model, 10, 10)
```

[]
```
# compute the area under curve (auc)
train_auc, test_auc = compute_auc(model, train, test)
```

[]
```
print('Precision: train %.2f, test %.2f.' % (train_precision, test_precision))
print('AUC: train %.2f, test %.2f.' % (train_auc, test_auc))
```
```
Precision: train 0.60, test 0.22.
AUC: train 0.94, test 0.90.
```

Surprise

```
# install the surprise package
%pip install surprise
```
```
Collecting surprise
  Downloading https://files.pythonhosted.org/packages/61/de/e5cba8682201fcf9c3719a6fdda95693468ed061945493dea2dd37c5618b/surprise-0.1-py2.py3-none-any.whl
Collecting scikit-surprise
  Downloading https://files.pythonhosted.org/packages/97/37/5d334adaf5ddd65da99fc65f6507e0e4599d092ba048f4302fe8775619e8/scikit-surprise-1.1.1.tar.gz (11.8MB)
     |████████████████████████████████| 11.8MB 32.1MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.0.1)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.19.5)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.4.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.15.0)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp37-cp37m-linux_x86_64.whl size=1617629 sha256=8f702499278e55f2d052507244acdceb369bf9a0718506669afadd9e626827d3
  Stored in directory: /root/.cache/pip/wheels/78/9c/3d/41b419c9d2aff5b6e2b4c0fc8d25c538202834058f9ed110d0
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.1 surprise-0.1
```

[]
```
# useful libraries
from surprise import SVD, NMF, SlopeOne, KNNBasic, CoClustering
from surprise.model_selection import cross_validate
```

```
from surprise import Reader
from surprise import Dataset
from surprise import accuracy
import pandas as pd
from surprise.model_selection import train_test_split
```

```
[]
def get_data():
  url='https://drive.google.com/file/d/11TzGIc0uonkBiDM-qjueTfFyb8YRH4L1/view?usp=sharing'
  file_id=url.split('/')[-2]
  dwn_url='https://drive.google.com/uc?id=' + file_id
  data = pd.read_csv(dwn_url)
  return data
```

```
[]
def split_data(data, test_size):
  _train, _test = train_test_split(data, test_size=test_size)
  return _train, _test
```

```
[]
def format_data(data, columns):
  _reader = Reader(rating_scale=(1, 5))
  _data = Dataset.load_from_df(data[columns], _reader)
  return _data
```

```
[]
def train_model(train, test, model):
  model.fit(train)
  _prediction = model.test(test)
  return _prediction
```

```
[]
def compute_rmse(prediction):
  _rmse = accuracy.rmse(prediction)
  return _rmse
```

```
[]
# load ratings dataset
ratings = get_data()
ratings.columns = ['user', 'item', 'rating', 'timestamp']

# print the first 5 rows
ratings.head()
```

```
[]
```

```
# format data to be readeable from surprise library
ratings = format_data(ratings, ['user', 'item', 'rating'])
```

```
[]
# split data into train and test sets: 80% training, 20% test of total observations
train, test = split_data(ratings, 0.2)
```

```
[]
# lenght of train set
train.n_ratings
80668
```

```
[]
# lenght of test set
len(test)
20168
```

```
[]
# sanity check
train.n_ratings + len(test) == len(ratings.df)
True
```

```
[]
# train different models belonging to Surprise package
models = [SVD(), NMF(), SlopeOne(), KNNBasic(), CoClustering()]
models_name = ['SVD', 'NMF', 'SlopeOne', 'KNNBasic', 'CoClustering']
model_rmse = pd.DataFrame()
predictions_dict = dict()
index = 0

# predict new values after training the model and get the computed RSME
for model in models:
  index += 1
  model_name = models_name[index-1]
  prediction = train_model(train, test, model)
  rmse = compute_rmse(prediction)
  predictions_dict[model_name] = rmse

model_rmse = model_rmse.append(predictions_dict, ignore_index=True)
model_rmse
```

```
[]
```

TensorFlow Recommender

```
# install tensorflow-recommenders library and update the datasets
!pip install -q tensorflow-recommenders
!pip install -q --upgrade tensorflow-datasets
```

```
# useful imports
import pandas as pd
from typing import Dict, Text
import numpy as np
import os
import pprint
import tensorflow as tf
import tensorflow_datasets as tfds
import tensorflow_recommenders as tfrs
from tensorflow.keras.layers.experimental.preprocessing import StringLo
okup
from tensorflow.keras.layers import Embedding
from tensorflow_recommenders.metrics import FactorizedTopK
from tensorflow.keras.optimizers import Adagrad, Adam
```

[]

```
def get_data():
  _ratings = tfds.load('movielens/100k-ratings', split="train")
  _movies = tfds.load('movielens/100k-movies', split="train")
  return _ratings, _movies
```

[]

```
# split data into train and test sets: 80% training, 20% test
def split_data(data):
  tf.random.set_seed(42)
  _shuffled_data = data.shuffle(100_000, seed=42, reshuffle_each_iterat
ion=False)
  _train = _shuffled_data.take(80_000)
  _test = _shuffled_data.skip(80_000).take(20_000)
  return _train, _test
```

[]

```
# cache train and test data
def cache_data(train, test):
  _train = train.shuffle(100_000).batch(8192).cache()
  _test = test.batch(4096).cache()
  return _train, _test
```

[]

```
def select_features(ratings, movies):
  _ratings = ratings.map(lambda x: {
```

```
    "movie_title": x["movie_title"],
    "user_id": x["user_id"]
  })
  _movies = movies.map(lambda x: x["movie_title"])
  return _ratings, _movies
```

```
[]
def build_vocabularies(ratings, movies):
  _user_ids_vocabulary = StringLookup(mask_token=None)
  _user_ids_vocabulary.adapt(ratings.map(lambda x: x["user_id"]))
  _movie_titles_vocabulary = StringLookup(mask_token=None)
  _movie_titles_vocabulary.adapt(movies)
  return _user_ids_vocabulary, _movie_titles_vocabulary
```

```
[]
class MovieLensModel(tfrs.Model):

  def __init__(
      self,
      user_model: tf.keras.Model,
      movie_model: tf.keras.Model,
      task: tfrs.tasks.Retrieval):
    super().__init__()

    self.user_model = user_model
    self.movie_model = movie_model
    self.task = task

  def compute_loss(self, features: Dict[Text, tf.Tensor], training=Fals
e) -> tf.Tensor:
    user_embeddings = self.user_model(features["user_id"])
    movie_embeddings = self.movie_model(features["movie_title"])

    return self.task(user_embeddings, movie_embeddings)
```

```
[]
class OverridedMovielensModel(tf.keras.Model):

  def __init__(
      self,
      user_model,
      movie_model):
    super().__init__()
```

```
    self.movie_model: tf.keras.Model = movie_model
    self.user_model: tf.keras.Model = user_model
    self.task: tf.keras.layers.Layer = task

  def train_step(self, features: Dict[Text, tf.Tensor]) -> tf.Tensor:

    # set up a gradient tape to record gradients.
    with tf.GradientTape() as tape:

      # loss computation.
      user_embeddings = self.user_model(features["user_id"])
      positive_movie_embeddings = self.movie_model(features["movie_titl
e"])
      loss = self.task(user_embeddings, positive_movie_embeddings)

      # handle regularization losses as well.
      regularization_loss = sum(self.losses)

      total_loss = loss + regularization_loss

    gradients = tape.gradient(total_loss, self.trainable_variables)
    self.optimizer.apply_gradients(zip(gradients, self.trainable_variab
les))

    metrics = {metric.name: metric.result() for metric in self.metrics}
    metrics["loss"] = loss
    metrics["regularization_loss"] = regularization_loss
    metrics["total_loss"] = total_loss

    return metrics

  def test_step(self, features: Dict[Text, tf.Tensor]) -> tf.Tensor:

    # loss computation.
    user_embeddings = self.user_model(features["user_id"])
    positive_movie_embeddings = self.movie_model(features["movie_title"
])
    loss = self.task(user_embeddings, positive_movie_embeddings)

    # handle regularization losses as well.
    regularization_loss = sum(self.losses)
```

```
    total_loss = loss + regularization_loss

    metrics = {metric.name: metric.result() for metric in self.metrics}
    metrics["loss"] = loss
    metrics["regularization_loss"] = regularization_loss
    metrics["total_loss"] = total_loss

    return metrics
```

```
[]
def build_user_model(user_ids_vocabulary):
  _user_model = tf.keras.Sequential([
            user_ids_vocabulary,
            Embedding(
                user_ids_vocabulary.vocabulary_size(),
                64)
              ])
  return _user_model

def build_movie_model(movie_titles_vocabulary):
  _movie_model = tf.keras.Sequential([
            movie_titles_vocabulary,
            Embedding(
                movie_titles_vocabulary.vocabulary_size(),
            64)
          ])
  return _movie_model

def task_objective(movies, movie_model):
  _task = tfrs.tasks.Retrieval(
            metrics=FactorizedTopK(
                  movies.batch(128).map(
                      movie_model)
                  )
            )
  return _task
```

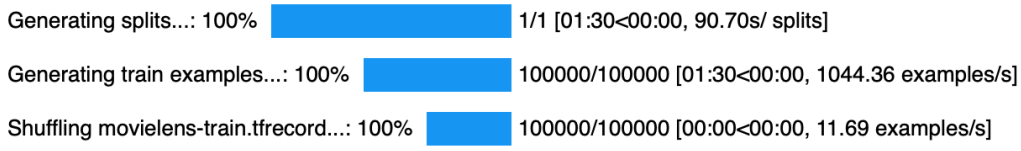
```
[]
def build_retrieval_model(user_model, movie_model, task, cached_train,
epochs):
  model = MovieLensModel(user_model, movie_model, task)
  model.compile(optimizer=Adagrad(learning_rate=0.1))
```

```
  model.fit(cached_train, epochs=epochs)
  return model
```

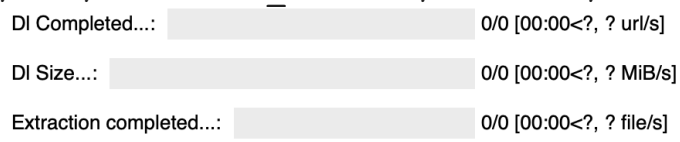
[]

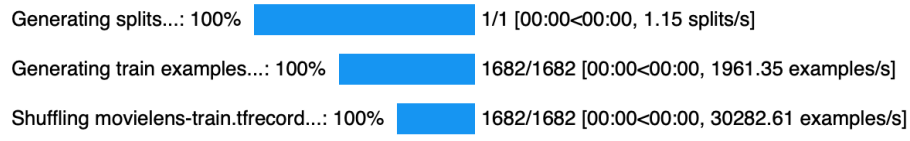
```
# get input data from tensorflow datasets
data = get_data()
```

```
Downloading and preparing dataset 4.70 MiB (download: 4.70 MiB, generated:
32.41 MiB, total: 37.10 MiB) to /root/tensorflow_datasets/movielens/100k-
ratings/0.1.0...
```




```
Dataset movielens downloaded and prepared to
/root/tensorflow_datasets/movielens/100k-ratings/0.1.0. Subsequent
calls will reuse this data.
Downloading and preparing dataset 4.70 MiB (download: 4.70 MiB,
generated: 150.35 KiB, total: 4.84 MiB) to
/root/tensorflow_datasets/movielens/100k-movies/0.1.0...
```






```
Dataset movielens downloaded and prepared to
/root/tensorflow_datasets/movielens/100k-movies/0.1.0. Subsequent calls
will reuse this data.
```

[]

```
ratings_data = data[0]
movies_data = data[1]
```

[]

```
# ratings data exploration
for i in ratings_data.take(1).as_numpy_iterator():
```

```
  pprint.pprint(i)
{'bucketized_user_age': 45.0,
 'movie_genres': array([7]),
 'movie_id': b'357',
 'movie_title': b"One Flew Over the Cuckoo's Nest (1975)",
 'raw_user_age': 46.0,
 'timestamp': 879024327,
 'user_gender': True,
 'user_id': b'138',
 'user_occupation_label': 4,
 'user_occupation_text': b'doctor',
 'user_rating': 4.0,
 'user_zip_code': b'53211'}
```

[]

```
# movie data exploration
for i in movies_data.take(1).as_numpy_iterator():
  pprint.pprint(i)
{'movie_genres': array([4]),
 'movie_id': b'1681',
 'movie_title': b'You So Crazy (1994)'}
```

[]

```
# select basic features: user_id, movie_title
ratings, movies = select_features(ratings_data, movies_data)
```

[]

```
# split data into train and test sets: 80% training, 20% test
train, test = split_data(ratings)
```

[]

```
# print some train set values
for i in train.take(5).as_numpy_iterator():
  pprint.pprint(i)
{'movie_title': b'Postman, The (1997)', 'user_id': b'681'}
{'movie_title': b'Clueless (1995)', 'user_id': b'442'}
{'movie_title': b'Maltese Falcon, The (1941)', 'user_id': b'932'}
{'movie_title': b'His Girl Friday (1940)', 'user_id': b'506'}
{'movie_title': b'Quiz Show (1994)', 'user_id': b'18'}
```

[]

