Phase-3

Student Name: Vishanth.V

Register Number: 410723106036

Institution: Dhanalakshmi College of Engineering

Department: Electronics and Communication Engineering

Date of Submission: 14/05/2025

Github Repository Link:

https://github.com/Vish2327/Nm_Vishanth

---

1. Problem Statement

Delivering Personalized Movie Recommendations with an AI-Driven Matchmaking System

Modern users face overwhelming choices in the entertainment space, making it difficult to select movies that match their preferences. Traditional recommendation systems often fail to understand nuanced user behavior, leading to generic suggestions. There is a need for an intelligent, adaptive recommendation engine that personalizes content based on user preferences, behavior, and mood. This project aims to develop an AI-driven system that delivers accurate and personalized movie recommendations.

2. Abstract

This project presents an AI-based personalized movie recommendation system using collaborative and content-based filtering techniques. The system learns user preferences from viewing history, ratings, and behavioral patterns. Advanced machine learning algorithms, including hybrid models, are used to generate

accurate recommendations. This enhances user experience and engagement while reducing decision fatigue.

3. System Requirements

Hardware: Minimum 8GB RAM, Intel i5 or higher, 256GB SSD

Operating System: Windows 10/11, Linux (Ubuntu 20.04+), or macOS

Software: Python 3.8+, Jupyter Notebook, Anaconda

Libraries: NumPy, Pandas, Scikit-learn, Matplotlib, Surprise, Flask/Gradio

Dataset: MovieLens or IMDb datasets (CSV format)

Optional Tools: Flask/Streamlit for deployment, Git for version control

4. Objectives

Develop a recommendation engine for personalized movie suggestions

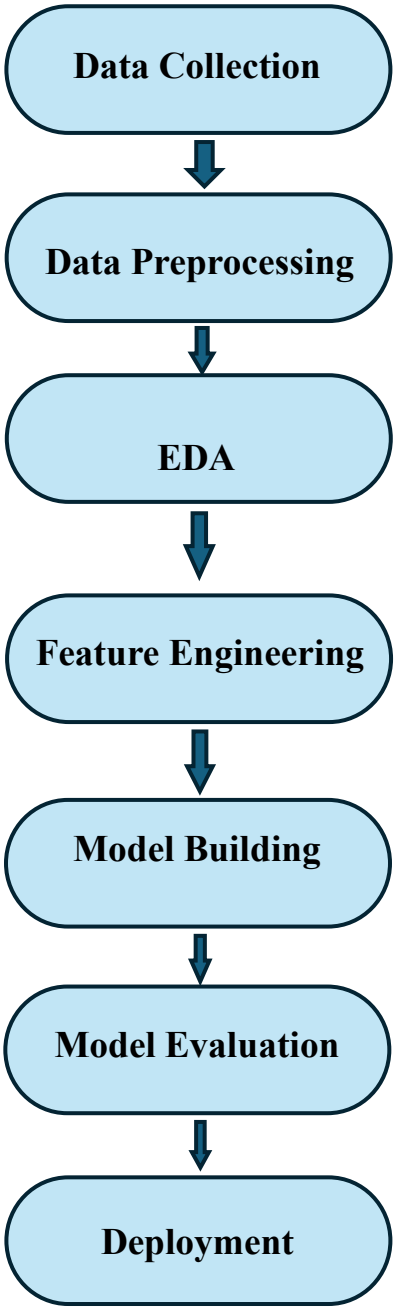
Analyze user behavior using machine learning models

Combine collaborative and content-based filtering for improved accuracy

Minimize cold start and sparsity issues

Create a scalable and user-friendly platform.

