Università degli Studi di Verona

Big Data

Report del progetto

Mattia Zorzan - VR464472

10 dicembre 2021

Indice

1 Descrizione del Progetto

Il progetto si propone di eseguire alcune interrogazioni sul dataset MovieLens. Nello specifico queste vengono eseguite sulla variante **MovieLens 1M**, di dimensione minore.
Di seguito un elenco delle interrogazioni richieste:

- **EXPLORATORY ANALYSIS**
  - *Number of ratings* for each **movie** (and its *distribution*)
  - *Number of ratings* for each **user** (and its *distribution*)
  - *Average* score received by each **movie**
  - *Average* score given by each **user**
  - Top **K** *movies* with at least **R** *ratings*
- **ADVANCED QUERIES**
  - Find if there is a *correlation* between the *standard deviation* of the ratings a movie has received, and the *number of ratings*
  - Find the *evolution over time* (with a granularity of **N** months) of the *number of ratings* and the *average rating*: do high rated movies maintain their ratings? Are low rated movies "abandoned" after a while?
  - Find how the *text* of each movie changes as we progressively **remove** the ratings from users that rated more and more movies. For instance, we can identify *different groups of users* (who rated less than 10 movies, who rated between 11 and 30 movies, ...) and we can compute the *average rating* considering all the groups, then only the groups of users with at least 11 ratings, and so forth
  - Is it possible to identify *groups of similar movies* based on the ratings they received from the users? For instance, if movies `m1` and `m2` have both obtained 5 stars from users `u1` and `u2`, they may be considered similar

Era possibile scegliere se esguire tutte le interrogazioni oppure solo un sottinsieme di esse.

2 Descrizione delle Interrogazioni

Per comodità, in questo report è stata omessa la descrizione delle soluzioni per l'*exploratory analysis*, ritengo che il codice sia auto-esplicativo nei confronti delle operazioni eseguite per ottenere i risultati. Verranno trattate solo le *advanced queries*.
In aggiunta a questo report è possibile trovare un breve commento alle query direttamente nel Notebook, espresso in tramite celle *Markdown*.

2.1 Query 1

Per dimostrare la correlazione tra *standard deviation* ed il numero dei *ratings* si è in primo luogo ottenuto il numero di valutazioni per ogni film presente nel dataset. Limitando (in 4 *DataFrame* diversi) il numero di film presi in considerazione si può a questo punto vedere come la diminuzione dei campioni presi in considerazione "*ammorbidisca*" gradualmente la curva nel *distplot*. I 4 *DataFrame* sono:

1. **1M Samples:** Prende in considerazione l'intero dataset
2. **100K Samples:** Prende in considerazione i primi 100.000 film del risultato del conteggio
3. **100K Samples:** Prende in considerazione i primi 1.000 film del risultato del conteggio
4. **100K Samples:** Prende in considerazione i primi 100 film del risultato del conteggio

Il campione può essere considerato "*randomico*" in quanto non viene fatto alcun tipo di ordinamento sui risultati della query, che vengono già restituiti in ordine sparso.
Considerando il *distplot* dei dati la forma della distribuzione potrebbe sembrare **gaussiana**, per verificare ciò è stato eseguito uno **skewness** test su tutti e 4 i *DataFrame*, dando risultati nulli o negativi. Non è quindi una distribuzione normale.
Osservando l'andamento del grafico è possibile riconoscere un andamendo iperbolico, riconducibile ad una distribuzione **paretiana**. Non avendo trovato alcun test di *paretianità* già implementato è stata generata una distribuzione paretiana generica tramite il metodo `genpareto()` di *scipy*. Accostando i due grafici la somiglianza risulta evidente.

2.2 Query 2

Per comodità nell'analisi è stato deciso di riferirsi ai quartili, quindi `N=4`, per il calcolo delle medie. Come primo passo è stata creata una **UDF** (*User Defined Function*) che andasse ad etichettare ogni riga del dataset come segue:

- **Q1:** Etichetta di tutte le *Row* aventi campo `Date` compreso tra *Aprile* e *Giugno 2000*
- **Q2:** Etichetta di tutte le *Row* aventi campo `Date` compreso tra *Lugio* e *Settembre 2000*
- **Q3:** Etichetta di tutte le *Row* aventi campo `Date` compreso tra *Ottobre* e *Dicembre 2000*
- **Q4:** Etichetta di tutte le *Row* aventi campo `Date` compreso tra *Gennaio* e *Marzo 2001*
- **Remaining:** Etichetta di tutte le *Row* aventi campo `Date` successivo a *Aprile 2001* (compreso)

