Relazione del progetto d'esame

Implementazione di una Random Forest e di algoritmi di Collaborative Filtering sul dataset MovieLens

Giulia Clerici

CdLM in Informatica
Corso di Metodi Statistici per l'Apprendimento

***Sommario*—Il presente elaborato ha l'obiettivo di illustrare il lavoro svolto riguardo l'implementazione di diversi algoritmi di Recommendation System applicati al dataset MovieLens. Nello specifico, il progetto sviluppato ha previsto l'utilizzo una Random Forest, un algoritmo di scomposizione a valori singolari (SVD) e due diversi algoritmi di K-Nearest Neighbors (k-NN). Tali algoritmi sono stati applicati a due differenti sottoinsiemi di diversa cardinalità del dataset MovieLens, in modo tale da valutare la performance degli algoritmi anche a fronte della numerosità del dataset considerato. I risultati ottenuti sono stati valutati osservando principalmente i valori di RMSE.**

I. Introduzione

Il progetto sviluppato intende implementare un algoritmo di Recommendation System a partire dalla collezione di dati MovieLens, contenente informazioni riguardanti le valutazioni di diversi utenti rispetto a diversi film. Un approfondimento della struttura di tali dati è presentato nella sezione III.
Gli algoritmi utilizzati sono stati l'algoritmo di Random Forest, K-Nearest Neighbors e Scomposizione a valori singolari. Il tutto è stato sviluppato utilizzando il software Weka e implementando del codice in linguaggio Python, avvalendosi di opportune librerie specificatamente indirizzate all'apprendimento automatico e al collaborative filtering.

II. Inquadramento dell'argomento, notazioni e definizioni rilevanti

A. Recommendation System

Vista la natura del dataset proposto, è evidente l'intenzione di voler implementare un sistema di raccomandazione. Tali sistemi si occupano di predire un rating, o preferenza, di un particolare utente riguardo un certo oggetto, in questo caso un film, al fine di poter raccomandare all'utente oggetti che possano rispecchiare le sue preferenze. I sistemi di raccomandazione si possono suddividere in due categorie principali: i sistemi content-based e i sistemi di collaborative filtering.
In entrambi i casi vi sono due tipi di entità: gli utenti e gli oggetti. Per comodità, nel presente elaborato gli oggetti sono anche riferiti con il termine film. A partire dai dati disponibili, viene creata una matrice di utilità, contenente per ogni coppia utente-film un valore, che rappresenterà il rating che quell'utente ha assegnato a tale film. Gli utenti sono posti sulle righe della matrice, mentre i film sulle colonne. Dunque, ogni riga della matrice rappresenterà le diverse valutazioni che un utente ha assegnato ai diversi film. Mentre ogni colonna rappresenterà i diversi rating che un film ha ricevuto da parte di diversi utenti. È bene notare come alcune posizioni della matrice siano evidentemente vuote, vista l'impossibilità di avere per ogni utente una valutazione per ciascun film e data la necessità di implementare un sistema che si occupi di predire tali valutazioni per consigliare nuovi oggetti all'utente.
È bene sottolineare come vi possano essere due tipologie di valutazione. Infatti una valutazione può essere esplicita, come nel caso del presente elaborato, qualora agli utenti venga chiesto di valutare con un certo punteggio i film visti. Tuttavia, in alcuni sistemi le valutazioni possono anche essere implicite, nel caso in cui vengano inferite a partire dal comportamento dell'utente. Ciò significa che non sono gli utenti stessi ad esprimere una valutazione immettendo un punteggio, ma è il sistema che assegna un valore pari a 1 o 0, rispettivamente nel caso in cui l'utente abbia preso visione dell'oggetto in questione o nel caso in cui l'utente ignori tale oggetto. A partire da questa matrice di utilità, è possibile implementare le due tipologie di sistemi di raccomandazione.

