MACHINE LEARNING

A Thesis

SUBMITTED BY

Batch – B Group - 7

Nishanth.S CB.EN.U4AIE22149

Sarvesh.K CB.EN.U4AIE22153

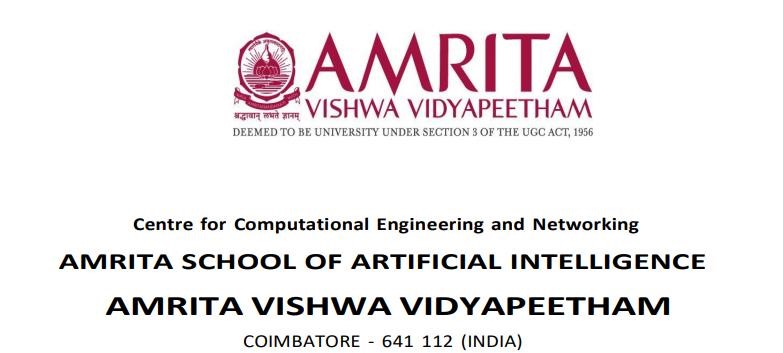
Shruthi.R CB.EN.U4AIE22154

Praatosh.B CB.EN.U4AIE22160

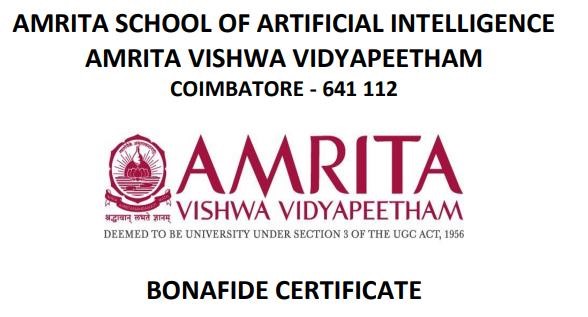
*In partial fulfilment for the award of the degree of*

**BACHELOR OF TECHNOLOGY**

**IN CSE(AI)**



**June- 2024**



This is to certify that the thesis entitled Recommendation System using SVT and BPR submitted by Nishanth.S(CB.EN.U4AIE22149), Sarvesh.K (CB.EN.U4AIE22153), Shruthi.R(CB.EN.U4AIE22154), Praatosh.B(CB.EN.U4AIE22160) of Batch B of the year 2022-2026 for the award of the Degree of Bachelor of Technology in the “CSE(AI) ” is a Bonafide record of the work carried out by them under our guidance and supervision at Amrita School of Artificial Intelligence, Coimbatore.

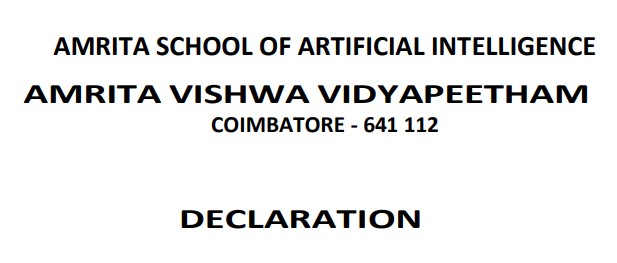
**Dr. Abhishek S**

Project Guide

**Dr. K.P. Soman**

Professor and Head CEN

Submitted for the university examination held on 04th June 2024



We, the Group 7 of Batch-B, hereby declare that this thesis, is the record of the original work done by me under the guidance of Dr. Abhishek S, Assistant Professor, Centre for Computational Engineering and Networking, Amrita School of Artificial Intelligence, Coimbatore. To the best of my knowledge this work has not formed the basis for the award of any degree/diploma/ associate ship/fellowship/or a similar award to any candidate in any University.

BY:

Nishanth.S CB.EN.U4AIE22149

Sarvesh.K CB.EN.U4AIE22153

Shruthi.R CB.EN.U4AIE22154

Praatosh.B CB.EN.U4AIE22160

**Place: Ettimadai**

**Date: 04-06-2023**

Table of Contents

[Acknowledgement 5](#_Toc168221164)

[PART A : COMPARITIVE ANALYSIS OF GROUND WATER LEVEL PREDICTION 7](#_Toc168221165)

[ABSTRACT 7](#_Toc168221166)

[INTRODUCTION 8](#_Toc168221167)

[METHODOLOGY 8](#_Toc168221168)

[Linear Regression 8](#_Toc168221169)

[Polynomial Regression 10](#_Toc168221170)

[Results and Discussion 12](#_Toc168221171)

[Linear Regression 12](#_Toc168221172)

[Polynomial Regression 12](#_Toc168221173)

[CONCLUSION: 13](#_Toc168221174)

[FUTURE WORK: 14](#_Toc168221175)

[PART B : MOVIE RECOMMENDATION SYSTEM 15](#_Toc168221176)

[ABSTRACT 15](#_Toc168221177)

[INTRODUCTION 15](#_Toc168221178)

[RECOMMENDATION SYSTEM 17](#_Toc168221179)

[Using SVT 19](#_Toc168221180)

[KNN 25](#_Toc168221181)

[CODE 27](#_Toc168221182)

[Recommendation using SVT 27](#_Toc168221183)

[Comparisons 30](#_Toc168221184)

[LITERATURE SURVEY 31](#_Toc168221185)

[References 33](#_Toc168221186)

# Acknowledgement

This project has been possible due to the sincere and dedicated efforts of many. First, we would like to thank the dean of our college for giving us the opportunity to get involved in a project and express our skill. We thank our Machine Learning teacher, Dr. Abhishek.S for his guidance and support without which this project would have been impossible. Finally, we thank our parents and our classmates who encouraged us throughout the project.

Table of Figures

[Figure 1.General form of Linear Regression 9](#_Toc168221187)

[Figure 2. Linear vs Polynomial 10](#_Toc168221188)

[Figure 3. Result from Linear regression 12](#_Toc168221189)

[Figure 4.Result from Polynomial Regression 12](#_Toc168221190)

[Figure 5. Plot comparison using MSE and R-square 13](#_Toc168221191)

[Figure 6.Collaborative Filtering 17](file:///D:\AMRITA\Sem4\ML\Final\ML_Final.docx#_Toc168221192)

[Figure 7.Content-Based Filtering 18](file:///D:\AMRITA\Sem4\ML\Final\ML_Final.docx#_Toc168221193)

[Figure 8. Hybrid Methods of Filtering 18](file:///D:\AMRITA\Sem4\ML\Final\ML_Final.docx#_Toc168221194)

[Figure 9. SVT Algorithm 21](#_Toc168221195)

[Figure 10. SVT Recommendation Output 30](file:///D:\AMRITA\Sem4\ML\Final\ML_Final.docx#_Toc168221196)

# PART A : COMPARITIVE ANALYSIS OF GROUND WATER LEVEL PREDICTION

## ABSTRACT

The principal aim of this ambitious project is to generate the most accurate and insightful predictions regarding the groundwater levels in Chennai city. To attain this objective, we have undertaken a rigorous comparative analysis utilizing two distinguished machine learning algorithms: Polynomial Regression, and Linear Regression. Each methodology offers a unique approach to modelling and forecasting, providing a comprehensive perspective on Chennai's groundwater dynamics.