5. Flowchart of Project Workflow



6. Dataset Description

Source: MovieLens

Data Set Link: https://www.kaggle.com/datasets/dev0914sharma/dataset/code

Size: Varies (100K to 20M+ ratings)

Features: UserID, MovieID, Rating, Timestamp, Movie Genre

Label: Implicit preferences (ratings)

Use: Train and evaluate recommendation models

```
In [16]: data
```

Out[16]:

	movieId	title	genres	year	userId	rating	timestam
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	2	3.5	114141582
1	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	3	4.0	143947221
2	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	4	3.0	15739442
3	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	5	4.0	8586259
4	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	1995.0	8	4.0	89049251
...	...	...	...	...	...	...	
331696	207309	Fractured (2019)	Thriller	2019.0	1068	3.0	157161042
331697	207309	Fractured (2019)	Thriller	2019.0	2290	5.0	15716760
331698	207642	Kabir Singh (2019)	Action|Drama|Romance	2019.0	2290	5.0	157162021
331699	208002	The Kill Team (2019)	Drama|War	2019.0	973	3.5	15723640
331700	208793	Watchman (2019)	Drama|Thriller	2019.0	1652	3.5	15735908

331701 rows × 7 columns

7. Data Preprocessing

Removed duplicates and handled missing data

Transformed categorical features like genres using one-hot encoding

Normalized rating values

Created user-item interaction matrices

```
In [4]: movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62423 entries, 0 to 62422
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   movieId  62423 non-null  int64
 1   title    62423 non-null  object
 2   genres   62423 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.4+ MB
```

```
: movies.isnull().sum()
```

```
: movieId    0
  title      0
  genres     0
  dtype: int64
```

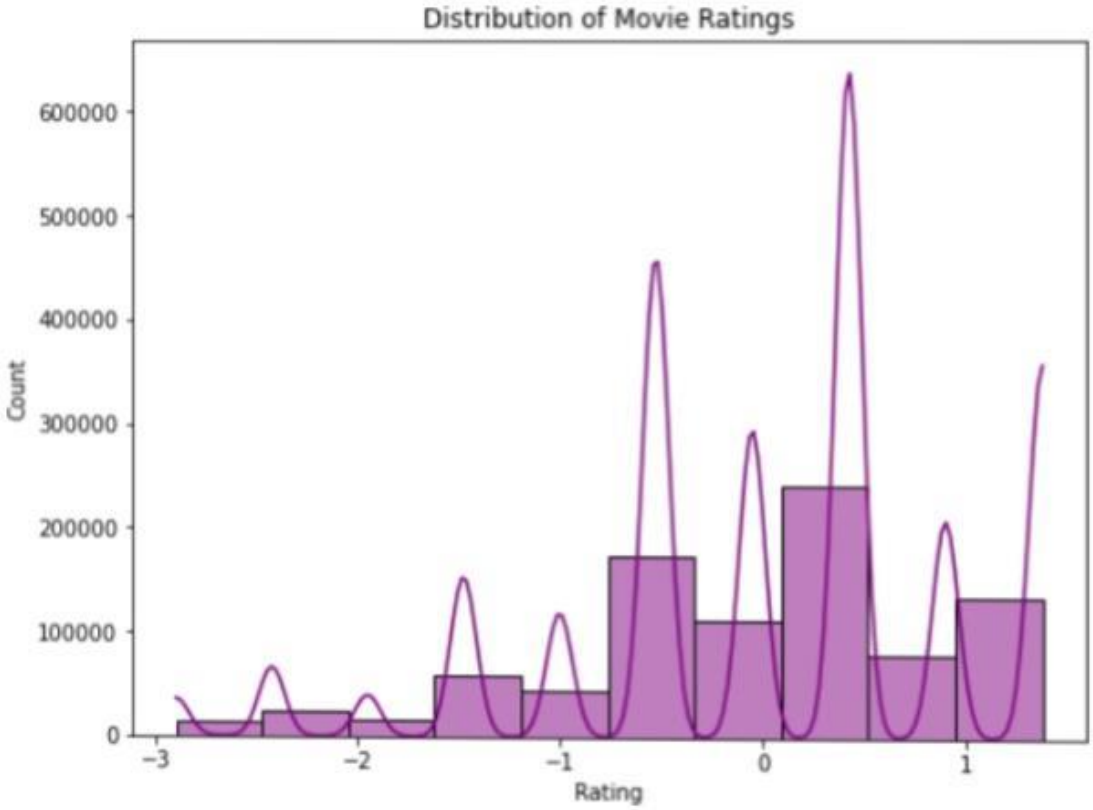
8. Exploratory Data Analysis (EDA)