B. Content-based system

Per quanto concerne i sistemi content-based, questi si basano sulla creazione di profili per film e utenti. Il principio sottostante consiste nel predire ad un utente un film con proprietà simili ad un film che ha ricevuto una valutazione positiva da parte dell'utente stesso. Ciò su cui si pone attenzione sono le proprietà degli oggetti, ossia dei film. La similarità tra due film è calcolata tramite una misura di similarità basata sulle caratteristiche dei due film. Infatti, ogni film è caratterizzato da un vettore in cui ogni componente rappresenta una caratteristica e utilizzando una misura di similarità, come ad esempio la similarità di Jaccard o la similarità del coseno, sugli elementi dei vettori rappresentanti i diversi oggetti allora è possibile ottenere un valore indicante quanto due oggetti siano simili. Dopodiché vi è la necessità di creare i profili degli utenti che consistono in vettori pesati aventi le stesse componenti dei vettori rappresentanti i profili dei film. La creazione dei profili degli utenti è dettata dalla combinazione della valutazione data dall'utente rispetto ai film che ha visto e dei valori delle varie componenti dei profili dei suddetti film. Una volta ottenuti i profili dei film e quelli degli utenti, è possibile stimare una predizione riguardo la valutazione di un utente rispetto ad un film andando ad utilizzare una misura di similarità tra i vettori costituenti i due profili [1].
Dunque, tramite la creazione di profili dei film e profili degli utenti, è possibile stabilire una misura di similarità che andrà

a costituire la preferenza predetta dal sistema per un utente nei confronti di un determinato film.

C. Collaborative filtering

L'altra categoria di sistemi di raccomandazione, di cui si discute nel presente elaborato, sono i sistemi di filtri collaborativi, anche detti sistemi di collaborative filtering. Tale categoria comprende l'insieme di algoritmi volti a predire la valutazione di un utente riguardo un determinato oggetto a partire dai dati riguardanti le passate preferenze di altri utenti con gusti simili. Il principio di fondo è che se un utente A e un utente B esprimono la medesima valutazione riguardo lo stesso oggetto, allora è più probabile che l'utente A possa esprimere una preferenza su un oggetto, mai visto in precedenza, in modo simile a come è stato valutato dall'utente B. Questo è un metodo che si basa unicamente sulle passate valutazioni da parte degli utenti nei confronti dei diversi oggetti, senza occuparsi di creare esplicitamente un profilo per ogni utente o oggetto, come invece accade nel metodo content-based. Due dei principali approcci di collaborative filtering che sono stati utilizzati nel presente elaborato sono gli algoritmi basati su k-Nearest Neighbors (k-NN) e un algoritmo basato sulla fattorizzazione di matrice.

L'utilizzo dell'algoritmo di k-NN in ambito di collaborative filtering serve ad identificare i k utenti più simili all'utente preso in considerazione, nel caso l'algoritmo sia *user-based*. Qualora l'algoritmo sia *item-based*, le entità prese in considerazione non sono gli utenti, ma i film. [2]. Si analizza qui il caso di un algoritmo K-NN *user-based*, come viene sviluppato nel presente elaborato. Dunque, al fine di predire un rating di un determinato utente rispetto ad un certo film, si utilizza l'algoritmo k-NN in modo tale da selezionare i k utenti *nearest neighbors* che mostrano di avere gusti simili all'utente di riferimento. Ciò significa che vengono selezionati i k utenti che mostrano di valutare in modo simile i medesimi film. Una volta ottenuti i *k nearest neighbors*, il rating da predire viene calcolato combinando il rating di ognuno dei k utenti rispetto a quel determinato film con la misura di similarità tra tale utente e l'utente di riferimento. Tale procedimento e le formule utilizzate sono discusse in modo approfondito nella sezione IV. Due importanti problematiche di questo approccio riguardano la sparsità e la scalabilità. Infatti, nei casi in cui vi sia un'elevata quantità di oggetti all'interno del database, raramente si ha a disposizione un numero cospicuo di utenti che abbiano espresso valutazioni su una sostanziosa percentuale di oggetti. Ciò implica una difficoltà nel recuperare una determinata quantità di utenti che mostrino una certa similarità con l'utente di riferimento, con il rischio di ridurre l'accuratezza dell'algoritmo. Mentre per quanto riguarda la scalabilità, questo approccio prevede una potenza computazionale che cresce con la grandezza del training set. Qualora i dataset siano particolarmente numerosi, si presenta la necessità di possedere risorse computazionali adeguate.

Un'altra tipologia di algoritmi utilizzati nei sistemi di raccomandazione sono gli algoritmi basati sulla fattorizzazione di matrice, ossia la fattorizzazione di una matrice in un prodotto di matrici. Una di queste scomposizioni matriciali particolarmente diffusa in ambito di collaborative filtering è la scomposizione a valori singolari (*Singular Value Decomposition - SVD*). Tale metodo prevede la scomposizione di una matrice M di dimensioni $d \times m$ nel prodotto di tre matrici, di cui una matrice unitaria U di dimensioni $d \times d$, una matrice diagonale Σ di dimensioni $d \times m$ che conterrà i valori singolari della matrice originale ed una matrice trasposta V^{T} di dimensioni $m \times m$.