```
# cache train and test data
cached_train, cached_test = cache_data(train, test)
```

[]

```
# build vocabularies to convert user ids and movie titles
# into integer indices for embedding model layers
user_ids_vocabulary, movie_titles_vocabulary = build_vocabularies(ratin
gs, movies)
```

[]

```
# define the user model
user_model = build_user_model(user_ids_vocabulary)
```

[]

```
# define the movie model
movie_model = build_movie_model(movie_titles_vocabulary)
```

[]

```
# define the task objectives
task = task_objective(movies, movie_model)
```

[]

```
# create a retrieval general model
model = build_retrieval_model(user_model, movie_model, task, cached_tra
in, 3)
```

```
Epoch 1/3
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-
packages/tensorflow/python/ops/array_ops.py:5049: calling gather (from
tensorflow.python.ops.array_ops) with validate_indices is deprecated and
will be removed in a future version.
Instructions for updating:
The `validate_indices` argument has no effect. Indices are always validated
on CPU and never validated on GPU.
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-
packages/tensorflow/python/ops/array_ops.py:5049: calling gather (from
tensorflow.python.ops.array_ops) with validate_indices is deprecated and
will be removed in a future version.
Instructions for updating:
The `validate_indices` argument has no effect. Indices are always validated
on CPU and never validated on GPU.
10/10 [==============================] - 26s 2s/step -
factorized_top_k/top_1_categorical_accuracy: 2.2500e-04 -
factorized_top_k/top_5_categorical_accuracy: 0.0019 -
factorized_top_k/top_10_categorical_accuracy: 0.0058 -
factorized_top_k/top_50_categorical_accuracy: 0.0571 -
factorized_top_k/top_100_categorical_accuracy: 0.1291 - loss: 69627.9737 -
regularization_loss: 0.0000e+00 - total_loss: 69627.9737
Epoch 2/3
10/10 [==============================] - 22s 2s/step -
factorized_top_k/top_1_categorical_accuracy: 0.0015 -
factorized_top_k/top_5_categorical_accuracy: 0.0150 -
factorized_top_k/top_10_categorical_accuracy: 0.0325 -
factorized_top_k/top_50_categorical_accuracy: 0.1562 -
factorized_top_k/top_100_categorical_accuracy: 0.2825 - loss: 67024.4595 -
regularization_loss: 0.0000e+00 - total_loss: 67024.4595
Epoch 3/3
10/10 [==============================] - 22s 2s/step -
factorized_top_k/top_1_categorical_accuracy: 0.0016 -
factorized_top_k/top_5_categorical_accuracy: 0.0212 -
factorized_top_k/top_10_categorical_accuracy: 0.0449 -
factorized_top_k/top_50_categorical_accuracy: 0.1977 -
factorized_top_k/top_100_categorical_accuracy: 0.3336 - loss: 65622.7805 -
regularization_loss: 0.0000e+00 - total_loss: 65622.7805
```

[]

```
# evaluate the model on test set
model.evaluate(cached_test, return_dict=True)
```

```
5/5 [==============================] - 7s 849ms/step -
factorized_top_k/top_1_categorical_accuracy: 3.0000e-04 -
factorized_top_k/top_5_categorical_accuracy: 0.0045 -
factorized_top_k/top_10_categorical_accuracy: 0.0131 -
factorized_top_k/top_50_categorical_accuracy: 0.1015 -
factorized_top_k/top_100_categorical_accuracy: 0.2113 - loss: 31256.1960 -
regularization_loss: 0.0000e+00 - total_loss: 31256.1960
{'factorized_top_k/top_100_categorical_accuracy': 0.2113499939441681,
 'factorized_top_k/top_10_categorical_accuracy': 0.013050000183284283,
 'factorized_top_k/top_1_categorical_accuracy': 0.0003000000142492354,
 'factorized_top_k/top_50_categorical_accuracy': 0.1014999970793724,
 'factorized_top_k/top_5_categorical_accuracy': 0.0044999998062849045,
 'loss': 28389.564453125,
 'regularization_loss': 0,
 'total_loss': 28389.564453125}
```

MovieLens 1m

LensKit

```
# install lenskit package
%pip install lenskit
```

```
Requirement already satisfied: lenskit in /usr/local/lib/python3.7/dist-packages (0.12.3)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.7/dist-packages (from lenskit) (1.1.5)
Requirement already satisfied: csr<0.3,>=0.2 in /usr/local/lib/python3.7/dist-packages (from lenskit) (0.2.0)
Requirement already satisfied: numba<0.53,>=0.51 in /usr/local/lib/python3.7/dist-packages (from lenskit) (0.51.2)
Requirement already satisfied: scipy>=1.2 in /usr/local/lib/python3.7/dist-packages (from lenskit) (1.4.1)
Requirement already satisfied: cffi>=1.12.2 in /usr/local/lib/python3.7/dist-packages (from lenskit) (1.14.5)
Requirement already satisfied: pyarrow>=0.15 in /usr/local/lib/python3.7/dist-packages (from lenskit) (3.0.0)
Requirement already satisfied: binpickle>=0.3.2 in /usr/local/lib/python3.7/dist-packages (from lenskit) (0.3.3)
Requirement already satisfied: numpy>=1.16 in /usr/local/lib/python3.7/dist-packages (from lenskit) (1.19.5)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24->lenskit)
(2018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.24-
>lenskit) (2.8.1)
Requirement already satisfied: llvmlite<0.35,>=0.34.0.dev0 in /usr/local/lib/python3.7/dist-packages (from
numba<0.53,>=0.51->lenskit) (0.34.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (from numba<0.53,>=0.51-
>lenskit) (56.1.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-packages (from cffi>=1.12.2->lenskit)
(2.20)
Requirement already satisfied: pickle5; python_version < "3.8" in /usr/local/lib/python3.7/dist-packages (from
binpickle>=0.3.2->lenskit) (0.0.11)
Requirement already satisfied: msgpack>=1.0 in /usr/local/lib/python3.7/dist-packages (from binpickle>=0.3.2-
>lenskit) (1.0.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3-
>pandas>=0.24->lenskit) (1.15.0)
```

[ ]

```
# useful libraries
from lenskit.datasets import MovieLens
from lenskit.algorithms.bias import Bias
from lenskit.batch import predict
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import pandas as pd
import numpy as np
```

[ ]

```
def get_data(choose):
  _data = pd.DataFrame()
  _url = ''
```

```
  # 100K dataset
  if choose == 1:
    _url = 'https://drive.google.com/file/d/1lTzGIc0uonkBiDM-qjueTfFyb8YRH4L1/view?usp=sharing'
  # 1M dataset: 1 milione di righe
  if choose == 2:
    _url = 'https://drive.google.com/file/d/1kPi2tM1rCmZgYDwq1lzitRDT-bDiZr3k/view?usp=sharing'
  _file_id = _url.split('/')[-2]
  _dwn_url ='https://drive.google.com/uc?id=' + _file_id
  _data = pd.read_csv(_dwn_url, sep=';')
  _data = _data.sample(frac=1).reset_index(drop=True)
  return _data
```

[]

```
def split_data(data, test_size):
  _test, _train = train_test_split(data, test_size=test_size)
  return _test, _train
```

[]

```
def train_model(train, test, model):
  model.fit(train)
  _prediction = predict(model, test)
  return _prediction
```

[]

```
def compute_rmse(data):
  _rmse = mean_squared_error(data['prediction'], data['rating'], squared=False)
  return _rmse
```

[]

```
# get input dataset
ratings = get_data(2) # 1M records
ratings.columns = ['user', 'item', 'rating', 'timestamp']
ratings.head()
```

	user	item	rating	timestamp
0	474.0	7414.0	2.5	1.080569e+09
1	232.0	32298.0	3.0	1.218170e+09
2	294.0	518.0	2.0	9.665967e+08
3	297.0	70.0	2.0	9.008720e+08
4	91.0	4105.0	4.0	1.112712e+09

[]

```
len(ratings)
1008360
```

[]

```
# split data into train and test sets: 80% training, 20% test
train, test = split_data(ratings, 0.2)
```

[]

```
# shape of train set
train.shape
(806688, 4)
```

[]

```
# shape of train set
test.shape
(201672, 4)
```

[]

```
# sanity check
len(train) + len(test) == len(ratings)
True
```

[]

```
# train the model
prediction = train_model(train, test, Bias())
```

[]

```
# print the first 5 prediction results
prediction.head()
```

	user	item	rating	timestamp	prediction
533369	291.0	1.0	4.0	1.453052e+09	4.414183
533369	291.0	1.0	4.0	1.453052e+09	4.414183
533369	291.0	1.0	4.0	1.453052e+09	4.414183
533369	291.0	1.0	4.0	1.453052e+09	4.414183
533369	291.0	1.0	4.0	1.453052e+09	4.414183

[]

```
# compute the rmse between predicted and actual values
rmse = compute_rmse(prediction)

rmse
0.802027068669124
```

LightFM

```
# install LightFM library
!pip install lightfm
```

```
Collecting lightfm
  Downloading
https://files.pythonhosted.org/packages/5e/fe/8864d723daa8e5afc74080ce510c30f7ad52facf6a157d4b42dec83dfab4/lightfm-
1.16.tar.gz (310kB)
     |████████████████████████████████| 317kB 3.9MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from lightfm) (1.19.5)
Requirement already satisfied: scipy>=0.17.0 in /usr/local/lib/python3.7/dist-packages (from lightfm) (1.4.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from lightfm) (2.23.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from lightfm) (0.22.2.post1)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->lightfm)
(2020.12.5)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->lightfm)
(2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages
(from requests->lightfm) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->lightfm)
(3.0.4)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->lightfm)
(1.0.1)
Building wheels for collected packages: lightfm
  Building wheel for lightfm (setup.py) ... done
  Created wheel for lightfm: filename=lightfm-1.16-cp37-cp37m-linux_x86_64.whl size=705344
sha256=c8e568eb9ae71d2ea2aa76857288b6e6d0ff61d44554a82ff4bc22c37833050a
  Stored in directory: /root/.cache/pip/wheels/c6/64/d4/673c7277f71ac4c5ad4835b94708c01b653ef2d3aa78ef20aa
Successfully built lightfm
Installing collected packages: lightfm
Successfully installed lightfm-1.16
```

[ ]

```
def get_data(choose):
  data = pd.DataFrame()
  # 100K dataset
  if choose == 1:
    url='https://drive.google.com/file/d/1lTzGIc0uonkBiDM-
qjueTfFyb8YRH4L1/view?usp=sharing'
    file_id = url.split('/')[-2]
    dwn_url ='https://drive.google.com/uc?id=' + file_id
    data = pd.read_csv(dwn_url)
  # 1M dataset: 1 milione di righe
  if choose == 2:
    # download largest MovieLens dataset (1 billion of records)
    data = fetch_movielens()
  return data
```

[ ]

```
def split_data(data):
  _train, _test = train = data['train'], data['test']
  return _train, _test
```

[ ]

```
def train_model(train, test, model, epochs, k):
  model.fit_partial(train, epochs=epochs)
  _train_precision = precision_at_k(model, train, k=k).mean()
  _test_precision = precision_at_k(model, test, k=k, train_interactions
=train).mean()
```

```
  return _train_precision, _test_precision
```

[]

```
def compute_auc(model, train, test):
  _train_auc = auc_score(model, train).mean()
  _test_auc = auc_score(model, test).mean()
  return _train_auc, _test_auc
```

[]

```
# get input dataset
ratings = get_data(2) # 1M records
ratings
```

```
{'item_feature_labels': array(['Toy Story (1995)', 'GoldenEye (1995)',
'Four Rooms (1995)', ...,
       'Sliding Doors (1998)', 'You So Crazy (1994)',
       'Scream of Stone (Schrei aus Stein) (1991)'], dtype=object),
 'item_features': <1682x1682 sparse matrix of type '<class
'numpy.float32'>'
   with 1682 stored elements in Compressed Sparse Row format>,
 'item_labels': array(['Toy Story (1995)', 'GoldenEye (1995)', 'Four Rooms
(1995)', ...,
       'Sliding Doors (1998)', 'You So Crazy (1994)',
       'Scream of Stone (Schrei aus Stein) (1991)'], dtype=object),
 'test': <943x1682 sparse matrix of type '<class 'numpy.int32'>'
   with 9430 stored elements in COOrdinate format>,
 'train': <943x1682 sparse matrix of type '<class 'numpy.int32'>'
   with 90570 stored elements in COOrdinate format>}
```

[]

```
# split data into train and test sets
train, test = split_data(ratings)
```

[]

```
# lenght of train set
train.shape[0] * train.shape[1]
```

```
1586126
```

[]

```
# lenght of test set
test.shape[0] * test.shape[1]
```

```
1586126
```

[]

```
# train the model and get precisions
model = LightFM(learning_rate=0.05, loss='warp')
train_precision, test_precision = train_model(train, test, model, 20, 2
0)
```

[]

