Personalized Product Recommendation System

Team Name: 686157-UZE300G6

Created By:

Rishabh Srivastava

Link to the video:
https://drive.google.com/drive/folders/1sTLA4PhCmSgjT_92HNmmdp3N0roCa_Pc?usp=sharing

1 Data Understanding

1.1 Data Source

The dataset used for the project is MovieLens 20M. widely used and popular dataset in the field of recommender systems and machine learning. It contains movie ratings and user information collected from the MovieLens website. It contains around 20 million ratings. The dataset includes the following information:

Movie Data: It includes details about various movies, such as their titles, genres, and release years. This information helps in understanding the characteristics of the movies being rated.

User Data: User information is also provided, including user IDs. This information allows for tracking individual user preferences and behavior.

Ratings: The core of the dataset consists of user ratings given to movies. Each rating is associated with a user and a movie, and the ratings range typically from 1 to 5.

2 Data Preparation

2.1 Data Cleaning

Despite being a popular dataset, there were many inconsistencies in MovieLens Dataset. So, traditional data cleaning methods were applied to the dataset, like handling missing and duplicate values, checking for consistent formatting, and data integrity checks etc.

2.2 Train, Validation, and Test split

A split of 0.7,0.2, 0.1 was used for training, evaluating and testing the model.

3 Methodology

For the project, hybrid recommendation systems are used to encounter the above problem. A hybrid recommendation system is a type of recommender system that combines multiple recommendation techniques to provide more accurate and diverse recommendations to users.

The hybrid recommendation system can divided into two parts:

1. **Content-Boosted Collaborative Filtering**:

This approach combines collaborative filtering, which relies on user-item interactions, with content-based filtering, which uses item features and user profiles to make recommendations.

By incorporating item attributes or content information, the system can address the "cold start" problem where new items or users have limited interaction data.

2. **Collaborative-Boosted Content Filtering**:

This approach is the inverse of the content-boosted method. It integrates content-based recommendations with collaborative filtering techniques.

It helps overcome the limitations of content-based approaches when there is limited content information available.

I tried 2 types of weighted hybrid systems two variants of each of the content and collaborative filtering.

Non-Deep Learning Based Hybrid system

In this hybrid system, for a given user:

In collaborative filtering:

- ➔ Get all the users who have watched atleast 60% of the movies as the given user
- ➔ Get correlation between these users based on the ratings given by them on the movies
- ➔ Select all the similar user by thresholding the correlation value.
- ➔ Then calculate a weighted score = correlation value*rating, for all the movies. And select the movies with highest weighted scores for recommendation.

In content-based filtering:

- Select the movie recently most liked by the user, based on the rating given by him.
- Now create a correlation matrix between the movie recently liked by the user and all the other movies based on the ratings
- Again calculate the weighted score = correlation value* Average rating. And select the movies with highest weighted scores for recommendation.

Score from both type of equally weighted and final top recommendation are given.

Deep Learning Based Hybrid systems

In collaborative filtering:

- I loaded and pre-processed the data, and performed the split.
- Then the product id were encoded using label encoder.
- A simple DNN was used in this case, to output a feature vector of size of number of movies.
- The model was trained and the epoch with lowest validation loss was chosen. The results on the test set is shown later.
- General metrics like accuracy, precision and recall was used for evaluating.

In content filtering :

- We used BERT llm to encode the movie name into feature vectore
- Then cosine similarity was used to get most identical movie names.
- Finally a feature vector of size num of movies was obtained.

Finally, both feature vectors were combined with equal weights to get the final score.

And the system recommends top 20 movies based on final score.

Following are some results:

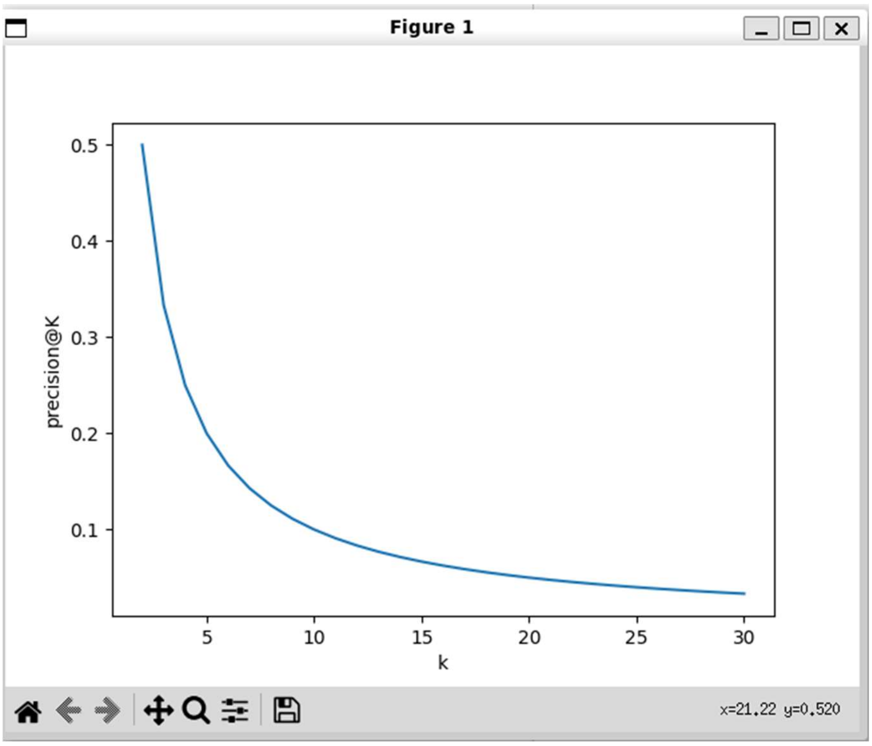


Figure 1
0.5
0.4
0.3
0.2
0.1
precision@K
5
10
15
20
25
30
k
x=21.22 y=0.520

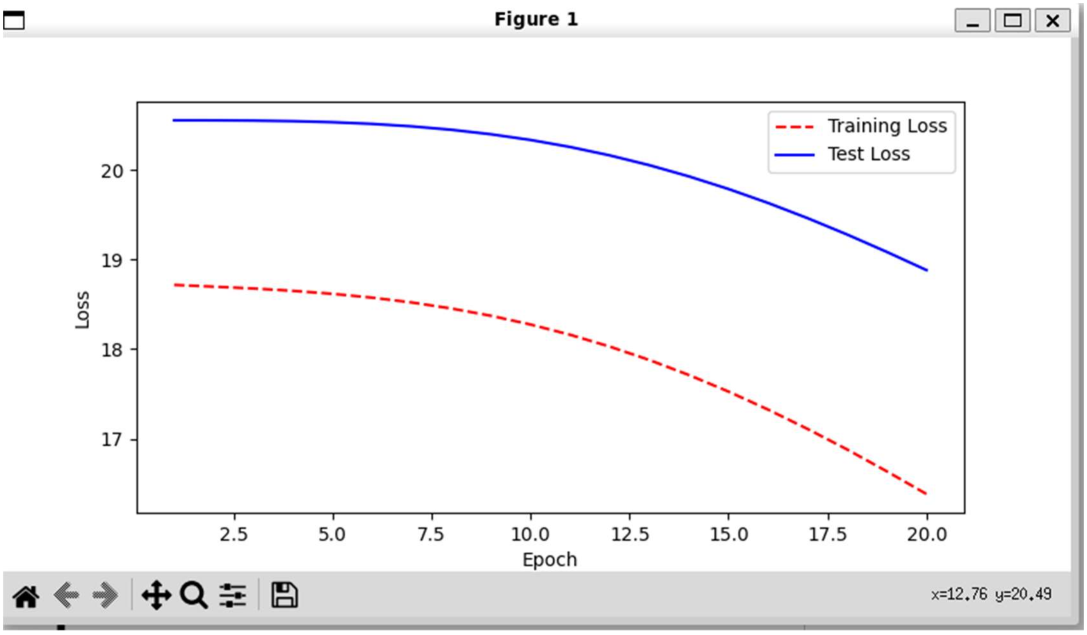
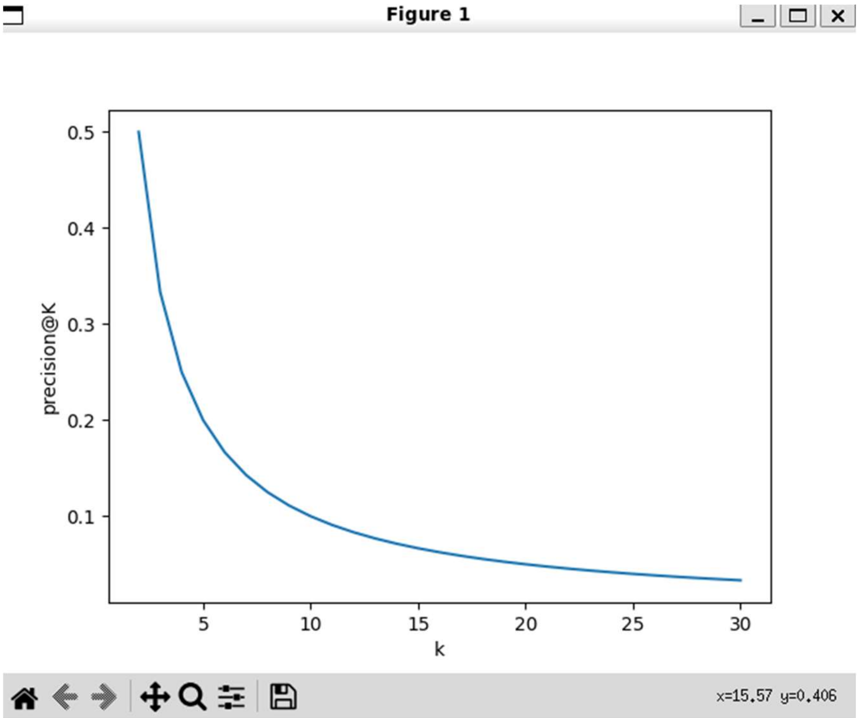