In our quest for precision and reliability, we meticulously sourced high-quality datasets from reputable websites. These datasets span from January 2011 to December 2023 and include essential variables such as historical rainfall records, historical groundwater levels, and temperature data. Ensuring the integrity and breadth of our data, we established a solid foundation for our machine learning models.

To enhance the interpretability and utility of our findings, we employed sophisticated data visualization techniques. These visualizations elucidate complex patterns within the data, making it more accessible for stakeholders. Detailed graphs, charts, and interactive plots present our comparative analysis in a visually appealing and analytically rigorous manner, facilitating easier inference and supporting informed decision-making for sustainable water management in Chennai.

Moreover, we utilized evaluation metrics like R-squared (R²) and Mean Squared Error (MSE) to assess the performance and precision of our models. R-squared measures how well the model's predictions match the actual data, while Mean Squared Error indicates the average prediction error, ensuring our models are fine-tuned for optimal accuracy.

By leveraging advanced machine learning algorithms, comprehensive data visualization, and rigorous evaluation metrics, our project aspires to deliver precise and actionable insights into Chennai's groundwater levels, contributing significantly to urban water resource management and sustainable practices in metropolitan cities globally.

## INTRODUCTION

Faced with numerous challenges, major metropolitan cities across the globe are grappling with the pressing issue of groundwater depletion. In this context, we have meticulously selected Chennai as the focal point of our comprehensive study. This thriving metropolis serves as the quintessential sample space for our project, enabling us to delve deep into the intricacies of groundwater dynamics.

To forecast the future groundwater levels in Chennai, we have employed a sophisticated array of parameters. These encompass historical rainfall data, which provides crucial insights into precipitation patterns over the years, historical groundwater levels that reveal long-term trends and fluctuations, and the city's temperature records, which influence evaporation rates and water demand. By integrating these diverse datasets, our analysis aspires to present a robust and precise forecast of Chennai's groundwater levels, contributing valuable information for sustainable water management and urban planning.

## METHODOLOGY

### Linear Regression

Linear regression is a type of supervised machine learning algorithm that computes the linear relationship between the dependent variable and one or more independent features by fitting a linear equation to observed data. The primary objective while using linear regression is to locate the best-fit line, which implies that the error between the predicted and actual values should be kept to a minimum. There will be the least error in the best-fit line.

The best Fit Line equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a unit change in the independent variable(s).

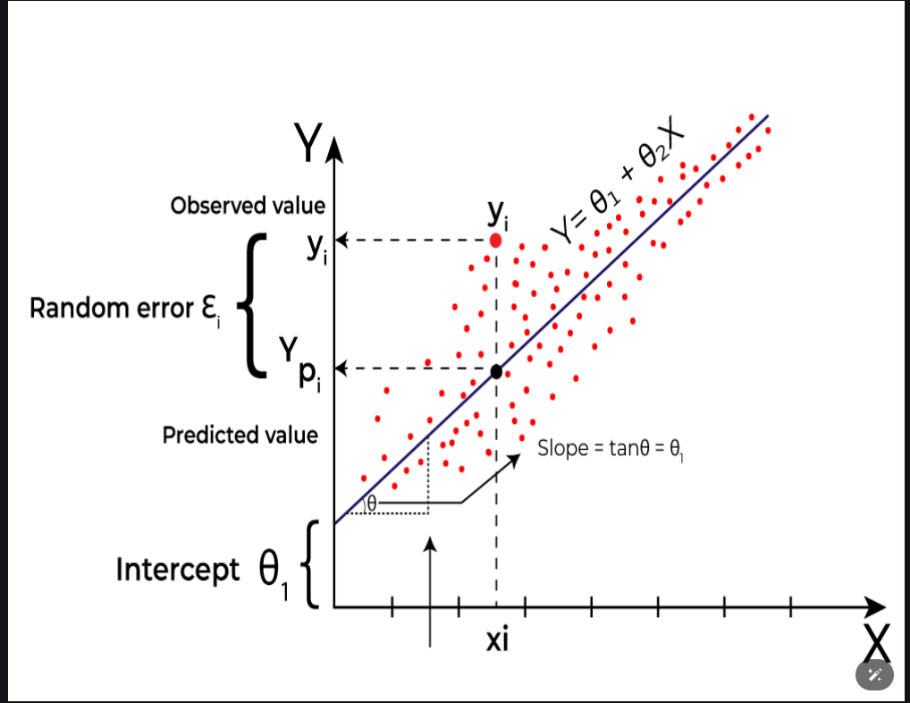


Figure 1.General form of Linear Regression

Here Y is called a dependent or target variable and X is called an independent variable also known as the predictor of Y.

There are two types of linear regression they are Simple Linear Regression and Multiple Linear Regression.

i) Simple Linear Regression: This is the simplest form of linear regression, and it involves only one independent variable and one dependent variable. The equation for simple linear regression is:

𝑦=𝛽0+𝛽1𝑋

where:

Y is the dependent variable

X is the independent variable

β0 is the intercept

β1 is the slope

ii) Multiple Linear Regression: This involves more than one independent variable and one dependent variable. The equation for multiple linear regression is:

𝑦=𝛽0+𝛽1𝑋+𝛽2𝑋+………𝛽𝑛𝑋

where:

Y is the dependent variable

X1, X2, …, Xp are the independent variables

β0 is the intercept

β1, β2, …, βn are the slopes

For this project, we have done using Multiple Linear Regression since for the Dependent variable Predicted Ground water level we will be using multiple Independent variables (i.e) Average Rainfall and Average Temperature of each month starting from 2011 to 2023.

Polynomial Regression

Polynomial Regression is a form of regression analysis in which the relationship between the independent variables and dependent variables are modeled in the nth degree polynomial. Polynomial Regression is a special case of Linear Regression where we fit the polynomial equation on the data with a curvilinear relationship between the dependent and independent variable.

Generally, Polynomial Regression is used in the cases where relation between independent variable and Dependent variable is not linear. While simple linear regression models the relationship as a straight line, polynomial regression allows for more flexibility by fitting a polynomial equation to the data. When the relationship between the variables is better represented by a curve rather than a straight line, polynomial regression can capture the non-linear patterns in the data.

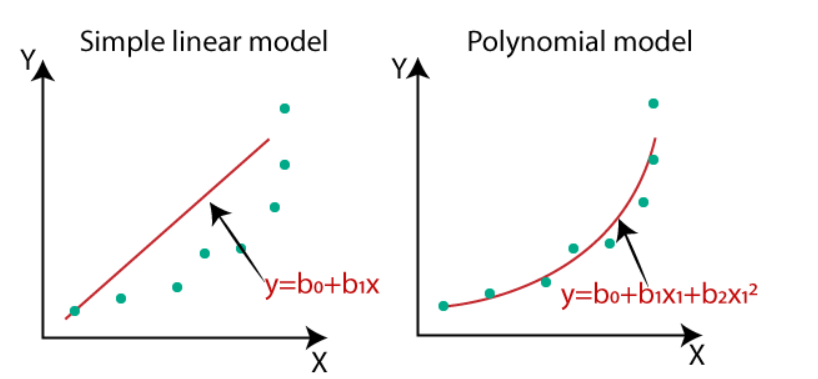


Figure 2. Linear vs Polynomial

This is when polynomial regression model is in the Quadratic form where maximum degree of x is 2.