```
# compute the area under curve (auc)
```

```
train_auc, test_auc = compute_auc(model, train, test)
```

[]

```
print('Precision: train %.2f, test %.2f.' % (train_precision, test_prec
ision))
print('AUC: train %.2f, test %.2f.' % (train_auc, test_auc))
```

```
Precision: train 0.59, test 0.18.
AUC: train 0.94, test 0.91.
```

Surprise

```
# install the surprise package
%pip install surprise
```

```
Collecting surprise
  Downloading https://files.pythonhosted.org/packages/61/de/e5cba8682201fcf9c3719a6fdda95693468ed061945493dea2dd37c5618b/surprise-0.1-
py2.py3-none-any.whl
Collecting scikit-surprise
  Downloading https://files.pythonhosted.org/packages/97/37/5d334adaf5ddd65da99fc65f6507e0e4599d092ba048f4302fe8775619e8/scikit-surprise-
1.1.1.tar.gz (11.8MB)
     |████████████████████████████████| 11.8MB 28.8MB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.0.1)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.19.5)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.4.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise->surprise) (1.15.0)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp37-cp37m-linux_x86_64.whl size=1617650
sha256=14faf26ca1898b8fdd1da0e111ddf8f8728bf57b8396764875dccca27b046856
  Stored in directory: /root/.cache/pip/wheels/78/9c/3d/41b419c9d2aff5b6e2b4c0fc8d25c538202834058f9ed110d0
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.1 surprise-0.1
```

[]

```
# useful libraries
from surprise import SVD, NMF, SlopeOne, KNNBasic, CoClustering
from surprise.model_selection import cross_validate
from surprise import Reader
from surprise import Dataset
from surprise import accuracy
import pandas as pd
from surprise.model_selection import train_test_split
```

[]

```
def get_data(choose):
  _data = pd.DataFrame()
  _url = ''
  # 100K dataset
  if choose == 1:
    _url = 'https://drive.google.com/file/d/1lTzGIc0uonkBiDM-
qjueTfFyb8YRH4L1/view?usp=sharing'
  # 1M dataset: 1 milione di righe
  if choose == 2:
    _url = 'https://drive.google.com/file/d/1BvLz4mbbyry75yPBuqS1e_Rn_Q
zGLNHK/view?usp=sharing'
  _file_id = _url.split('/')[-2]
```

```
  _dwn_url ='https://drive.google.com/uc?id=' + _file_id
  _data = pd.read_csv(_dwn_url, sep=';')
  _data = _data.sample(frac=1).reset_index(drop=True)
  return _data
```

[]

```
def split_data(data, test_size):
  _train, _test = train_test_split(data, test_size=test_size)
  return _train, _test
```

[]

```
def format_data(data, columns):
  _reader = Reader(rating_scale=(1, 5))
  _data = Dataset.load_from_df(data[columns], _reader)
  return _data
```

[]

```
def train_model(train, test, model):
  model.fit(train)
  _prediction = model.test(test)
  return _prediction
```

[]

```
def compute_rmse(prediction):
  _rmse = accuracy.rmse(prediction)
  return _rmse
```

[]

```
# load ratings dataset
ratings = get_data(2) # 1M records
ratings.columns = ['user', 'item', 'rating', 'timestamp']
# print the first 5 rows
ratings.head()
```

[]

```
# format data to be readeable from surprise library
ratings = format_data(ratings, ['user', 'item', 'rating'])
```

[]

```
# split data into train and test sets: 80% training, 20% test of total
observations
train, test = split_data(ratings, 0.2)
```

[]

```
# train different models belonging to Surprise package
models = [SVD(), NMF(), SlopeOne(), KNNBasic(), CoClustering()]
models_name = ['SVD', 'NMF', 'SlopeOne', 'KNNBasic', 'CoClustering']
model_rmse = pd.DataFrame()
predictions_dict = dict()
index = 0
# batch size
k = 0
# predict new values after training the model and get the computed RSME
for model in models:
  index += 1
  model_name = models_name[index-1]
  prediction = train_model(train, test, model)
  print(prediction)
  rmse = compute_rmse(prediction)
  predictions_dict[model_name] = rmse.mean()