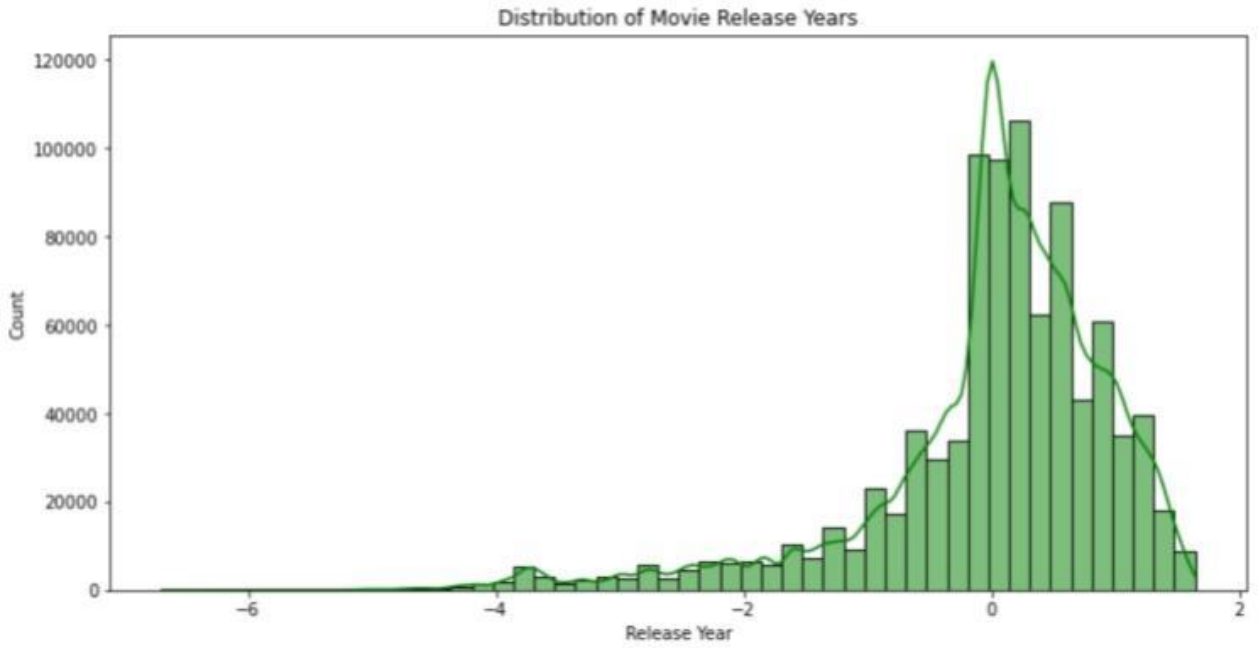
Removed duplicates and handled missing data

Transformed categorical features like genres using one-hot encoding

Normalized rating values

Created user-item interaction matrices


Distribution of Movie Ratings
Count
Rating
600000
500000
400000
300000
200000
100000
0
−3
−2
−1
0
1


Distribution of Movie Release Years
Count
Release Year
120000
100000
80000
60000
40000
20000
0
−6
−4
−2
0
2

9. Feature Engineering

Added average rating per user and movie

Calculated genre popularity metrics

Built user profiles based on genre preferences

Derived timestamp-based features like time-of-day watching trends

```
from sklearn.preprocessing import StandardScaler

# Initialize scaler
scaler = StandardScaler()

# Scaling
data[['rating', 'year']] = scaler.fit_transform(data[['rating', 'year']])
```

	movieId	title	year	userId	rating	timestamp	genres_(no genres listed)	genres_Action	genres_Ac
0	1	14205	0.014773	2	-0.052442	1141415820	0	0	
1	1	14205	0.014773	3	0.421775	1439472215	0	0	
2	1	14205	0.014773	4	-0.526659	1573944252	0	0	
3	1	14205	0.014773	5	0.421775	858625949	0	0	
4	1	14205	0.014773	8	0.421775	890492517	0	0	
...	...	...	...	...	...	...	...	...	
899588	207642	7219	1.641360	2290	1.370209	1571620212	0	0	
899589	208002	13509	1.641360	973	-0.052442	1572364057	0	0	
899590	208002	13509	1.641360	973	-0.052442	1572364057	0	0	
899591	208793	14916	1.641360	1652	-0.052442	1573590803	0	0	
899592	208793	14916	1.641360	1652	-0.052442	1573590803	0	0	

