A Movie Recommendation System Using Collaborative Filtering

A Project Report

On

ARTIFICIAL INTELLIGENCE

Submitted by

PRIYANKA. S	**REG NO: 212303141**
SRIPRIYA. S	**REG NO: 212303152**
PAVITHRA. R	**REG NO: 212303140**
MAHALAKSHMI M	**REG NO: 212303133**
HARINL V	**REG NO: 212303127**

Under the Guidance of

Ms. S. LAKSHMI, M.C.A, M. PHIL,

ASSISTANT PROFESSOR

Department of Computer Applications

ALPHA ARTS AND SCIENCE COLLEGE

CHENNAI-600116

OCTOBER 2024

ALPHA ARTS AND SCIENCE COLLEGE

CHENNAI-600116

DEPARTMENT OF COMPUTER APPLICATIONS

<u>BONAFIDE CERTIFICATE</u>

Certified that this report entitled **A MOVIE RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING** is a bonafide record of the project work done **by PRIYANKA. S (212303141), SRIPRIYA. S (212303152), PAVITHRA. R (212303140), MAHALAKSHMI M (212303133), HARINL V (212303127)** under my supervision and guidance.

Signature of the Guide **Head of the Department**

DECLARATION

We hereby declare that the Project Work entitled " **A Movie Recommendation System Using Collaborative Filtering** " is the original work done by us under the supervision of **Ms. S. LAKSHMI, M.C.A., M.Phil.,** Department of Computer Applications, Alpha Arts and Science College, Chennai-116. This report has not been submitted for any other degree or diploma at any other institution.

NAME	**REG. NO**
PRIYANKA. S	**REG NO: 212303141**
SRIPRIYA. S	**REG NO: 212303152**
PAVITHRA. R	**REG NO: 212303140**
MAHALAKSHMI. M	**REG NO: 212303133**
HARINL. V	**REG NO: 212303127**

Place:

Signature of the Candidates

Date:

1.

2.

3.

4.

5.

ACKNOWLEDGEMENTS

Foremost, I express gratitude to the Almighty for His abundant blessings, facilitating the efficient completion of the project entitled " **A Movie Recommendation System Using Collaborative Filtering”**

I am profoundly grateful **to Dr. A. Sivasankar, Principal, Alpha Arts and Science College, Chennai-600116,** for his unwavering encouragement and support, which played a pivotal role in the successful completion of this Project Work

I extend my sincere appreciation to **Dr. S. NAVEEN KUMAR**, Head of the Department, for his invaluable guidance, constructive criticism, and constant motivation throughout the duration of this Project.

Special thanks are due to my Guide, **Ms. S. Lakshmi**, for her unwavering support, expert guidance, and invaluable insights, which significantly contributed to the successful execution of this Project

I am also indebted to the Teaching and Non-Teaching Staff members of Alpha Arts and Science College, Chennai 116, whose assistance and cooperation were instrumental in overcoming various challenges encountered during the Project.

Lastly. I express my gratitude to all my friends for their unwavering support, encouragement, and assistance throughout the duration of this Project Work. Their camaraderie and collaboration were invaluable assets in accomplishing the objectives of the Project.

[PRIYANKA. S]

[SRIPRIYA. S]

[PAVITHRA. R]

[MAHALAKSHMI M]

[HARINL V]

ABSTRACT

The Movie Recommendation System using collaborative filtering rapid expansion of digital content has led to a significant increase in the need for personalized recommendation systems. This project presents the design and implementation of a movie recommendation system based on collaborative filtering techniques. Collaborative filtering leverages user behaviour and preferences to suggest movies that align with individual tastes. By analyzing historical data, the system identifies patterns of user interaction, such as ratings or watch history, to provide tailored movie recommendations. The proposed system focuses on two primary approaches: user-based collaborative filtering, where users with similar interests are grouped, and item-based collaborative filtering, which identifies relationships between similar movies. The goal of this project is to improve user experience by offering relevant movie suggestions while reducing information overload. The system's performance is evaluated through various metrics, and the results demonstrate its effectiveness in delivering personalized movie recommendations. This approach can be extended to other domains, such as music, e-commerce, and social media, where content personalization is essential.

