



	<b>Recommendation System</b>

# Recommendation System

# **Meet The Team**

**-SALMA SAEED (Team Leader)**

**-GANNA TAHER**




# Introduction

Recommendation systems help users discover movies they'll likely enjoy by analyzing both user preferences (ratings) and movie features (genres, actors). Using collaborative filtering, the system recommends movies based on the tastes of similar users. With content-based filtering, it suggests movies with similar features to those the user has liked before. Combining both methods, the system provides personalized movie suggestions that match the user's interests and viewing habits.



# Project Background

With the growth of streaming platforms, users struggle to find movies. Recommendation systems help by suggesting personalized content based on user ratings and movie features. They use Collaborative Filtering (similar users' preferences) and Content-Based Filtering (similar movies). These systems enhance user experience with relevant suggestions



# Project Planning

**A**

Data Collecting &  
Preprocessing

**(Salma Saeed)**

**B**

ML & Modeling

**(Salma Saeed)**

**C**

Advanced Techniques  
& MLOPs

**(Ganna Taher)**

# Project Goals

**1.**

**Enhance User  
Experience.**

**2.**

**Increase Content  
Discovery.**

# Process

**1.**

- **Collect Data.**
- **Clean Data.**
- **Data Preprocessing.**
- **Transform.**

**2.**

- **ML & Modeling**
- **Select Algorithms.**
- **Train Models.**
- **Evaluate.**

**3.**

**Advanced Techniques  
Generative Models.**

**4.**

**MLOPs  
MLflow.  
Prompt Engineering.**

# Data Used



**Data**

**User Ratings.**

**Movie Details.**



# Ratings File

- Structure: Each row corresponds to a user rating for a specific movie.

Columns:

- UserID: Unique identifier for the user.
- MovieID: Unique identifier for the movie.
- Rating: Numeric rating given by the user.

# Movies File

- Structure: Each row contains information about a specific movie.

Columns

- MovieID: Unique identifier for the movie.
- Title: Title of the movie.
- Genres: Comma-separated list of genres (Action, Comedy, Drama).
- Release Year: The year the movie was released

# Data Collection

- Start with collecting Movielens dataset from kaggle.

## Preprocessing on Data

- Big data we use a sample of data in anylisis part.
- Use Data Types with Less Memory like float32.
- One Hot Encoder genre column (from (x|y) to x y ).
- Clip rating values to be between (0.5 , 5 ) as (min,max)..

**SALMA**

# Analysis on Data

- **Basic Analysis :**

Number of ratings: 32000204

Number of unique movieId's: 84432

Number of unique users: 200948

Average ratings per user: 159.25

Average ratings per movie: 379.01

Number of movies with no genres listed: 7080

**SALMA**

- **Basic Descriptive Analysis**

Distribution of ratings (most 3,4)

- **Basic Statistical Analysis**

Find out which movies have the most ratings.(count).

Average Rating per Movie.