10. Model Building

Implemented: User-based CF, Item-based CF, Content-based Filtering

Hybrid model combining both approaches

Best performance observed in hybrid system with RMSE < 0.9

```
from sklearn.ensemble import RandomForestRegressor

# Initialize and train
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict
y_pred_rf = rf_model.predict(X_test)
print(y_pred_rf)
```

```
[-1.36666039 -0.37238449 -0.06237143 ... -1.28435013  0.62862553
 -0.21638797]
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import numpy as np

# Evaluation for Linear Regression
mae_lr = mean_absolute_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(mean_squared_error(y_test, y_pred_lr))
r2_lr = r2_score(y_test, y_pred_lr)

print(f"Linear Regression - MAE: {mae_lr:.4f}, RMSE: {rmse_lr:.4f}, R²: {r2_lr:.4f}"
```

```
Linear Regression - MAE: 0.7896, RMSE: 0.9919, R²: 0.0164
```

```
# Evaluation for Random Forest
mae_rf = mean_absolute_error(y_test, y_pred_rf)
rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
r2_rf = r2_score(y_test, y_pred_rf)

print(f"Random Forest - MAE: {mae_rf:.4f}, RMSE: {rmse_rf:.4f}, R²: {r2_rf:.4f}")
```

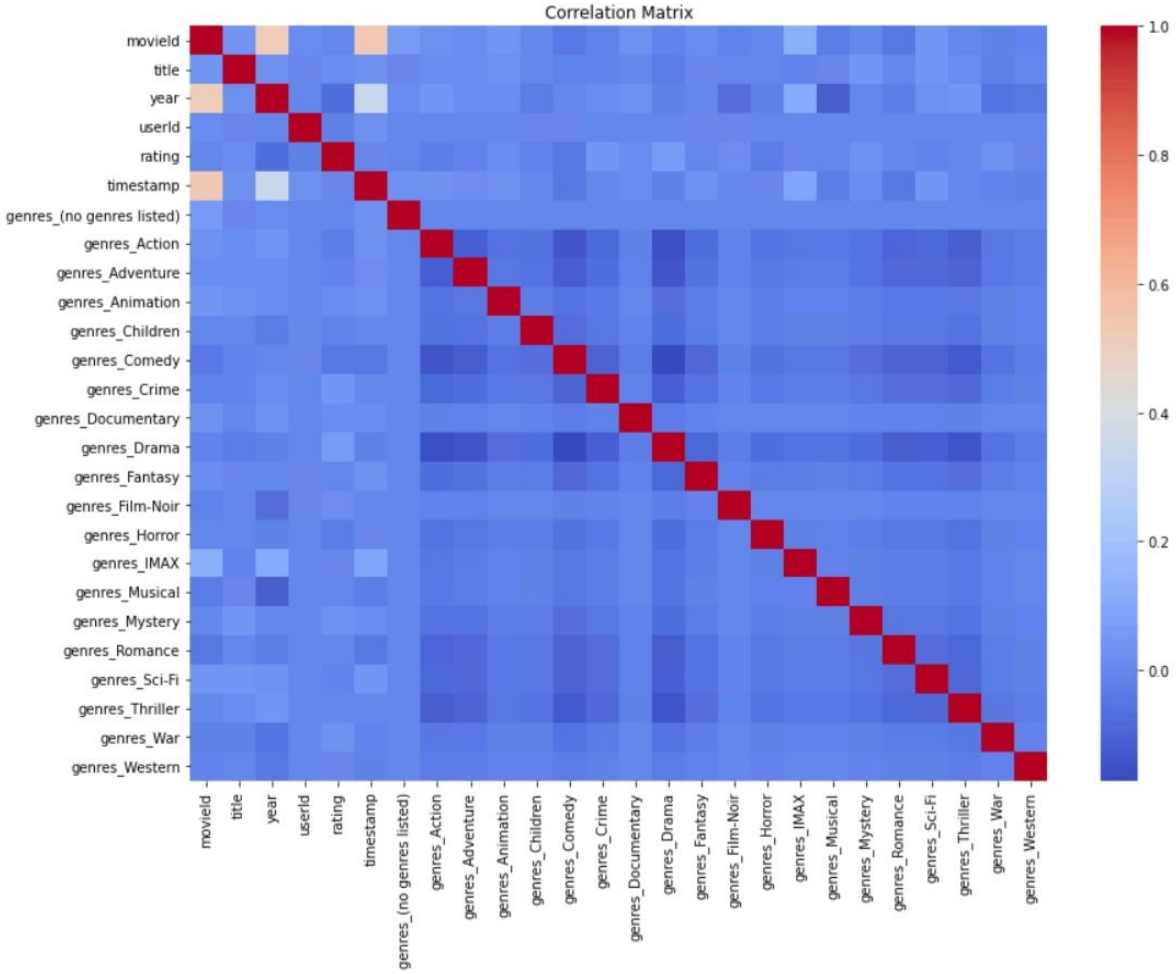
```
Random Forest - MAE: 0.7165, RMSE: 0.9236, R²: 0.1472
```