$$M = U \times \Sigma \times V^{\mathrm{T}}$$

SVD ha l'obiettivo di scoprire l'esistenza di dimensioni latenti che possano spiegare i rating osservati, trasportando sia gli utenti che i film nella stessa dimensione latente in modo tale che siano direttamente confrontabili [3]. La dimensione latente cerca di spiegare le valutazioni che caratterizzano sia utenti che film su fattori automaticamente inferiti dalle valutazioni degli utenti. Ogni utente u è associato ad un vettore di features p_u, mentre ogni film i è associato ad un vettore di features q_i. La predizione di un rating è ottenuta calcolando il prodotto interno tra il vettore p_u ed il vettore q_i.

Al fine di migliorare la predizione di un rating, vengono calcolati i cosiddetti baseline estimates, stime di base, che permettono di prendere in considerazioni le tendenze di un utente nell'assegnare valutazioni più alte o basse della media e le tendenze di un film di ricevere valutazioni più alte o basse della media. Tali stime prevedono il calcolo di tre fattori; μ, b_u e b_i. Il fattore μ rappresenta la media delle valutazioni su tutti i film da parte di tutti gli utenti. La componente b_u rappresenta la deviazione osservata delle valutazioni espresse dell'utente u rispetto alla media, mentre b_i costituisce la deviazione osservata delle valutazioni ricevute dall'oggetto i rispetto alla media.

Gli algoritmi qui presentati saranno analizzati più approfonditamente nella sezione IV.

III. Descrizione del dataset

I dati sottoposti ai modelli implementati sono stati attinti dal dataset MovieLens, ossia una raccolta di dati inerenti alla valutazione di pellicole cinematografiche da parte di utenti. Questo dataset, raccolto dal gruppo di ricerca GroupLens dell'Università del Minnesota [4], prevede nella sua interezza la presenza di 26 000 000 di valutazioni da parte di 270 000 utenti per 45 000 film. Per ragioni pratiche e al fine di evitare costi computazionali onerosi, sono stati presi in considerazioni due sottoinsiemi del datataset originale su cui testare i diversi algoritmi di apprendimento, entrambi in formato file csv ed aventi la medesima struttura.

In primo luogo è stato preso in considerazione un file in formato csv denominato *ratings_small.csv*, già presente all'interno del materiale disponibile, composto da 100 004 istanze. A partire da questo file ne è stato generato un ulteriore, denominato *ratings_smaller.csv*, selezionando i dieci utenti le cui valutazioni rispetto ai film presentano una maggiore variabilità. Quest'ultimo file, che presenta 456 istanze, ha

costituito il secondo dataset posto in ingresso ai modelli.
Come affermato in precedenza, entrambi i datatset, *ratings_small.csv* e *ratings_smaller.csv*, presentano la medesima struttura. Ogni istanza è costituita da quattro attributi. Un primo attributo, denominato *userId*, è un numero intero che costituisce un identificativo dell'utente. Il secondo attributo è *movieId*, costituito da un numero intero che identifica il film. Un ulteriore attributo rappresenta l'informazione temporale indicante quando è stata effettuata la valutazione ed è chiamato *timestamp*. Infine è presente un attributo rappresentante la valutazione che l'utente dà ad un determinato film, denominata *rating*, che costituisce l'etichetta del dato. Tale rating può assumere valori da 0.5 a 5, compresi i valori intermedi spaziati di 0.5, per un totale di dieci valori possibili.
Dunque, si giunge ad avere due file csv, *ratings_small.csv* e *ratings_smaller.csv*, contenenti i due dataset rispettivamente dalle dimensioni di 100 004 e 456 istanze, che verranno entrambi sottoposti all'esecuzione di ognuno degli algoritmi implementati.

IV. Il modello implementato

Il progetto presentato è stato sviluppato grazie all'utilizzo di WEKA [5], un software di data mining, e della libreria Surprise [6] in Python, specifica per applicazioni di apprendimento automatico e sistemi di raccomandazione. Grazie a tali strumenti sono stati implementati diversi algoritmi, a cui sono stati posti in ingresso entrambi i dataset esposti nella sezione III. Dopodiché, la performance di tali algoritmi è stata stimata effettuando una 10-fold cross-validation.
In primo luogo, è stata utilizzata una RandomForest, ossia un metodo di apprendimento che produce un ensemble di alberi di decisione. Per quanto riguarda i parametri, la percentuale di bag size, ossia la porzione dei dati da sottoporre ad ogni albero facente parte dell'ensemble, è stata posta al 100%. Il numero di iterazioni è stato pari a 100 e il numero minimo di istanze per foglia è stato posto a 1, senza porre limiti sulla profondità dell'albero. La predizione del rating è effettuata facendo una media pesata degli output degli alberi di decisione appartenenti all'ensemble [7].
Un altro algoritmo utilizzato è stato il metodo di fattorizzazione matriciale di scomposizione a valori singolari (SVD). Qui la predizione di un rating di un utente *u* rispetto ad un film *i*, denotata dalla scrittura $\hat{r}_{\mathrm{ui}}$, viene calcolata utilizzando la seguente formula 1.