- **Genre Analysis**

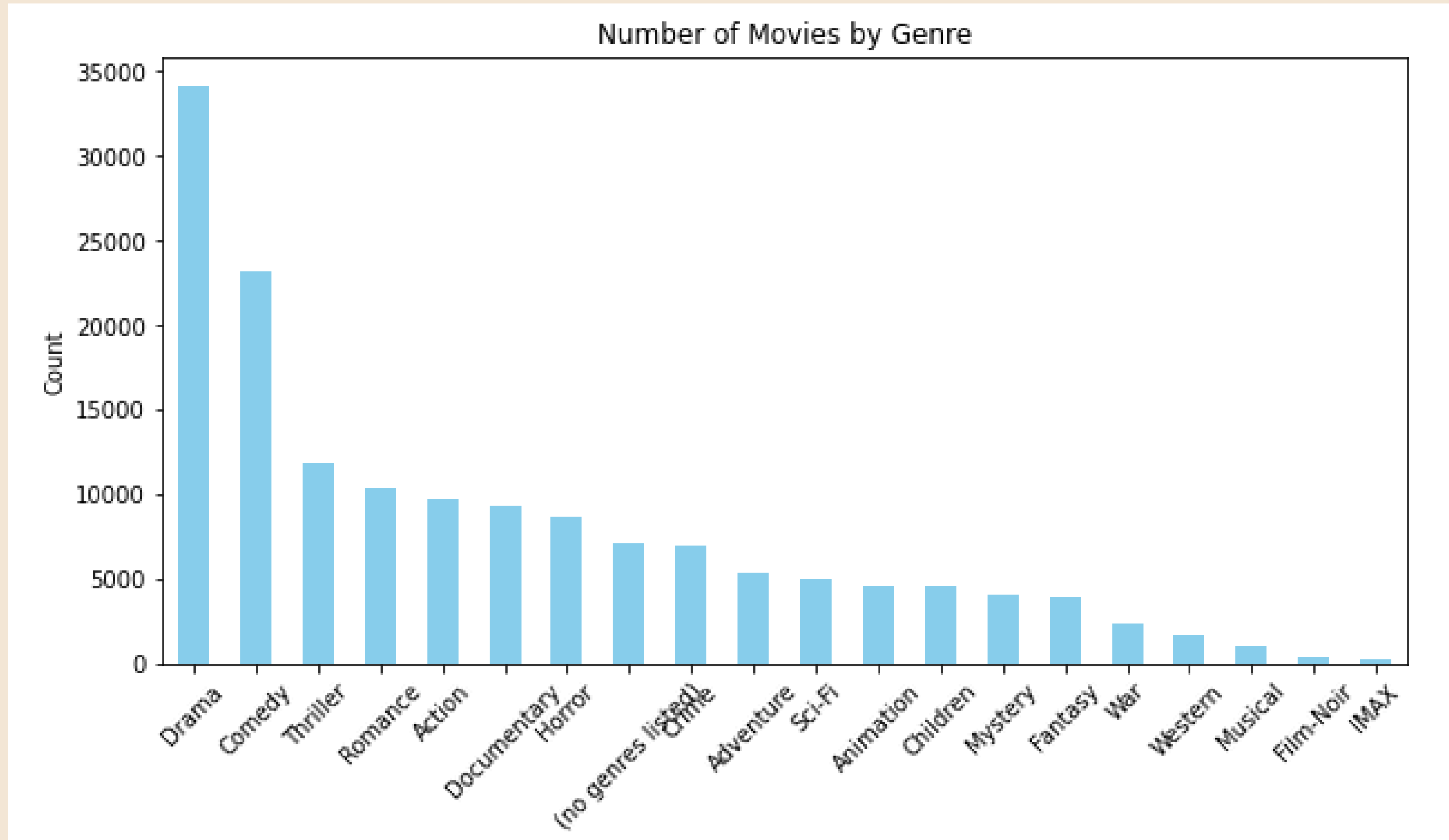
Count the number of movies for each genre.