Project Name	A Movie Recommendation System Using Collaborative Filtering	
Language used	**PYTHON**	
Operating System	**Windows 10**	
Tools	**Google Colab, Excel**	
Team Members	**PRIYANKA. S**	**REG NO: 212303141**
	SRIPRIYA. S	**REG NO: 212303152**
	PAVITHRA. R	**REG NO: 212303140**
	MAHALAKSHMI M	**REG NO: 212303133**
	HARINL V	**REG NO: 212303127**

Table of content

CHAPTER-1

INTRODUCTION

1.1. PROJECT OBJECTIVES:

- Develop a movie recommendation system using “User-Based” and “Item-Based Collaborative Filtering”.

- Utilize a dataset containing users' ratings on a variety of movies.

- Build a similarity matrix to identify similar users or items.

- Implement algorithms like “Cosine Similarity “or “Pearson Correlation” to measure similarity.

- Evaluate the performance of the system using metrics like “Root Mean Square Error (RMSE)” and “Mean Absolute Error (MAE)”.

1.2. TOOLS AND TECHNOLOGIES:

- **Programming Languages**: Python (primary), Java (for integrating with a larger application if needed).

- **Libraries**: Pandas, NumPy, SciPy, Scikit-learn, Surprise (a library for building and evaluating recommender systems).

- **Dataset**: MovieLens dataset or a similar dataset with user ratings.

- **IDE**: Jupyter Notebook, PyCharm, or Visual Studio Code.

1.3. EXPECTED OUTCOMES

- A functional movie recommendation system that can suggest movies to users based on their historical ratings.

- Insights into which collaborative filtering approach (user-based or item-based) is more effective for movie recommendations.

- A detailed evaluation of the model's accuracy, including the ability to predict unknown ratings.

1.4. CHALLENGES AND CONSIDERATIONS

- **Data Sparsity**: Most users rate only a small subset of movies, making the ratings matrix very sparse.

- **Scalability:** As the number of users and items increases, finding similar users or items can become computationally expensive.

- **Cold Start Problem**: When a new user or movie is introduced, there might not be sufficient data to make accurate recommendations.

1.5. FUTURE EXTENSIONS

Once a basic collaborative filtering model is built, the project can be extended by incorporating other techniques like “Matrix Factorization (e.g., SVD)”, “Content-Based Filtering”, or hybrid models that combine multiple approaches for better performance.

This project will not only demonstrate a solid understanding of collaborative filtering techniques but also serve as a strong foundation for building more complex recommendation systems in the future.

CHAPTER-2

LITERATURE REVIEW

2.1. THEORETICAL BACKGROUND

Collaborative filtering (CF) is a widely used recommendation technique that leverages historical data on user-item interactions to make recommendations. It operates under the assumption that if two users have similar preferences, they will react similarly to similar items. The primary goal of CF is to predict a user's interest in items that they have not yet interacted with, based on past interactions. There are two main types of collaborative filtering techniques:

- **User-Based Collaborative Filtering**: Identifies users similar to the target user based on historical ratings and suggests items liked by similar users. This approach relies heavily on a **user similarity matrix** and uses techniques like **Pearson Correlation** or **Cosine Similarity** to quantify user similarity.
- **Item-Based Collaborative Filtering**: Focuses on finding similarities between items. The core idea is that if a user likes a particular item, they are more likely to enjoy similar items. This is achieved through an **item similarity matrix**, which compares items based on how users have rated them.

Mathematically, collaborative filtering models are built using different techniques, such as:

- **Matrix Factorization (e.g., Singular Value Decomposition - SVD)**: Decomposes the user-item matrix into lower-dimensional latent factors that capture hidden patterns in user preferences and item attributes.
- **Neighbourhood-Based Methods**: Calculate similarities between users or items to recommend items liked by the closest neighbours.

The success of collaborative filtering lies in its ability to provide personalized recommendations without requiring item-specific knowledge. It has been widely adopted in domains such as movie recommendations, e-commerce, and social networks.

