Project Title: Recommender Systems

## **Table of Contents**

1. Introduction
2. Task 0: Initial Setup
3. Task 1: User-based Collaborative Filtering
4. Task 2: Item-based Filtering
5. Task 3: A Better Recommender System

* 3.1 Develop a New Solution
* 3.2 Evaluate Performance

1. Task 4: Presentation
2. Conclusion
3. References

## **1. Introduction**

The project focuses on the implementation and evaluation of recommender systems, a crucial aspect of data science widely employed in real-world applications. The primary objective is to develop and assess user-based collaborative filtering (Task 1), item-based collaborative filtering (Task 2), and propose an enhanced recommender system (Task 3) using the MovieLens 1M dataset. The MovieLens 1M dataset provides a comprehensive collection of user ratings for movies, serving as the foundation for building and testing various recommendation algorithms.

**Project Goals**

1. Implement user-based collaborative filtering using KNN and study the impact of different K values on the prediction accuracy.

2. Develop and evaluate item-based collaborative filtering employing KNN, comparing the performance of different similarity metrics.

3. Propose a novel solution for movie recommendation, either based on existing publications (Option 1) or through an original algorithm (Option 2), and assess its effectiveness.

4. Compare the performance of the proposed solution with two baseline methods (Movie Average and KNN-based Collaborative Filtering) using evaluation metrics such as Average Precision (AP) and Normalized Discounted Cumulative Gain (NDCG).

**Dataset: MovieLens 1M**

The dataset, MovieLens 1M, serves as the cornerstone of this project. It contains a vast collection of user ratings for movies, enabling the exploration and implementation of various recommender system algorithms. The dataset provides essential information such as user ratings, movie details, and user preferences, essential for training and evaluating the recommendation models.

By leveraging this dataset, the project aims to contribute insights into the effectiveness of different collaborative filtering approaches and propose an innovative solution to enhance the overall quality of movie recommendations.

## **2. Task 0: Initial Setup**

**Dataset Download**

To initiate the project, the first step involves downloading the MovieLens 1M dataset. This dataset, available in the form of “ml-1m.zip”, contains essential information about user ratings for movies. The README.txt file within the dataset provides comprehensive details about its structure and contents.

**The following steps outline the dataset download process:**

1. Download the Dataset: Obtain the “ml-1m.zip” file from the provided source.

2. Extract the Dataset: Unzip the contents of `ml-1m.zip` to access the necessary files and folders for analysis.

**Jupyter Notebook Creation**

Creating a Jupyter Notebook provides a structured environment for developing, implementing, and documenting the code. Here are the steps for setting up the Jupyter Notebook:

1. Create a New Jupyter Notebook: Open your preferred Python environment (preferably Anaconda) and create a new Jupyter Notebook named `assignment3.ipynb`.

2. Organize Code Sections: Utilize the provided template (`assignment3-copy1.ipynb`) to organize the code into distinct sections, ensuring clarity and readability.

3. Add Comments: Throughout the code, include detailed comments to explain the implementation of various tasks. This is crucial for understanding the logic behind each step.

**Code Organization**

Effectively organizing the code within the Jupyter Notebook enhances readability and simplifies the debugging process. Follow these guidelines for code organization:

1. Cell Structure: Divide the code into cells, each dedicated to a specific task or analysis.

2. Commenting: Add comments to elucidate the purpose and functionality of each code segment.

3. Remove Unnecessary Lines: Clean the code by removing any redundant or unnecessary lines to maintain conciseness.

4. Kernel Restart & Run All: Before submission, ensure the code runs seamlessly by executing the "Restart & Run All" option from the Jupyter Notebook's Kernel menu.

## **3. Task 1: User-based Collaborative Filtering**

### **3.1 Implementation:**

User-based collaborative filtering using K-nearest neighbours (KNN) involves the following steps:

**1. Data Preparation:**

* + Loading Data:
  + Load user data (e.g., UserID, Gender, Age).
  + Load movie data (e.g., MovieID, Title, Genres).
  + Load rating data (e.g., UserID, MovieID, Rating).
  + Merging Data:
  + Merge user data with rating data based on the 'UserID' column.
  + Merge movie data with the combined user-rating data based on the 'MovieID' column.
  + Select relevant columns to create a Data Frame with UserID, MovieID, Title, and Rating.

**2. Creating User-Movie Matrix:**

* + Pivoting Data:
  + Using the pandas `pivot\_table` function to create a matrix where rows represent users, columns represent movies, and values represent ratings.
  + Filling missing values with zeros or another appropriate value.

**3. Building the KNN Model:**

* + Set Up Parameters:
  + Define a range of k values.
  + Iterative Process for Different k Values:
  + For each k:
  + Initialize KNN model with chosen k and Euclidean metric.
  + Fit model with user-item matrix.
  + Find (k+1) nearest neighbours for a random user.
  + Predict user ratings by averaging nearest neighbours’ ratings.
  + Calculate RMSE for user predictions.
  + Update best k and RMSE if current combination is better.