The general form of a polynomial regression equation of degree n is:



where:

y is the dependent variable.

x is the independent variable.

β0,β1,…,βn are the coefficients of the polynomial terms.

n is the degree of the polynomial.

ϵ represents the error term.

There are mainly three types of Polynomial Regression they are Linear, Quadratic and Cubic Polynomial Regression.

i) Linear Polynomial Regression: Possible when maximum degree of x is 1.

y= β0+ β1x+ ϵ

where:

y is the dependent variable

x is the independent variable

β0 and β1 are the coefficients

ϵ represents the error term.

ii) Quadratic Polynomial Regression: Possible when maximum degree of x is 2.

y= β0+ β1x + β2x2 + ϵ

where:

y is the dependent variable

x is the independent variable

β0, β1 and β2 are the coefficients

ϵ represents the error term.

iii) Cubic Polynomial Regression: Possible when maximum degree of x is 3.

y= β0+ β1x + β2x2 + β3x3 + ϵ

where:

y is the dependent variable

x is the independent variable

β0, β1, β2 and β3 are the coefficients

ϵ represents the error term.

In this project we are using Quadratic Polynomial Regression the degree of the polynomial refers to the highest power of the independent variables that will be included in the model. This is distinct from the number of independent variables in your dataset. The degree of the polynomial (in this case, degree = 2) dictates the highest power of the variables and their combinations that will be included. For degree 2, the polynomial features generated will include:

- All original features: (x1, x2 )

- All squared terms: (x1^2 , x2^2)

- All interaction terms: (x1.x2)

Using degree 2 allows your model to capture more complex relationships than a simple linear model. By including squared and interaction terms, the model can better fit data where the relationship between the independent variables and the dependent variable (y) is not purely linear. This can be especially useful if there are curves in the data that a linear model cannot capture.

Results and Discussion

Here in results part, we will be displaying the Mean Square Error (MSE) and R-square error of each regression model. This is done to evaluate the performance and quality of the model. MSE quantifies the average squared difference between the actual and predicted values, providing a measure of the model's prediction accuracy. R² provides a measure of how well the independent variables explain the variability of the dependent variable.

If your MSE is low, it means that on average the predicted values are close to the actual values and if your R² is high, it means that a large portion of the variability in values is explained by the model, suggesting a good fit.

Linear Regression

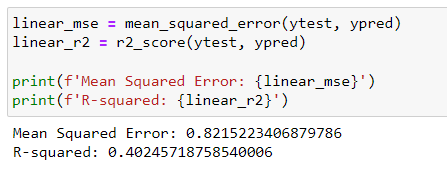


Figure . Result from Linear regression

### Polynomial Regression

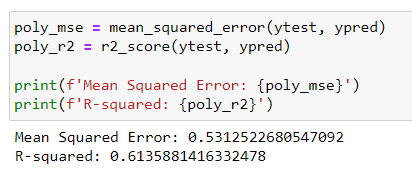


Figure .Result from Polynomial Regression

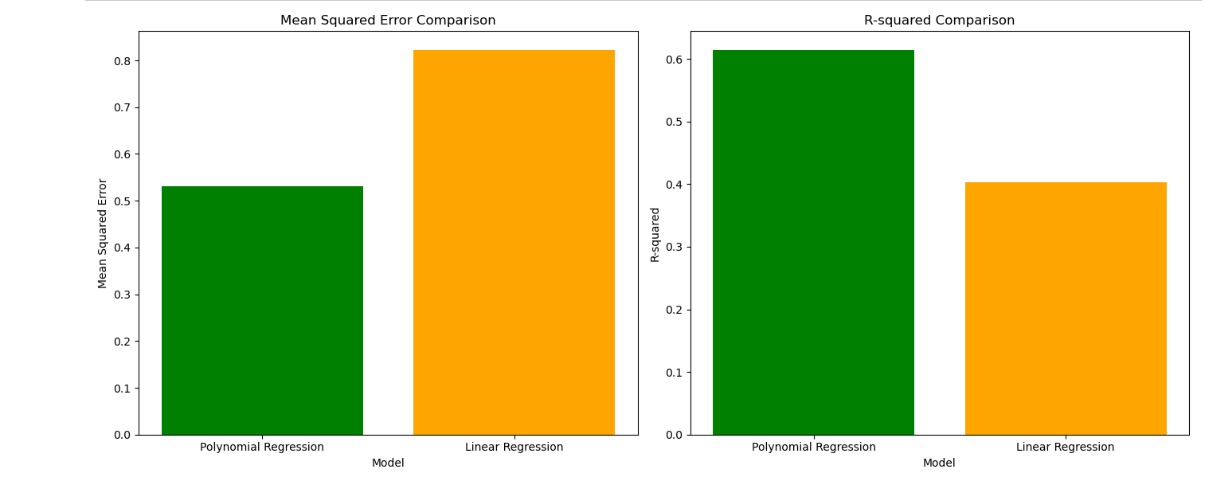


Figure 5. Plot comparison using MSE and R-square

## CONCLUSION:

This project effectively employed Polynomial Regression and Linear Regression to predict groundwater levels in Chennai from January 2011 to December 2023. By sourcing high-quality datasets encompassing historical rainfall records, temperature data, and groundwater levels, we ensured a solid foundation for our analysis. Our use of sophisticated data visualization techniques enabled us to elucidate complex patterns and relationships within the data, making our findings accessible and comprehensible for stakeholders.

Through the application of rigorous evaluation metrics such as R-squared and Mean Squared Error, we meticulously assessed and fine-tuned our models to achieve optimal accuracy and reliability. The comparative analysis of the three machine learning algorithms provided a comprehensive perspective on groundwater dynamics, highlighting the strengths and nuances of each approach.

Ultimately, our project delivers precise and actionable insights into the groundwater levels of Chennai, offering valuable information for sustainable water management and informed decision-making. This work not only contributes significantly to urban water resource management in Chennai but also sets a precedent for addressing similar challenges in metropolitan cities globally.

## FUTURE WORK:

This project lays the groundwork for numerous exciting avenues of expansion, each promising to enhance the depth and breadth of our analysis and its applicability. Future iterations could integrate additional predictors such as land use changes, population growth, industrial water usage, and conservation policies to refine the predictive accuracy of groundwater levels. Moving from monthly to daily or weekly data would capture short-term variations and trends, offering more granular insights into groundwater dynamics. Expanding the scope to include spatial data would allow for a more detailed understanding of groundwater variations across different regions of Chennai, facilitating targeted water management strategies.