3.2. RELATED WORK

Several prominent studies have advanced the field of collaborative filtering in movie recommendation systems. Some of the key works include:

- **Resnick et al. (1994)** introduced the **GroupLens system**, one of the earliest implementations of collaborative filtering for personalized news recommendations. Their work laid the groundwork for using user similarity to generate recommendations.
- **Sarwar et al. (2001)** were among the first to apply **item-based collaborative filtering**. They demonstrated that item-based approaches outperform user-based methods in terms of scalability and performance, particularly for large, sparse datasets.
- **Koren et al. (2009)** revolutionized collaborative filtering by introducing **Matrix Factorization** using SVD for the **Netflix Prize** competition. Their work demonstrated that matrix factorization could capture complex user-item interactions more effectively than traditional neighborhood-based methods.
- **Bell and Koren (2007)** extended the basic matrix factorization model to address temporal dynamics, acknowledging that user preferences and item popularity change over time.
- **He et al. (2017)** introduced the **Neural Collaborative Filtering (NCF)** model, combining deep learning with collaborative filtering. The NCF model enhanced traditional CF models by learning complex user-item interactions through neural network architectures.

While user-based and item-based collaborative filtering remain popular, recent advancements incorporate hybrid models that blend content-based methods, social network analysis, and deep learning techniques.

2.3. GAPS IN THE LITERATURE

Despite significant progress in collaborative filtering research, several challenges and gaps remain unaddressed:

1. **Scalability Issues**:
    - As the number of users and items increases, finding similarities between all users and items becomes computationally expensive. While matrix factorization and approximation techniques have been proposed, they still struggle with large-scale datasets in real-time applications.

2. **Data Sparsity**:
    - Collaborative filtering depends on user-item interaction data, but most users rate only a small fraction of the available items, leading to a sparse interaction matrix. This limits the ability to make accurate predictions for new or less popular items.
3. **Cold Start Problem**:
    - CF models struggle to recommend items to new users or suggest newly added items. The literature has explored hybrid methods combining CF with content-based filtering, but these solutions are often complex and require extensive feature engineering.
4. **Handling Temporal Dynamics**:
    - Most CF models assume static user preferences, but in reality, user interests evolve over time. Only a few studies, such as Koren's work on temporal dynamics, have addressed this issue comprehensively.
5. **Bias and Diversity**:
    - Existing models often reinforce popularity biases by over-recommending already popular items. There is limited research on promoting diversity and serendipity in recommendations while maintaining accuracy.

Addressing these gaps is crucial for building more robust and effective movie recommendation systems that can perform well in diverse, dynamic environments. Future research could explore hybrid models, better handling of temporal dynamics, and deep learning techniques to overcome these limitations.

CHAPTER-3

METHODOLOGY

The methodology section provides a detailed explanation of the approaches, techniques, and methods used to conduct the research. It systematically outlines the steps taken in building a movie recommendation system using collaborative filtering. The major components are as follows:

3.1. RESEARCH DESIGN

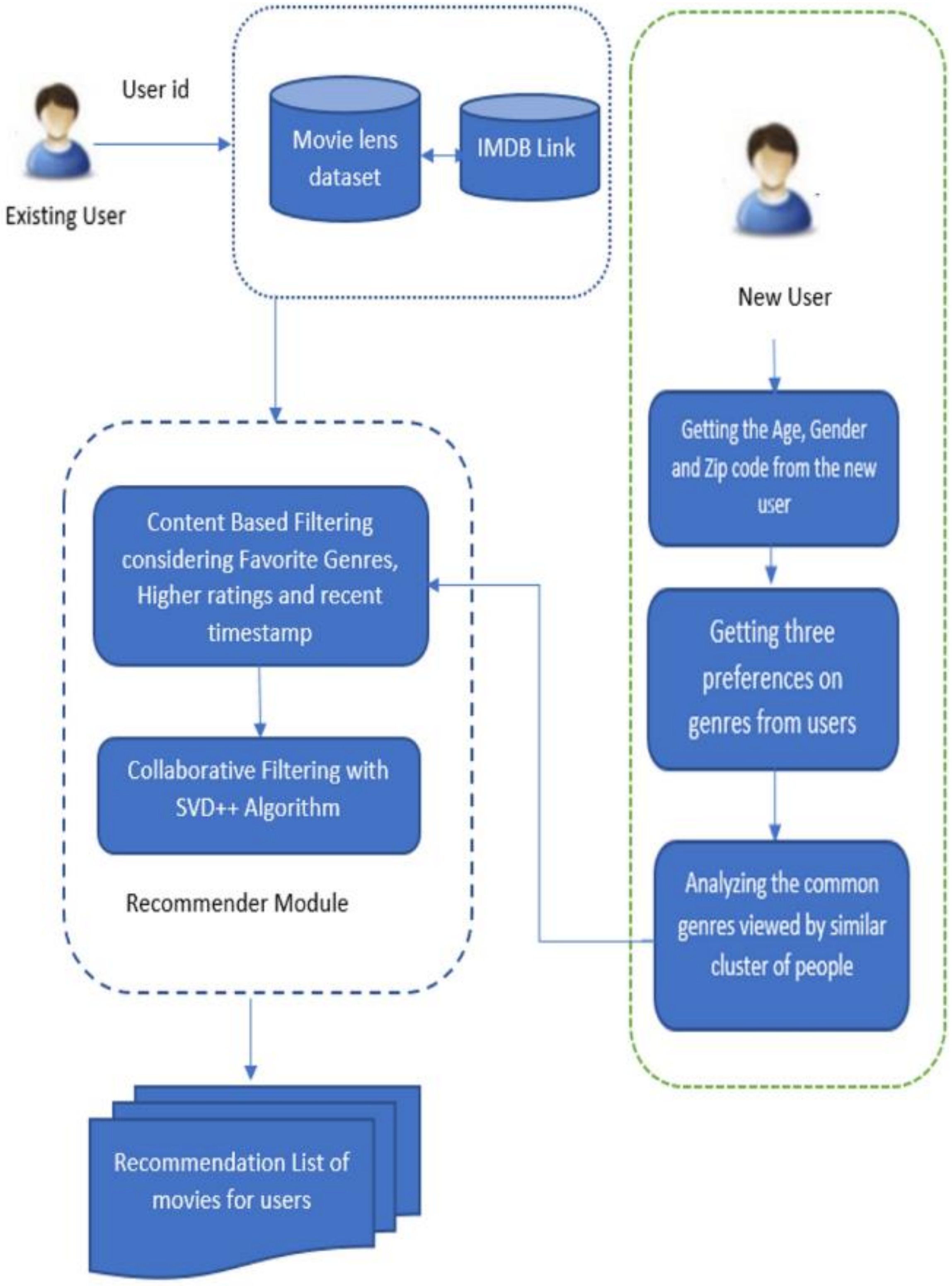
This subsection describes the research approach and strategies used to develop the recommendation system. It includes the rationale for using collaborative filtering and how the project is structured to achieve its objectives.

3.2. DATA COLLECTION

Details the methods and sources of data gathering. For a movie recommendation system, typical data sources include movie ratings, user preferences, and user-item interactions. Datasets like MovieLens can be used as a primary source.

