Muhammad Afzal Hashmi

F2021266252

Movie Recommendation System

Introduction

With a vast number of movies available, choosing what to watch can be challenging. This project aims to create a system that recommends movies to users based on their past ratings using only basic Python libraries.

Objective

Develop a model to suggest movies to users based on their preferences using NumPy and pandas.

Dataset Description

Name: MovieLens Dataset

Source: MovieLens

Features:

userId: Unique user identifier

movieId: Unique movie identifier

rating: Rating given by the user

timestamp: When the rating was given

Methodology

Data Cleaning:

Handle missing values and duplicates using pandas.

Exploratory Data Analysis (EDA):

Use pandas for data manipulation and simple visualizations (e.g., histograms, bar charts).

Model Development:

Collaborative Filtering:

User-based: Calculate similarity between users using cosine similarity or Pearson correlation with NumPy.

Item-based: Calculate similarity between movies using the same methods.

Matrix Factorization:

Implement Singular Value Decomposition (SVD) manually with NumPy.

Model Evaluation:

Calculate Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) manually using NumPy.

Implement cross-validation using pandas.

Tools and Technologies

Python Libraries:

Data Handling: pandas, NumPy

Visualization: Basic plotting with pandas (e.g., pandas.DataFrame.plot)

Expected Outcomes

A functional movie recommendation system.

Insights into user preferences and movie popularity.

Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse import csr_matrix
from sklearn.metrics.pairwise import cosine_similarity

# Load datasets
genome_scores = pd.read_csv('genome-scores.csv')
genome_tags = pd.read_csv('genome-tags.csv')
links = pd.read_csv('links.csv')
movies = pd.read_csv('movies.csv')
ratings = pd.read_csv('ratings.csv')
tags = pd.read_csv('tags.csv')

# Convert timestamps to datetime
ratings['timestamp'] = pd.to_datetime(ratings['timestamp'], unit='s')
tags['timestamp'] = pd.to_datetime(tags['timestamp'], unit='s')

# Reduce dataset size for testing purposes
small_ratings = ratings.head(100000)  # Adjust this number as needed

# Merge datasets for analysis
```

```
data = pd.merge(small_ratings, movies, on='movieId')
tags['tag'].fillna('')
data = pd.merge(data, tags[['userId', 'movieId', 'tag']], on=['userId',
'movieId'], how='left')

# Basic data analysis
average_ratings =
data.groupby('title')['rating'].mean().sort_values(ascending=False)
rating_counts =
data.groupby('title')['rating'].count().sort_values(ascending=False)

# Create user-item interaction matrix
user_item_matrix = data.pivot_table(index='userId', columns='title',
values='rating', fill_value=0)
user_item_matrix_sparse = csr_matrix(user_item_matrix.values)

# Calculate cosine similarity between users
user_similarity_sparse = cosine_similarity(user_item_matrix_sparse)
user_similarity_df = pd.DataFrame(user_similarity_sparse,
index=user_item_matrix.index, columns=user_item_matrix.index)

# Function to get movie recommendations for a user
def recommend_movies(user_id, num_recommendations=5):
    similarity_scores = user_similarity_df[user_id]
    similar_users = similarity_scores.sort_values(ascending=False).index[1:]
    similar_user_movies = user_item_matrix.loc[similar_users].sum(axis=0)
    user_movies = user_item_matrix.loc[user_id]
    movies_to_recommend = similar_user_movies[user_movies == 0]
    recommended_movies =
movies_to_recommend.sort_values(ascending=False).head(num_recommendations)
    return recommended_movies.index.tolist()

# Example: Recommend movies for user with ID 1
recommended_movies = recommend_movies(user_id=1)
print("\n\nRecommended movies for user 1:")
print(recommended_movies)

# Visualization with Matplotlib Subplot

# Create a figure and subplots using Matplotlib's subplot
fig = plt.figure(figsize=(10, 10))

# Distribution of Average Ratings
ax1 = fig.add_subplot(221)
average_ratings.hist(bins=50, color='skyblue', edgecolor='black', ax=ax1)
```

```
ax1.set_title('Distribution of Average Ratings')
ax1.set_xlabel('Average Rating')
ax1.set_ylabel('Frequency')

# Distribution of Rating Counts
ax2 = fig.add_subplot(222)
rating_counts.hist(bins=50, color='salmon', edgecolor='black', ax=ax2)
ax2.set_title('Distribution of Rating Counts')
ax2.set_xlabel('Number of Ratings')
ax2.set_ylabel('Frequency')

# Top 10 Movies by Average Rating
ax3 = fig.add_subplot(223)
top_10_avg_ratings = average_ratings.head(10)
top_10_avg_ratings.plot(kind='barh', color='purple', ax=ax3)
ax3.set_title('Top 10 Movies by Average Rating')
ax3.set_xlabel('Average Rating')
ax3.set_ylabel('Movie Title')
ax3.invert_yaxis()

# Top 10 Movies by Rating Count
ax4 = fig.add_subplot(224)
top_10_rating_counts = rating_counts.head(10)
top_10_rating_counts.plot(kind='barh', color='orange', ax=ax4)
ax4.set_title('Top 10 Movies by Rating Count')
ax4.set_xlabel('Number of Ratings')
ax4.set_ylabel('Movie Title')
ax4.invert_yaxis()

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
```