$$\hat{r}_{\mathrm{ui}} = \mu + b_{\mathrm{u}} + b_{\mathrm{i}} + q_{\mathrm{i}}^{\mathrm{T}} p_{\mathrm{u}} \tag{1}$$

Al fine di stimare le predizioni ignote, si minimizza l'errore quadratico regolarizzato dato dalla seguente formula 2.

$$\sum_{r_{\mathrm{ui}} \in R_{\mathrm{train}}} (r_{\mathrm{ui}} - \hat{r}_{\mathrm{ui}})^2 + \lambda(b_{\mathrm{i}}^2 + b_{\mathrm{u}}^2 + \|q_{\mathrm{i}}\|^2 + \|p_{\mathrm{u}}\|^2) \tag{2}$$

La minimizzazione viene effettuata utilizzando il metodo di discesa stocastica del gradiente , tramite le seguenti formule.

$$b_{\mathrm{u}} \leftarrow b_{\mathrm{u}} + \gamma(e_{\mathrm{ui}} - \lambda b_{\mathrm{u}})$$
$$b_{\mathrm{i}} \leftarrow b_{\mathrm{i}} + \gamma(e_{\mathrm{ui}} - \lambda b_{\mathrm{i}})$$
$$p_{\mathrm{u}} \leftarrow p_{\mathrm{u}} + \gamma(e_{\mathrm{ui}} \cdot q_{\mathrm{i}} - \lambda p_{\mathrm{u}})$$
$$q_{\mathrm{i}} \leftarrow q_{\mathrm{i}} + \gamma(e_{\mathrm{ui}} \cdot p_{\mathrm{u}} - \lambda q_{\mathrm{i}})$$

Dove $e_{\mathrm{ui}} = r_{\mathrm{ui}} - \hat{r}_{\mathrm{ui}}$, γ indica il learning rate, λ il termine di regolarizzazione, mentre b_u e b_i sono i *baseline estimates*, che vengono inizializzati a zero.
È stata utilizzata una 10-fold cross-validation interna per la scelta dei parametri dell'algoritmo di SVD. Tramite tale tecnica sono stati impostati i seguenti valori sia per il predittore costruito utilizzando il dataset *ratings_smaller.csv* sia per quello costruito partendo dai dati contenuti in *ratings_small.csv*. Il valore delle epoche è stato posto a 10, il learning rate al valore 0.005 e il termine di regolarizzazione a 0.4, per entrambi i dataset.
Un altro algoritmo utilizzato è stato K-Nearest Neighbors, visto in due varianti, entrambe user-based.
La prima variante, chiamata KNNBasic, è la più semplice e calcola la predizione del rating di un utente *u* per un film *i* andando a selezionare i k utenti più simili all'utente *u* e combinando la misura di similarità tra l'utente *u* ed un secondo utente *v*, tra i k utenti selezionati, con il valore di rating espresso dall'utente *v* per quel determinato film *i*. Nello specifico la formula utilizzata per le predizioni è la seguente 3:

$$\hat{r}_{\mathrm{ui}} = \frac{\sum\limits_{v \in N_{\mathrm{i}}^{\mathrm{k}}(u)} sim(u,v) \cdot r_{\mathrm{vi}}}{\sum\limits_{v \in N_{\mathrm{i}}^{\mathrm{k}}(u)} sim(u,v)}. \tag{3}$$

Aggiungendo a tale formula la presenza dei termini di bias dell'utente e del film, si giunge alla seconda variante dell'algoritmo K-NN utilizzata, denominata KNNBaseline. Tale approccio è definito dall'equazione 4;

$$\hat{r}_{\mathrm{ui}} = b_{\mathrm{ui}} + \frac{\sum\limits_{v \in N_{\mathrm{i}}^{\mathrm{k}}(u)} sim(u,v) \cdot (r_{\mathrm{vi}} - b_{\mathrm{vi}})}{\sum\limits_{v \in N_{\mathrm{i}}^{\mathrm{k}}(u)} sim(u,v)} \tag{4}$$