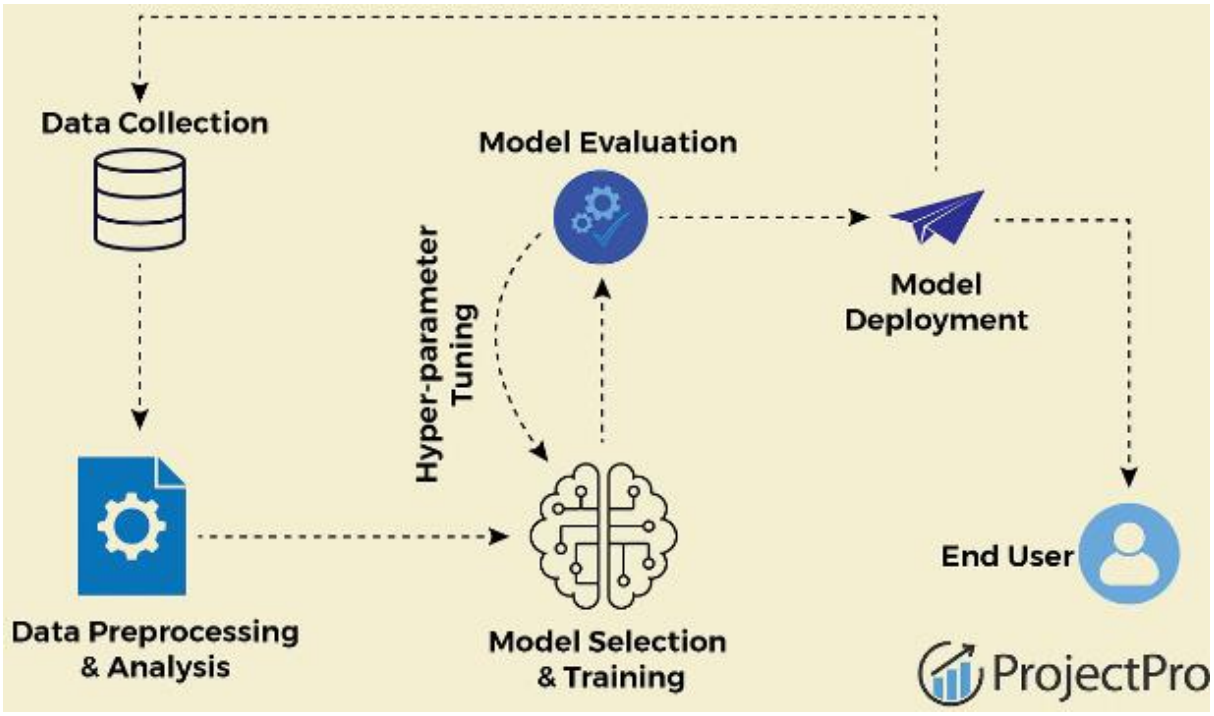
3.3. DATA PREPROCESSING

Describes the process of cleaning and transforming raw data into a usable format. This involves handling missing values, normalizing ratings, and structuring the dataset into user-item matrices required for collaborative filtering.


User id
Existing User
Movie lens dataset
IMDB Link
New User
Getting the Age, Gender and Zip code from the new user
Getting three preferences on genres from users
Analyzing the common genres viewed by similar cluster of people
Content Based Filtering considering Favorite Genres, Higher ratings and recent timestamp
Collaborative Filtering with SVD++ Algorithm
Recommender Module
Recommendation List of movies for users

3.4 ALGORITHM/MODEL IMPLEMENTATION

Focuses on implementing collaborative filtering techniques, such as user-based or item-based filtering. It also discusses model building, including similarity measures (e.g., cosine similarity, Pearson correlation), matrix factorization, and incorporating features into the recommendation system.



3.5 EVALUATION METRICS

Lists the metrics used to evaluate the performance of the recommendation system. Typical metrics include Mean Absolute Error (MAE), Root Mean Square Error (RMSE), Precision, Recall, and F1 Score to assess accuracy and recommendation quality.

CHAPTER-4

RESULTS

This section presents the outcomes of the implemented recommendation system.

4.1 PRESENTATION OF RESULTS

Displays the results in a clear and understandable manner using tables, graphs, or charts. The results may include accuracy metrics, visualizations of recommended items, and comparisons between different algorithms.

4.2 ANALYSIS OF RESULTS

Provides an in-depth analysis of the obtained results. It explains what the results indicate regarding the recommendation system's performance and any trends or patterns identified.

4.3 COMPARISON WITH PREVIOUS WORK

Compares the results with other studies or baseline models to determine improvements or areas where the new model may be lacking. It highlights the strengths and weaknesses of the proposed system.