In seguito all'etichettatura delle *Row* si è implementata un'ulteriore **UDF**, che andasse a valutare la "frequenza" nei quartili di appartenenza alla categoria **High Rated** o alla categoria **Low Rated**. Per differenzaiare le due classi si è deciso di utilizzare il voto a *3 stelle* come threshold, se per la maggior parte dei quartili il film è etichettato come **High Rated** allora la sua etichetta "globale" sarà quella, **Low Rated** altrimenti.
In seguito a quest'operazione, possiamo vedere che in media i film etichettati come **High Rated** sono ottengono in media il doppio delle recensioni dopo l'anno rispetto agli altri.
Con la seconda cella Jupyter invece si va ad analizzare la stabilità nel numero di rating dopo l'anno.
Dallo *scatterplot* possiamo vedere che per entrambe le categorie c'è una tendenza all'abbandono, quasi totale per i film **Low Rated** mentre più mitigata per i film **High Rated**.
Segno evidente di questo è la grande presenza di film "apprezzati" nel range [40, 100] (pallini *blu*), sono invece quasi del tutto assenti nello stesso range i film "poco apprezzati" (pallini *arancioni*).

2.3 Query 3

La prima operazione necessaria per lo svolgimento dell'interrogazione è l'etichettatura dei dati per ottenere le varie categorie degli utenti, basate sul *numero dei rating* da loro espressi.
Per fare ciò è stata definita una **UDF** che divide gli utenti nelle seguenti categorie:

- **1.** Utenti che hanno espresso al massimo 30 ratings
- **2.** Utenti che hanno espresso tra i 31 ed i 100 ratings
- **3.** Utenti che hanno espresso tra i 101 ed i 200 ratings
- **4.** Utenti che hanno espresso oltre 200 ratings

Modificando gli estremi nei costrutti `if` all'interno del corpo della **UDF** è anche possibile adattare le categorie alle proprie esigenze, rendendo quindi possibile "personalizzare l'analisi".
In seguito a questo si sono definiti i vari *DataFrame pandas* contenti le medie come segue:

- **all-users:** Le *medie* dei ratings considerando ogni categoria d'utente esistente
- **no-c4:** Le *medie* dei ratings considerando solo gli utenti che hanno espresso tra 0 e 200 ratings
- **no-c34:** Le *medie* dei ratings considerando solo gli utenti che hanno espresso tra 0 e 100 ratings
- **no-c234:** Le *medie* dei ratings considerando solo gli utenti che hanno espresso tra 0 e 30 ratings

Ottenuti questi dati è stato eseguito il *plotting* dei dati, dal *lineplot* possiamo vedere la variazione delle medie nelle varie categorie, si può infatti notare l'aumento di valori estremi man mano che le varie categorie d'utente vengono rimosse dal dataset.
Nel secondo invece si possono confrontare le densità di probabilità delle varie categoria, sempre più simili ad una linea *spezzata* rimuovendo le varie categorie.

2.4 Query 4

Al fine di completare quest'interrogazione è stato negessario una grossa riorganizzazione dei dati. Questi sono stati riportati nel seguente formato, ogni cella del *DataFrame* rappresenta il rating dato da un utente ad un dato film, quindi ogni entry è esprimibile come `rating[MovieID][UserID]`. Quest'operazione avviene in due parti:

1. Tramite pySpark ed una **UDF** ottengo un *DataFrame* in cui ogni *Row* è composta da uno `UserID` ed un oggetto `tuple` di *Python* `(MovieID, Rating)`

2. Il *DataFrame* del punto precedente viene esportato come *DataFrame pandas*, raggruppato per utente e poi utilizzato per la creazione del *DataFrame pandas* su cui andremo ad eseguire l'operazione di *clustering*, organizzato come da descrizione precedente

Non è possibile però che ogni utente avesse espresso una valutazione per ogni singolo film, i campi vuoti sono quindi stati riempiti con il valore medio tramite il metodo `mean()` dei *DataFrame pandas*.
Viene a questo punto eseguito il clustering tramite l'oggetto `KMeans` di *scikit-learn*, in seguito a questo viene utilizzato l'oggetto `PCA` per rendere *plottabili* in 2 dimensioni i dati clusterizzati, ottentndo lo *scatterplot* visibile alla fine del Jupyter Notebook.