**SCTR’s Pune Institute of Computer Technology Dhankawadi, Pune**



**PROJECT REPORT ON**

MOVIE RECOMMENDATION SYSTEM

**SUBMITTED BY**

Name: Sakshi Rathi Class: TE 3

Roll no: 31359

Name: Vaishnavi Raut

Class: TE

**Under the guidance of**

**Prof. Ratnamala S Paswan**

**DEPARTMENT OF COMPUTER ENGINEERIN**

ACADEMIC YEAR 2022-23



**DEPARTMENT OF COMPUTER ENGINEERING**

**SCTR’s Pune Institute of Computer Technology Dhankawadi, Pune**

**Maharashtra 411043**

**CERTIFICATE**

This is to certify that the SPPU Curriculum-based Project report entitled

**“MOVIE RECOMMENDATION SYSTEM”**

Submitted by

*Sakshi Rathi*

*Seat no: T190054454*

*&*

*Vaishnavi Raut*

*Seat no: T190054455*

has satisfactorily completed the curriculum-based Project under the guidance of *Prof* Ratnamala S Paswantowards the partial fulfillment of third year Computer Engineering Semester VI, Academic Year 2022-23 of Savitribai Phule Pune University.

**Prof Ratnamala S Paswan Dr. Geetanjali Kale**

Project Guide Head

PICT, Pune of Computer Engineering Department PICT, Pune

Place:

Date:

**ACKNOWLEDGEMENT**

I would like to express my gratitude and appreciation to all those who gave me the opportunity to complete this report. Special thanks to my guide **Prof Ratnamala S Paswan** ma’am whose insightful suggestions and encouragement helped me in successfully completing the Project and in writing this report. I also sincerely thanks for the time spent proofreading and correcting my many mistakes.

Many thanks to the PICT faculty, **Prof Ratnamala S Paswan**, and head of the Computer department for providing me the opportunity to complete Project and present this report by providing all the resources required.

Sakshi Rathi

**Contents**

1. [Title 3](#_TOC_250012)
2. [Introduction 3](#_TOC_250011)
3. [Problem Statement 4](#_TOC_250010)
4. [Objectives and Scope 4](#_TOC_250009)

[4.1 Objectives . . . . . . . . . . . . . . . . . . . . . . . . 4](#_TOC_250008)

[4.2 Scope . . . . . . . . . . . . . . . . . . . . . . . . . . 4](#_TOC_250007)

1. Theory
2. [System Architecture 7](#_TOC_250002)
3. Methodological Details
4. [Outcome/ results of Project 11](#_TOC_250000)
5. Conclusion 13

**List of Figures**

1.TYPES OF MODEL. . . . . . . . . . . . . . . . . . . . . . . 5

2.ACCESS API KEY . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .7

3. ARCHITECTURE. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

# 1 ABSTARCT

Each of us needs entertainment to recharge our spirits and energy in this fast-paced world. Our confidence for work is restored by entertainment, and we work more ardently as a result.

We might view our favourite films or listen to our favourite music to reenergize ourselves. Since finding chosen movies will take more and more time, which one cannot afford to waste, we can use more reliable movie recommendation algorithms to watch good movies online. In this project, collaborative filtering based on content is used with a Support Vector Machine classifier and a genetic algorithm to enhance the quality of a movie recommendation system.

* Our dataset includes various attributes like
* **Title**: Movie Title.
* **Overview**: Abstract of the Movie.
* **Popularity**: Movie popularity rating as per TMDB.
* **Vote\_average**: Votes average out of 10.
* **Vote\_count**: Number of votes from the users.
* **Release\_date**: Date of release of the movie.
* **Keywords**: Keywords for the movie by TMDB in the list.
* **Genres**: Movie Genres in the list.
* **Cast**: Cast of the movie on the list.
* **Crew**: Crew of the movie in the list.

# 2.Title

# MOVIE RECOMMENDATION SYSTEM

# 3.Problem Statement

Develop a movie recommendation model using the scikit-learn library in python.

# 4 Objectives and Scope

## Objectives

* Increasing the Recommendation System's Accuracy
* Enhance the caliber of the movie suggestion system.
* Increasing Scalability.
* Improving the client experience.

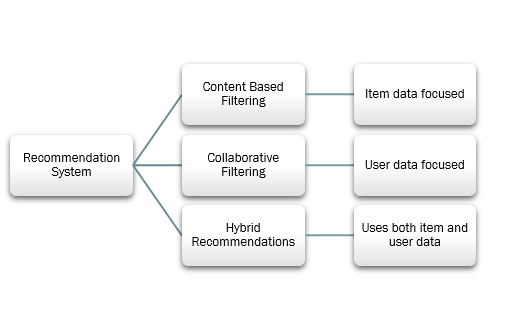
## Scope

The goal of this project is to give people reliable movie suggestions. The project's objective is to make movie recommendation systems better than pure techniques in terms of accuracy, quality, and scalability.