Number of movies with no genres listed: 7080

**SALMA**

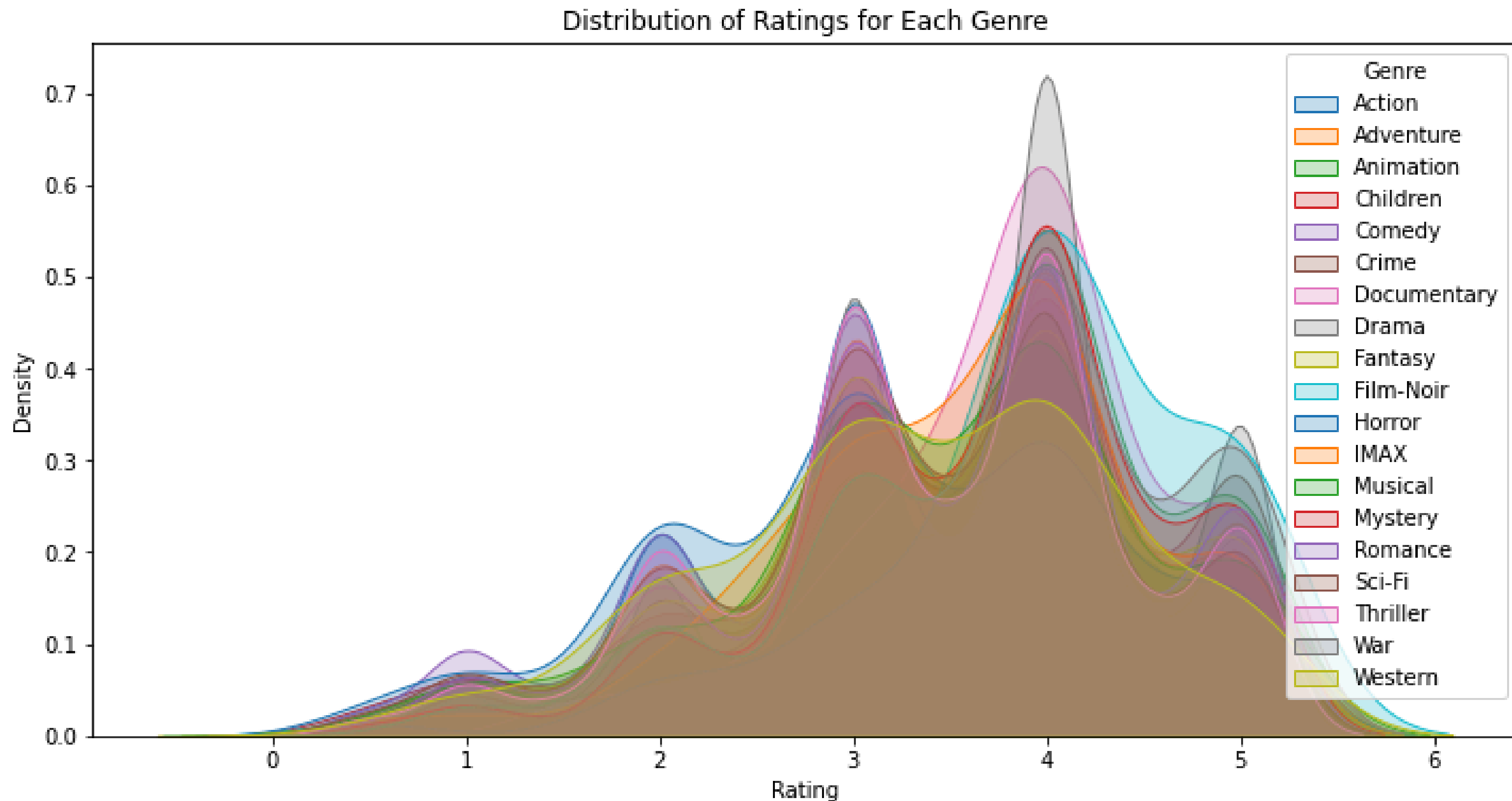
# Visualization

- Plot genre Distribution Using Bar Chart.



**SALMA**

- Plot the distribution of ratings for each Genre



# User Analysis

- Find out which most\_active\_users.(count-> user\_id 28).

There is a dataframe contained the number of ratings each user has given in a dataset of movie ratings.

- Calculate the average rating given by each user (for all movie he watched).

**SALMA**

# Movie Analysis

- Find Lowest and Highest rated movies & number of people who rated movies rated movie lowest.
- Find Highest rated movies & number of people who rated movies rated movie Highest.

\*Calculate Movie Statistics: (count , mean\_rating)

**SALMA**



# Movie Recommendation Based on User's Watched Movies

Objective: Recommend movies tailored to a user based on their viewing history.

Approach: Use collaborative filtering techniques to find similarities among user ratings.

Process: Input the user ID and analyze the movies they have watched to identify similar films.

Output: A list of recommended movie titles that align with the user's preferences.

Advantage: Enhances user satisfaction by providing personalized movie suggestions based on their unique tastes.

**SALMA**

- **Movie Recommendation Based on Another Movie**

Objective: Recommend movies similar to a specific movie chosen by the user.

Approach: Utilize the K-Nearest Neighbors (KNN) algorithm to find movies with similar ratings.

Process: Input the selected movie ID and retrieve a list of similar movies from the user-item data.

Output: A curated list of recommended movie titles based on similarities to the chosen movie.

Advantage: Provides personalized movie suggestions, enhancing user experience and engagement.

- **Movie Recommendation Based on Genre**

Objective: Suggest movies to users based on a specific genre they are interested in.

Approach: Utilize the SVD algorithm from the Surprise library to predict ratings for movies within the chosen genre.

Process: Filter movies by the specified genre and predict their ratings for a given user.

Output: A list of recommended movie titles sorted by predicted ratings within the chosen genre.

Advantage: Provides tailored recommendations, enhancing user engagement by focusing on genres they enjoy.

# GANs

- **Normalizing Data**

We normalize data by **MINMAXSCALER** to make it in range of ( 0 , 1 ) to suit rating which between ( 0.5 , 5 ) and suit **Sigmoid** function.

- **Convert Data to Tensorflow**

- **Shuffle Data and Prefetch for Performance**

- **Create Iterator**

To get batch batch not get all at the same time .

**GANNA**

- **Define Generator and Discriminator**
- **Train GANs**

We use **optimizer Adam** to adjust weights during train ,use **Binary Cross Entropy Loss** to evaluate the performance ,get into loop to repeat training : train Discriminator, generate fake data by generator, measure loss , train Generator and print results after 1000 epochs.

