Deciphering Amazon's User based Recommendation System

Chuks Chiazor

October 2023

1 Introduction

In the age of data-driven decision-making, building a recommender system is a vital skill for any data scientist. This article offers a step-by-step guide to creating a movie recommender system using Amazon's movie ratings dataset. By the end of this guide, you'll gain not just technical know-how but also insights into the practical applications of such models, particularly in the educational sector. Unveiling the patterns within Amazon's movie ratings requires not just theoretical knowledge but practical, executable code. Herein, we shall detail how Python's libraries are utilized to transform raw data into actionable insights.

1.1 Data Loading and Exploration

First, we initialize the data analysis process by loading and exploring the dataset. We start by importing Pandas, a cornerstone library for data manipulation. Exploration begins by examining the first few rows, the shape, size, and a statistical summary of the dataset. The .head() method gives us a glimpse into the dataset, while .shape reveals its size.

```
import pandas as pd
amazon = pd.read_csv("path_to/Amazon.csv")
print(amazon.head()
print("Dataset␣shape:", amazon.shape)
```

1.2 Data Cleaning and Preprocessing

To ensure the integrity of our analysis, we replace missing values and refine our DataFrame. Here, fillna addresses missing values, crucial for maintaining data integrity. Dropping irrelevant columns like 'userid' focuses our analysis on movie ratings.

```
Amazon_filtered = amazon.fillna(value=0)
Amazon_filtered1 = Amazon_filtered.drop(columns='user_id')
```

1.3 Diving Deep into Views and Ratings

Here we analyze the movie popularity by determining the movie with the maximum views which entails summing up the ratings and finding the max. Summing ratings per movie gives us a popularity metric. The argmax() function identifies the movie with the highest viewership.

```
Amazon_max_views = Amazon_filtered1.sum()
max_views_index = Amazon_max_views.argmax()
print("Most␣viewed␣movie␣index:", max_views_index)
```

1.4 Average Ratings: The Quest for Quality

Calculating the average rating across movies provides insight into overall viewer satisfaction.

```
average_ratings = Amazon_max_views.mean()
print("Average␣rating:", average_ratings)
```

1.5 Building the Recommender Model: SVD Algorithm

1. Data Formatting for Surprise: We set the stage for recommendation by preparing our data for the Surprise library.

```
from surprise import Reader, Dataset
reader = Reader(rating_scale=(-1, 10))
data = Dataset.load_from_df(melt_df.fillna(0), reader)
```

2. Model Training and Evaluation: This system is developed to recommend movies based on user preferences. We use the Surprise library, a go-to for recommender systems.

```
from surprise import SVD
from surprise.model_selection import train_test_split,
cross_validate

trainset, testset = train_test_split(data, test_size=0.25)
algo = SVD()
algo.fit(trainset)
predictions = algo.test(testset)
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3,
verbose=True)
```

3. Making Predictions: The SVD algorithm, a powerful tool for matrix factorization, predicts user ratings for movies.

```
user_id = 'A1CV1WROP5KTTW'
movie = 'Movie6'
rating = 5
algo.predict(user_id, movie, r_ui=rating)
```

The result is something like this:

```
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.2849  0.2784  0.2827  0.2820  0.0027
MAE (testset)     0.0430  0.0424  0.0420  0.0425  0.0004
Fit time          81.84   79.53   79.48   80.28   1.10
Test time         7.78    6.13    5.49    6.47    0.96
{'test_rmse': array([0.28486078, 0.2784318 , 0.2826823 ]), 'test_mae': array([0.04298138, 0.04243755, 0.041962  ]),
'fit_time': (81.84026455879211, 79.53197860717773, 79.4800181388855), 'test_time': (7.775432348251343, 6.1267905235
29053, 5.494760751724243)}
```

2 Conclusion (Model Application)

The outlined script does more than just unravel the intricacies of Amazon's movie rating data; it arms with a versatile analytical toolkit. This methodology outlined here transcends entertainment. It can aswell be adapted to various data landscapes. In education for example, a similar recommender system can suggest personalized learning materials, courses, or even extracurricular activities, enhancing student engagement and learning outcomes.

2.1 Education example:

Utilizing the Surprise library to recommend educational resources based on student preferences.

1. Dataset:

```
import pandas as pd

# Suppose 'education_data.csv' contains columns 'student_id',
'resource_id', and 'rating'
education_data = pd.read_csv("path_to/education_data.csv")
print(education_data.head()
```

2. Analyzing Resource Popularity:

Identify the most popular or highly rated educational resources.

```
resource_popularity = education_filtered.groupby
'resource_id').sum()
most_popular_resource =
resource_popularity['rating'].idxmax()
print("Most␣popular␣resource:
{most_popular_resource}"
```

3. Building an Educational Recommender Model:

Utilize the Surprise library to recommend educational resources based on user preferences.

```
resource_popularity =
education_filtered.groupby('resource_id').sum()
most_popular_resource =
resource_popularity['rating'].idxmax()
print("Most␣popular␣resource:␣{most_popular_resource}"
```

4. Recommending Educational Resources Predict ratings for a specific user and educational resource, demonstrating the model's applicability in an educational setting.

```
Student_id = 'student123'
resource = 'course456'
predicted_rating = algo.predict(Student_id, resource).est
print("Predicted␣rating␣for␣resource
{resource}␣by␣user␣{user_id}:␣{predicted_rating}"
```

Check out the source code and dataset on my Github. I look forward to hearing any feedback or questions.