**GDSC AI/ML- Task 1 Made By: Om Parekh**

**RECOMMENDER SYSTEM WITH VARIOUS FILTERING TECHNIQUES**

**My approach for the problem:**

**I created two individual filters namely the collaborative filter and content based filter and thought of making the hybrid filter by combining the two but could not complete it due to the lack of time. I have made the two filters using PyTorch and deep learning techniques and below I have discussed about them in detail.**

**Collaborative Filtering:**

**Introduction**

In this project, we have implemented a **Collaborative Filtering-based Movie Recommendation System** using **Deep Learning (PyTorch)**. The system recommends movies based on user ratings using a **Neural Network Model** that learns user and movie embeddings.

**Implementations in the Project:**

✅ Build a **deep learning model** for **movie recommendation**  
✅ Use **Collaborative Filtering** to predict user preferences  
✅ Train the model on **movie ratings data**  
✅ Return **movie titles** as recommendations

**Dataset Used**

I used  Tmdb metadata which contained 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

I have used two datasets for this project:

**ratings.csv**

* Contains user ratings for movies.
* Columns used: **userId, movieId, rating**

**movies\_metadata.csv**

* Contains movie information (title, genres, etc.).
* Columns used: **id, title**

# ****Implementation Steps****

## Load and Preprocess the Data

First, we **load the dataset** and preprocess it by **encoding** userId and movieId as continuous integers (0,1,2...) for compatibility with PyTorch.

This step ensures that the data is formatted correctly for our deep learning model.

## Define the Collaborative Filtering Model

I defined a **Neural Network Model** that learns embeddings for users and movies. These embeddings capture latent features representing user preferences and movie characteristics.

The model consists of**:**

* **Embedding layers** for users & movies
* **Fully connected layer** to predict ratings

## Training the Model

We train the model for 20 epochs using the **Mean Squared Error (MSE) loss function** and **Adam optimizer**.

The model learns how users interact with movies based on their past ratings.

## Evaluate Model Performance

We test the model using **unseen data** and calculate the **test loss**.

Lower test loss means better recommendations.

## Movie Recommendation Function

We define a function that predicts the **top N movie recommendations** for a user.

This function:

* Predicts ratings for all movies
* Selects the top N movies
* Returns the movie titles instead of IDs

## Load Movie Titles and Get Recommendations

## This step ensures that we return actual movie names, not just IDs.

# Results

For a given user, the model recommends movies with **their titles** instead of just similarity scores.

**Conclusion**

In this project, we successfully built a deep learning-based movie recommendation system using collaborative filtering.

**Key Takeaways:**

✅ Used **Collaborative Filtering** to predict user ratings.  
✅ Implemented a **Neural Network** with **User & Movie Embeddings**.  
✅ Trained the model using **PyTorch**.  
✅ Returned **actual movie titles** instead of just numerical IDs.

**This system can be improved further by:**

* Incorporating **Deep Learning-based feature extraction**
* Using **Hybrid Filtering (Collaborative + Content-Based)**
* Training on a **larger dataset**

**Content-Based Filtering**:

**Introduction**

The objective of this project is to develop a **content-based movie recommendation system** using a **deep learning-based feature embedding approach**. This system utilizes **movie metadata** (such as genres, cast, crew, and keywords) to generate meaningful feature embeddings and recommend similar movies.

### Implementations in the Project****:****

✅ Extract meaningful feature representations using **Deep Learning**  
✅ Train a **Neural Network Model** to learn **movie embeddings**  
✅ Compute **movie similarity** using **cosine similarity and k-nearest neighbors (KNN)**  
✅ Provide **personalized recommendations** based on movie content

**Dataset Used**

I used  Tmdb metadata which contained 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

I have used two datasets for this project:

**ratings.csv**

* Contains user ratings for movies.
* Columns used: **userId, movieId, rating**

**movies\_metadata.csv**

* Contains movie information (title, genres, etc.).
* Columns used: **id, title**

**Keywords.csv**

* **Contains about the type of movie and the keywords contained in it.**
* **Columns used: id,keywords**

## Implementation Steps

### Data Preprocessing

The first step is to filter the dataset and convert categorical text-based features into **numerical encodings** using **Label Encoding**.

This step ensures that movie metadata is structured properly for deep learning.

### Defining the Deep Learning Model

We create a **deep learning-based embedding model** that learns feature representations for movies.

**T**his model learns a unique embedding for each movie based on its features**.**

### ****Training the Model****

The model is trained using the **Mean Squared Error (MSE) Loss** and the **Adam optimizer**.

**The model learns how movies relate to each other based on their metadata.**

### ****Generating Movie Embeddings****

After training, we extract the **movie embeddings** from the model.

**These embeddings are used for similarity-based recommendations.**

### ****Finding Similar Movies Using k-Nearest Neighbors (KNN)****

We use **cosine similarity** and **KNN** to find the top 10 similar movies.

**This function finds movies that are similar to the input movie.**

## Results & Conclusion

### ****Key Findings:****

🎯 **A deep learning model** was successfully used to **generate movie embeddings**.  
🎯 **Cosine similarity** with **KNN** was effective in **finding similar movies**.  
🎯 The model provides **highly relevant movie recommendations** based on metadata.

### ****Future Improvements:****

🚀 **Increase embedding dimensions** for better feature learning.  
🚀 **Train on a larger dataset** for more accurate recommendations.  
🚀 **Incorporate user ratings** to make it a hybrid recommendation system.

**Refrences**:

Dataset Link : <https://drive.google.com/drive/folders/1t7PO3jjYMXv-cJt6XS70T9ObvOS8sYMi?usp=sharing>

PyTorch Documentation [- https://pytorch.org/docs/stable/index.html](-%20https:/pytorch.org/docs/stable/index.html)