Dove $b_{\mathrm{ui}} = \mu + b_{\mathrm{u}} + b_{\mathrm{i}}$. Per entrambi gli algoritmi KNNBasic e KNNBaseline è stata eseguita una 10-fold cross-validation interna per la stima dei parametri. Il numero di epoche è stato posto pari a 20, il coefficiente di apprendimento γ pari a 0.005 ed il termine di regolarizzazione λ a 0.4. Per il parametro *k*, seguendo le specifiche della libreria utilizzata, è stato imposto che il limite massimo al valore di *k* fosse 40 e il limite minimo fosse 1. Dunque, l'algoritmo utilizza al più 40 *nearest neigbors*, tenendo presente che il numero di *k* varia in base al valore della misura di similarità tra gli utenti. Infatti, vengono presi in considerazione sono gli utenti la cui similarità sia positiva, scartando quelli con similarità negativa, tenendo presente che la misura di similarità è posta a zero qualora i due utenti non abbiamo valutato un certo

numero di film in comune. Maggiori precisazioni sulla misura di similarità utilizzata sono esposte tra poco. Perciò, il valore di *k* varia in base all'istanza considerata, entro i limiti indicati. Per quanto concerne la misura di similarità, è stato utilizzato l'indice di correlazione di Pearson comprendente i termini di bias, definito dalla seguente equazione 5.

$$pearson\_sim(u,v) = \frac{\sum\limits_{i \in I_{\mathrm{uv}}} (r_{\mathrm{ui}} - b_{\mathrm{ui}}) \cdot (r_{\mathrm{vi}} - b_{\mathrm{vi}})}{\sqrt{\sum\limits_{i \in I_{\mathrm{uv}}} (r_{\mathrm{ui}} - b_{\mathrm{ui}})^2} \cdot \sqrt{\sum\limits_{i \in I_{\mathrm{uv}}} (r_{\mathrm{vi}} - b_{\mathrm{vi}})^2}} \tag{5}$$

È stato posto a 5 il numero minimo di film che due utenti debbano avere entrambi valutato per evitare che la similarità tra i due utenti sia posta a zero.
I termini di bias, inizializzati a zero, sono stati calcolati risolvendo il seguente problema di minimizzazione:

$$\min_{b_*} \sum_{(u,i) \in K} (r_{\mathrm{ui}} - \mu - b_{\mathrm{u}} - b_{\mathrm{i}})^2 + \lambda_1 (\sum_u b_{\mathrm{u}}^2 + \sum_i b_{\mathrm{i}}^2) \tag{6}$$

Utilizzando il metodo di discesa stocastica del gradiente tramite le seguenti formule IV.

$$b_{\mathrm{u}} \leftarrow b_{\mathrm{u}} + \gamma(e_{\mathrm{ui}} - \lambda b_{\mathrm{u}})$$
$$b_{\mathrm{i}} \leftarrow b_{\mathrm{i}} + \gamma(e_{\mathrm{ui}} - \lambda b_{\mathrm{i}})$$

Dove $e_{\mathrm{ui}} = (r - \mu - b_{\mathrm{u}} - b_{\mathrm{i}})$.
Tutti gli algoritmi presentati sono stati sperimentati su entrambi i dataset, *ratings_small.csv* e *ratings_smaller.csv*, visti in precedenza nella sezione III.

V. Risultati ottenuti

A seguito della selezione degli algoritmi da utilizzare e dei rispettivi parametri, tutti gli algoritmi sono stati sottoposti ad una 10-fold cross-validation esterna al fine di misurare la performance di ogni algoritmo presentato. Tale operazione è avvenuta per entrambi i dataset. Inoltre, il confronto tra i diversi algoritmi avviene valutando le misure dell'errore medio assoluto (*Mean Absolute Error - MAE)* e della radice dell'errore quadratico medio (*Root Mean Square Error - RMSE*). Maggiore attenzione verrà posta sui valori di RMSE in quanto misura utilizzata nella valutazione e nel confronto degli algoritmi proposti per la competizione Netflix Prize, indice di riferimento in ambito di sistemi di collaborative filtering.