**4. Predicting Ratings:**

* + Once the (k)-nearest neighbours are identified, the code aggregates the ratings given by these neighbours.
  + The predicted rating for each item is calculated by taking the average of the ratings given by the (k) neighbours for that item.
  + This process is applied for each item that the user has not rated.
  + The code is using a user-based collaborative filtering approach, specifically KNN, to predict ratings for a given user by leveraging the ratings of its most similar users. The prediction is based on the average ratings of the user's neighbours.

### **3.2 Evaluation:**

**1. Studying Different K Values:**

* The code explores a range of (k) values to understand their impact on user-based collaborative filtering predictions.
* For each (k) value, the code computes predictions using the k-nearest neighbours (KNN) algorithm with Euclidean metric.
* This process involves finding the nearest neighbours for a randomly selected user and predicting their ratings based on the average ratings of these neighbours.

**2. RMSE for Evaluation:**

* + Root Mean Squared Error (RMSE) is calculated for each set of predictions.
  + RMSE is a metric that measures the average difference between the actual ratings given by users and the predicted ratings generated by the collaborative filtering model.
  + By evaluating RMSE for different (k) values, the code assesses how well the model performs with varying numbers of neighbours.

**3. Results and Findings:**

* + The code identifies that the optimal (k) value, which minimizes RMSE, is 7.
  + The corresponding RMSE score for (K = 7) is 2.77493.
  + This RMSE score is the lowest among the tested (k) values (3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25, 27, 29), indicating that using 7 nearest neighbours yields the most accurate predictions.
  + The findings suggest that, in this specific scenario, a (k) value of 7 leads to the best balance between capturing local patterns and avoiding sensitivity to noise in the data.

## **4. Task 2: Item-based Filtering**

### **4.1 Implementation:**

**1. Data Preparation:**

* + The item-based collaborative filtering implementation begins with creating the movie features DataFrame by pivoting the original DataFrame, resulting in a matrix where rows represent movies, columns represent users, and values represent ratings. Missing values are filled with zeros.
  + A sparse matrix is generated from the movie features DataFrame using the `csr\_matrix` function for efficient computations during the KNN process.

**2. Parameter and Metric Choices:**

* + `k\_values`: A range of values for the number of neighbors (e.g., [3, 5, 7, ... 29]). In this case, the optimal value is found to be 25.
  + `similarity\_metrics`: Two similarity metrics, cosine similarity and Euclidean distance, are considered.

**3. Model Training and Prediction:**

* + The implementation includes a loop over each combination of `k` and similarity metric to explore different scenarios.
  + For each combination, a KNN model is created with the specified parameters. The model is trained on the movie features matrix.
  + Neighbors for a randomly selected movie (index 3174) are found by measuring the similarity between items using the chosen metric and considering the top `k` neighbors.
  + Ratings are predicted for the selected movie based on the average of the ratings from its nearest neighbors.

**4. Results:**

* + The best combination of hyperparameters (`k` and similarity metric) is determined based on the lowest Root Mean Squared Error (RMSE). In this case, the best combination is found to be K=25 and Similarity Metric=cosine, with an RMSE of 2.8376.
  + Results for each combination of `k` and similarity metric are stored, including the RMSE value.

### **4.2 Evaluation:**

**1. Choice of Metrics:**

* + Two similarity metrics, cosine similarity and Euclidean distance, are considered. These metrics provide different perspectives on the similarity between items.
  + The evaluation focuses on identifying the combination of `k` and similarity metric that minimizes RMSE, indicating the most accurate predictions.

**2. Performance Evaluation:**

* + RMSE is used as the evaluation metric to quantify the prediction accuracy by calculating the square root of the mean squared differences between predicted and actual ratings.
  + The best combination of hyperparameters (K=25, Similarity Metric=cosine) is identified, providing optimal performance for item-based collaborative filtering using KNN.
  + The results table presents an overview of RMSE values for different combinations, demonstrating the effectiveness of the chosen parameters in making accurate movie recommendations.

## 5. Task 3: A Better Recommender System

### 5.1 Develop a New Solution

#### 5.1.1 Option 1: Solution from Publications

Describe the chosen solution, provide citations, and explain the implementation in Python with comments.

#### 5.1.2 Option 2: Proposed Algorithm

Explain the proposed algorithm, justify its originality, and provide a strong list of references.

### 5.2 Evaluate Performance

#### 5.2.1 Recommendation to Users

Explain the process of recommending Top-30 movies to selected users and the evaluation metrics used (AP and NDCG).

#### 5.2.2 Results Comparison

Present a comparison of Movie Average, KNN-based Collaborative Filtering, and the new solution's performance using visualizations.

---

## 6. Task 4: Presentation

Summarize key results and findings in slides with the following sections:

- Cover page/slide & project info

- Task 1 results

- Task 2 results

- Task 3.1 details and citations

- Literature review (if applicable)

- Algorithm details for Option1RecSys or Option2RecSys

- Task 3.2 key results and visualizations

- List of references

---

## 7. Conclusion

Summarize the overall findings and the effectiveness of the proposed recommender system.

---

## 8. References

List all the references used in the project, including publications, articles, and relevant documentation.