Implementing advanced machine learning techniques like Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNNs), or hybrid models could further improve prediction accuracy by capturing complex temporal and spatial dependencies. Developing models to simulate various future scenarios, such as climate change impacts, extreme weather events, or policy interventions, would provide valuable foresight for proactive water resource planning and management. Leveraging Internet of Things (IoT) devices to gather real-time data on groundwater levels, precipitation, and temperature could facilitate dynamic modeling and instantaneous decision-making support. Additionally, creating stakeholder engagement and decision support systems would ensure that the insights derived from our models are effectively utilized for sustainable water management and informed policymaking.

# PART B : MOVIE RECOMMENDATION SYSTEM

## ABSTRACT

In this report, we present a movie recommendation system that employs K-Nearest Neighbors (KNN) and Singular Value Thresholding (SVT) to enhance the accuracy and relevance of recommendations. Movie recommendation systems have become essential in guiding users to find content tailored to their preferences. Our approach integrates collaborative filtering techniques, specifically utilizing KNN for identifying similar users or items, and SVT for matrix completion and noise reduction.

## INTRODUCTION

Recommendations have emerged as a significant element of web-based experiences in the digitalization process, influencing how users engage with products and services on websites like e-commerce websites, video-sharing platforms and social media. These systems try to infer the user’s interests and provide recommendations with items that the user might find appealing or valuable. The project focuses on two prominent techniques used in the development of the recommendation systems: - the Singular Value Thresholding (SVT) and Bayesian Personalized Ranking (BPR)

Singular Value Thresholding (SVT)

Singular Value Thresholding is a mathematical technique based on linear algebra and matrix theory. It is useful in the context of matrix completion problems, where the goal is to fill in the missing entries of a partially observed matrix. In the case of recommendation systems, user-item interaction data is often represented as a sparse matrix, with rows corresponding to users, columns to items, and entries indicating interactions such as ratings. SVT utilizes the principles of singular value decomposition (SVD) to decompose the matrix into its constituent components, and then applies a thresholding operation to retain only the significant singular values. This process helps in reconstructing the original matrix with improved accuracy, thereby enhancing the quality of recommendations. SVT is especially effective in scenarios where explicit feedback, such as user ratings, is available.

Bayesian Personalized Ranking (BPR)

Bayesian Personalized Ranking is an advanced machine learning approach designed for handling implicit feedback data, where user preferences are inferred from indirect actions like clicks, views, or purchases, rather than explicit ratings. BPR optimizes for personalized ranking by focusing on the relative order of items rather than their absolute scores. It employs a pairwise ranking method, where the model is trained to distinguish between pairs of items, ensuring that preferred items are ranked higher than non-preferred ones. The underlying assumption of BPR is that for a given user, the observed interactions are more relevant than the unobserved ones. By utilizing a Bayesian framework, BPR integrates probabilistic assumptions and employs stochastic gradient descent (SGD) for efficient optimization, making it particularly suitable for large-scale recommendation tasks.\

Objectives and Scope

The objective of this project is to explore the mathematical foundations and practical applications of SVT and BPR in building effective recommendation systems. We aim to:

- Understand the theoretical underpinnings of SVT and BPR.

- Implement these techniques using real-world datasets.

- Evaluate their performance in terms of accuracy and computational efficiency.

- Discuss the strengths and limitations of each approach.

By combining mathematical rigor with computational techniques, this project aims to offer a thorough understanding of how SVT and BPR can be employed to enhance recommendation systems, ultimately improving user experience across various domains.

## RECOMMENDATION SYSTEM

Recommendation systems, are a subclass of information filtering systems that seek to predict the preferences or ratings that a user would give to an item. These systems are essential in various domains, such as e-commerce, streaming services, social media, and more, where they enhance user experience by suggesting relevant items. The primary goal of a recommendation system is to filter and present items that are likely to interest the user, thereby improving user satisfaction and engagement.

There are two types of filtering possible in a recommendation system.

1. Collaborative Filtering: This approach relies on past interactions between users and items. It can be further divided into:

- User-based Collaborative Filtering: Recommends items by finding similar users who have similar preferences.

- Item-based Collaborative Filtering: Recommends items that are similar to those the user has liked in the past.

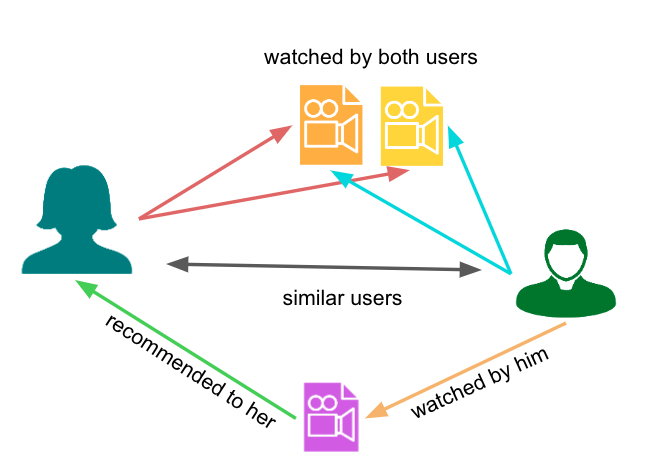


Figure 6.Collaborative Filtering

2. Content-based Filtering: This approach uses the features of items and users to recommend items similar to those the user has liked based on their attributes.