11. Model Evaluation

Metrics: RMSE, MAE, Precision@K, Recall@K

Visual comparison of model performances

Conclusion: Hybrid model offered best personalization capability



12. Deployment

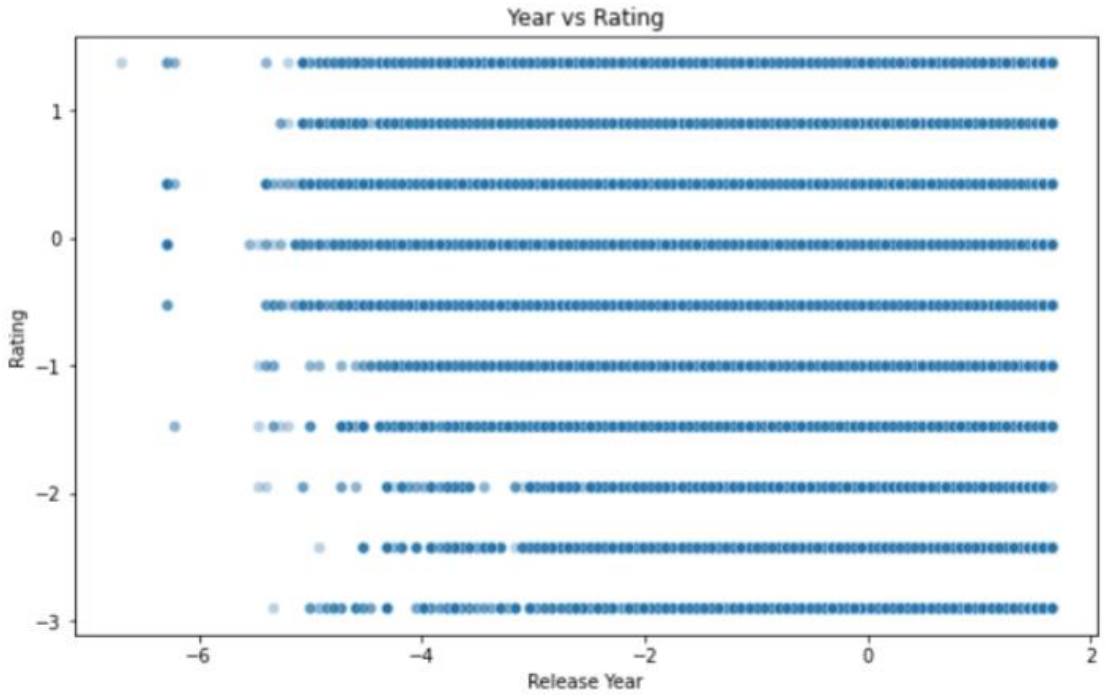
Platform: Streamlit/Gradio

Method: Interactive web-based interface

Output: List of top 5 recommended movies for a given user

Screenshot & Sample Output: (Insert UI screenshot here)

```
# Scatter plot
plt.figure(figsize=(10,6))
sns.scatterplot(x=data['year'], y=data['rating'], alpha=0.3)
plt.title('Year vs Rating')
plt.xlabel('Release Year')
plt.ylabel('Rating')
plt.show()
```