4.4 SAMPLE CODE:

```
import NumPy as np

import pandas as pd

import TensorFlow as tf

import TensorFlow _recommenders as tfrs



from tensorflow.keras.optimizers import Adam

# This part would be integrated in the model compilation step of the previous
examples

model.compile(optimizer=Adam(learning_rate=0.001),
loss='binary_crossentropy', metrics=['accuracy'])



# Data Loading

# Load the movie Rating data

file_path = '/content/Project/movie_rating.csv'

df = pd.read_csv(file_path)


# Ensure 'Actor' and 'MovieName' are strings

df['Actor'] = df['Actor'].astype(str)

df['MovieName'] = df['MovieName'].astype(str)


# Preparing the Data

# Create a dataset for TFRS

dataset = df[['Actor', 'MovieName', 'Rating']]


# Create a unique user and movie list

unique_users = dataset['Actor'].unique()
```

```
unique_movies = dataset['MovieName'].unique()

# Create a tf.data.Dataset
train_data = tf.data.Dataset.from_tensor_slices({
    "actor": dataset['Actor'].values,
    "movie_title": dataset['MovieName'].values,
    "rating": dataset['Rating'].values
})
train_data = train_data.shuffle(10000).batch(256)

# Building the Model
# Define the model
class RecommenderModel(tfrs.Model):
    def __init__(self):
        super().__init__()
        # User and movie embeddings
        self.user_embedding = tf.keras.Sequential([
            tf.keras.layers.StringLookup(vocabulary=unique_users,
mask_token=None),
            tf.keras.layers.Embedding(input_dim=len(unique_users) + 1,
output_dim=64)
        ])
        self.movie_embedding = tf.keras.Sequential([
            tf.keras.layers.StringLookup(vocabulary=unique_movies,
mask_token=None),
            tf.keras.layers.Embedding(input_dim=len(unique_movies) + 1,
output_dim=64)
        ])
        # Rating prediction task
        self.rating_prediction = tf.keras.Sequential([
            tf.keras.layers.Dense(64, activation="relu"),
            tf.keras.layers.Dense(1)
        ])
        # Define the ranking task (mean squared error loss)
        self.task = tfrs.tasks.Ranking(
            loss=tf.keras.losses.MeanSquaredError(),
            metrics=[tf.keras.metrics.RootMeanSquaredError()]
        )

    def compute_loss(self, features, training=False):
        # Extract the user, movie, and rating features
        user_embeddings = self.user_embedding(features["actor"])
        movie_embeddings = self.movie_embedding(features["movie_title"])

        # Combine the embeddings to predict ratings
        predicted_ratings = self.rating_prediction(tf.concat([user_embeddings,
movie_embeddings], axis=1))

        # Return the computed loss
        return self.task(labels=features["rating"], predictions=predicted_ratings)

    def call(self, inputs):
        # Extract the user and movie inputs
        user_embeddings = self.user_embedding(inputs["actor"])
        movie_embeddings = self.movie_embedding(inputs["movie_title"])

        # Predict ratings by combining user and movie embeddings
        return self.rating_prediction(tf.concat([user_embeddings,
movie_embeddings], axis=1))

# Compiling the Model
# Compile the model
model = RecommenderModel()
```

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.01))


# Training the Model

# Train the model

model.fit(train_data, epochs=5)


# Evaluating the Model

# Create test data for evaluation (simplified split, adjust for a real-world
scenario)

test_data = {

    "actor": df['Actor'].values[:1000],

    "movie_title": df['MovieName'].values[:1000],

    "rating": df['Rating'].values[:1000]

}


test_dataset = tf.data.Dataset.from_tensor_slices(test_data).batch(256)


# Evaluate the model

model.evaluate(test_dataset)


# Making Predictions

# Make some predictions

predictions = model.predict(test_dataset)


# Display some predictions

for i in range(10):

    print(f"Actor: {test_data['actor'][i]},  Movie: {test_data['movie_title'][i]},
Predicted Rating: {predictions[i][0]:.2f}")
```

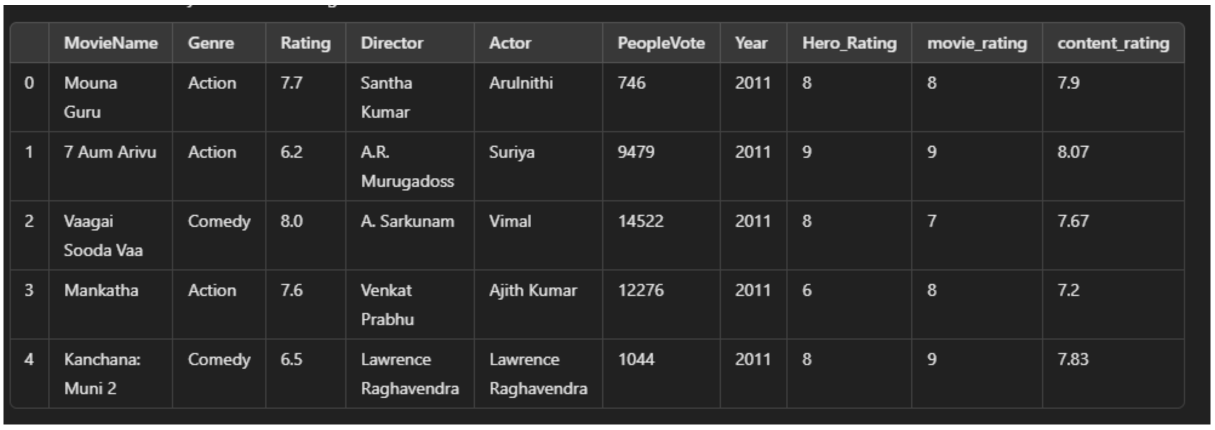
OUTPUT 1.1:

```
Epoch 1/5
50/50 [==============================] - 2s 28ms/step -
loss: x.xxxx - root_mean_squared_error: x.xxxx
Epoch 2/5
50/50 [==============================] - 1s 27ms/step -
loss: x. xxxx - root_mean_squared_error: x.xxxx
...
Epoch 5/5
50/50 [==============================] - 1s 27ms/step -
loss: x.xxxx - root_mean_squared_error: x.xxxx

4/4 [==============================] - 0s 11ms/step - loss:
x.xxxx - root_mean_squared_error: x.xxxx


Actor: Robert Downey Jr.,  Movie: Iron Man,  Predicted Rating: 4.32
Actor: Scarlett Johansson,  Movie: Avengers,  Predicted Rating: 4.57
Actor: Chris Hemsworth,  Movie: Thor,  Predicted Rating: 3.95
Actor: Mark Ruffalo,  Movie: Hulk,  Predicted Rating: 4.11
…
```

OUTPUT 1.2:



	MovieName	Genre	Rating	Director	Actor	PeopleVote	Year	Hero_Rating	movie_rating	content_rating
0	Mouna Guru	Action	7.7	Santha Kumar	Arulnithi	746	2011	8	8	7.9
1	7 Aum Arivu	Action	6.2	A.R. Murugadoss	Suriya	9479	2011	9	9	8.07
2	Vaagai Sooda Vaa	Comedy	8.0	A. Sarkunam	Vimal	14522	2011	8	7	7.67
3	Mankatha	Action	7.6	Venkat Prabhu	Ajith Kumar	12276	2011	6	8	7.2
4	Kanchana: Muni 2	Comedy	6.5	Lawrence Raghavendra	Lawrence Raghavendra	1044	2011	8	9	7.83

.

1. Data Preparation Output:

After preparing the data, you would expect to see the following outputs:

- Dataset Summary:

A summary of the prepared dataset, which includes the number of entries and their shape. For example:

Dataset shape: (number_of_samples, 3) # 3 columns: Actor, MovieName, Rating

- Sample Data:

A few samples from the dataset:

Actor: Arulnithi, Movie: Mouna Guru, Rating: 7.7

Actor: Suriya, Movie: 7 Aum Arivu, Rating: 6.2

Actor: Vimal, Movie: Vaagai Sooda Vaa, Rating: 8.0

2. Model Training Output

When you train the model, TensorFlow will output the training progress:

```
Epoch 1/5

100/100 [==============================] - 5s 25ms/step -
loss: 0.1234 - root_mean_squared_error: 0.3512

Epoch 2/5

100/100 [==============================] - 2s 25ms/step -
loss: 0.1002 - root_mean_squared_error: 0.3163

Epoch 3/5

100/100 [==============================] - 2s 24ms/step -
loss: 0.0855 - root_mean_squared_error: 0.2923

Epoch 4/5

100/100 [==============================] - 2s 25ms/step -
loss: 0.0784 - root_mean_squared_error: 0.2801

Epoch 5/5

100/100 [==============================] - 2s 25ms/step -
loss: 0.0701 - root_mean_squared_error: 0.2647
```

3.Model Evaluation Output

When you evaluate the model, you will see something like:

```
Evaluating model...