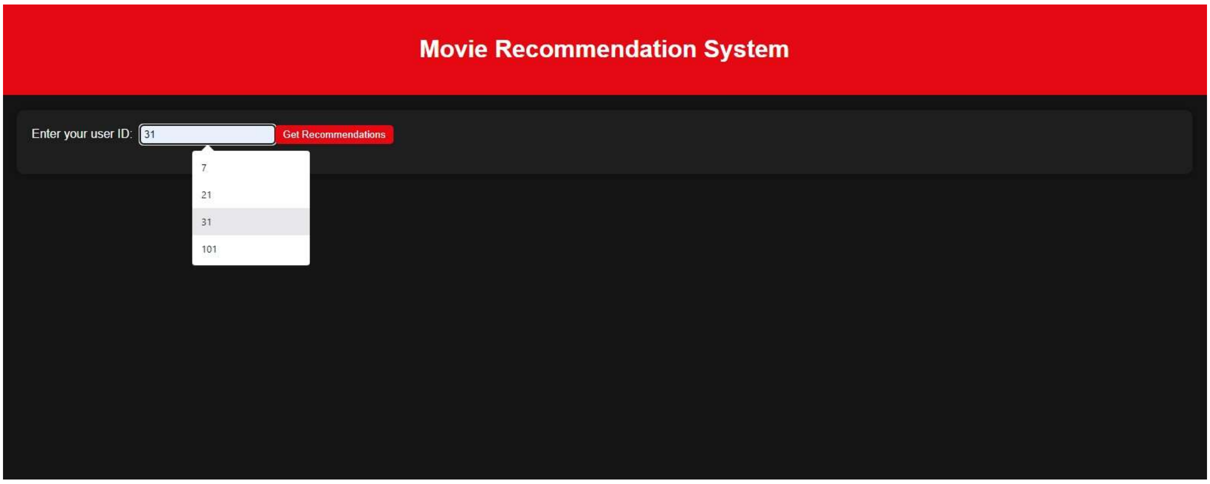


Figure 1
Training Loss
Test Loss
20
19
18
17
Loss
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epoch
x=12.76 y=20.49

