**Report Summary**

## **Movie Recommendation System**

**Monalisa Mondal**

**MS in Computer Science**

[**Monalisa.mondal@pace.edu**](mailto:Monalisa.mondal@pace.edu)

**Pace University**

1. **Introduction**

In today's digital world, choosing a movie can be overwhelming with so many options available. I created this **Movie Recommendation System** to make it easier for users to discover movies they’ll love based on their preferences and past ratings. This system uses a mix of **collaborative filtering and content-based filtering** to provide personalized recommendations. It is built using **Python, Pandas, and Scikit-Learn**, with data from the **MovieLens dataset**.

**2. Why I Built This Project**

My goal with this project was to:  
\* Create a **smart movie recommendation engine** that helps users find relevant movies easily.  
\* Explore different recommendation techniques (**content-based filtering, collaborative filtering, and hybrid models**).  
\* Evaluate the system’s performance using **Precision@K, Recall@K, and RMSE** to ensure quality recommendations.

**3. Dataset I Used**

To make this recommendation system work, I used the **MovieLens dataset**, which provides real-world movie ratings from users. It contains:

* **ratings.csv** – User ratings for different movies.
* **movies.csv** – Movie titles, release years, and genres.  
  This dataset is **freely available** at [**MovieLens**](https://grouplens.org/datasets/movielens/).

**4. How It Works**

This system generates recommendations using two key techniques:

**4.1 Content-Based Filtering**

* This method recommends movies based on **their features** (genres, metadata, etc.).
* **How I did it:**  
  \* Extracted **movie genres** and structured them into a usable format.  
  \* Used **cosine similarity** to find movies that are most alike.  
  \* Recommended movies similar to those the user has already watched and liked.

**4.2 Collaborative Filtering**

* Instead of relying on movie attributes, this technique **analyzes user behavior** to find patterns.
* **Two approaches I implemented:**
  + **User-User Similarity:** Finds users with similar movie tastes and recommends movies they enjoyed.
  + **Item-Item Similarity:** Recommends movies that are similar to those the user has already rated.
* The system uses **cosine similarity** to measure relationships between users and movies.

**4.3 Hybrid Model (Best of Both Worlds!)**

Since both techniques have their strengths and weaknesses, I combined them into a **hybrid approach** to improve accuracy and variety in recommendations.

**5. Steps I Took to Build This**

**5.1 Data Preprocessing**

Before making recommendations, I had to clean and prepare the data:  
\* **Checked for missing values** and removed incomplete records.  
\* **Normalized ratings** using **MinMaxScaler** to ensure uniformity.  
\* **Transformed genres** into a machine-readable format.

**5.2 Training the Model & Evaluating Its Performance**

* I trained the model using **movie ratings and metadata**.
* To measure how well it performs, I used:
  + **Precision@K & Recall@K** – Determines how relevant recommendations are.
  + **RMSE (Root Mean Square Error)** – Measures accuracy of predicted ratings.

**6. What I Observed**

After running multiple tests, here’s what I found:  
\* **Collaborative filtering** works best when users have a long history of ratings.  
\* **Content-based filtering** is useful for users with fewer ratings.  
\* The **hybrid model** produces better recommendations overall.  
\* The system can be extended beyond movies – it could work for **books, music, e-commerce, and more!**

**7. How I Can Make It Even Better**

I have a few ideas to take this project further:  
**Deep Learning-Based Recommendations** – Implementing **Neural Collaborative Filtering (NCF)** to improve accuracy.  
**Hybrid Model Optimization** – Fine-tuning how content-based and collaborative filtering work together.  
**Deploying as a Web App** – Making it accessible through a **Flask or Django-based web interface**.  
**Real-Time Recommendations** – Using **live data processing** to update suggestions dynamically.

**8. Final Thoughts**

This was a **fun and rewarding project** that allowed me to dive deep into **recommendation systems**. I successfully built a **Movie Recommendation System** that provides **personalized movie suggestions** using **collaborative filtering and content-based filtering**. There’s always room for improvement, and I look forward to **implementing deep learning and real-time recommendations** to take it to the next level!

**9. Technologies I Used**

**Python** – The backbone of the project  
**Pandas** – For data processing  
**Scikit-Learn** – For machine learning algorithms  
**NumPy** – For numerical computations  
**MovieLens Dataset** – Real-world movie ratings data  
**Jupyter Notebook** – For writing and testing the code