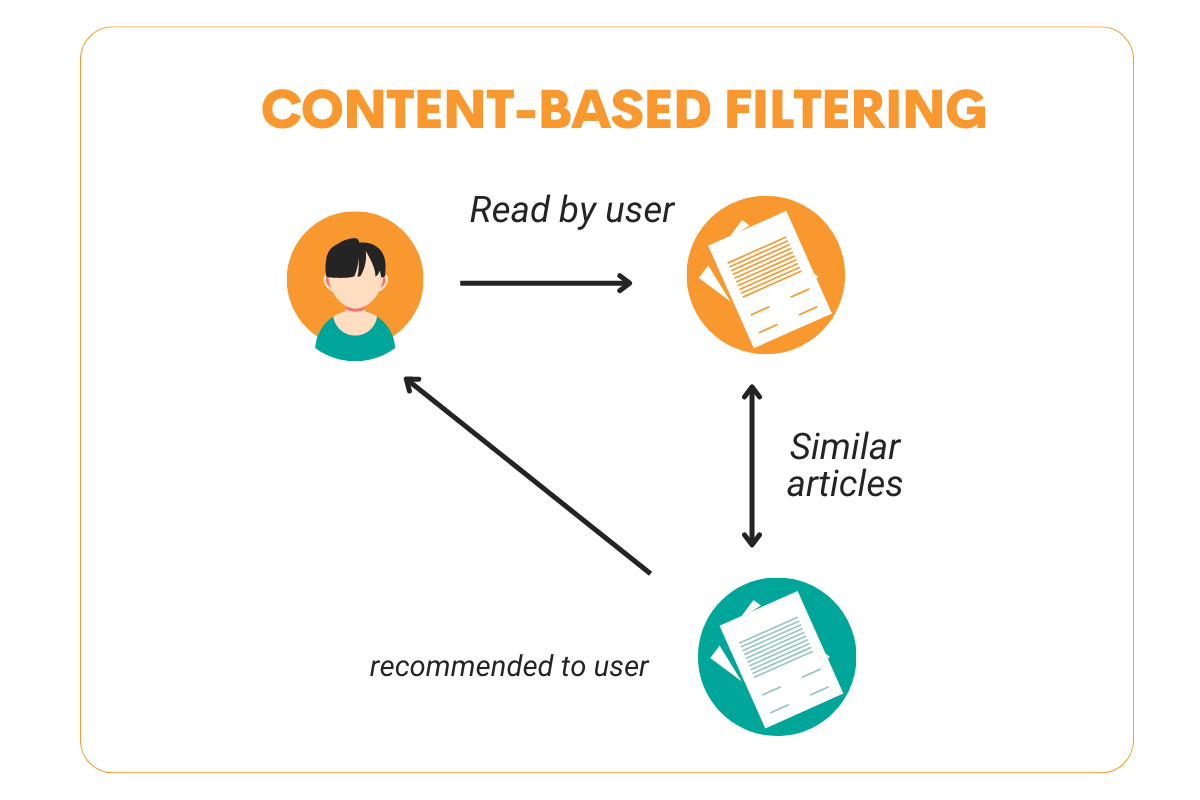


Figure 7.Content-Based Filtering

3. Hybrid Methods: These combine collaborative filtering and content-based filtering to leverage the strengths of both approaches.

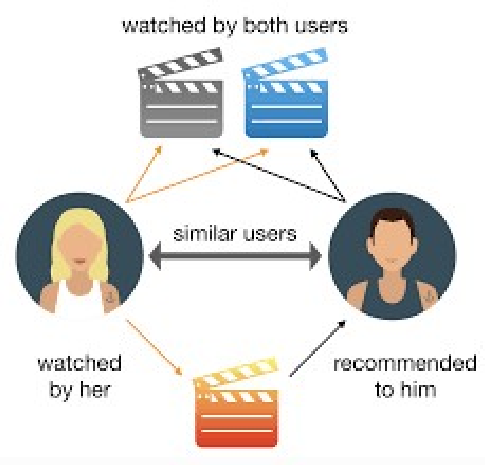


Figure 8. Hybrid Methods of Filtering

Recommendation systems can use explicit feedback, such as user ratings and reviews, or implicit feedback, such as clicks, views, and purchase history, to generate suggestions. Advanced techniques involve machine learning algorithms and matrix factorization methods to improve the accuracy and relevance of recommendations.

In this project, we use two datasets:

(i) Ratings.csv, which contains 1486636 values, 371659 rows and 4 columns of the names: 'userId', 'movieId', 'rating', 'timestamp'.

(ii) Movies.csv which contains 37260 values, 12420 rows, 3 columns of the names: 'movieId', 'title', 'genres'.

### Using SVT

The SVT algorithm is a simple first-order and easy-to-implement algorithm that is efficient at addressing problems in which the optimal solution has low rank. The algorithm is iterative and produces a sequence of matrices {Xk ,Yk} and at each step, it performs a soft-thresholding operation on the singular values of the matrix Yk.

There are two features making this attractive for low-rank matrix completion problems.

The first is that the soft-thresholding operation is applied to a sparse matrix. The soft-thresholding operation is a nonlinear transformation applied to the elements of a matrix or a vector. It is often used in tasks such as denoising or feature selection. The soft-thresholding operation shrinks the magnitude of each element toward zero by a certain threshold λ, while preserving the sign of the element. If the magnitude of an element is less than λ, it is set to zero. Here, this operation is being applied to each non-zero element of the matrix, effectively shrinking the non-zero elements toward zero if their magnitude is below a certain threshold.

The second is that the rank of the iterates {Xk} is empirically nondecreasing; that is as the iterations progress, the rank of the matrices in the sequence {Xk} tends to increase or stay the same, but never decreases. Both these facts allow the algorithm to make use of very minimal storage space and keep the computational cost of each iteration low.

#### Mathematical Concepts Involved

The key concepts behind SVT includes:

* **Singular Value Decomposition (SVD):**

SVD factorises a matrix Y into three matrices Y = UΣVT, where,

U : m X r orthogonal matrix

Σ : r X r diagonal matrix with non-negative real numbers(singular values) in the diagonals.

V : n X r orthogonal matrix.

R : Rank of the matrix.

* **Low Rank Matrix Approximation:**

Given a matrix M, we have to find a low rank approximation X that closely matches M at the observed entries. This is formulated as

minXrank(X) subjected to PΩ(X) = PΩ(M), where ||X||\* is the nuclear norm (sum of singular values) of X.

* **Singular Value Thresholding:**

SVT is an iterative approach to solve the nuclear norm minimization problem. The idea is to apply a soft-thresholding operator to the singular values of the matrix. The thresholding operator Dτ applied to a matrix Y can be defined as

Dτ(Y) = UΣτVT, where, Στ is obtained by applying soft thresholding to the singular values σi of Σ.

Στ = diag(max(σi – τ,0))

#### Mathematical Formulas Involved

* Projection Operator:

PΩ(M) keeps the entries in Ω and zeroes out the others

* SVD:

Y = UΣVT

* Soft Thresholding:

Dτ(Y) = U diag(max(σi – τ,0))VT

* Frobenius Norm:

||A||F =

#### Algorithm in Detail

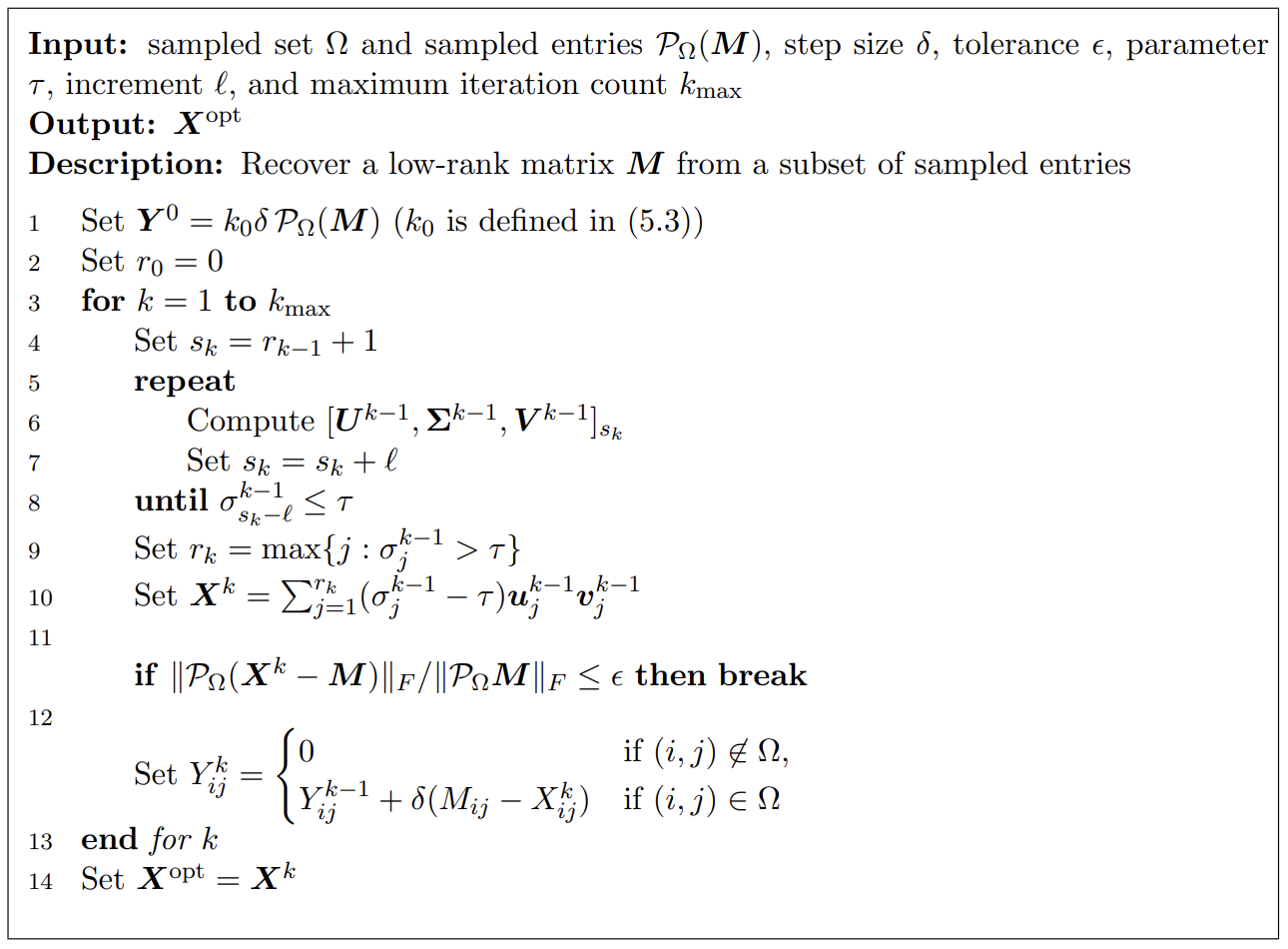


Figure 9. SVT Algorithm

Let us use the iterative SVT algorithm for recovering a low-rank matrix M from a subset of its sampled entries.

Input Parameters:

1. Ω : Set of indices where the entries of matrix M are sampled.
2. PΩ(M) : Matrix containing the sampled entries of M at indices Ω.
3. δ : Step size
4. ε : Tolerance
5. τ : Parameter for singular value thresholding
6. *l* : Increment for rank
7. *kmax* :Maximum number of iterations.

Output Parameters:

1. Xopt : The estimated low rank matrix

Initialization:

1. Y0 = k0 δ PΩ(M) : Initialize Y0 with a scaled version of the sampled matrix.
2. R0 = 0 : Initialize the rank r0.

Iterative Process:

1. For each iteration k from 1 to kmax:
   * 1. Set sk = rk-1 + 1 : Increment the rank by 1.
     2. Repeat the loop:

Compute the partial SVD of Yk-1:[Uk-1, Σk-1, Vk-1]sk.

Increment the rank sk = sk + *l.*

* + 1. Until the smallest singular value σk-1sk-*l* <= τ.
    2. Update rk to the largest index where σk-1j > τ.

Matrix Approximation:

1. Compute Xk as the sum of the significant singular values and their corresponding singular vectors : Xk = Σrkj=1(σk-1j – τ) uk-1j(vk-1j)T.

Convergence check:

1. If the normalized Frobenius norm difference between the sampled entries of Xk and M is less than or equal to ε, break the loop.

Update Y:

1. Update Y based on whether the indices are in Ω or not:
   * 1. If (i, j) ∉ Ω, set Ykij = 0.
     2. If (i, j) ∈ Ω, update Ykij using δ and the differences between Mij and Xkij

Final Output:

1. Set Xopt = Xk.

#### SVT Implementation in Python (Code and Result)

import numpy as np

# Function for SVT matrix completion with given dimensions and rank

def svt\_matrix\_completion(m, n, r, X, mask, tau=1.0, delta=1.0, max\_iter=100, tol=1e-4):

    Y = np.zeros((m, n))

    for i in range(max\_iter):

        U, S, Vt = np.linalg.svd(Y, full\_matrices=False)

        S\_threshold = np.maximum(S - tau, 0)[:r]  # Only keep top r singular values

        X\_hat = np.dot(U[:, :r], np.dot(np.diag(S\_threshold), Vt[:r, :]))

        Y = Y + delta \* (mask \* (X - X\_hat))

        if np.linalg.norm(mask \* (X - X\_hat), 'fro') < tol:

            break

    return X\_hat

# Example usage with the provided conditions

m = 10  # Number of rows

n = 10  # Number of columns

r = 5   # Rank of the low-rank matrix

# Generate example data

X = np.random.randn(m, n)  # Example original matrix

mask = np.random.randint(0, 2, size=(m, n)).astype(bool)  # Example mask matrix

print("Original Matrix:")

print(X)

# Print rank of original matrix

print("Rank of Original Matrix:", np.linalg.matrix\_rank(X))