A. Risultati ottenuti sul dataset ratings_smaller.csv

In prima istanza, sono presentati i risultati ottenuti sul dataset *ratings_smaller.csv*. Per quanto concerne l'algoritmo di RandomForest, l'errore medio assoluto ottenuto è stato pari a 1.401 e la radice dell'errore quadratico medio ha raggiunto il valore di 1.7512.
L'algoritmo di SVD ha ottenuto un valore di MAE pari a 1.5896 e un valore di RMSE pari a 1.7253.
L'algoritmo KNNBasic ha conseguito un MAE di 1.7329 e un valore di RMSE di 1.8818.
Infine, l'algoritmo KNNBaseline ha raggiunto un valore di MAE di 1.6050 ed un valore di RMSE di 1.8012. A seguito è riportata una tabella riassuntiva di tali risultati.

Algoritmi	MAE	RMSE
RandomForest	1.401	1.7512
SVD	1.5896	1.7253
KNNBasic	1.7329	1.8818
KNNBaseline	1.6050	1.8012

Tabella I: Tabella riportante le misure di MAE e RMSE ottenute sul dataset *ratings_smaller.csv* dai diversi algoritmi utilizzando una 10-fold cross-validation.

Le misure di MAE e RMSE ottenute dai diversi algoritmi sono osservabili tramite l'istogramma in figura 1. Concentrandosi sui valori di RMSE, misura che, rispetto alla misura MAE, conferisce un maggior peso ad errori più elevati, è evidente come l'algoritmo di SVD ottenga il risultato migliore, seguito in ordine dall'algoritmo di Random Forest, KNNBaseline ed infine da KNNBasic.
Nel grafico 2 viene mostrato un violin plot dei valori RMSE dei diversi algoritmi implementati. Qui per ogni algoritmo è possibile notare la distribuzione dei valori di RMSE, insieme alla media e alla mediana di tali valori. Tale grafico supporta l'affermazione fatta in precedenza, secondo cui l'algoritmo di SVD si conferma avere una prestazione migliore rispetto agli altri tre algoritmi, in quanto avente media inferiore e mediana coincidente con la media.
Ad ogni modo, i risultati ottenuti non risultano essere pienamente soddisfacenti se confrontati con il risultato ottenuto dal sistema Pragmatic Chaos di Bell e Koren [8] [9], il sistema di raccomandazione vincitore del Netflix Prize nel 2009, che ottenne un valore di RMSE=0.8567 [10]. Infatti, normalizzando il valore di RMSE con la formula (7) riportata di seguito, se il risultato di Bell e Koren ottiene un valore di RMSE normalizzato (NRMSE) pari a 21,4175%, la miglior performance ottenuta dagli algoritmi qui implementati presenta un NRMSE=38,34%.

$$NRMSE = 100 \cdot \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{r}_{\mathrm{i}} - r_{\mathrm{i}})^2}}{r_{\max} - r_{\min}} \tag{7}$$

B. Risultati ottenuti sul dataset ratings_small.csv

Dopo aver preso visione dei risultati ottenuti sul dataset più ridotto dei due presentati, i medesimi algoritmi sono stati valutati sull'insieme di dati più numeroso al fine di esaminare le diverse prestazioni degli algoritmi su dataset di cardinalità differente. Anche in questo caso, le misure utilizzate nell'analisi della performance degli algoritmi sono stati l'errore medio assoluto e la radice dell'errore quadratico medio.
La performance dell'algoritmo di RandomForest ha conseguito un valore di MAE pari a 0.7082 e un valore di RMSE uguale a 0.9294.
Per quanto riguarda l'algoritmo di scomposizione a valori

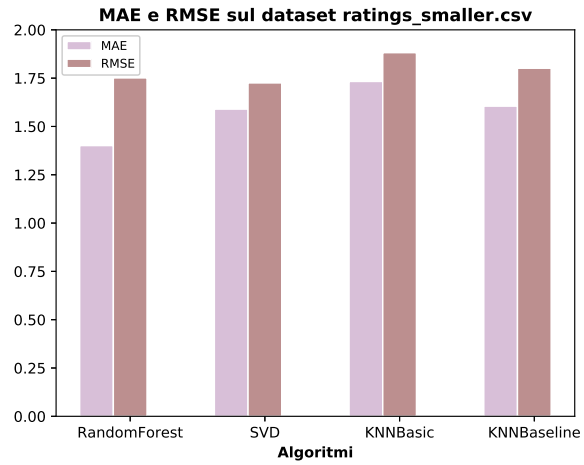

Figura 1: Istogramma rappresentante i valori di MAE e RMSE per gli algoritmi di Random Forest, SVD, KNNBasic e KNN-Baseline, implementati nell'elaborato proposto, addestrati sul dataset *ratings_smaller.csv*.