**GANNA**

- Output of Training GANs

```
➤ Epoch 0: D Loss: 33.22731018066406, G Loss: 0.6961937546730042
Epoch 1000: D Loss: 8.103785975599465e-10, G Loss: 20.243759155273438
Epoch 2000: D Loss: 9.434722447743127e-11, G Loss: 22.39232063293457
Epoch 3000: D Loss: 3.994088393355355e-11, G Loss: 23.251279830932617
Epoch 4000: D Loss: 2.3861308762196387e-11, G Loss: 23.76618766784668
```

- Print Fake Data

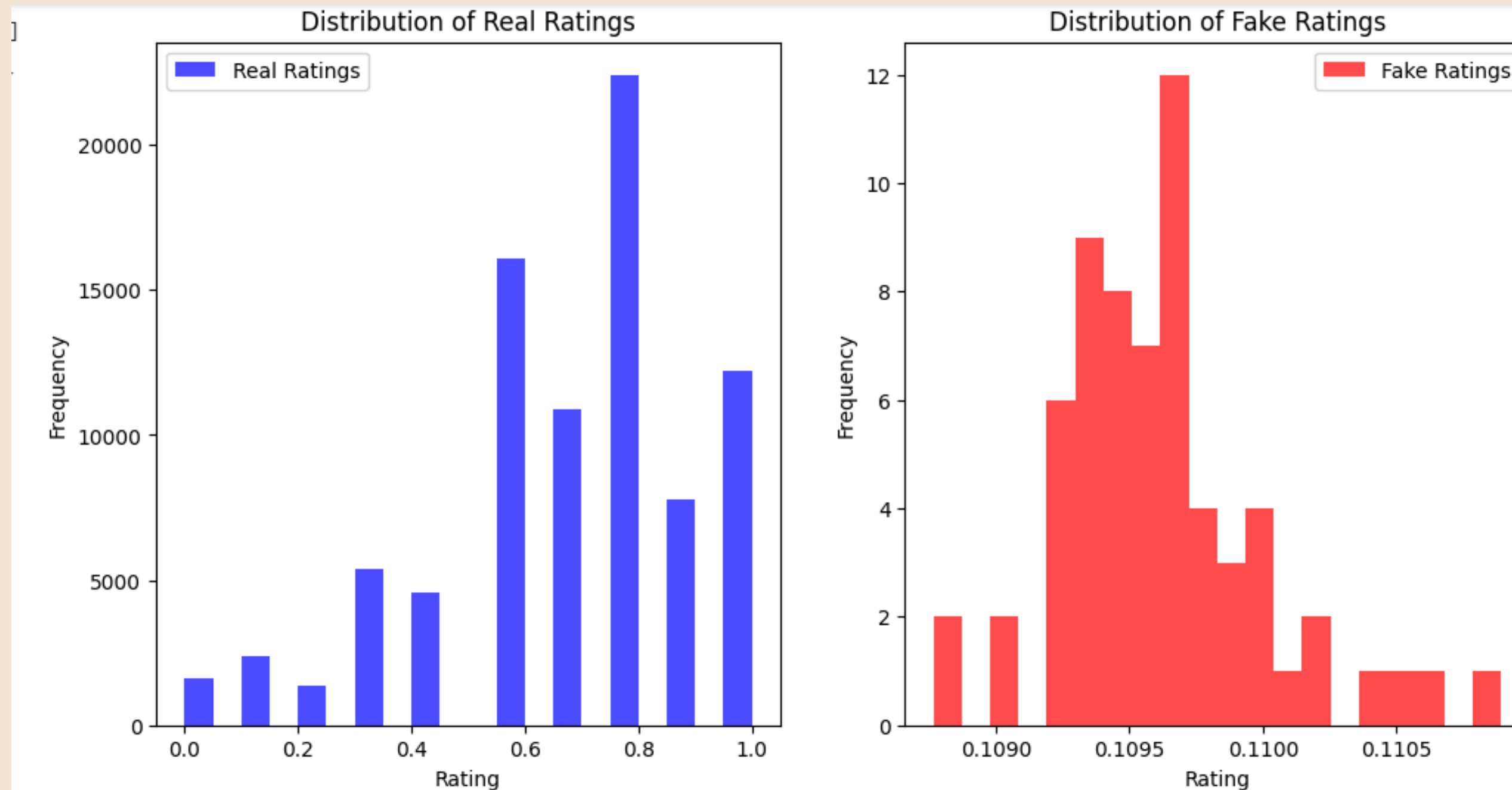
```
[13] Fake Data Generated by the Generator:
↳ [[0.35389587 0.9999976 0.10929426]
    [0.35328117 0.99999976 0.10923776]
    [0.35338527 0.99999995 0.10949679]
    [0.3536296 0.99999845 0.10904401]
    [0.3540845 0.9999987 0.11059201]
    [0.3534806 0.9999989 0.10945074]
    [0.35287195 1. 0.10876599]
    [0.3541294 0.9999919 0.11014752]
    [0.35413784 0.9999994 0.11036736]
    [0.35354424 0.99999857 0.10964604]
    [0.35373318 0.9999932 0.10925864]]
```