# Perform SVT matrix completion

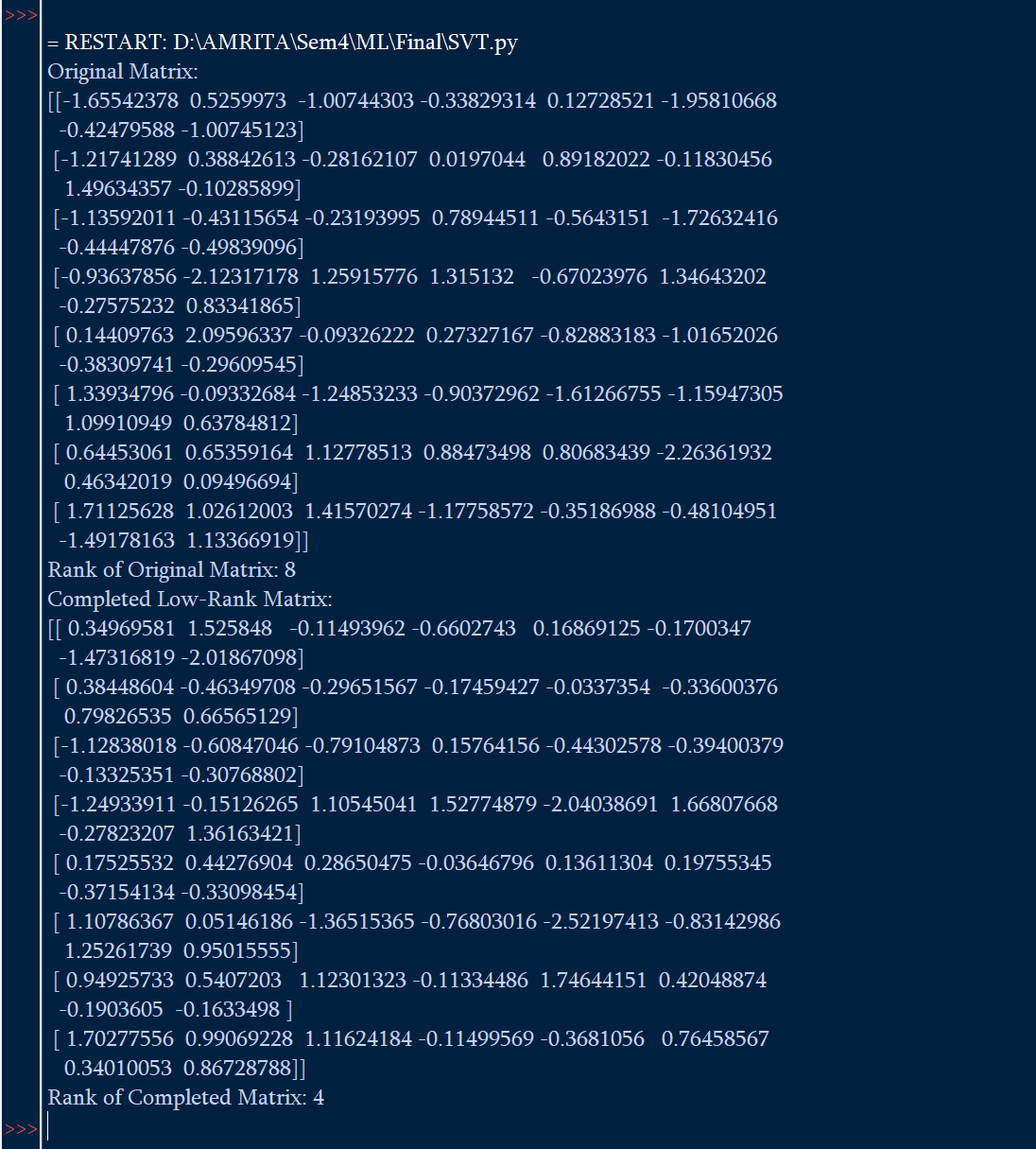
X\_completed = svt\_matrix\_completion(m, n, r, X, mask)

print("Completed Low-Rank Matrix:")

print(X\_completed)

# Print rank of completed matrix

print("Rank of Completed Matrix:", np.linalg.matrix\_rank(X\_completed))



### KNN

K-Nearest Neighbours is an instance-based machine learning algorithm that can be used for both classification and regression tasks. It is a type of lazy learning where the function is only approximated locally and all computations are deferred until function evaluation.

1. Instance-Based Learning: KNN does not build an explicit model. Instead, it memorizes the training dataset, and computations are done at the time of prediction.
2. Lazy Learning: It doesn't learn a discriminative function from the training data but memorizes the training dataset instead.
3. Non-Parametric: It makes no assumptions about the underlying data distribution.

How KNN Works?

1. Choose the number of K neighbours: Determine the number 𝐾*K* of nearest neighbours to consider.
2. Compute Distance: For a given test point, compute the distance between the test point and all the training points. Common distance metrics include Euclidean, Manhattan, and Minkowski distances.
3. Identify Neighbour**s**: Select the 𝐾*K* training points that are closest to the test point.
4. Vote or Average: For classification, each of the 𝐾*K* neighbours votes for their class, and the majority class is assigned to the test point. For regression, the average of the 𝐾*K* nearest neighbours’ values is assigned to the test point.

Distance Metrics

1. Euclidean Distance:

D(p,q) =

2. Manhattan Distance:

D(p,q) =

For Classification,

1. Compute Distance: Calculate the distance between the test sample and all training samples.

2. Sort Neighbours: Sort the computed distances in ascending order.

3. Select K Neighbours: Select the first K samples from the sorted list.

4. Vote: Each of the K samples votes for their class. The class with the most votes is assigned to the test sample.

For Regression,

1. Compute Distance: Calculate the distance between the test sample and all training samples.

2. Sort Neighbours: Sort the computed distances in ascending order.

3. Select K Neighbours: Select the first K samples from the sorted list.

4. Average: Compute the average of the values of the K nearest neighbours and assign this average to the test sample.

Choosing the Value of K

The value of 𝐾*K* is crucial and can affect the performance of the algorithm:

* Small K: Can lead to overfitting. The model captures noise in the data.
* Large K: Can lead to underfitting. The model becomes too smooth and misses important patterns.

Advantages and Disadvantages

Advantages:

1. Simplicity: Easy to understand and implement.
2. No Training Phase: Since it's a lazy learner, there's no explicit training phase.
3. Versatility: Can be used for both classification and regression tasks.

Disadvantages:

1. Computationally Intensive: Especially for large datasets, as it requires calculating the distance of each test sample to all training samples.
2. Storage Intensive: Needs to store the entire training dataset.
3. Sensitive to Irrelevant Features: Performance can degrade with noisy or irrelevant features.
4. Choice of K: The performance heavily depends on the choice of 𝐾*K*.

## CODE

### Recommendation using SVT

import pandas as pd

import numpy as np

import tkinter as tk

from tkinter import ttk

import time

from sklearn.neighbors import NearestNeighbors

# Load movies and ratings data

movies\_file = 'D:/AMRITA/Sem4/ML/Final/movies.csv'

ratings\_file = 'D:/AMRITA/Sem4/ML/Final/ratings.csv'

movies\_df = pd.read\_csv(movies\_file)

ratings\_df = pd.read\_csv(ratings\_file)

# Prepare the data

data = {

    'user\_id': list(ratings\_df['userId']),

    'movie\_id': list(ratings\_df['movieId']),

    'rating': list(ratings\_df['rating'])

}

df = pd.DataFrame(data)

# Convert the dataframe to a user-item matrix

user\_item\_matrix = df.pivot\_table(index='user\_id', columns='movie\_id', values='rating').fillna(0)

# Mask for observed entries

mask = user\_item\_matrix > 0

# Matrix completion with SVT

def svt\_matrix\_completion(X, mask, tau=1.0, delta=1.0, max\_iter=5, tol=1e-4):

    m, n = X.shape

    Y = np.zeros((m, n))

    for i in range(max\_iter):

        U, S, Vt = np.linalg.svd(Y, full\_matrices=False)

        S\_threshold = np.maximum(S - tau, 0)

        X\_hat = np.dot(U, np.dot(np.diag(S\_threshold), Vt))

        Y = Y + delta \* (mask \* (X - X\_hat))

        if np.linalg.norm(mask \* (X - X\_hat), 'fro') < tol:

            break

    return X\_hat

X = user\_item\_matrix.values

# Measure runtime of SVT matrix completion

start\_time\_svt = time.time()

X\_completed = svt\_matrix\_completion(X, mask.values)

end\_time\_svt = time.time()

svt\_runtime = end\_time\_svt - start\_time\_svt

completed\_matrix = pd.DataFrame(X\_completed, columns=user\_item\_matrix.columns, index=user\_item\_matrix.index)