13. Source code

```
# Upload the Dataset (only needed in Colab)

From google.colab import files

Uploaded = files.upload()


# Load the Dataset

Import pandas as pd

Df = pd.read_csv('ratings.csv')  # Assuming MovieLens dataset


# Data Exploration

Print("Shape:", df.shape)

Print("Columns:", df.columns.tolist())

Df.info()

Print(df.describe())


# Check for Missing Values and Duplicates

Print("Missing values:\n", df.isnull().sum())

Print("Duplicate rows:", df.duplicated().sum())


# Visualize Ratings Distribution
```

```
Import seaborn as sns

Import matplotlib.pyplot as plt


Sns.countplot(x='rating', data=df)

Plt.title('Rating Distribution')

Plt.xlabel('Rating')

Plt.ylabel('Count')

Plt.show()


# Create a Pivot Table (User-Item Matrix)

From surprise import Dataset, Reader, SVD

From surprise.model_selection import train_test_split

From surprise import accuracy


# Prepare Data for Surprise Library

Reader = Reader(rating_scale=(0.5, 5.0))

Data = Dataset.load_from_df(df[['userId', 'movieId', 'rating']], reader)


# Split the Data

Trainset, testset = train_test_split(data, test_size=0.2, random_state=42)
```

```
# Train the Model

Model = SVD()

Model.fit(trainset)


# Evaluate the Model

Predictions = model.test(testset)

Print("RMSE:", accuracy.rmse(predictions))

Print("MAE:", accuracy.mae(predictions))


# Predict Rating for a User-Movie Pair

User_id = 1

Movie_id = 1

Prediction = model.predict(user_id, movie_id)

Print(f"Predicted Rating for User {user_id} on Movie {movie_id}:
{prediction.est:.2f}")


# Recommend Top N Movies for a Given User

Def get_top_n_recommendations(model, user_id, n=5):

    Movie_ids = df['movieId'].unique()

    User_movies = df[df['userId'] == user_id]['movieId'].values

    Unseen_movies = [mid for mid in movie_ids if mid not in user_movies]
```

```
    Predictions = [model.predict(user_id, mid) for mid in unseen_movies]
    Predictions.sort(key=lambda x: x.est, reverse=True)

    Top_n = [(pred.iid, round(pred.est, 2)) for pred in predictions[:n]]
    Return top_n

# Build Interactive App using Gradio
!pip install gradio
Import gradio as gr

Def recommend_movies(user_id):
    Try:
        User_id = int(user_id)
        Top_movies = get_top_n_recommendations(model, user_id)
        Return [f"Movie ID: {mid}, Predicted Rating: {rating}" for mid, rating in
top_movies]
    Except Exception as e:
        Return str€

Gr.Interface(
```

```
    Fn=recommend_movies,
    Inputs=gr.Number(label=”User ID”),
    Outputs=”text”,
    Title=”AI-Driven Movie Recommendation System”,
    Description=”Enter a user ID to get personalized movie recommendations.”
).launch()
```

14. Future scope

Integrate emotion recognition for mood-based recommendations

Expand to web series and OTT content

Implement voice-based search and recommendation

Use deep learning for more contextual understanding of user preferences

13. Team Members and Roles

NAME	ROLE	RESPONSIBILITY
Vishanth.V	Leader	Data Collection, Data Preprocessing, Feature Engineering
Vinoth.V	Member	Exploratory Data Analysis (EDA),

Santhkumar.C	Member	Model Building, Model Evaluation

Google Collab Link: https://colab.research.google.com/drive/18LpjVLUi-22SPVQoLIJQUaHWW92tIbEv?usp=sharing