Report Progetto FIA

Giugno,Luglio 2023

Contents

1 Introduzione

Negli ultimi anni le piattaforme di streaming online per la visione di contenuti cinematografici sono sempre più ampiamente utilizzate, e conseguentemente vengono sviluppati diversi sistemi che mirano a consigliare ad un determinato utente un contenuto che possa piacergli.

Abbiamo quindi puntato alla creazione di un sistema di raccomandazione intelligente che possa suggerire dei contenuti cinematografici ad uno specifico utente sulla base delle sue preferenze, fornite da diversi dati raccolti precedentemente.

PEAS	
Performance	La misura di performance dell'agente 'e la sua capacit'a di predire quanto più possibile una v
Enviroment	L'ambiente in cui opera l'agente è una possibile piattaforma con accesso a dati su film e uten Completamente osservabile , perchè si ha visibilità su tutti i dati in ogni momento Stocastico, in quanto le info relative a film e utenti possono essere soggetti a cambiamenti e a Episodico, in quanto ogni volta la predizione sarà un evento atomico e dipenderà dalle inform Statico, perchè nel momento della predizione l'ambiente è invariato Discreto, perchè vengono forniti un numero limitato di percezioni Singolo, lavora un solo agente
Actuators	Vengono fornite delle predizioni su determinati film
Sensors	Vengono selezionati dei film sul quale effettuare la predizione

2 Descrizione dell'agente

2.1 Obiettivi

Lo scopo del progetto è quello di realizzare un agente intelligente che sia in grado di predirre un rating positivo riguardo un film che l'utente ancora deve visionare, in modo da suggerire un contenuto apprezzabile.

2.2 Specifica PEAS

2.3 Analisi del problema

Abbiamo inizialmente optato per risolvere il problema sulla selezione e valutazione dei film con un algoritmo di clustering. Questa opzione ci ha portato a fuorviare da una soluzione ammissibile per diversi motivi:

- Costruire cluster di film consigliabili era piuttosto difficile, in quanto utilizzando una metrica di similarità riuscivamo a creare un solo cluster ottimale di film consigliabili.

- La mancanza di un dataset di info sugli utenti ci ha reso ulteriormente difficile la selezione di film in quanto dovevamo inserire info manuali.

- Risultava altrettanto complessa la valutazione del modello.

Pertanto abbiamo cercato un dataset che potesse darci info riguardo una serie di utenti, e valutato di cambiare tecnica spostandoci su algoritmi di classificazione e regressione che si sposassero meglio con i dataset (film e utenti) che abbiamo deciso di utilizzare. I dataset si abbinano molto bene in quanto è possibile convergere e creare nuove tabelle tra le votazioni di un utente e le rispettive pellicole votate.

L'idea è stata quindi di combinare i due dataset e lavorare su un lavoro di predizione sulle votazioni che gli utenti avrebbero assegnato ai film sulla base di quelli precedentemente da loro votati.

3 Raccolta, analisi e preprocessing dei dati

3.1 Scelta del dataset

Venendo al dataset necessario per la creazione del modello di machine learning, le possibili strade da seguire erano due:

1. Creare un dataset da zero
2. Cercare dei dataset sulla rete

La prima opzione era molto dispendiosa in termini di:

• Tempo • Possibile inconsistenza dei dati, dovendone generare di nuove o casuali

Si è deciso pertanto di ricercare un dataset su kaggle, e abbiamo trovato movielens : https://www.kaggle.com/datasets/prajitdatta/movielens-100k-dataset

3.2 Analisi del dataset

Il dataset fornito da kaggle ci ha offerto una serie di dati raccolti dall'università del Minnesota per un progetto al quale hanno lavorato. Il dataset consiste principalmente in:

* 100,000 votazioni (1-5) da 943 utenti su 1682 movies. * Ogni utente ha votato almeno 20 films.

* Semplici info demografiche degli utenti (età, sesso, occupazione, cod.postale)

Il dataset è suddiviso in diversi file rappresentativi di diverse informazioni: Tabella relazionale che mette in relazione l'id utente con quello di un film

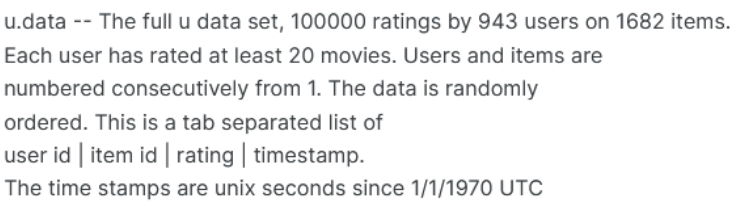

Figure 1:

votato.

Tabella che conta il numero di utenti, film, e votazioni del dataset.

Tabella che contiene info riguardo i films: genere, titolo, id e data di rilascio.