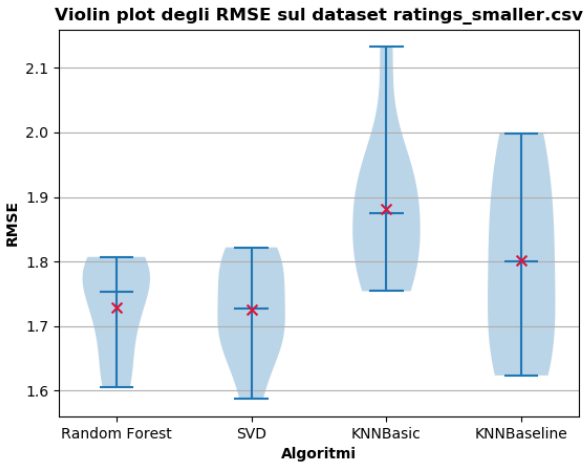

Figura 2: Nel grafico sovrastante è rappresentato un violin plot che pone a confronto i valori di RMSE degli algoritmi implementati a partire dal dataset *ratings_smaller.csv*. Qui il simbolo *x* in rosso rappresenta la media, mentre la linea blu, che si posiziona all'interno del plot tra le due estremità, rappresenta la mediana.

singolari, questo ha ottenuto i valori di 0.7048 per l'errore medio assoluto e di 0.9084 per la radice dell'errore quadratico medio.
L'algoritmo KNNBasic ha acquisito un valore MAE uguale a 0.7387 e un valore di RMSE pari a 0.9615.
Infine, l'algoritmo KNNBaseline ha ottenuto un valore di MAE di 0.6827, mentre il valore di RMSE raggiunto è pari a 0.8920.
È presentata di seguito una tabella riassuntiva dei risultati raggiunti sul dataset *ratings_small.csv*.

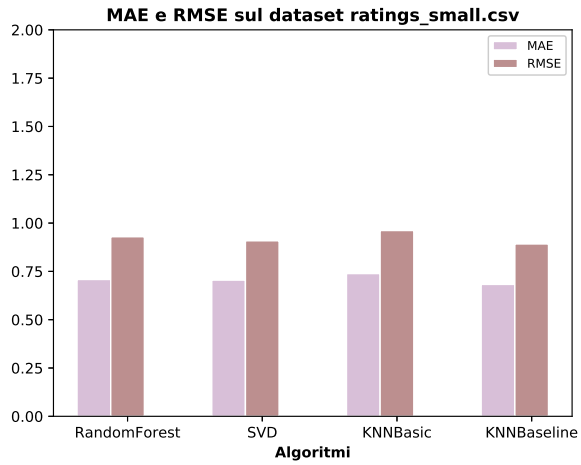

Figura 3: Istogramma rappresentante i valori di MAE e RMSE per gli algoritmi di Random Forest, SVD, KNNBasic e KNN-Baseline, implementati nell'elaborato proposto, addestrati sul dataset *ratings_small.csv*.

Algoritmi	MAE	RMSE
RandomForest	0.7082	0.9294
SVD	0.7048	0.9084
KNNBasic	0.7387	0.9615
KNNBaseline	0.6827	0.8920

Tabella II: Tabella riportante le misure di MAE e RMSE ottenute sul dataset *ratings_small.csv* dai diversi algoritmi utilizzando una 10-fold cross-validation.

Le misure di MAE e RMSE ottenute dai diversi algoritmi sono osservabili in figura 3. Concentrandosi sui valori di RMSE come in precedenza, è evidente come l'algoritmo di KNNBaseline ottenga il risultato migliore, seguito in ordine dall'algoritmo di SVD, Random Forest ed infine da KNN-Basic. In questo caso, i risultati ottenuti risultano essere soddisfacenti se confrontati con il risultato ottenuto da Bell e Koren (RMSE=0.8567).
Ponendo attenzione al violin plot in figura 4, è possibile notare come l'algoritmo KNNBaseline dimostri effettivamente di avere una performance migliore rispetto agli altri tre algoritmi, che anche in tale grafico confermano ciò che si evince dall'istogramma precedentemente osservato in figura 3.
Normalizzando i valori di RMSE con la medesima formula 7 utilizzata in precedenza, la performance ottenuta dall'algoritmo KNNBaseline ha conseguito un NRMSE=19,8223%, competitivo rispetto al risultato ottenuto dal sistema Pragmatic Chaos (NRMSE=21,4175%). A fronte di tale confronto, bisogna, tuttavia, tenere in considerazione il fatto che i due sistemi, quello qui presentato e quello implementato da Bell e Koren, si avvalgono di dataset differenti, i cui rating prevedono scale di valutazioni diverse.