model_rmse = model_rmse.append(predictions_dict, ignore_index=True)
model_rmse
```

[]

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)

Done computing similarity matrix.
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

	CoClustering	KNNBasic	NMF	SVD	SlopeOne
0	0.855808	0.725167	0.693429	0.520548	0.661144

BIBLIOGRAFIA

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., & Dean, J. et al. (2016). *TensorFlow: A System for Large-Scale Machine Learning*. Usenix.org. Retrieved from https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. (2007). *Google news personalization: scalable online collaborative filtering*. In Proceedings of the 16th International Conference on World Wide Web (WWW'07). ACM, New York, NY, 271–280. DOI:http://dx.doi.org/10.1145/1242572.1242610

Ameen, N., Tarhini, A., Reppel, A. and Anand, A., (2021). *Customer experiences in the age of artificial intelligence.* Computers in Human Behavior, (114).

Ashok Mahant, M., & Pellakuri, V. (2021). *Innovative supervised machine learning techniques for classification of data. Materials Today: Proceedings.* DOI:https://doi.org/10.1016/j.matpr.2020.11.092

Aggarwal S., Goswami D., Hooda M., Chakravarty A., Kar A., Vasudha (2020) *Recommendation Systems for Interactive Multimedia Entertainment*. In: Hemanth J., Bhatia M., Geman O. (eds) Data Visualization and Knowledge Engineering. Lecture Notes on Data Engineering and Communications Technologies, vol 32. Springer, Cham. DOI:https://doi.org/10.1007/978-3-030-25797-2_2

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. (2000). *Application of Dimensionality Reduction in Recommender System—A Case Study.* Technical Report. DTIC Document. Retrieved from https://bit.ly/3cb7RGT

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. (2001). *Item-based collaborative filtering recommendation algorithms.* In Proceedings of the 10th International Conference on World Wide Web (WWW'01). ACM, New York, NY, 285–295. DOI:http://dx.doi.org/10.1145/371920.372071

Buunk, Bram and Thomas Mussweiler. (2001). "*New Directions in Social Comparison Research*," European Journal of Social Psychology, 31(5): 467–475.

Chen, C. and Cheng, C., (2009). *Understanding consumer intention in online shopping: a respecification and validation of the DeLone and McLean model*. Behaviour & Information Technology, 28(4), pp.335-345. DOI: 10.1080/01449290701850111

Campello, R., da Costa, A., Fressato, E., Neto, F., & Manzato, M. (2018). *Case recommender | Proceedings of the 12th ACM Conference on Recommender Systems*.

Dl.acm.org. Retrieved 23 May 2021, from https://dl.acm.org/doi/pdf/10.1145/3240323.3241611

Casagranda, P., & Metta, S. (2016). *Leggi questo articolo, una tua amica lo ha trovato interessante | Un'introduzione alle opportunità e criticità dei recommender system per la personalizzazione dei contenuti audiovisivi.* Crit.rai.it. Retrieved from http://www.crit.rai.it/eletel/2016-2/162-4.pdf

Chen Y., Harper F.M., Konstan J., Xin Li S. (2010), *"Social comparisons and contributions to online communities: a field experiment on MovieLens"*, The American Economic Review, Vol. 100, n.4

Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl. (2003). *Is seeing believing?: How recommender system interfaces affect users' opinions.* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'03). ACM, New York, NY, 585–592. DOI:http://dx.doi.org/10.1145/642611.642713

Delmedico, A. (2019). *Sistemi di raccomandazione e comportamento del consumatore: confronto tra user-based e item- based collaborative filtering* (Laurea Magistrale). Luiss Guido Carli.

Dwivedi, S., & Roshni, V. K. (2017). *Recommender system for big data in education.* In 2017 5th National Conference on E-Learning & E-Learning Technologies (ELELTECH) (pp. 1-4). IEEE. Retrieved from https://ieeexplore.ieee.org/abstract/document/8074993

Festinger, Leon. 1954. *"A Theory of Social Comparison*," Human Relations, 7: 117–140.

George Karypis. (2001). *Evaluation of item-based top-N recommendation algorithms*. In Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM'01). ACM, New York, NY, 247–254. DOI:http://dx.doi.org/10.1145/502585.502627

Gosmar, D. (2021). *Machine Learning: il sesto chakra dell'intelligenza artificiale* (3rd ed.). Independently published.

Guzzi, P. (2019). Computing Languages for Bioinformatics: Python. Encyclopedia Of Bioinformatics And Computational Biology, 1(195-198). https://doi.org/https://doi.org/10.1016/B978-0-12-809633-8.20366-X

Hao, J., & Ho, T. (2019). *Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language.* Journal of Educational and Behavioral Statistics. Retrieved 18 May 2021, from https://doi.org/10.3102/1076998619832248

Harper, F. and Konstan, J., (2016). *The MovieLens Datasets*. ACM Transactions on Interactive Intelligent Systems, 5(4), pp.1-19. DOI: 10.1145/2827872

Hu, W., Chen, C., Ye, F., Zheng, Z., & Du, Y. (2021). *Learning deep discriminative representations with pseudo supervision for image clustering*. Information Sciences, 568, 199-215. https://doi.org/10.1016/j.ins.2021.03.066

Hug, N., (2020). Surprise: *A Python library for recommender systems.* Journal of Open Source Software, 5(52), 2174. https://doi.org/10.21105/joss.02174

Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). *Recommendation systems: Principles, methods and evaluation*. Egyptian Informatics Journal, 16(3), 261-273. DOI:https://doi.org/https://doi.org/10.1016/j.eij.2015.06.005

Ives, B., Olson, M. and Baroudi, J., (1983). *The measurement of user information satisfaction*. Communications of the ACM, 26(10), pp.785-793. DOI: 10.1145/358413.358430

Jafari-Marandi, R. (2021). *Supervised or unsupervised learning? Investigating the role of pattern recognition assumptions in the success of binary predictive prescriptions*. Neurocomputing, 434, 165-193. DOI: https://doi.org/10.1016/j.neucom.2020.12.063

John O'Donovan and Barry Smyth. (2005). *Trust in recommender systems*. In Proceedings of the 10th International Conference on Intelligent User Interfaces (IUI'05). ACM, New York, NY, 167–174. DOI:http://dx.doi.org/10.1145/1040830.1040870

Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. (1997). *GroupLens: Applying collaborative filtering to Usenet news.* Communications of the ACM 40, 3, 77–87. DOI:http://dx.doi.org/10.1145/245108.245126

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

Kumar, V. and Ayodeji, O., (2021). *E-retail factors for customer activation and retention: An empirical study from Indian e-commerce customers.* Journal of Retailing and Customer Services, (Volume 59).

Lang, F., Liang, L., Huang, K., Chen T., Zhu S. (2021). *Movie Recommendation System for Educational Purposes Based on Field-Aware Factorization Machine*. Mobile Netw Appl. DOI: https://doi.org/10.1007/s11036-021-01775-9

Liu, F., Du, P., Weng, F., Qu, J., (2007). *Use clustering to improve neural network in financial time series prediction.* Third International Conference on Natural Computation (ICNC 2007). IEEE, pp. 89-93.

Mao, Y., Mokhov, S. A., & Mudur, S. P. (2021). *Application of Knowledge Graphs to Provide Side Information for Improved Recommendation Accuracy.* Retrievede from https://arxiv.org/pdf/2101.03054.pdf

Michael D. Ekstrand. (2020). *LensKit for Python: Next-Generation Software for Recommender Systems Experiments*. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM '20). Association for Computing Machinery, New York, NY, USA, 2999–3006. DOI: https://doi.org/10.1145/3340531.3412778

Michael D. Ekstrand, Daniel Kluver, F. Maxwell Harper, and Joseph A. Konstan. (2015). *Letting users choose recommender algorithms: An experimental study*. In Proceedings of the 9th ACM Conference on Recommender Systems (RecSys'15). ACM, New York, NY, 11–18. DOI:http://dx.doi.org/10.1145/2792838.2800195

Mukund Deshpande and George Karypis. (2004). *Item-based top-N recommendation algorithms*. ACM Transactions on Information Systems 22, 1, 143–177. DOI:http://dx.doi.org/10.1145/963770.963776

Nick Pentreath. (2015). *Machine Learning with Spark*. Packt Publishing Ltd, Birmingham, UK

Noguera Torres, A. D. P. (2021). *Sistema de recomendación de matrícula de cursos electivos para estudiantes de Ingeniería Electrónica e Ingeniería de Telecomunicaciones de la UNAD.* Universidad Nacional Abierta y a Distancia UNAD, Yopal, Retrieved from https://repository.unad.edu.co/bitstream/handle/10596/40465/adnoguerat.pdf?sequence=1

Ortega, F., Mayor, J., López-Fernández, D., & Lara-Cabrera, R. (2021). CF4J 2.0: *Adapting Collaborative Filtering for Java to new challenges of collaborative filtering based recommender systems. Knowledge-Based Systems*, 215. https://doi.org/https://doi.org/10.1016/j.knosys.2020.106629

Pappas, I., Kourouthanassis, P., Giannakos, M. and Chrissikopoulos, V., (2016). *Explaining online shopping behavior with fsQCA: The role of cognitive and affective perceptions*. Journal of Business Research, 69(2), pp.794-803. DOI: https://doi.org/10.1016/j.jbusres.2015.07.010

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. (1994). *GroupLens: An open architecture for collaborative filtering of Netnews.* In Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW'94). ACM, New York, NY, 175–186. DOI:http://dx.doi.org/10.1145/192844.192905

Paolo Massa and Paolo Avesani. (2007). *Trust-aware recommender systems*. In Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys'07). ACM, New York, NY, 17–24. DOI:http://dx.doi.org/10.1145/1297231.1297235

Paullier, A. and Sotelo, R., (2020). *A Recommender Systems' algorithm evaluation using the Lenskit library and MovieLens database*s. 2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), DOI:10.1109/bmsb49480.2020.9379914

Raschka, S., Patterson, J., & Nolet, C. (2020). *Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence*. Information, 11(4), 193. MDPI AG. Retrieved from http://dx.doi.org/10.3390/info11040193

Rashidi, R., Khamforoosh, K., & Sheikhahmadi, A. (2020). *An analytic approach to separate users by introducing new combinations of initial centers of clustering*. Physica A: Statistical Mechanics And Its Applications, 551. Retrieved from https://doi.org/https://doi.org/10.1016/j.physa.2020.124185

Ricci, F., Rokach, L., & Shapira, B. (2011). *Recommender Systems Handbook.*

Sharma, N., Sharma, R., & Jindal, N. (2021). *Machine Learning and Deep Learning Applications-A Vision*. Global Transitions Proceedings, 2, 24-28. DOI: https://doi.org/10.1016/j.gltp.2021.01.004

Shinde P. P., Shah S., (2018) *A review of machine learning and deep learning applications*. Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).

Subramaniyaswamy, V., Logesh, R., Chandrashekhar, M., Challa, A., & Vijayakumar, V. (2017). *A personalised movie recommendation system based on collaborative filtering*. International Journal Of High Performance Computing And Networking, 10(1/2), 54. DOI:https://doi.org/10.1504/ijhpcn.2017.083199

Suls, Jerry, Rene Martin, and Ladd Wheeler. (2002). "*Social Comparison: Why, with Whom, and with What Effect?,*" Current Directions in Psychological Science, 11: 159–163.

Toby Segaran. (2007). *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly Media, Inc., Sebastopol, CA.

Tsai, H. and Huang, H., (2007). *Determinants of e-repurchase intentions: An integrative model of quadruple retention drivers.* Information & Management, 44(3), pp.231-239. DOI:10.1016/j.im.2006.11.006

Tyrväinen, O., Karjaluoto, H. and Saarijärvi, H., (2020). *Personalization and hedonic motivation in creating customer experiences and loyalty in omnichannel retail*. Journal of Retailing and Consumer Services, (57).

Vanam, M., Amirali Jiwani, B., Swathi, A., & Madhavi, V. (2021). *High performance machine learning and data science based implementation using Weka. Materials Today: Proceedings.* DOI:https://doi.org/10.1016/j.matpr.2021.01.470

Vitale F. (2019). *Recommendation System: Implementazione Di Algoritmi Collaborativi Per Servizi Video-Streaming On Demand (VOD).* Tesi Luiss Guido Carli. Retrieved from http://tesi.luiss.it/25923/1/692341_VITALE_FABRIZIO.pdf

Walek, B., & Fojtik, V. (2020). *A hybrid recommender system for recommending relevant movies using an expert system.* Expert Systems With Applications, (158). DOI:https://doi.org/10.1016/j.eswa.2020.113452

Wellmann, J., Croucher, A., & Regenauer-Lieb, K. (2012). *Python scripting libraries for subsurface fluid and heat flow simulations with TOUGH2 and SHEMAT*. Computers & Geosciences, 43, 197-206. DOI:https://doi.org/https://doi.org/10.1016/j.cageo.2011.10.011

Wette, K. (2020). *SWIGLAL: Python and Octave interfaces to the LALSuite gravitational-wave data analysis libraries*. Softwarex, 12. DOI:https://doi.org/https://doi.org/10.1016/j.softx.2020.100634

Wu, P., Chang, X., Yuan, W., Sun, J., Zhang, W., Arcucci, R., & Guo, Y. (2021). *Fast data assimilation (FDA): Data assimilation by machine learning for faster optimize model state*. Journal Of Computational Science, 51. DOI: https://doi.org/10.1016/j.jocs.2021.101323

Xu, Y., Zhou, Y., Sekula, P., & Ding, L. (2021). *Machine learning in construction: From shallow to deep learning*. Developments In The Built Environment, 6. DOI:https://doi.org/10.1016/j.dibe.2021.100045

Zhang, T., 2021. *The Value of IT-Enabled Retailer Learning: Can Personalized Product Recommendations (PPRs) Improve Customer Store Loyalty in Electronic Markets?*. Hdl.handle.net. Retrieved from http://hdl.handle.net/1903/2820

SITOGRAFIA

Bellec, A. (2020). *Le 4 fasi di una personalizzazione di successo*. Kameleoon. Retrieved 6 May 2021, from https://www.kameleoon.com/it/blog/4-fasi-personalizzazione-successo

Bernardini, F. (2021). *Che cosa è il recommender system: il caso Netflix*. Neuragate. Retrieved 27 April 2021, from https://www.neuragate.it/intelligenza-artificiale/che-cosa-e-il-recommender-system-il-caso-netflix/

Boise State University. *Loading Data — LensKit*. Lkpy.readthedocs.io. (2019). Retrieved 1 June 2021, from https://lkpy.readthedocs.io/en/stable/datasets.html#movielens-data-sets

Choudhury, A. (2020). *Top Open Source Recommender Systems In Python For Your ML Project*. Analytics India Magazine. Retrieved 25 May 2021, from https://analyticsindiamag.com/top-open-source-recommender-systems-in-python-for-your-ml-project/

Cisco. (2020). *Cisco Annual Internet Report* (2018–2023). White Paper. Retrieved 30 April 2021, from https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

Clayton, R. (2021). *Cos'è il machine learning?*. Oracle.com. Retrieved 24 April 2021, from https://www.oracle.com/it/data-science/machine-learning/what-is-machine-learning/

Colab. *The MovieLens Dataset*. Colab.research.google.com. Retrieved 31 May 2021, from https://bit.ly/3vaJ1ON

CRM Team (2021). *Machine Learning applicato al Marketing*. Retrieved 26 April 2021, from https://www.crmteam.it/machine-learning-applicato-al-marketing/

Free journal (2020). *Sistema di raccomandazione - software libero Che cos'è*. ww.it.freejournal.info. Retrieved 7 May 2021, from https://amp.ww.it.freejournal.info/6966909/1/sistema-di-raccomandazione.html

Google Colaboratory. Colab.research.google.com. Retrieved 10 June 2021, from https://colab.research.google.com/.

Google Developers. *Classification: ROC Curve and AUC | Machine Learning Crash Course*. Retrieved 1 June 2021, from https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Govoni, L. *L'Overfitting e l'Underfitting nel machine learning* | Lorenzo Govoni. Lorenzo Govoni. Retrieved 1 June 2021, from https://www.lorenzogovoni.com/overfitting-e-underfitting-machine-learning/

Govoni, L. *Machine Learning e principio di funzionamento*. Lorenzo Govoni Business e Tecnologia. Retrieved 25 April 2021, from https://www.lorenzogovoni.com/machine-learning-e-funzionamento/

Hug, N. (2015). *prediction_algorithms package — Surprise*. Surprise.readthedocs.io. Retrieved 1 June 2021, from https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html

Ichi.pro. *Approfondisci il sistema di raccomandazione di Netflix*. Retrieved 5 May 2021, from https://ichi.pro/it/approfondisci-il-sistema-di-raccomandazione-di-netflix-57277795941192

Intelligenza Artificiale. *Machine learning (apprendimento automatico) - Cos'è e come funziona?*. Retrieved 20 April 2021, from https://www.intelligenzaartificiale.it/machine-learning/

Kameleoon. *Cos'è la personalizzazione*?. Retrieved 6 May 2021, from https://www.kameleoon.com/it/personalizzazione#personalization%20types

Kemp, S. (2020). *Digital 2020: October Global Statshot*. Data reportal. Retrieved 30 April 2021, from https://datareportal.com/reports/digital-2020-october-global-statshot

Kirk J., 2017. *TensorRec: A Recommendation Engine Framework in TensorFlow.* Medium. Retrieved 23 May 2021, from https://medium.com/hackernoon/tensorrec-a-recommendation-engine-framework-in-tensorflow-d85e4f0874e8

LensKit — LensKit 0.12.3 documentation. Lkpy.readthedocs.io. Retrieved 10 June 2021, from https://lkpy.readthedocs.io/en/stable/.

Leonardi, A. (2019). *Il valore del Machine Learning nel Marketing.* AI4Business. Retrieved 26 April 2021, from https://bit.ly/2TwY1ZA

Meng, M. (2020). *Movie Recommendation System based on MovieLens*. Towards data science. Retrieved 22 May 2021, from https://towardsdatascience.com/movie-recommendation-system-based-on-movielens-ef0df580cd0e

Minini, A. *Apprendimento senza supervisione*. Andreaminini.com. Retrieved 23 April 2021, from http://www.andreaminini.com/ai/machine-learning/apprendimento-senza-supervisione

Minini, A. *Le librerie e i moduli del linguaggio Python*. Retrieved 21 May 2021, from https://bit.ly/3bChU7D

Moody, J. (2019). *What does RMSE really mean?.* Towards Data Science. Retrieved 1 June 2021, from https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e

MovieLens. GroupLens. Retrieved 1 June 2021, from https://grouplens.org/datasets/movielens/

MovieLens. *Movielens.org.* Retrieved 1 June 2021, from https://movielens.org/

Nardini, D. (2019). *Le 5 migliori librerie di Python per il Machine Learning*. Pulp Learning - Tutto sul Machine Learning. Retrieved 11 May 2021, from https://bit.ly/2QtzPpP

Narkhede, S. (2018). *Understanding AUC - ROC Curve*. Towards data science. Retrieved 1 June 2021, from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Netflix. *Funzionamento del sistema di consigli di Netflix*. Retrieved 5 May 2021, from https://help.netflix.com/it/node/100639

Provino, A. (2020). *Clustering: tutto quello che devi sapere | Edizione 2021*. Machine Learning & Data Science Blog. Retrieved 23 April 2021, from https://andreaprovino.it/clustering/

Retta, L. *The power of collaborative filtering*. Dynamic Yield. Retrieved 7 May 2021, from https://www.dynamicyield.com/lesson/collaborative-filtering/

Rocket To Ride. (2019). *I sistemi di raccomandazione* - Recommender System. Retrieved 27 May 2021, from http://www.rockettoride.com/r-tutorial-archive/i-sistemi-di-raccomandazione-recommender-system

Santucci, U. *Sistema di raccomandazion*e. Retrieved 3 May 2021, from http://www.umbertosantucci.it/atlante/sistema-di-raccomandazione/

Sas. *Machine Learning: che cos'è e perché è importante*. Sas.com. Retrieved 25 April 2021, from https://www.sas.com/it_it/insights/analytics/machine-learning.html

Shekhar, Y. (2020). *ML - Content Based Recommender System*. GeeksforGeeks. Retrieved 6 May 2021, from https://www.geeksforgeeks.org/ml-content-based-recommender-system/

Stecher, M. (2018). *La storia dell'intelligenza artificiale, da Turing ad oggi*. CyberLaws. Retrieved 20 April 2021, from https://www.cyberlaws.it/2018/la-storia-dellintelligenza-artificiale-da-turing-ad-oggi/

Tarkoff, R. *How AI/ML are Driving Customer Experience and the Experience Economy* - The Six Five Summit. The Six Five Summit. Retrieved 25 April 2021, from https://thesixfivesummit.com/session/how-ai-ml-are-driving-customer-experience-and-the-experience-economy/

TensorFlow. Retrieved 10 June 2021, from https://www.tensorflow.org.

Tre., L. (2021). *Con l'intelligenza artificiale i profitti aziendali possono crescere anche del 20%.* Ilsole24ore.com. Retrieved 26 April 2021, from https://www.ilsole24ore.com/art/con-l-intelligenza-artificiale-profitti-aziendali-possono-crescere-anche-20percento-ADetU0GB

Velocci, S. (2020). *Machine Learning: Cos'è e perché è Importante.* start2impact. Retrieved 17 April 2021, from https://www.start2impact.it/blog/programmazione/cose-machine-learning/

Welcome to LightFM's documentation! — LightFM 1.15 documentation. Making.lyst.com. Retrieved 10 June 2021, from https://making.lyst.com/lightfm/docs/home.html.

Welcome to Python.org. Python.org. Retrieved 10 June 2021, from https://www.python.org.

Wikipedia. *Sistema di raccomandazione*. It.wikipedia.org. Retrieved 4 May 2021, from https://bit.ly/2S4CB5s

Wikipedia. *Python (programming language)*. The Reader Wiki, Reader View of Wikipedia. Retrieved 18 May 2021, from https://thereaderwiki.com/en/Python_(programming_language)
https://lkpy.readthedocs.io/en/stable/

LUISS

Dipartimento
di Impresa e Management

Cattedra Machine Learning & Object Driven Marketing

The power of personalization: Un confronto sperimentale sulle previsioni del dataset MovieLens

Prof. Luigi Laura
RELATORE

Prof. Marco Querini
CORRELATORE

Ilaria Ampola Matr.727501
CANDIDATO

Anno Accademico 2020/2021

INDICE

CAPITOLO 1. MACHINE LEARNING……..120
CAPITOLO 2. RECOMMENDATION SYSTEM……. 122
CAPITOLO 3. MOVIELENS: CONFRONTO SPERIMENTALE TRA LIBRERIE PYTHON…….126
BIBLIOGRAFIA……135
SITOGRAFIA…….137

RIASSUNTO

1. MACHINE LEARNING

Ad oggi, lo sviluppo dell'AI e del Machine Learning deriva dalla grande quantità di dati disponibili, cresciuti considerevolmente in maniera esponenziale insieme alla capacità di elaborazione computazionale, rendendo possibile il verificarsi di complicati calcoli in tempi ragionevoli. Tali dati dell'utente derivano da qualsiasi attività compiuta in rete, da profili social, ricerche, elementi salvati e sono proprio questi la fonte principale dello sviluppo economico.
Di conseguenza, il Machine Learning e l'intelligenza artificiale diventano essenziali per effettuare predizioni che possono rivelarsi molto utili per tutta la comunità mondiale.
Alcuni esempi possono riguardare la guida autonoma, suggerimenti di offerte online (Amazon) o musica (Spotify) o film (Netflix), conoscere il pensiero dei propri clienti attraverso i social network, fino a raggiungere livelli maggiori con intercettazione di una frode o problemi bancari.
Intelligenza artificiale ed apprendimento automatico vengono spesso utilizzati insieme in maniera assoluta, ma sono nettamente differenti. Infatti, il machine Learning è contenuto all'interno della macro-sfera dell'AI (figura1) e ne costituisce la sua maggiore area di interesse.

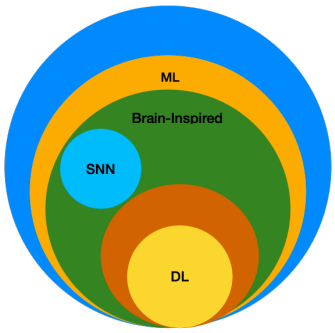


Figura 1. Correlazione tra intelligenza artificiale, machine learning e deep learning

L'AI riguarda l'intelligenza delle macchine e il loro modo di simulare funzioni cognitive come l'apprendimento e la risoluzione dei problemi. Il Machine Learning, invece, riguarda quell'insieme di algoritmi che automatizzano la costruzione di modelli analitici, consentendo ai sistemi di apprendere e svilupparsi senza programmazione diretta. Consiste in un autoapprendimento basato su algoritmi, di conseguenza il sistema si sviluppa e si addestra sulla base della sua esperienza, di cluster per parallelizzare le operazioni e dello storico di dati. Proprio per questo motivo, una delle funzionalità più importanti dell'apprendimento automatico riguarda l'intensità di elaborazione e la ripetitività, in quanto più un modello viene esposto ai dati e più è in grado di riproporsi adattandosi in maniera completamente autonoma, perché affinché l'apprendimento e il processo decisionale venga migliorato è necessario l'addestramento ripetitivo. Le tecnologie di apprendimento automatico svolgono un ruolo

importante, soprattutto quando l'elaborazione di grandi quantità di dati apporta un valore aggiunto significativo al risparmio di tempo e alla massimizzazione delle risorse di elaborazione. L'obiettivo principale del Machine Learning è quello di estrarre delle feature (valori che utilizzo come input per l'apprendimento) il più possibile distintive e rappresentative all'interno dei dati relativi ad un fenomeno che desideriamo analizzare attraverso modelli (o algoritmi di varia natura) per apprenderne i meccanismi (ovvero i *datapoint* che consistono in dati che permettono al computer di imparare) ed effettuare delle predizioni (*label*) massimizzandone l'accuratezza, ricavandone modelli di apprendimento. Con l'utilizzo di questi modelli è possibile ricavare algoritmi di apprendimento, utili per risolvere un problema specifico, in quanto indicano alla macchina le operazioni che si possono effettuare. Sono proprio gli algoritmi i "motori" che alimentano il ML. L'esecuzione di una attività di machine learning avviene sulla base di cinque passaggi principali: raccolta dati, preparazione dei dati, addestramento del modello, valutazione del modello e miglioramento delle prestazioni. Successivamente viene scelto l'algoritmo appropriato e la sua rappresentazione come modello. La parte successiva dei dati viene utilizzata per controllare l'accuratezza del modello attraverso le tracce degli output dei dati, utile per convalidare la precisione nella selezione dell'algoritmo. A questo punto vengono introdotte più variabili per verificarne l'efficienza e selezionare un ulteriore modello. A seconda del tipo di algoritmo utilizzato in funzione della metodologia per apprendere i dati e fare previsioni, si possono classifica differenti sistemi di apprendimento automatico in tre macro-categorie:

- *Apprendimento supervisionato*: L'algoritmo viene addestrato e testato da un set di dati che viene già etichettato e possiede un output predefinito. Infatti, si ottiene l'apprendimento supervisionato quando l'algoritmo utilizza variabili di input (X) e prevede o classifica dal database di addestramento variabili di output (Y), per apprendere ed estrarre la funzione di mappatura che dall'input genera l'output: Y=f(X)
- *Apprendimento non supervisionato***:** si possiede una variabile di input (i dati), ma nessuna variabile di output corrispondente. L'obiettivo principale è quello di analizzare i dati a disposizione per ricavare informazioni utili e crearne schemi o relazioni, analizzando i dati senza utilizzare una categorizzazione, ma eseguendo predizioni attraverso metodi induttivi. L'apprendimento non supervisionato viene utilizzato anche per la preelaborazione dei dati come l'estrazione di feature, la selezione di feature e il ricampionamento.
- *Apprendimento con rinforzo*: tale apprendimento si basa su due condizioni: esito ritardato e ricerca per tentativi ed errori. Utilizza distinte tipologie di logiche. La prima riguarda le predizioni, quindi ricevere dati e paramenti in input per fornire

output adeguati. La seconda riguarda l'algoritmo usato che apprende e si adatta ai cambiamenti ambientali attraverso un sistema di valutazione, agendo sulla base di ricompense (valore reale positivo) se l'azione compiuta è corretta e l'obiettivo viene raggiunto o è vicino; oppure penalità (valore reale negativo) se ciò non accade.

Con maggiore riferimento all'ambito del digital marketing, le tecniche di apprendimento automatico permettono di trasformare in valore monetario tutte le informazioni provenienti dai contatti e dalle comunicazioni multicanale. Grazie a queste tecniche, si possono estrarre intuizione ed evidenze utili dai big data provenienti da più fonti per profilare il cliente e conoscerlo in profondità, così da costruire una customer journey mirata, attraverso la personalizzazione di contenuti che permettono di aumentare l'engagement di clienti acquisiti e prospect, la fidelizzazione, strumenti di marketing, promozioni mirate e il tasso di risposta alla call-to-action, cercando di ottimizzare tutte le campagne promozionali e politiche di pricing. Alcune applicazioni riguardano: Marketing Automation, social media marketing e pagine web. È possibile affermare che l'utilizzo del machine learning offre alle aziende un vantaggio competitivo, in quanto è capace di automatizzare e velocizzare i processi decisionali, ma anche accelerare il time to value, aumentando la visibilità aziendale e la collaborazione.
L'uso dell'apprendimento automatico trova molte applicazioni anche a livello quotidiano, si intende in quelle situazioni in cui si utilizza la tecnologia (esempio: riconoscimento vocale). Alcuni settori che hanno incrementato il proprio business utilizzando tali tecnologie possono essere: servizi finanziari, marketing e vendite, pubblica amministrazione, settore sanitario e servizio dei trasporti.

2. RECOMMENDATION SYSTEM

Un *recommender system* è un programma di filtraggio dei contenuti, sulla base di preferenze, interessi o atteggiamenti osservati e registrati, che crea delle raccomandazioni personalizzate per l'utente per supportare il suo processo decisionale e consigliare prodotti, informazioni e servizi adeguati, facendogli ottenere solo le risorse più pertinenti alle sue esigenze. Gli effetti conseguenti a tale sistema non diventano vantaggiosi solo per l'utente o il cliente di un prodotto/servizio al quale viene facilitata la scelta, ma anche per l'azienda che offre quel determinato servizio o prodotto. I sistemi di raccomandazione utilizzano un'analisi di un tipo specifico di dati per prevedere le valutazioni degli utenti per i singoli articoli. Successivamente, (sulla base di questa analisi), creano raccomandazioni e modificano il contenuto della pagina visualizzata in modo che corrisponda il più possibile alle preferenze dell'utente. Questo è uno dei motivi per cui una vasta gamma di aziende e applicazioni web ha recentemente implementato sistemi di analisi del comportamento degli utenti rispetto alla raccomandazione

dei prodotti, servizi o informazioni più adatti. L'obiettivo è, ovviamente, aumentare le vendite e i profitti di queste società. Un recommendation system è costituito da un insieme di elementi essenziali: items, users, transactions e rating. Questi sistemi giocano un ruolo essenziale nel mondo della vendita online e il loro potenziale (con il crescente numero di persone che acquistano online) è ancora in aumento. Un'esperienza di acquisto più personalizzata e coinvolgente può ridurre il tasso di abbandono. È necessario che il sistema di raccomandazione possieda delle caratteristiche specifiche affinché sia efficace per gli utenti (Aggarwal, 2016): rilevanza, diversità, novità e serendipity. I sistemi di raccomandazione cercano di bilanciare queste caratteristiche per aumentare i benefici degli algoritmi a favore di utenti e aziende. Alcuni benefici riguardano: riduzione dei costi di transazione per la ricerca e la selezione di articoli in un ambiente di shopping online, miglioramento del processo decisionale e della qualità delle decisioni, aumento dei ricavi per gli e-commerce, approfondimento delle ricerche scientifiche grazie alla disponibilità di biblioteche online. Ogni tipologia di raccomandazione ha bisogno di seguire tre fasi specifiche, rappresentate in figura, per fornire i propri suggerimenti personalizzati.

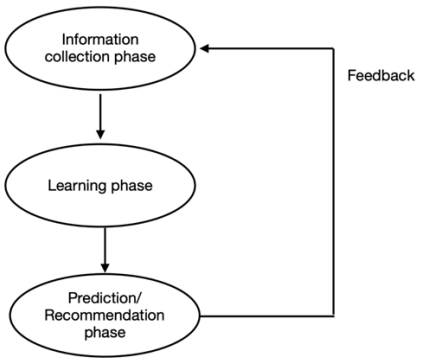


Figura 2. Fase di raccomandazione

- *Information collection phase*: raccoglie i dati rilevanti da un utente per generare un profilo utente. Ciò include attributi, comportamenti e contenuti a cui l'utente accede. Questo sistema può essere costruito solo quando ha abbastanza dati rilevanti sul profilo utente. Il meccanismo principale di questo è attraverso il feedback esplicito, implicito e di apprendimento.
- *Learning phase*: l'algoritmo del sistema di apprendimento viene distribuito per garantire che il modello sia in grado di filtrare gli attributi dell'utente dal feedback raccolto nelle fasi precedenti.
- *Prediction/Recommendation phase*: il modello prevede le scelte dell'utente o consiglia gli elementi che l'utente molto probabilmente preferirebbe. Le due fonti considerate

sono o un dataset fisso dalla fase di raccolta delle informazioni, o un modello basato sulla memoria che si basa sulle inferenze del sistema dal comportamento degli utenti.

Per riuscire a misurare l'efficacia di un sistema di raccomandazione, si possono effettuare tre diverse tipologie di valutazione:

- User studies. Metodo esplicito, che si basa sulla valutazione di raccomandazioni sviluppate da algoritmi diversi, dove quello con la media dei voti più alta viene considerato l'algoritmo migliore per quel sito.
- Valutazione online. Metodo implicito, dove il sistema analizza il comportamento dell'utente
- Valutazione offline da parte di esperti. Metodo che utilizza dataset a cui vengono eliminate delle informazioni. In questo caso si devono prevedere le informazioni mancanti per sviluppare raccomandazioni basate sui dati mancanti e i dati a disposizione.

Un algoritmo che possiede una grande quantità di dati, derivati da un traffico elevato su un sito, sarà sicuramente molto più preciso ed elaborato. È possibile raccogliere informazioni sia in maniera diretta, come il log in ad un sito web, sia in maniera indiretta, come l'accettazione di cookie. È utile ricordare che non è sempre possibile assegnare un rating a tutti gli items. Per questo motivo i sistemi di raccomandazione devono utilizzare adeguati metodi di filtraggio delle informazioni, per includere tutti i prodotti valutati sia in maniera esplicita che implicita e suggeriti all'utente. Le tecniche di filtraggio (Figura 3) si possono suddividere in: collaborative filtering, content- based filtering e hybrid filtering.

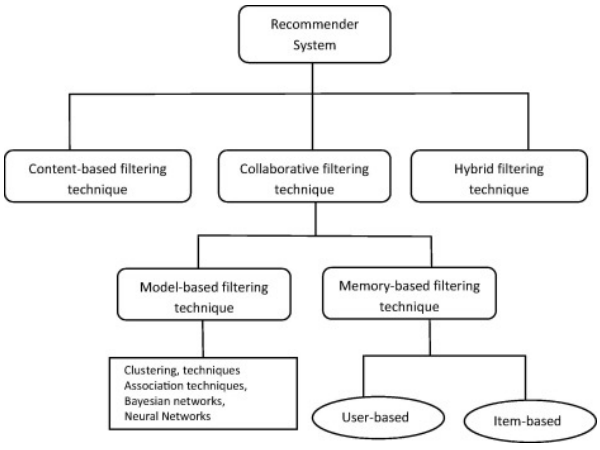


Figura 3. Tecniche di raccomandazione

Il *filtraggio collaborativo* raccomanda l'elemento all'utente specifico in base alla valutazione dell'altro utente nel sistema. Gli approcci basati sul contenuto eseguono previsioni in base alle caratteristiche dell'elemento della loro cronologia passata, ad esempio consigliando un film che è stato classificato come "Film d'azione" a un utente a cui piacciono i film d'azione. Ogni

elemento e ogni utente viene descritto da un vettore di funzionalità oppure da un incorporamento. Di conseguenza, il sistema costruisce un database di matrice elemento-utente, basandosi sulla raccolta di un grande quantità di dati riguardanti comportamenti, attività e preferenze di ogni utente per calcolare la somiglianza tra i vari profili, abbinarli tra di loro e formulare sia previsioni che raccomandazioni. I dati utilizzati si basano esclusivamente su interazioni passate (dati storici) tra gli utenti e gli elementi target. Infatti, generalmente viene utilizzata una matrice dove le righe rappresentano gli utenti e le colonne gli elementi, che dovranno essere sufficienti per fare una previsione corretta. Inoltre, la tecnica di filtraggio collaborativo può essere suddivisa in *model based* e *memory based*.

Il *filtraggio basato sui contenuti* rappresenta una delle tecniche di raccomandazione di maggior successo. Per contenuti si intendono gli attributi degli elementi che vengono analizzati per generare nuove previsioni su caratteristiche, comportamenti e preferenze dell'utente. La tecnica utilizzata consiste in un algoritmo dipendente dal dominio, dove si incrocia il contenuto di un item (come parole chiave e tags) con l'user-profile (incluse le preferenze aggiornate in real time).

La tecnica di *filtraggio ibrido* combina diverse tecniche di raccomandazione al fine di ottenere una migliore ottimizzazione del sistema per evitare alcune limitazioni e problemi del sistema di raccomandazione puri. L'idea di base è che una combinazione di algoritmi può fornire raccomandazioni più accurate ed efficaci rispetto a un singolo algoritmo, poiché in questo modo è possibile superare gli svantaggi o i limiti di un algoritmo utilizzandone un altro.

Netflix è un esempio di applicazione di un sistema di raccomandazione ibrido, che suggerisce agli utenti film o serie tv per migliorare la *customer experiences*, abbassare il tasso di abbandono e mantenere gli utenti attivi sulla piattaforma per molto tempo sulla base delle preferenze dell'utente stesso e degli altri utenti simili. Negli anni '90 Netflix si basava su un sistema di abbonamenti, inviando direttamente a casa dei clienti il DVD richiesto. Solo successivamente inizia ad aggiungere consigli personalizzati sui film, utilizzando un sistema di raccomandazione. Il suo sistema raccoglie vari dati dell'utente prima di suggerire un titolo: le interazioni con il servizio della piattaforma come visione e valutazione; legame con abbonati che hanno simili preferenze; informazioni sui film come titolo, anno di uscita del film, attori, genere; orario di visione; dispositivi utilizzati per la visione, tempo trascorso sulla piattaforma. Tutti questi elementi costituiscono dati input fondamentali per l'algoritmo che li elabora per la realizzazione dell'home page di ogni utente, luogo dove si crea proprio l'esperienza di raccomandazione con copertine di film in linea con i suoi gusti. Circa l'80% delle ore di visione sono influenzate dall'implementazione degli algoritmi di raccomandazione". Tale personalizzazione viene poi completata con l'aggiunta dei video più popolari in maniera simile per ogni utente. La pagina principale è formata da diverse sezioni, suddivise tra righe di video

a tre livelli di personalizzazione (es. continua a guardare, i titoli del momento, film consigliati in base a ciò che hai visto, commedie premiate, ecc.) e ciascuna di essa utilizza algoritmi differenti. Ovviamente Netflix utilizza un ulteriore sistema di raccomandazione per decidere il tipo di riga da mostrare, dove ad esempio la categoria “consigliati perché hai guardato questo film” rappresenta un content-based recommender system, mentre la categoria “migliori film storici” rappresenta un community-based recommender system.

3. MOVIELENS: CONFRONTO SPERIMENTALE TRA LIBRERIE PYTHON

Dal 1990, gli algoritmi dei sistemi di raccomandazione sono stati ampiamente utilizzati dalle aziende più differenti per aumentare i propri ricavi, sfruttando sia dati impliciti che espliciti forniti dai propri utenti al fine di migliorare l’interazione tra essi e il sistema. Tecnologie di alto livello come Google, Amazon, Facebook e Spotify utilizzano appunto sistemi di raccomandazioni e hanno sviluppato i propri algoritmi, strumenti necessari per la creazione del loro valore e vantaggio competitivo. È possibile trovare un'ampia varietà di algoritmi nella letteratura del sistema di raccomandazione per il calcolo delle previsioni e l'ottenimento di elenchi di raccomandazioni. Le prestazioni di questi algoritmi possono essere valutate mediante numerose metriche a seconda dell'obiettivo aziendale che si intende ottimizzare. Tuttavia, le prestazioni possono variare drasticamente a seconda dell'algoritmo selezionato e della metrica selezionata.

Lo scopo di questo studio è quello di eseguire un confronto sperimentale di alcune librerie Python per i sistemi di raccomandazione che utilizzano il dataset MovieLens, per rispondere alla seguente domanda di ricerca: “Come varia l'accuratezza delle previsioni nei sistemi di raccomandazione?”

La scelta del linguaggio di programmazione è una decisione fondamentale nella progettazione del software scientifico. Linguaggi di scripting di alto livello, di cui Python e Octave sono due esempi, forniscono un livello superiore di astrazione dall'architettura della macchina, consentendo allo sviluppatore di concentrarsi sull'algoritmo, riducendo i tempi di sviluppo e facilitando la prototipazione rapida di nuove idee

Python è un linguaggio di programmazione di alto livello molto utilizzato in ambito machine learning, scritto da Guido van Rossum 1991, che si caratterizza per la sua programmazione multi-paradigma. È un linguaggio interpretato che mira a enfatizzare la leggibilità del codice utilizzando una sintassi semplice e ad evitare casi speciali ed eccezioni. Utilizza un sistema di tipo dinamico e supporta più paradigmi di programmazione. Gli interpreti Python sono disponibili per molti sistemi operativi e supportano l’integrazione con altri linguaggi e programmi. Python è un linguaggio versatile, potente e di facile apprendimento con un'ampia comunità di utenti. È disponibile gratuitamente con una licenza open source e funziona su tutte

le principali piattaforme informatiche. A differenza dei linguaggi di codifica tradizionali come *Fortran* o *C / C ++,* gli script Python non hanno bisogno di essere compilati, il che semplifica il loro sviluppo. Python offre una sintassi pulita e strutture dati moderne e flessibili orientate agli oggetti. Le librerie di estensione sono disponibili per un'ampia gamma di attività, incluso il calcolo scientifico. La filosofia di progettazione di Python enfatizza la leggibilità del codice con il suo notevole uso di rientranze significative. I suoi costrutti linguistici e il suo approccio orientato agli oggetti mirano ad aiutare i programmatori a scrivere codice chiaro e logico per progetti su piccola e grande scala.

Piuttosto che avere tutte le sue funzionalità integrate nel suo nucleo, Python è stato progettato per essere altamente estensibile attraverso l'uso di moduli contenenti funzioni utili per la stesura del programma. Questa modularità compatta lo ha reso particolarmente popolare come mezzo per aggiungere interfacce programmabili alle applicazioni esistenti. La visione di Van Rossum di un piccolo linguaggio di base con una grande libreria standard e un interprete facilmente estensibile derivava dalle sue frustrazioni con ABC, che adottò l'approccio opposto. Le librerie di Python vengono definite come "collezioni di metodi e funzioni che permettono di svolgere delle azioni senza scrivere il codice di ogni passaggio." In questo modo si riesce a semplificare il codice, riducendo il numero di operazioni. Inoltre, le librerie sono utili per costruire un modello automatico di successo, grazie all'elevata quantità di elementi che contengono. Di seguito verranno presentate alcune principali librerie di Python per lo sviluppo dei sistemi di raccomandazione tradizionali:

- *LensKit*: è un toolkit open source, realizzato dal gruppo di ricerca GroupLens, per la creazione, la ricerca e l'apprendimento dei sistemi di raccomandazione. Questa libreria è composta da una serie di strumenti specifici da utilizzare in algoritmi di raccomandazione con caratteristiche particolari per implementazioni modulari di diversi algoritmi, in particolare gli algoritmi di raccomandazione implementati con filtri collaborativi.
- *Crab*: è un framework Python per la creazione di motori di raccomandazione integrati con i pacchetti scientifici Python (numpy, scipy, matplotlib). Ha il supporto per algoritmi di raccomandazione come il filtraggio collaborativo basato sugli utenti e sugli elementi.
- *Surprise*: è uno scikit Python per la creazione e l'analisi di sistemi di raccomandazione che gestiscono dati di valutazione espliciti. Surprise è stato progettato per essere utile ai ricercatori che desiderano esplorare rapidamente nuove idee di raccomandazione supportando la creazione di algoritmi di previsione personalizzati, ma può anche servire come risorsa di apprendimento per studenti e utenti meno esperti grazie alla sua documentazione dettagliata.

- *Rexy:* è un vero e proprio sistema di raccomandazione open source, basato su un concetto generale di tag-prodotto-utente e una struttura flessibile, progettata per essere adattabile con vari schemi di dati.
- *TensorFlow:* è un sistema di machine learning e uno strumento open source che consente di creare, ottimizzare e distribuire sistemi di apprendimento automatico di grandi dimensioni e arbitrari. In TensorFlow, un processo di apprendimento automatico è espresso come un "grafico" che mostra come i dati fluiscono attraverso il sistema, per rappresentare il calcolo, lo stato condiviso e le operazioni che modificano quello stato.
- *TensorRec:* gestisce la manipolazione dei dati, il punteggio e la classificazione per generare consigli. È un algoritmo di raccomandazione che utilizza una semplice API esterna per l'addestramento e la previsione che assomiglia ai comuni strumenti di machine learning in Python.
- *LightFM:* È un modello ibrido che consente inoltre di incorporare metadati sia degli elementi che degli utenti nei tradizionali algoritmi di fattorizzazione a matrice. Rappresenta ogni utente e oggetto come la somma delle rappresentazioni latenti delle loro caratteristiche, consentendo così la generalizzazione dei consigli a nuovi elementi (tramite le caratteristiche dell'oggetto) e ai nuovi utenti (tramite le caratteristiche dell'utente.
- *CaseRecommender*: è stato sviluppato per fornire flessibilità ed estensibilità negli ambienti di ricerca, pur mantenendo alte prestazioni, fornendo una varietà di raccomandazioni e algoritmi di clustering, nonché funzioni per la manipolazione dei dati. Il vantaggio principale di Case Recommend è la possibilità di integrare algoritmi di clustering e ensemble con motori di raccomandazione, facilitando lo sviluppo di approcci più accurati ed efficienti.
- *Spotlight*: è un sistema di raccomandazione Python che utilizza PyTorch per creare modelli di raccomandazioni sia profondi che superficiali. Fornendo una serie di elementi costitutivi per le rappresentazioni delle funzioni di perdita e utilità per il recupero o la generazione di set di dati di raccomandazione.

Sono disponibili numerosi set di dati per la ricerca di raccomandazioni. Tra questi, il set di dati *MovieLens* è probabilmente uno dei più popolari. MovieLens è un sistema di raccomandazione di film basato sul web non commerciale. È stato creato nel 1997 e gestito da GroupLens, un laboratorio di ricerca presso l'Università del Minnesota, al fine di raccogliere dati sulla classificazione dei film a scopo di ricerca. I set di dati MovieLens sono ampiamente utilizzati nell'istruzione, nella ricerca e nell'industria. Sono stati rilasciati quattro set di dati MovieLens, noti come 100k, 1m, 10m e 20m, che riflettono il numero approssimativo di valutazioni in ciascun set di dati. Le valutazioni e altri dati vengono attribuiti a ID utente

anonimi (questi ID non vengono mappati tra i set di dati). I film sono elencati insieme ai loro titoli in MovieLens, insieme a zero o più generi (gli ID dei film vengono mappati tra i set di dati). Sono inclusi solo gli utenti con almeno 20 valutazioni. Dal lancio di MovieLens, le valutazioni sono state espresse come un valore "stella", che è un modello standard dell'interfaccia utente per consentire agli utenti di inserire le preferenze.

In questo studio, nello specifico, è stato utilizzato per sviluppo e test il dataset MovieLens100K contenente 100 mila righe, composto da: 100.000 valutazioni (da 1 a 5) effettuate da 943 utenti su 1682 film; e il dataset MovieLens1M, contenente 1.000.209 valutazioni anonime di circa 3.900 film realizzato da 6.040 utenti che si sono uniti a MovieLens nel 2000.

Le librerie Python prese in considerazioni ed analizzate per questo studio sono: LightFM, Surprise, LensKit e TensorFlow.

Il codice sorgente di tutte le analisi segue i seguenti passi:

4. Installazione e/o import di librerie necessarie;
5. Caricamento del dataset;
6. Suddivisione del dataset:
   - 80% per il training dell'algoritmo/modello
   - 20% per il test del modello sui nuovi dati, per capire come lo stesso si comporta sui dati nuovi e valutarne la sua capacità di generalizzazione dei pattern appresi durante la fase di training;
7. Check della dimensionalità dei dati dopo lo splitting: TRAINING + TEST = dimensione DATASET iniziale;
8. Addestramento del modello sui dati di training;
9. Calcolo di performance metrics sui dati di test.

La suddivisione del dataset iniziale MovieLens segue la seguente impostazione:

- *Shuffle*, ovvero mescolamento del dataset iniziale in modo tale da avere un set di dati quanto più randomico possibile in termini di ordinamento (ad esempio per mescolare l'identificativo utente).
- *Training e test.* Il primo 80% del dataset viene utilizzato per il training, il rimanente 20% per il test dell'algoritmo.

La fase successiva per la creazione del modello riguarda l'utilizzo della funzione *predict*. Tale passaggio riguarda il momento in cui il modello mette in prova la sua conoscenza sui dati di test, applicando il sistema di raccomandazione appena implementato sul nuovo set. Le predizioni ottenute, in questo caso, riguardano *ratings* dei vari film per tutti gli utenti (overall del sistema). Per quantificare la bontà di questi outputs e valutarne l'accuratezza delle predizioni, sono state utilizzate due differenti metriche, in base alla libreria in esame:

- *RMSE (Root Mean Square Error)*: Si utilizza per misurare l'errore di un modello nella previsione dei dati quantitativi.
- *AUC (Area Under The Curve)*: Rappresenta il grado o la misura di separabilità della curva di probabilità.

Per la libreria LightFM ottenere i dati di MovieLens è molto semplice, in quanto il dataset consiste in una delle funzioni fornite da LightFM stesso. Vengono effettuati gli split dei dati in modo che ci sia un 80% di training e un 20% di test. Successivamente viene addestrato il modello e se ne osserva l'accuratezza. Basandosi sulle caratteristiche di questa libreria e sulla sua originaria costruzione, non è stato possibile calcolare la RMSE bensì la AUC. Si utilizza il modello WARP per ottimizzare le prestazioni in modo che si ottenga un input migliore in termini di precisione.

