Collaborative Filtering Notebook

What is Collaborative Filtering?

Collaborative Filtering is a technique used in recommendation systems that relies on past interactions and the principle of 'people like you also liked this'. It works by finding patterns in user-item interactions. There are two main types: user-based and item-based collaborative filtering. In this notebook, we are analyzing user ratings of movies to understand preferences and eventually generate recommendations based on collective user behavior.

Notebook Walkthrough

Cell 1 (Code Cell)

```
# Collaborative Filtering
import pandas
user_item_details =
pandas.read_csv("https://coding.co.ke/datasets/movie_ratings.csv")
user_item_details
```

This cell begins by importing the `pandas` library, a powerful tool for data analysis and manipulation in Python.
Then, it loads a dataset of user ratings for movies using `pandas.read_csv()` from an online source.
The dataset likely contains at least the following columns: `user_id`, `movie_id`, and `rating`.
The data is stored in a DataFrame named `user_item_details` which will be used to analyze how different users have rated different movies.

Cell 2 (Code Cell)

```
movie_titles =
pandas.read_csv("https://coding.co.ke/datasets/movie_titles.csv")
movie_titles
```

This cell reads another dataset named `movie_titles.csv` from the same source.
This dataset contains metadata about each movie, including its `movie_id` and human-readable `title`.
This dataset will later be used to replace numeric movie IDs with meaningful names, enhancing readability and analysis.

Cell 3 (Code Cell)

```
# merge the two datasets
data = pandas.merge(user_item_details, movie_titles, on =
'movie_id')
data.head()
```

This cell merges the two previously loaded datasets (`user_item_details` and `movie_titles`) using `movie_id` as the common key.
The resulting DataFrame, `data`, now includes the user ID, movie ID, rating, and the title of the movie for each rating record.
This merged dataset forms the foundation for further analysis, such as grouping and similarity calculations.

Cell 4 (Code Cell)

```
# Find average rating per movie
data.groupby('title')['rating'].mean().sort_values(ascending=False).
head(5)
```

This cell performs a group-by operation on the merged dataset `data`, grouping it by `title` (movie name).
It then computes the **average rating** for each movie using `.mean()` on the `rating` column.
The results are sorted in descending order to list the highest-rated movies at the top.
The top 5 entries from this sorted result are displayed using `.head(5)`.
This gives insight into which movies are most favored by users based on average score.

Cell 5 (Code Cell)

```
# How manny times was it rated
# Find average rating per moview
data.groupby('title')['rating'].count().sort_values(ascending=False).
tail(5)
```

This cell also performs a group-by operation on `data` by movie `title`, but instead of computing the mean, it uses `.count()` to determine how many times each movie was rated.
It helps identify movies with fewer interactions, which may be less reliable for making recommendations.
Sorting is done in descending order, and `.tail(5)` shows the **least frequently rated movies** at the bottom of the sorted list.

Cell 6 (Code Cell)

```
mean_ratings =
pandas.DataFrame(data.groupby('title')['rating'].mean())
mean_ratings['num_of_ratings'] =
pandas.DataFrame(data.groupby('title')['rating'].count())
mean_ratings.head(5)
```

Explanation: This cell performs a step in the data analysis pipeline.

Cell 7 (Code Cell)

```
# Find which movie did each user rate, and what was the rating  -
PIVOT TABLE
pivot   = data.pivot_table(index = 'user_id', columns = 'title', values =
'rating')
pivot.head(5)
```

Explanation: This cell performs a step in the data analysis pipeline.

Cell 8 (Code Cell)

```
# Lets correlate movies using the pivot
selected_movie = pivot['Godfather, The (1972)']

# find similar movies to this one based ratings
similar = pivot.corrwith(selected_movie)
similar_df  = pandas.DataFrame(similar, columns = ['Correlations'])
similar_df.sort_values('Correlations', ascending = False).head(50)
```

Explanation: This cell performs a step in the data analysis pipeline.

Cell 9 (Code Cell)

```
# Show by movies that only had over 200 ratings
similar_df = similar_df.join(mean_ratings['num_of_ratings'])
similar_df.head(5)
```

Explanation: This cell performs a step in the data analysis pipeline.

Cell 10 (Code Cell)

```
# Take only the one with over 200 ratings
similar_df[similar_df['num_of_ratings'] >
200].sort_values('Correlations', ascending = False).head(20)
```

Explanation: This cell performs a step in the data analysis pipeline.