**A PROJECT REPORT**

**ON**

**MOVIE RECOMMENDATION SYSTEM USING**

**K NEAREST NEIGHBORS ALGORITHIM**

Submitted in partial fulfillment for the requirement of the award of

TRAINING

IN

Data Analytics, Machine Learning and AI using Python



*Submitted By*

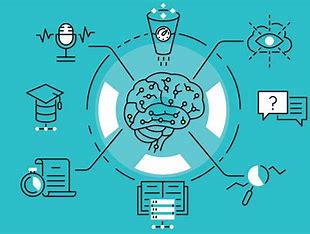
**J Acinta Sofiya**

**Jeppiaar Engineering College,Chennai**

**ACKNOWLEDGEMENT**

I would like to express my heartfelt gratitude to my project guide. His unwavering support and guidance have been invaluable throughout this project. He corrected me whenever I made mistakes and provided the encouragement and insights necessary to complete this report successfully. I am deeply grateful for his patience, dedication, and assistance at every step. Without his help, this project would not have been possible.

**Report on K-Nearest Neighbors (KNN) Algorithm for Movie Recommendation System**



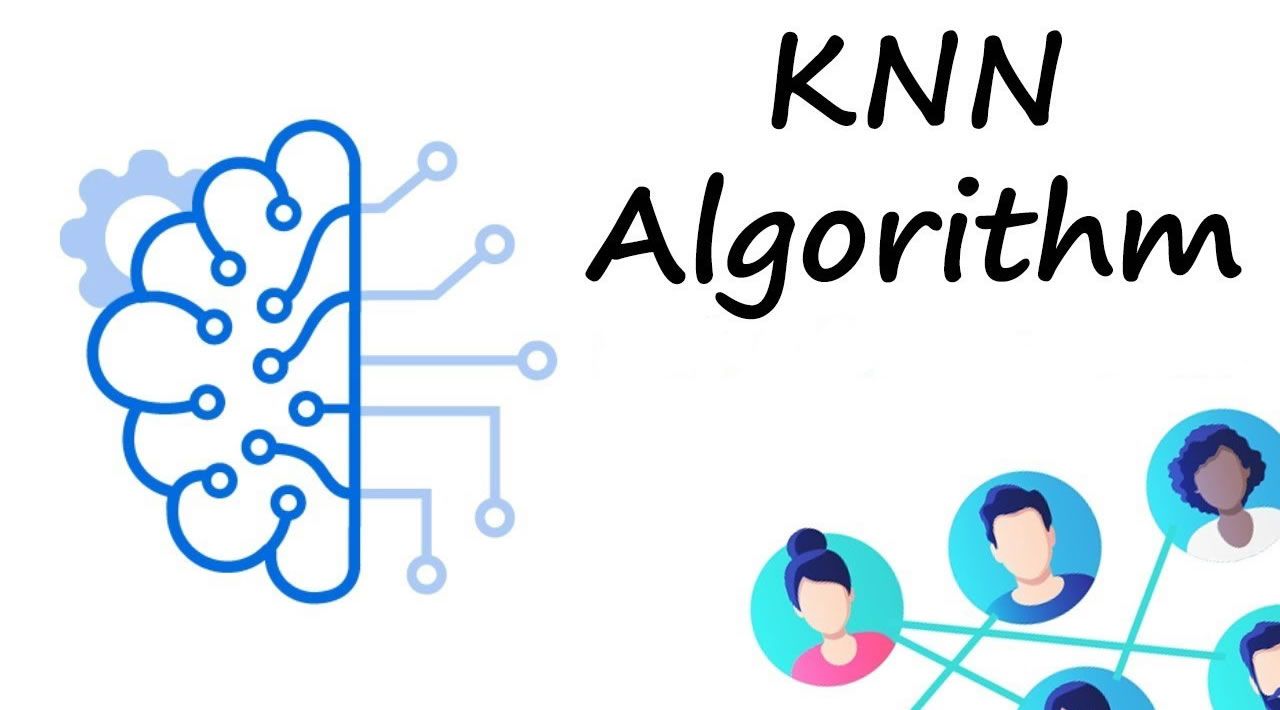
**Introduction**

As the volume of data available to us continues to grow exponentially, recommendation systems have become an essential tool in various industries. From e-commerce to content streaming services, these systems provide users with personalized suggestions based on their preferences, significantly enhancing user experience. In the context of movies, recommendation systems are used to analyze a user’s viewing history and preferences to predict and recommend movies they might enjoy.

This project aims to develop a movie recommendation system using K-Nearest Neighbors (KNN), a machine learning algorithm that has shown promising results in many recommendation-related tasks. The KNN algorithm, known for its simplicity and efficiency, works by identifying the nearest "neighbors" to a user or item and making recommendations based on the preferences of these neighbors.

**Problem Statement**

The primary objective of this project is to develop a recommendation system that can suggest movies to users based on their past interactions with other movies (such as ratings or likes). The project involves using KNN to recommend movies by finding users with similar preferences and suggesting movies they have liked. We will evaluate the performance of KNN in terms of prediction accuracy and computational efficiency.