```
Precision: train 0.60, test 0.22.
AUC: train 0.94, test 0.90.
```

L'AUC di training rappresenta la capacità dell'algoritmo di apprendere le relazioni nascoste tra i dati di training durante l'addestramento dell'algoritmo e corrisponde al 94%. Mentre la AUC di test evidenzia quanto l'algoritmo ha predetto bene i nuovi dati, quelli di test, e corrisponde al 90% di accuratezza rispetto ai valori reali.
Per quanto riguarda l'analisi del dataset da 1 milione di righe si seguono i medesimi passaggi.

```
Precision: train 0.59, test 0.18.
AUC: train 0.94, test 0.91.
```

Si ottiene un AUC per il training pari al 94% e un AUC per il test pari al 91% di accuratezza rispetto ai valori reali. Risultati molto buoni, quasi uguali allo studio del dataset da 100k.
Per la libreria Surprise, la quale ha al suo interno una serie di algoritmi e set di dati integrati, ho deciso di usare e analizzare gli algoritmi più comuni, come: CoClustering; SVD, KNN Basic, NMF, SlopeOne.
Per il dataset da 100mila righe, viene utilizzato il `train_test_split()` per campionare un trainset e un testset con determinate dimensioni e utilizzare l'opzione desiderata. Si utilizza il metodo che addestrerà l'algoritmo sul trainset e il metodo che restituirà le previsioni fatte dal testset. Successivamente si addestrano i diversi modelli appartenenti al pacchetto Surprise:

```
models = [SVD(), NMF(), SlopeOne(), KNNBasic(), CoClustering()]
models_name = ['SVD', 'NMF', 'SlopeOne', 'KNNBasic', 'CoClustering']
model_rmse = pd.DataFrame()
predictions_dict = dict()
index = 0
```

Per la misurazione delle prestazioni, utilizzando questo pacchetto, non è possibile calcolare altre metriche oltre al RMSE, in quanto la sua struttura non lo permette (avvolge altre librerie comunemente usate in Machine Learning). Dopo aver addestrato il modello e aver ottenuto

l'RMSE si prevedono i nuovi valori. Più precisamente, i risultati ottenuti da ciascun algoritmo sono i seguenti:

	CoClustering	KNNBasic	NMF	SVD	SlopeOne
0	0.944849	0.951764	0.924705	0.873799	0.904411

L'algoritmo *KNNBasic* mostra il miglior valore RMSE per il set di test di input, il suo comportamento approssima al meglio i valori previsti rispetto a quelli reali. I suoi valori predetti sono affidabili al 95% rispetto a quelli reali. Tuttavia, anche gli altri algoritmi hanno ottenuto buoni risultati in termini di affidabilità, superiori al 90%. Solamente SVD potrebbe essere migliorato, in quanto i suoi valori predetti risultano affidabili all'87% rispetto a quelli reali.
Per quanto riguarda le analisi condotte sul dataset da 1 milione di righe, vengono seguiti i medesimi messaggi precedentemente descritti. Viene caricato il "rating dataset" prevedendo che i rating contengano le seguenti colonne (*ratings.columns): user, item, rating, timestamp.* Successivamente i dati vengo formattati per essere leggibili dalla libreria ('user', 'item', 'rating') e divisi in 80% train e 20% test delle osservazioni totali. L'addestramento viene effettuato per i differenti algoritmi. Dopo aver addestrato il modello, si prevedono i nuovi valori e si ottiene l'RMSE.

	CoClustering	KNNBasic	NMF	SVD	SlopeOne
0	0.855808	0.725167	0.693429	0.520548	0.661144

Per questo dataset il migliore algoritmo è stato CoClustering con l'85% di RMSE. Tuttavia, nessun algoritmo ha raggiunto il 90% e i risultati ottenuti sono stati discreti, partendo dal 52% con SVD che rimane l'algoritmo con percentuale inferiore. Si può chiaramente affermare che la libreria Surprise è più accurata con il dataset più piccolo da 100mila riga.
Gli algoritmi di LensKit prevedono che un ratings rame contenga le seguenti colonne (in qualsiasi ordine): user, item e rating. Le performance dell'algoritmo sono buone e pari quasi a 89%.

```
# compute the rmse between predicted and actual values
rmse = compute_rmse(prediction)
rmse
```

```
0.8925411538190678
```

Lo studio del dataset da 1 milione segue la stessa analisi e prevede che il ratings rame contenga le seguenti colonne: *user, item, rating, timestamp*. Vengono suddivisi i dati in 80% training e 20% test e avviato il *sanity check.*
Si ottengono i seguenti risultati:

```
# compute the rmse between predicted and actual values
rmse = compute_rmse(prediction)
rmse
```

```
0.802027068669124
```

Le performance dell'algoritmo sono pari a 80%, quindi risultano inferiori a quelle calcolate dal dataset di 100k.

Per analizzare TensorFlow, potente libreria, sono state attuate le seguenti differenze tecniche:

- Il dataset è stato scaricato direttamente da TensorFlow, quindi non è stato fornito dall'esterno.
- Sono state selezionate solo le colonne di interesse ai fini dell'analisi, alleggerendone l'esecuzione del modello.
- Per la stessa motivazione del punto precedente, sono stati ripartiti i dati in batch.
- Sono stati *tokenizzati* alcuni dati per rendere idoneo il formato del modello.
- È stata eseguita una creazione e *overriding* di classi e funzioni usate dal modello.
- I modelli costruiti sono i seguenti: utente, film, genere (retrieval, che usa i precedenti).

Nel complesso il modello di TensorFlow Recommender non è andato benissimo e non è riuscito a generalizzare i nuovi dati ai fini della prediction. La performance del test mostra un risultato peggiore della fase di allenamento.