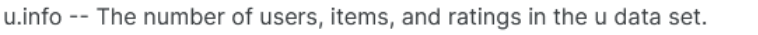

Figure 2:

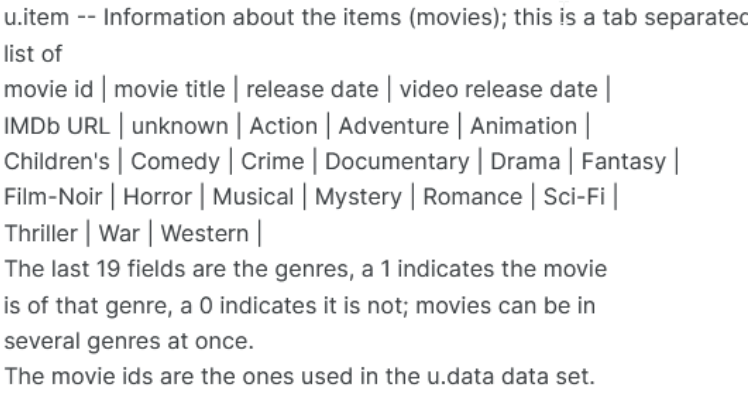

Figure 3:

Figure 4:

Una lista di tutti i generi contemplati dai film.

u.user -- Demographic information about the users; this is a tab
separated list of
user id | age | gender | occupation | zip code
The user ids are the ones used in the u.data data set.

Figure 5:

Info demografiche su ciascun utente.

3.3 Preprocessing dei dati

La formattazione dei dati è avvenuta in diverse fasi, cercando di adattare i dati quanto piu possibile all'algoritmo utilizzato.

Per come la tabella è strutturata, avevamo una colonna contenente i generi al quale apparteneva il film. Abbiamo estratto tutti i possibili generi e trasformato ognuno di essi in una specifica colonna. Dopodichè abbiamo utilizzato una codifica binaria per rappresentare l'appartenenza di un film a un determinato genere:

```
with open('./dataset/movies2', encoding = "ISO-8859-1") as content:
    colonne = ['movie_id', 'movie_title', 'release_date', 'video_release_date', 'imdb_url']
    generi = ['unknown', 'action', 'adventure', 'animation', 'children', 'comedy', 'crime', 'documentary', 'drama', 'fantasy',
          'film-noir',  'horror', 'musical', 'mystery', 'romance', 'sci-fi', 'thriller', 'war', 'western']
    colonneG = colonne + generi
    movies = pd.DataFrame(columns=colonneG)
    i = 0
    for x in content:
        x = x.split("|")
        x[-1] = x[-1][:-1]
        if x[1][-1] == ' ':
            x[1] = x[1][:-1]
        movies.loc[i] = [word if word!='' else "empty" for word in x]
        i = i + 1
movies['movie_id'] = movies['movie_id'].astype('int64')
movies[generi] = movies[generi].astype('category')
```

Figure 6:

Si è continuato con la pulizia dei dati, rimuovendo le colonne inutili ai fini del nostro algoritmo:

Successivamente abbiamo inserito la colonna target, contenente la variabile dipendente "like", necessaria per il supporto e la predizione del valore:

Effettuando il join tra due tabelle, film La tabella risultante avrà questo aspetto :

```
#@title Data cleaning: Tabelle con feature inutili
movies = movies.drop('video_release_date', axis=1)
movies = movies.drop('release_date', axis=1)
movies = movies.drop('imdb_url', axis=1)
```

Figure 7:

```
#@title Inserimento colonna target con relativo riempimento
movies['like'] = 0

for i in range(1600):
  if((i % 3) == 0):
    movies.loc[i, 'like'] = 1
  if((i % 8) == 0):
    movies.loc[i , 'like'] = 2
  if((i % 7) == 0):
     movies.loc[i , 'like'] = 3
  if((i % 5) == 0):
    movies.loc[i , 'like'] = 4
  if((i % 6) == 0):
    movies.loc[i , 'like'] = 5


movies.head(100)
```

Figure 8:

```
#@title Inserimento colonna target con relativo riempimento
movies['like'] = 0

for i in range(1600):
  if((i % 3) == 0):
    movies.loc[i, 'like'] = 1
  if((i % 8) == 0):
    movies.loc[i , 'like'] = 2
  if((i % 7) == 0):
     movies.loc[i , 'like'] = 3
  if((i % 5) == 0):
    movies.loc[i , 'like'] = 4
  if((i % 6) == 0):
    movies.loc[i , 'like'] = 5


movies.head(100)
```

Figure 9:

4 Algoritmo di clustering

5 Algoritmo di regressione e classificazione

6 Evoluzione del modello

Federico Pomposelli, Marco Gallo, Nicholas De Marco