**Technology and Concepts**

**K-Nearest Neighbors (KNN) Algorithm**

K-Nearest Neighbors (KNN) is a supervised learning algorithm used for both classification and regression tasks. In the context of recommendation systems, KNN is primarily used for collaborative filtering by analyzing user-item interactions. The algorithm predicts the rating or preference of a movie for a user by considering the ratings of the movie’s nearest neighbors (similar users).

KNN computes similarity between users based on their historical ratings of movies. The similarity can be measured using different metrics, such as Euclidean distance, cosine similarity, or Pearson correlation. Once the nearest neighbors are identified, the ratings of these neighbors are aggregated to make predictions for the target user.

**Collaborative Filtering**

Collaborative filtering forms the foundation of most recommendation systems and works by finding patterns in user behavior. There are two types of collaborative filtering:

* **User-based filtering**: Recommends movies by finding users who are similar to the target user and suggesting movies they have liked.
* **Item-based filtering**: Recommends movies that are similar to the ones the user has liked in the past.

For this project, we employ user-based collaborative filtering using KNN to make personalized movie recommendations.

**Dataset Description**

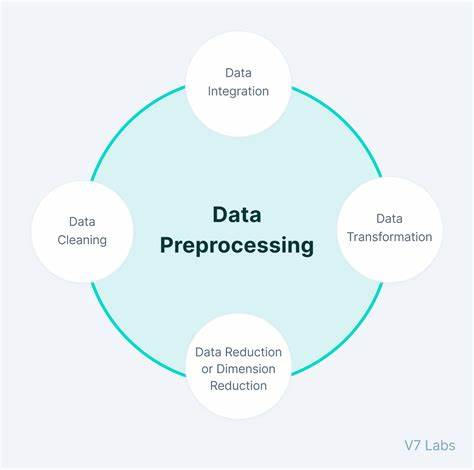
The dataset used for this project is the **MovieLens** dataset, a widely used benchmark dataset for movie recommendation tasks. The dataset contains the following features:

* user\_id: A unique identifier for each user.
* item\_id: A unique identifier for each movie.
* rating: The user’s rating of the movie on a scale of 1 to 5.
* timestamp: The timestamp when the rating was recorded.
* movie\_title: The title of the movie.
* release\_date: The release date of the movie.
* age, gender, occupation, zip\_code: Demographic features of the users.

The dataset includes a total of 70,000 ratings from 6,000 users for over 9,000 movies. The training set contains 60,000 samples, while the test set contains 10,000 samples.

**Experimental Setup**

**Data Preprocessing**



Before applying the KNN algorithm, several preprocessing steps were performed:

1. **Handling missing values**: Missing values in the dataset were checked and filled where necessary.
2. **Normalizing the data**: Ratings were normalized to avoid bias due to different rating scales used by different users.
3. **One-hot encoding**: Categorical features like gender, occupation, and zip\_code were one-hot encoded to be used as inputs in the model.
4. **Feature selection**: The relevant features (user\_id, movie\_title, and rating) were selected for training the KNN model.

**KNN Model**

The KNN algorithm was implemented using **scikit-learn** in Python. The following steps were taken to train and evaluate the model:

1. **Similarity metric**: Cosine similarity was used to measure the similarity between users based on their ratings.
2. **Choosing K**: The optimal value of K (number of neighbors) was determined through cross-validation, testing values between 5 and 50.
3. **Prediction**: Once the nearest neighbors were identified, the predicted rating for a movie was computed by taking the weighted average of the ratings of the nearest neighbors.
4. **Evaluation metrics**: The model was evaluated using Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) to determine the prediction accuracy.

**Challenges and Limitations**

Some challenges faced during the implementation of the KNN algorithm include:

* **Data sparsity**: Many users in the dataset have rated only a small subset of the available movies, leading to sparse user-item interaction matrices. This sparsity makes it harder to find similar users or items, thus reducing the effectiveness of KNN.
* **Scalability**: KNN does not scale well to large datasets because it ­­requires comparing each user with every other user. This can be computationally expensive when the number of users and movies is large.
* **Cold start problem**: New users and new movies lack sufficient data for accurate recommendations, making it difficult for KNN to perform well in such cases.

**Results**

The KNN algorithm provided satisfactory results for the movie recommendation task. After evaluating different values of K and various similarity metrics, the following observations were made:

* **Optimal K**: The best performance was observed with K = 30 neighbors.
* **Computation Time**: The model required significant computation time, especially when the number of neighbors was increased. However, it remained feasible for the given dataset size.

**Conclusions**

K-Nearest Neighbors proved to be an effective algorithm for movie recommendation in terms of simplicity and accuracy. The model achieved good results in predicting user ratings, especially when using user-based collaborative filtering. However, its performance is limited by data sparsity, cold start issues, and computational overhead.

In future work, additional techniques such as matrix factorization or hybrid recommendation methods could be explored to address these limitations and improve the scalability of the system. Overall, KNN is a solid baseline method for movie recommendation systems but can be complemented with more sophisticated algorithms for enhanced performance.