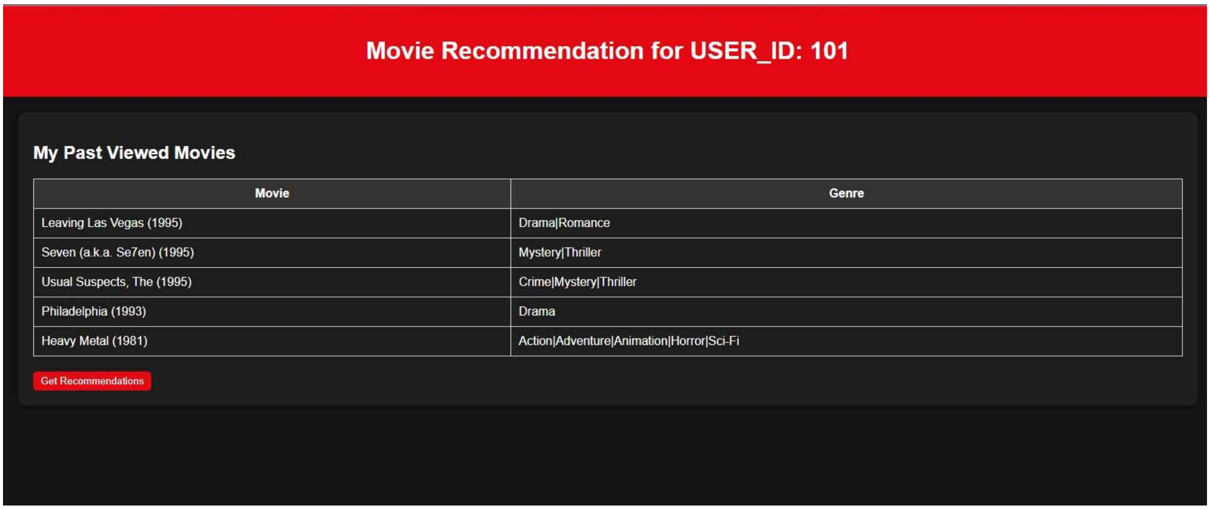


4. Web Application:

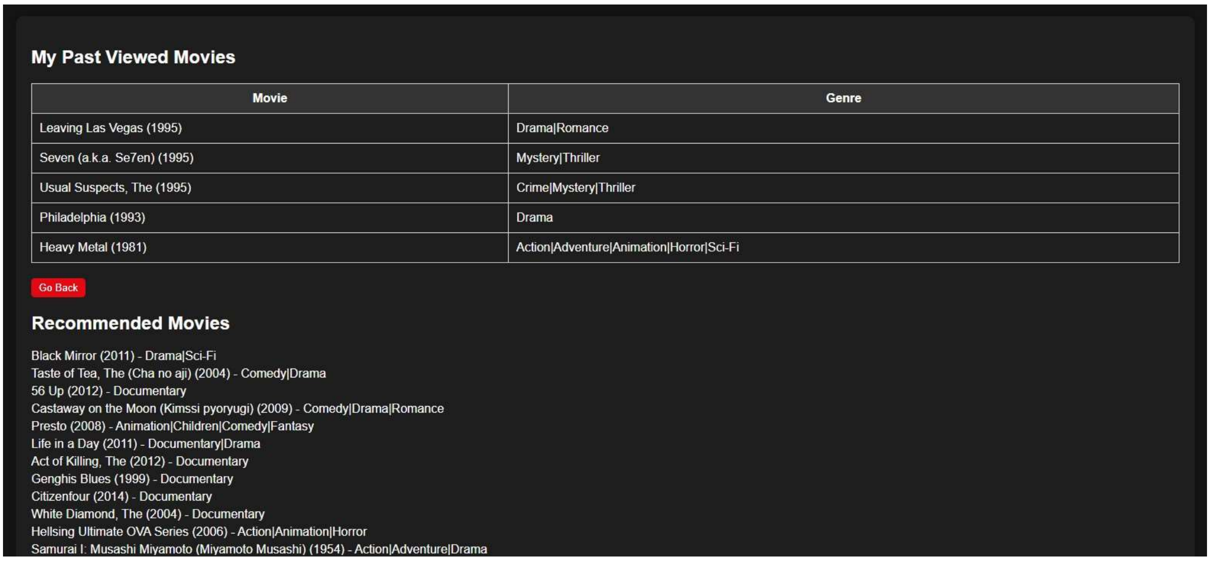
4.1 Landing page



4.2 User Page



4.3 Recommendation Page



5 .Future Work:

1. Dynamic Database

A) When hosted we can use the newer purchases of users to update the recommendations.

2 Selective Attention

3 Negative Sampling