GANNA

- **Evaluate Real and Fake Data**

We evaluate real and fake data by measure mean and standard deviation for each of them

- **Visualization the Real and Fake Data**



**GANNA**

- **Implement stopping criteria**

It used for if the model after specific number of layers it makes the model break then print the epochs and loss when it stopped

- **Save and Download Generator and Discriminator**

To implement them after trained model on machine model

**GANNA**



# MLOPs

## MLFlow

- Set Tracking on Local Server
- Set Experiment
- Start Run the MLflow

**GANNA**

- **Implement SVD model**
- **Record a parameter using log\_param**
- **Train SVD model**
- **Test model and implement it by log\_metric**
- **End MLflow run**

# Prompt Engineering

It based on chatbot gives user movies that he wanted when he input rating

- Merge Two Datasets
- Filter the Data set
- Giving Chatbot rat to get movies
- Rat should between 1 , 5
- Quit of the Chatbot when he want

**GANNA**

# Result

## Increased Engagement.



Users spend more  
time on the platform.

## Enhanced Discovery.



Users find a wider  
variety of movies

## Boosted Revenue.

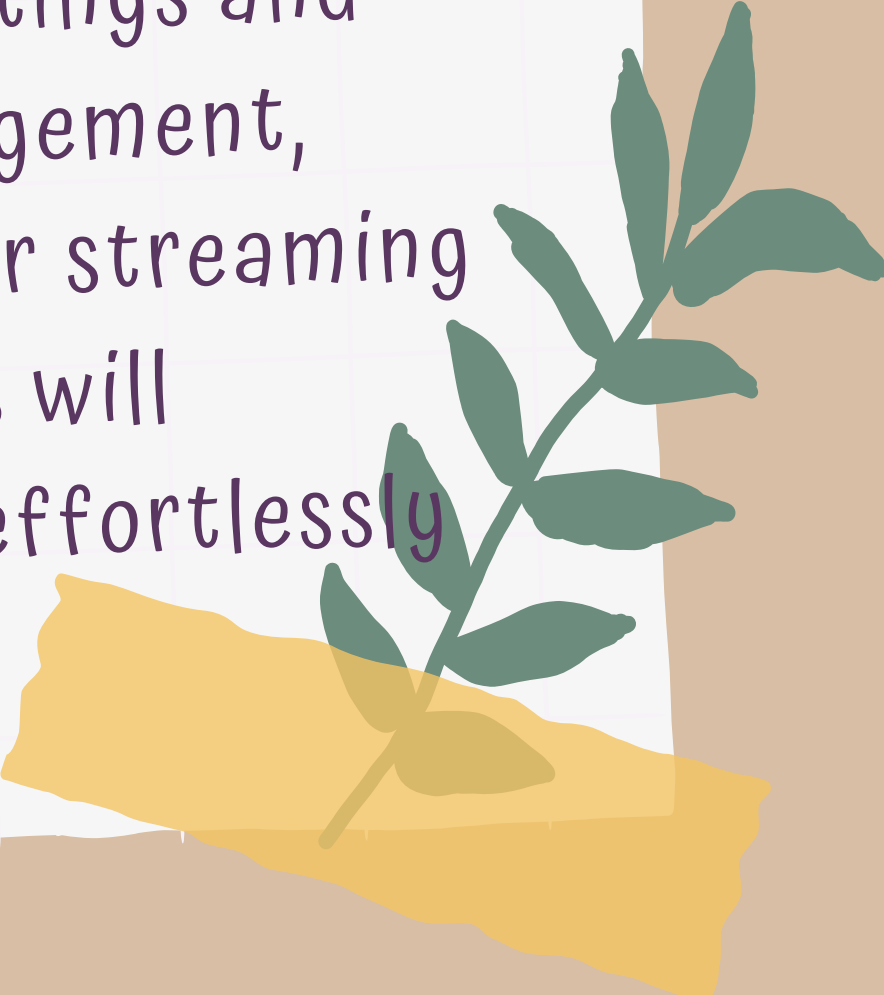


Higher  
subscriptions and  
ad income



# Conclusion

A recommendation system enhances user experience by offering personalized suggestions based on user ratings and movie metadata. This leads to increased user engagement, improved content discovery, and higher revenue for streaming platforms. Continuous advancements in algorithms will further optimize recommendations, helping users effortlessly find films that align with their preferences.





**Any  
Question ?**



**Thank  
you**