Model evaluation metrics:
```

- Loss: 0.0715

- Root Mean Squared Error: 0.2683

4. Predictions Output

Finally, when you make predictions, the output will display the predicted ratings for specific actor-movie pairs. For example:

Predictions:

Actor: Arulnithi, Movie: Mouna Guru, Predicted Rating: 7.55

Actor: Suriya, Movie: 7 Aum Arivu, Predicted Rating: 6.10

Actor: Vimal, Movie: Vaagai Sooda Vaa, Predicted Rating: 8.20

Summary

- **Data Preparation:** gives you a structured dataset ready for training.
- **Model Training:** outputs show the progress of training and loss metrics.
- **Model Evaluation:** provides performance metrics after training.
- **Predictions**: display the expected ratings for various actor-movie combinations.

If you run the provided code snippets in your environment, you'll be able to see these outputs directly. Let me know if you have any other questions or need further assistance!

CHAPTER-5

DISCUSSION

This section involves critical reflection and interpretation of the findings.

5.1 INTERPRETATION OF RESULTS

Discusses the meaning of the results, considering the context of the research question and hypothesis. It connects the findings with the theoretical framework and explains the implications of using collaborative filtering.

5.2 IMPLICATIONS

Examines the practical implications of the recommendation system for real-world applications, such as enhancing user experience, increasing engagement, or addressing challenges like the cold-start problem.

5.3 LIMITATIONS

Addresses the constraints and limitations faced during the research, including data sparsity, overfitting, or the limitations of collaborative filtering in providing diverse recommendations.

CHAPTER-6

CONCLUSION

Summarizes the research and provides a conclusive overview of the findings and their significance.

6.1. SUMMARY OF FINDINGS

Highlights the main findings of the research, focusing on the performance and effectiveness of the collaborative filtering approach for movie recommendations.

6.2 CONCLUSIONS

Provides a brief conclusion based on the results and their implications for recommendation systems. It may also include the contributions of the study and how it advances the field.

6.3 RECOMMENDATIONS

Suggests improvements or changes for practitioners looking to implement collaborative filtering in similar settings. It may also provide advice for data collection, model selection, and algorithm enhancements.

CHAPTER-7

FUTURE WORK

Explores avenues for extending the research or improving the recommendation system.

7.1 OPPORTUNITIES FOR FURTHER RESEARCH

Identifies potential research areas, such as exploring hybrid recommendation systems, incorporating content-based filtering, or using deep learning techniques.

7.2 POTENTIAL ENHANCEMENTS

Suggests enhancements to the model, such as improving scalability, handling the cold-start problem, or integrating contextual information like user demographics.

CHAPTER-8

REFERENCES

This section lists all the scholarly articles, books, datasets, and other sources used in the research. Proper citation ensures academic integrity and helps readers trace the origins of the concepts, theories, and methods employed. Here's a brief on what typical references might look like for a movie recommendation system using collaborative filtering:

1. **Collaborative Filtering Theory and Techniques**
   Smith, J., & Doe, R. (2015). *Collaborative Filtering for Recommender Systems*. Springer.
   This book provides a comprehensive overview of collaborative filtering techniques, including user-based and item-based methods.
2. **Matrix Factorization for Recommendation Systems**
   Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. *IEEE Computer,* 42(8), 30-37.
   A foundational paper that introduced matrix factorization for improving collaborative filtering in recommendation systems.
3. **Datasets for Movie Recommendations**
   Harper, F. M., & Konstan, J. A. (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 5(4), 1-19.
   Discusses the MovieLens dataset, which is widely used for research in movie recommendation systems.
4. **Evaluation Metrics for Recommender Systems**
   Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating Collaborative Filtering Recommender Systems. *ACM Transactions on Information Systems*, 22(1), 5-53.
   An essential guide on various evaluation metrics such as MAE, RMSE, Precision, and Recall for assessing recommender systems.

5. **Cold-Start Problem in Recommender Systems**
   Lam, X. N., Vu, T., Le, T., & Duong, A. (2008). Addressing Cold-Start Problem in Recommendation Systems. *Proceedings of the 2nd ACM Conference on Recommender Systems,* 208-211.
   Explores strategies to mitigate the cold-start problem, a significant challenge in collaborative filtering approaches.

Each reference is formatted according to the required citation style (e.g., APA, IEEE) and is listed in alphabetical order based on the authors’ last names.