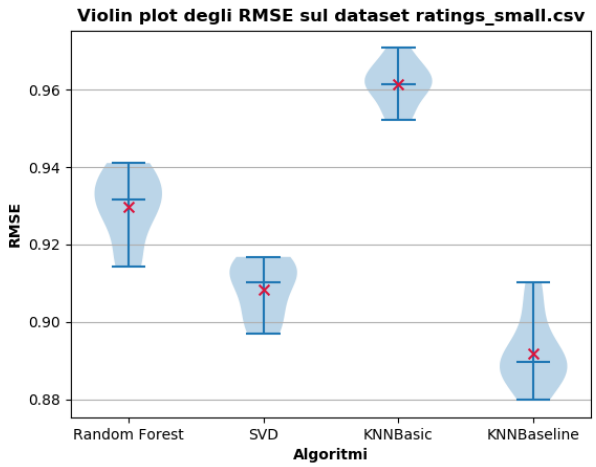

Figura 4: Nel grafico sovrastante è rappresentato un violin plot che pone a confronto i valori di RMSE degli algoritmi implementati a partire dal dataset *ratings_small.csv*. Qui il simbolo *x* in rosso rappresenta la media, mentre la linea blu che si posiziona all'interno del plot, tra le due estremità, rappresenta la mediana.

VI. Considerazioni critiche e conclusioni

Dopo aver analizzato i risultati ottenuti dai quattro algoritmi implementati su entrambi i dataset, si evince come le prestazioni degli algoritmi siano estremamente influenzate dalla cardinalità del dataset a disposizione. Infatti, i risultati migliori sono stati ottenuti utilizzando il dataset di cardinalità maggiore, *ratings_small.csv*. A fronte di un numero maggiore di esempi disponibili, i valori di RMSE si sono dimezzati rispetto ai valori ottenuti sul dataset di cardinalità inferiore.
Come è stato evidenziato nella sezione V, per il dataset *ratings_smaller.csv* la miglior performance è stata ottenuta dall'algoritmo SVD, mentre per il dataset *ratings_small.csv* il miglior risultato è stato ottenuto dall'algoritmo KNNBaseline. Il fatto che tali algoritmi dimostrino una performance migliore può essere dovuto al fatto che questi tengano conto dei termini di bias per utente e film, non considerati negli algoritmi di Random Forest e KNNBasic.
Dunque, si può affermare come nella costruzione di un sistema di raccomandazione sia fondamentale avere a disposizione un dataset numeroso, in grado di migliorare l'accuratezza del sistema prodotto. Inoltre, risulta importante tenere in considerazione i termini di bias per utenti e film, che permettono di accrescere ulteriormente la performance degli algoritmi. Avvalendosi di queste due accortezze, ci si aspetta che le prestazioni degli algoritmi utilizzati migliorino considerevolmente.

Riferimenti bibliografici

[1] R. A. and U. J., *Mining of Massive Datasets*. No. 3, Cambridge University Press, 2010. Cap. 9, pag. 307-341.
[2] S. B. K. G. K. J. and R. J., "Item-based collaborative filtering recommendation algorithms," *GroupLens Research Group/Army HPC Research Center Department of Computer Science and Engineering University of Minnesota, Minneapolis, MN 55455*, 2001.
[3] K. Y., "Factor in the neighbors: Scalable and accurate collaborative filtering.," *ACM Transactionson Knowledge Discovery from Data 4, 1–24.*, 2010.
[4] "Movielens - grouplens dataset." https://grouplens.org/datasets/movielens/.
[5] "Weka - data mining software." https://www.cs.waikato.ac.nz/ml/weka/.
[6] "Surprise library." https://surprise.readthedocs.io/en/stable/index.html.
[7] I. H. Witten, ed., *Data Mining with Weka*, Department of Computer Science, University of Waikato, New Zealand, 2013. Class 4 – Lesson 1 Classification boundaries.
[8] B. R. M. K. Y. and V. C., "The bellkor 2008 solution to the netflix prize.," tech. rep., ATT Labs Research, 2008.
[9] K. Y., "Matrix factorization techniques for recommender systems," *Yahoo Research, ATT Labs—Research*, 2009.
[10] "Netflix prize leaderboard." www.netflixprize.com/leaderboard.html.