# National Institute of Technical Teachers Training and Research(NITTTR) Project Work

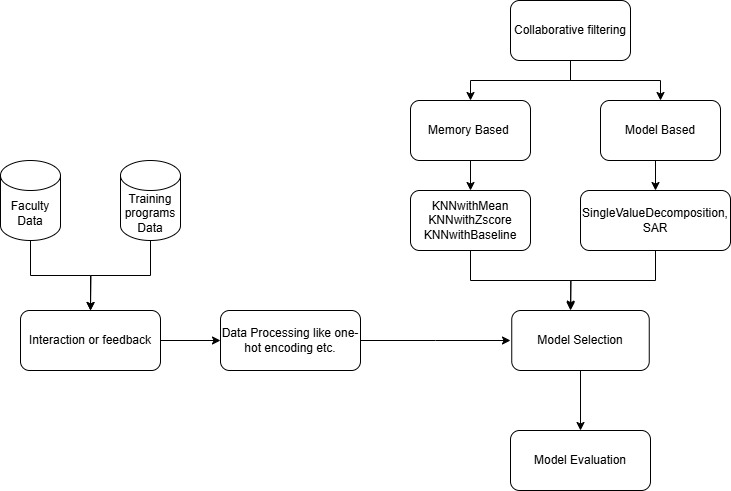


Project Title: **Training Program Recommendation System**

# Objective:

The objective of this project is to develop a recommendation system tailored for faculty members from diverse branches. The system aims to assist faculty in selecting relevant courses based on their training mode preference (online or offline), age, gender, and the number of courses completed within their respective branches. Furthermore, the system will recommend courses that faculty members from the same branch have completed, ensuring a collaborative learning environment. For faculty who have not completed any courses, the system will recommend courses from their department's domain. The system will also track and display the number of courses completed by each faculty member, providing insights into their continuous learning journey. This project seeks to enhance faculty development by facilitating personalized and relevant course recommendations, ultimately enriching their teaching and research capabilities.

# Technical Architecture:



**Data collection:**

**Required features to train the model:**

|  |  |
| --- | --- |
| **Faculty Dataset** | **Training Program Dataset** |
| User ID | Program Id |
| Dept ID | Program |
| Dept Name | Dept Id |
| Program ID | Instructor Info |
| Program Attended | Difficulty Level |
| User Age | Description |
| Gender | Duration |
| Mode | Rating |
| Duration | Total Enrollments |
| Profession |  |
| Interests |  |
| Publications on Interests |  |

**Data Processing and Splitting:**

* One Hot Encoding
* Standardization/normalization

Splitting:

* Random Splitting
* Chronological Splitting
* Stratified Splitting

**Model Selection:**

### Collaborative based filtering:

**1. Memory-based approach**: Utilizes entire user-item rating information to calculate similarity scores between items or users for making recommendations. These are further either of 2 types:

* **User-based**: Two users are considered similar if they rate items similarly. An item is recommended to a user, if another user i.e., similar to the user in question has liked the item

**Example**: If Ram and Sam have both liked similar training programs in the past, and Sam attended a new training program that Ram has not yet attended, the system will recommend that new training program to Ram.

* **Item-based**: Two items are considered similar if users similarly rate them. An item is recommended to a user, that is similar to the items the user has rated in the past

**Example**: If Ram attended a "Leadership" program and many users who attended "Leadership" also attended "AI Basics," the system will recommend "AI Basics" to Ram.

**Algorithms**:KNN with Mean,KNN with Zscore,KNN with baseline

**2. Model-based approach**: Utilizes user-item rating information to build a model & the model (not the entire dataset) is thereafter used for making recommendations. This approach is preferred in instances where time & scalability are a concern.

**Algorithm**:SVD

This project aims to build a recommendation system based on collaborative filtering & will tackle an example of both memory-based & model-based algorithms.

**Model Evaluation:**

Common evaluation metrics for recommendation systems include:

1. **Precision**: Measures the proportion of recommended items that are relevant.
2. **Recall**: Measures the proportion of relevant items that are recommended..
3. **Mean Absolute Error (MAE)**: Measures the average absolute difference between predicted and actual ratings.
4. **Root Mean Squared Error (RMSE)**: Measures the square root of the average squared differences between predicted and actual ratings.
5. **Hit Rate**: Measures the percentage of times a recommended item is among the top-N recommendations.
6. **Coverage**: Measures the percentage of items that can be recommended.

Train-Test Split

Split your data into training and testing sets to evaluate the performance of your models. This helps simulate how the system will perform in real-world scenarios.

### Evaluation for Memory-Based Collaborative Filtering

User-Based and Item-Based Collaborative Filtering

1. **Train-Test Split**: Divide the user-item interaction matrix into training and test sets.
2. **Similarity Calculation**: Compute user-user or item-item similarity using the training set.
3. **Generate Recommendations**: Use the similarity matrix to generate recommendations for the test set.
4. **Calculate Metrics**: Compare the recommendations with the actual interactions in the test set using precision, recall, F1 score, MAE, RMSE, etc.

### Evaluation for Model-Based Collaborative Filtering

Using Matrix Factorization (e.g., SVD)

1. **Train-Test Split**: Split your dataset into training and test sets.
2. **Model Training**: Train the matrix factorization model on the training set.
3. **Generate Predictions**: Use the model to predict ratings for the test set.
4. **Calculate Metrics**: Compare the predicted ratings with the actual ratings using MAE, RMSE, and other relevant metrics.

### Example Evaluation Process

Here’s a detailed example of how you can evaluate your recommendation system:

Step 1: Train-Test Split

Divide your data into training and test sets. For collaborative filtering, ensure that the split maintains a reasonable number of interactions for both users and items in both sets.

Step 2: Memory-Based Evaluation

For User-Based Collaborative Filtering:

* **Compute Similarity**: Calculate user-user similarity using the training set.
* **Generate Recommendations**: For each user in the test set, find similar users and recommend items they liked.
* **Evaluate**: Calculate precision, recall, F1 score, MAE, RMSE, etc., by comparing the recommended items with the actual items the user interacted with in the test set.

For Item-Based Collaborative Filtering:

* **Compute Similarity**: Calculate item-item similarity using the training set.
* **Generate Recommendations**: For each user in the test set, recommend items similar to those they interacted with.
* **Evaluate**: Use precision, recall, F1 score, MAE, RMSE, etc., to assess the recommendations.

Step 3: Model-Based Evaluation

Using Matrix Factorization (e.g., SVD):

* **Train Model**: Train the SVD model on the training set.
* **Generate Predictions**: Predict ratings for the interactions in the test set.
* **Evaluate**: Calculate metrics such as MAE and RMSE by comparing the predicted ratings with the actual ratings in the test set.