```
{'factorized_top_k/top_100_categorical_accuracy': 0.2113499939441681,
 'factorized_top_k/top_10_categorical_accuracy': 0.013050000183284283,
 'factorized_top_k/top_1_categorical_accuracy': 0.0003000000142492354,
 'factorized_top_k/top_50_categorical_accuracy': 0.1014999970793724,
 'factorized_top_k/top_5_categorical_accuracy': 0.0044999998062849045,
 'loss': 28389.564453125,
 'regularization_loss': 0,
 'total_loss': 28389.564453125}
```

Questo è causato da:

1) *Overfitting*: Si comporta meglio sui dati che ha visto, semplicemente perché può memorizzarli.
2) *Re-recommending*: Il modello consiglia nuovamente alcuni dei film già visti dagli utenti. Dovremmo migliorarlo con alcune tecniche speciali come regolarizzazione e livellamento.

Il file da 1 milione di righe ha seguito la medesima analisi e non ha avuto sostanziali modifiche, registrando un'accuratezza pari al 22%. Percentuale molto bassa che non si discosta dall'accuratezza del file da 100 mila righe. Di conseguenza è possibile affermare che anche in questo caso TensorFlow non è riuscito a generalizzare i nuovi dati ai fini della prediction.

Dall'analisi di ogni singolo algoritmo, KNN Basic risulta il migliore in termini di accuratezza prevedendo i risultati reali nel 95% dei casi. La differenza con l'algoritmo CoClustering risulta di pochissimi decimali e si può considerare anche molto buono in termini di accuratezza dei risultati reali. Per questi algoritmi è stato possibile raggiungere tali livelli di performance grazie

alle loro caratteristiche. Il *k-nearest neighbors* (k-NN) è tra gli algoritmi più semplici utilizzati in machine learning e si basa sulla similarità dei dati, sulle caratteristiche degli oggetti vicini a quello considerato. Riesce ad individuare k cluster e identificare i dati all'interno dei ratings. Di conseguenza, è possibile trovare i valori più simili a quelli reali. Inoltre, il KNN Basic è stato realizzato proprio per funzionare bene con dataset più piccoli. Per gli stessi motivi è possibile giustificare l'accuratezza dell'algoritmo CoClustering, in quanto raggruppa gli elementi sulla base della loro distanza reciproca. In generale, la maggior parte degli algoritmi ha registrato risultati molto buoni, superiori al 90% che suggeriscano quanto il dataset sia già stato ottimizzato per essere usato a scopo sperimentale. I vantaggi nell'utilizzo di un dataset più piccolo stanno proprio nelle dimensioni della quantità di dati, in quanto l'algoritmo performa meglio con dati più piccoli. MovieLens 100k avrà al suo interno un processo di *data cleaning* in grado di rilevare gli errori e imprecisioni per correggerli. Di conseguenza, la correzione dei dati può essere già avvenuta attraverso l'utilizzo di alcuni sistemi come l'approccio a *batch.*

Dall'analisi di ogni singolo algoritmo del dataset MovieLens 1milione il migliore è stato LightFM, come mostra la figura 32, con un valore di AUC pari al 91%, il quale ha approssimato con un'accuratezza del 91% i valori reali e presenti nel test set. In generale, questo studio non è stato complessivamente soddisfacente come quello del dataset da 100mila righe, in quanto quasi tutte le librerie hanno registrato una percentuale di accuratezza inferiore al 90%. La motivazione alla base di queste performance può essere collegata alla grande quantità di dati presenti. I dataset da 1 milione non sono del tutto ottimizzati, quindi è possibile che ci siano errori ed imprecisioni che causino problemi di *overfitting*. In questo modo si avranno performance migliori in fase di training e peggiori in fase di test. Sarebbe opportuno risolvere questo problema utilizzando un approccio a *batch* per tutti gli algoritmi, così da dividere i dati in porzioni più piccole.
Il dataset utilizzato come riferimento per il confronto sperimentale è molto utilizzato per analisi simili in ambito di sistemi di raccomandazione. Per questo studio specifico sono stati utilizzati solo due tipologie di dataset, ML100k e ML1m. Il set con 100k è la versione più vecchia dei set di dati MovieLens, contenete un piccolo set di dati con dati demografici, ma queste sue dimensioni più ridotte lo rendono veloce da scaricare, analizzare ed elaborare. Di conseguenza, ulteriori studi potrebbero affrontare lo stesso confronto utilizzando versione del dataset differenti e più elevate, come il ML25m. Tale set non è stato eseguito in questo studio a causa della sua grandezza e potenza. Infatti, dataset così grande contengono troppi dati e sono eccessivamente pesanti per essere supportati da un normale computer o da un sistema come Colab. Si suggerisce agli studi futuri di utilizzare un "*cloud*" per essere più veloci negli studi ed evitare di sovraccaricare il sistema. Alcune soluzioni possono essere: Amazon Web

Services, Azure di Microsoft e Google Cloud Platform. Gli studiosi, che sono interessati a combinare valutazioni o tag di dati con dati di contenuti cinematografici ricchi, dovrebbero prendere in considerazione l'utilizzo del set di dati di 20 (o 25) milioni in combinazione con risorse esterne. Questo set include una mappa degli ID MovieLens sugli ID IMDb4 ("Internet Movie Database"). Poiché gli ID dei film sono stabili tra i set di dati, il file di collegamento di 20m può essere utilizzato in combinazione con uno qualsiasi dei set di dati rilasciati in precedenza.

Le ampie modifiche all'esperienza utente di MovieLens hanno inevitabilmente portato a dati di valutazione meno "puliti" rispetto alla situazione in cui avesse mantenuto l'interfaccia costante. Le modifiche agli strumenti di ricerca, agli algoritmi di raccomandazione e alle funzionalità disponibili modificano il contenuto mostrato agli utenti, il che a sua volta influisce sui dati di valutazione risultanti. Tuttavia, MovieLens non è l'unico con questo problema, i siti di raccomandazione come Amazon e Netflix hanno anche introdotto sostanziali modifiche e perturbazioni dell'interfaccia attraverso i test A/B.

L'approccio usato in questo studio è quello di partire da un dataset pronto e strutturato, quindi ulteriori studi potrebbero migliorare questo metodo usando un sistema di *real time*. Sarebbe interessanti utilizzare come codice sorgente non dati pronti, ma *Data Lake.* In questo modo sarà possibile recuperare e organizzare i dati secondo il tipo di analisi che si intende effettuare.

I set di dati MovieLens includono dati solo da utenti con almeno 20 valutazioni, quindi sono intrinsecamente sbilanciati verso utenti "di successo". Cioè, gli utenti che sono meno interessati a valutare i film, non sono stati in grado di trovare abbastanza contenuti valutabili o non hanno apprezzato la loro esperienza iniziale nel sistema, non sono inclusi nei set di dati. È possibile che questi utenti siano fondamentalmente diversi dagli utenti nei set di dati. I dataset associano i timestamp a ciascuna valutazione, ma questi timestamp non rappresentano la data di consumo. Le valutazioni in MovieLens possono verificarsi in qualsiasi momento, anche molti anni dopo aver visto un film. Spesso, gli utenti inseriscono un gran numero di valutazioni in una singola sessione, riempiendo la cronologia delle valutazioni per soddisfazione personale o nella speranza di ottenere consigli più personalizzati.

Quindi i prossimi studi possono analizzare ulteriori dataset che superino questo limite o che abbiano impostazioni differenti, per valutarne anche approcci e algoritmi differenti. I ricercatori potrebbero prendere in considerazione set di dati alternativi che corrispondono meglio al sistema online che desiderano simulare oppure potrebbero valutare un approccio su più set di dati. Inoltre, future ricerche potrebbero prendere in considerazioni librerie differente per la creazione di sistemi di raccomandazioni. Sarebbe interessante analizzare anche lo stesso dataset su ulteriori sistemi per confrontarne i risultati e capire se esiste un livello maggiore di accuratezza delle previsioni.

BIBLIOGRAFIA

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., & Dean, J. et al. (2016). *TensorFlow: A System for Large-Scale Machine Learning.* Usenix.org. Retrieved from https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

Ashok Mahant, M., & Pellakuri, V. (2021). *Innovative supervised machine learning techniques for classification of data.* Materials Today: Proceedings. https://doi.org/10.1016/j.matpr.2020.11.092

Campello, R., da Costa, A., Fressato, E., Neto, F., & Manzato, M. (2018). *Case recommender | Proceedings of the 12th ACM Conference on Recommender Systems.* Dl.acm.org. Retrieved 23 May 2021, from https://dl.acm.org/doi/pdf/10.1145/3240323.3241611.

Casagranda, P., & Metta, S. (2016). *Leggi questo articolo, una tua amica lo ha trovato interessante | Un'introduzione alle opportunità e criticità dei recommender system per la personalizzazione dei contenuti audiovisivi.* Crit.rai.it. Retrieved from http://www.crit.rai.it/eletel/2016-2/162-4.pdf.

Delmedico, A. (2019). *Sistemi di raccomandazione e comportamento del consumatore: confronto tra user-based e item- based collaborative filtering* (Laurea Magistrale). Luiss Guido Carli.

Dwivedi, S., & Roshni, V. K. (2017). *Recommender system for big data in education*. In 2017 5th National Conference on E-Learning & E-Learning Technologies (ELELTECH) (pp. 1-4). IEEE. Retrieved from https://ieeexplore.ieee.org/abstract/document/8074993

Gosmar, D. (2021). *Machine Learning: il sesto chakra dell'intelligenza artificiale* (3rd ed.). Independently published.

Guzzi, P. (2019). *Computing Languages for Bioinformatics: Python.* Encyclopedia Of Bioinformatics And Computational Biology, 1(195-198). https://doi.org/https://doi.org/10.1016/B978-0-12-809633-8.20366-X

Hao, J., & Ho, T. (2019). *Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language.* Journal of Educational and Behavioral Statistics. Retrieved from https://doi.org/10.3102/1076998619832248.

Harper, F. and Konstan, J., (2016). *The MovieLens Datasets*. ACM Transactions on Interactive Intelligent Systems, 5(4), pp.1-19. DOI: 10.1145/2827872

Hug, N., (2020). *Surprise: A Python library for recommender systems*. Journal of Open Source Software, 5(52), 2174. https://doi.org/10.21105/joss.02174

Isinkaye, F., Folajimi, Y., & Ojokoh, B. (2015). R*ecommendation systems: Principles, methods and evaluation.* Egyptian Informatics Journal, 16(3), 261-273. DOI:https://doi.org/https://doi.org/10.1016/j.eij.2015.06.005

Jafari-Marandi, R. (2021). *Supervised or unsupervised learning? Investigating the role of pattern recognition assumptions in the success of binary predictive prescriptions.* Neurocomputing, 434, 165-193. DOI: https://doi.org/10.1016/j.neucom.2020.12.063

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

Noguera Torres, A. D. P. (2021). *Sistema de recomendación de matrícula de cursos electivos para estudiantes de Ingeniería Electrónica e Ingeniería de Telecomunicaciones de la UNAD*. Universidad Nacional Abierta y a Distancia UNAD, Yopal, Retrieved from https://repository.unad.edu.co/bitstream/handle/10596/40465/adnoguerat.pdf?sequence=1

Paullier, A. and Sotelo, R., (2020). *A Recommender Systems' algorithm evaluation using the Lenskit library and MovieLens databases*. 2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), DOI:10.1109/bmsb49480.2020.9379914

Shinde P. P., Shah S., *A review of machine learning and deep learning applications*. Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) (2018)

Vanam, M., Amirali Jiwani, B., Swathi, A., & Madhavi, V. (2021). *High performance machine learning and data science based implementation using Weka*. Materials Today: Proceedings. DOI:https://doi.org/10.1016/j.matpr.2021.01.470

Vitale F. (2019). *Recommendation System: Implementazione Di Algoritmi Collaborativi Per Servizi Video-Streaming On Demand (VOD).* Tesi Luiss Guido Carli. Retrieved from http://tesi.luiss.it/25923/1/692341_VITALE_FABRIZIO.pdf

Walek, B., & Fojtik, V. (2020). *A hybrid recommender system for recommending relevant movies using an expert system.* Expert Systems With Applications, (158). DOI:https://doi.org/10.1016/j.eswa.2020.113452

Wellmann, J., Croucher, A., & Regenauer-Lieb, K. (2012). *Python scripting libraries for subsurface fluid and heat flow simulations with TOUGH2 and SHEMAT*. Computers & Geosciences, 43, 197-206. DOI:https://doi.org/https://doi.org/10.1016/j.cageo.2011.10.011

Wette, K. (2020). *SWIGLAL: Python and Octave interfaces to the LALSuite gravitational-wave data analysis libraries.* Softwarex, 12. DOI:https://doi.org/https://doi.org/10.1016/j.softx.2020.100634

Xu, Y., Zhou, Y., Sekula, P., & Ding, L. (2021). *Machine learning in construction: From shallow to deep learning.* Developments In The Built Environment, 6. DOI:https://doi.org/10.1016/j.dibe.2021.100045

SITOGRAFIA

Boise State University. *Loading Data — LensKit. Lkpy.readthedocs.io.* (2019). Retrieved 1 June 2021, from https://lkpy.readthedocs.io/en/stable/datasets.html#movielens-data-sets

Choudhury, A. (2020). *Top Open Source Recommender Systems In Python For Your ML Project.* Analytics India Magazine. Retrieved 25 May 2021, from https://analyticsindiamag.com/top-open-source-recommender-systems-in-python-for-your-ml-project/

Clayton, R. (2021). *Cos'è il machine learning?.* Oracle.com. Retrieved 24 April 2021, from https://www.oracle.com/it/data-science/machine-learning/what-is-machine-learning/

Colab. *The MovieLens Dataset.* Colab.research.google.com. Retrieved 31 May 2021, from https://bit.ly/3vaJ1ON

Govoni, L. *Machine Learning e principio di funzionamento*. Lorenzo Govoni Business e Tecnologia. Retrieved 25 April 2021, from https://www.lorenzogovoni.com/machine-learning-e-funzionamento/

Hug, N. (2015). *prediction_algorithms package — Surprise.* Surprise.readthedocs.io. Retrieved 1 June 2021, from https://surprise.readthedocs.io/en/stable/prediction_algorithms_package.html

Ichi.pro. *Approfondisci il sistema di raccomandazione di Netflix*. Retrieved 5 May 2021, from https://ichi.pro/it/approfondisci-il-sistema-di-raccomandazione-di-netflix-57277795941192

Kirk J., 2017. *TensorRec: A Recommendation Engine Framework in TensorFlow*. Medium. Retrieved 23 May 2021, from https://medium.com/hackernoon/tensorrec-a-recommendation-engine-framework-in-tensorflow-d85e4f0874e8

Leonardi, A. (2019). *Il valore del Machine Learning nel Marketing*. AI4Business. Retrieved 26 April 2021, from https://bit.ly/2TwY1ZA

Moody, J. (2019). *What does RMSE really mean?.* Towards Data Science. Retrieved 1 June 2021, from https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e

MovieLens. *GroupLens.* Retrieved 1 June 2021, from https://grouplens.org/datasets/movielens/

MovieLens. *Movielens.org*. Retrieved 1 June 2021, from https://movielens.org/

Nardini, D. (2019). *Le 5 migliori librerie di Python per il Machine Learning*. Pulp Learning - Tutto sul Machine Learning. Retrieved 11 May 2021, from https://bit.ly/2QtzPpP

Narkhede, S. (2018). *Understanding AUC - ROC Curve*. Towards data science. Retrieved 1 June 2021, from https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Netflix. *Funzionamento del sistema di consigli di Netflix.* Retrieved 5 May 2021, from https://help.netflix.com/it/node/100639

Rocket To Ride. (2019). *I sistemi di raccomandazione - Recommender System*. Retrieved 27 May 2021, from http://www.rockettoride.com/r-tutorial-archive/i-sistemi-di-raccomandazione-recommender-system

Santucci, U. *Sistema di raccomandazione.* Retrieved 3 May 2021, from http://www.umbertosantucci.it/atlante/sistema-di-raccomandazione/

Wikipedia. *Sistema di raccomandazione*. It.wikipedia.org. Retrieved 4 May 2021, from https://bit.ly/2S4CB5s

Wikipedia. *Python (programming language).* The Reader Wiki, Reader View of Wikipedia. Retrieved 18 May 2021, from https://thereaderwiki.com/en/Python_(programming_language)