# Fit kNN model on completed\_matrix

k = 5  # Number of neighbors

knn\_model = NearestNeighbors(n\_neighbors=k, metric='cosine')

knn\_model.fit(completed\_matrix)

# Function to get k nearest neighbors for a given user

def get\_k\_nearest\_neighbors(user\_id):

    user\_ratings = completed\_matrix.loc[user\_id].values.reshape(1, -1)

    distances, indices = knn\_model.kneighbors(user\_ratings)

    return distances, indices

# Get recommendations from k nearest neighbors

def get\_recommendations\_from\_neighbors(indices, num\_recommendations):

    recommendations = []

    for neighbor\_index in indices.flatten():

        neighbor\_ratings = completed\_matrix.iloc[neighbor\_index]

        top\_recommendations = neighbor\_ratings.sort\_values(ascending=False).index[:num\_recommendations]

        recommendations.extend(top\_recommendations)

    return list(pd.Series(recommendations).unique())[:num\_recommendations]

def recommend\_movies():

    user\_id = int(user\_id\_entry.get())

    # Measure runtime of kNN recommendation

    start\_time\_knn = time.time()

    distances, indices = get\_k\_nearest\_neighbors(user\_id)

    recommended\_movie\_ids = get\_recommendations\_from\_neighbors(indices, 5)

    end\_time\_knn = time.time()

    knn\_runtime = end\_time\_knn - start\_time\_knn

    # Map movie IDs to movie titles

    recommended\_movies = movies\_df[movies\_df['movieId'].isin(recommended\_movie\_ids)]

    recommended\_movie\_titles = recommended\_movies['title'].tolist()

    # Display results

    result\_text = f"Top 5 Recommended Movies:\n"

    for title in recommended\_movie\_titles:

        result\_text += f"{title}\n"

    result\_text += f"\nSVT Runtime: {svt\_runtime:.2f} seconds\n"

    result\_text += f"kNN Recommendation Runtime: {knn\_runtime:.2f} seconds"

    result\_label.config(text=result\_text)

# Create the main window

root = tk.Tk()

root.title("Movie Recommendation System")

# Create and place the components

ttk.Label(root, text="Enter User ID:", font=("Helvetica", 12, "bold")).grid(row=0, column=0, padx=10, pady=10)

user\_id\_entry = ttk.Entry(root, font=("Helvetica", 12))

user\_id\_entry.grid(row=0, column=1, padx=10, pady=10)

recommend\_button = ttk.Button(root, text="Recommend Movies", command=recommend\_movies, style="Big.TButton")

recommend\_button.grid(row=1, column=0, columnspan=2, padx=10, pady=10)

result\_label = ttk.Label(root, text="", font=("Helvetica", 12))

result\_label.grid(row=2, column=0, columnspan=2, padx=10, pady=10)

# Customize button style

style = ttk.Style()

style.configure("Big.TButton", font=("Helvetica", 12, "bold"))

# Start the main event loop

root.mainloop()

## Result

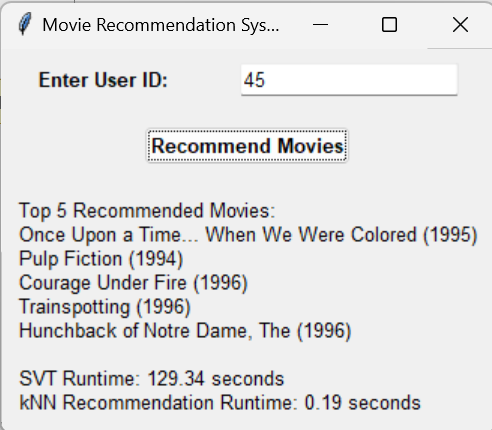


Figure 10. SVT Recommendation Output

### CONCLUSION

Using single value thresholding in recommendation systems helps identify and filter items that meet or exceed a certain relevance score, simplifying decision-making by categorizing items as relevant or not. The K-Nearest Neighbors (KNN) enhances recommendation quality by considering the preferences or ratings of the closest similar users or items, thus providing personalized recommendations based on the collective patterns of nearest neighbors. Combining these methods leverages the simplicity of thresholding to ensure basic relevance, while KNN refines and personalizes recommendations, improving overall user satisfaction and engagement by balancing computational efficiency with tailored accuracy.

### FUTURE WORK

For future work, exploring hybrid models that combine KNN with other recommendation techniques like collaborative filtering or matrix factorization could enhance recommendation accuracy and diversity. Developing adaptive thresholding mechanisms and incorporating contextual information will provide more personalized recommendations. Addressing scalability through optimization strategies, such as approximate nearest neighbor search, will make the system more efficient for large datasets. Additionally, integrating user feedback continuously, enhancing the explain ability of recommendations, and balancing accuracy with diversity and novelty will further improve user satisfaction and engagement.

## LITERATURE SURVEY

|  |  |  |  |
| --- | --- | --- | --- |
| TITLE | INFERENCE | AUTHOR | PUBLISHER |
| Clustering collaborative filtering recommendation system based on SVD algorithm | Use of SVD algorithm in recommendation systems for clustering user based on similarity | Qilong Ba, Xiaoyong Li, Zhongying Bai | 2018 9th International Conference on Information and Communication Systems (ICICS) |
| Fast Matrix-Completion-Based Approach for Recommendation Systems | Usage of probability completion model to fill in empty spaces and obtain a complete matrix same probability distribution as the original matrix | Meng Qiao, Zheng Shan, Fudong Liu, Xingwei Li, And Wenjie Sun | Cornel University |
| A review on matrix completion for recommender systems | Different matrix completion techniques and its pros and cons | Zhaoliang Chen, Shiping Wang | Springer Link |
| Recommendation Systems Explained: Understanding the Basics | Understanding what is recommendation systems | Francesco Casalegno | Website : Medium |
| Linear Algebra in Recommendation Systems of Machine Learning | Maths involved in recommendation system | Parth Shukla | Website : Pythonkitchen |
| Recommender System — Matrix Factorization | Use of matrix factorization in recommender systems | Denise Chen | Website : Medium |
| [Groundwater level prediction using machine learning models: A comprehensive review](https://www.sciencedirect.com/science/article/pii/S092523122200282X) | By reading and going through the paper we were able to infer the basic parameters for predicting the groundwater levels | Hai Tao , Mohammed Majeed Hameed Haydar Abdulameer Marhoon e f, Mohammad Zounemat-Kermani g, Salim Heddam. | Elsevier |
| [Groundwater-level prediction using multiple linear regression and artificial neural network techniques: a comparative assessment](https://srv2.freepaper.me/n/47S4H_EjH6R-jHTuNNqLzg/PDF/d8/d873d5c60adcd324432dc98fa533f6a7.pdf) | Got clearer insights on it upon reading how regression models can be incorporated into this. | Sasmita Sahoo & Madan K. Jha | Hydrogeology Journal |
| https://opencity.in | Got the Rainfall data from this website | Nikhil VJ |  |