In social networking sites, recommendation systems are employed as information filtering tools to reduce data overload. Therefore, there is a lot of room for research in this area to enhance the quality, accuracy, and scalability of movie recommendation systems. A very effective and crucial mechanism is the movie recommendation system. However, because of the limitations with a pure collaborative method, scalability concerns and poor recommendation quality also affect movie recommendation systems.

# Theory

## Recommendation Models



## Figure 1: Types of Recommendation Models

## Recommendation System

A recommender system is a simple algorithm whose aim is to provide the most relevant information to a user by discovering patterns in a dataset. The algorithm rates the items and shows the user the items that they would rate highly. An example of recommendation in action is when you visit Amazon and you notice that some items are being recommended to you or when Netflix recommends certain movies to you. They are also used by Music streaming applications such as Spotify and Deezer to recommend music that you might like.

**Different types of recommendation engines**

The most common types of recommendation systems are content-based and collaborative filtering recommender systems. In collaborative filtering, the behaviour of a group of users is used to make recommendations to other users. The recommendation is based on the preference of other users. A simple example would be recommending a movie to a user based on the fact that their friend liked the movie. There are two types of collaborative models Memory-based methods and Model-based methods. The advantage of memory-based techniques is that they are simple to implement and the resulting recommendations are often easy to explain. They are divided into two:

## User-based collaborative filtering: In this model, products are recommended to a user based on the fact that the products have been liked by users similar to the user. For example, if Derrick and Dennis like the same movies and a new movie come out that Derick like, then we can recommend that movie to Dennis because Derrick and Dennis seem to like the same movies.

## Item-based collaborative filtering: These systems identify similar items based on users’ previous ratings. For example, if users A, B, and C gave a 5-star rating to books X and Y then when a user D buys book Y they also get a recommendation to purchase book X because the system identifies book X and Y as similar based on the ratings of users A, B, and C.

## Model-based methods are based on Matrix Factorization and are better at dealing with sparsity. They are developed using data mining, machine learning algorithms to predict users’ rating of unrated items. In this approach techniques such as dimensionality reduction are used to improve accuracy. Examples of such model-based methods include Decision trees, Rule-based Model, Bayesian Model, and latent factor models.

## Content-based systems use metadata such as genre, producer, actor, musician to recommend items say movies or music. Such a recommendation would be for instance recommending Infinity War that featured Vin Diesel because someone watched and liked The Fate of the Furious. Similarly, you can get music recommendations from certain artists because you liked their music. Content-based systems are based on the idea that if you liked a certain item you are most likely to like something that is similar to it.

## Python libraries: For the computation and analysis we need certain python libraries which are used to perform analytics. Packages such as SKlearn, Numpy, pandas, Matplotlib, Flask framework, etc are needed.

## SKlearn: It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. NumPy: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.

## Pandas: Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Data frame.

## Flask: It is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug.

## Methodology

# Building the software

This movie recommendation system will be built by following these steps:

1. Install libraries
2. Download dataset
3. Pre-process dataset
4. Encode data
5. Perform vector search

## To create the movie recommendation system we have taken data from [TMDB API](https://www.themoviedb.org/settings/api1).

## 

**Reading Movies Data:**

**data=pd.read\_csv('tmdb.csv.zip',compression='zip',index\_col='id') data.head()**

A screenshot of a computer

Description automatically generated with medium confidence

**Cleaning Data**

As before applying any machine learning models or even exploring the data we need to clean the data:

**DEFINING IMPORTANT FUNCTIONS:**

From entire Crew ,we consider only Director:

def changeforCrew(obj):

    lst = []

    for i in ast.literal\_eval(obj):

        if i['job'] == 'Director':

            lst.append(i['name'])

            break

    return lst

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers.

def stemm(obj):

    lst = []

    for i in obj.split():

        lst.append(ps.stem(i))

    return ' '.join(lst)

Making new "tag" column in Dataframe

data['tag'] = data['overview'] + data['genres'] + data['keywords'] + data['cast'] + data['crew']

Data Modification

data['tag'] = data['tag'].apply(lambda x: ' '.join(word for word in x.split() if word not in swords))

ps = PorterStemmer()

data['tag'] = data['tag'].apply(stemm)

**Vectorization:**

from sklearn.feature\_extraction.text import CountVectorizer

cv = CountVectorizer(max\_features=5500, stop\_words='english')

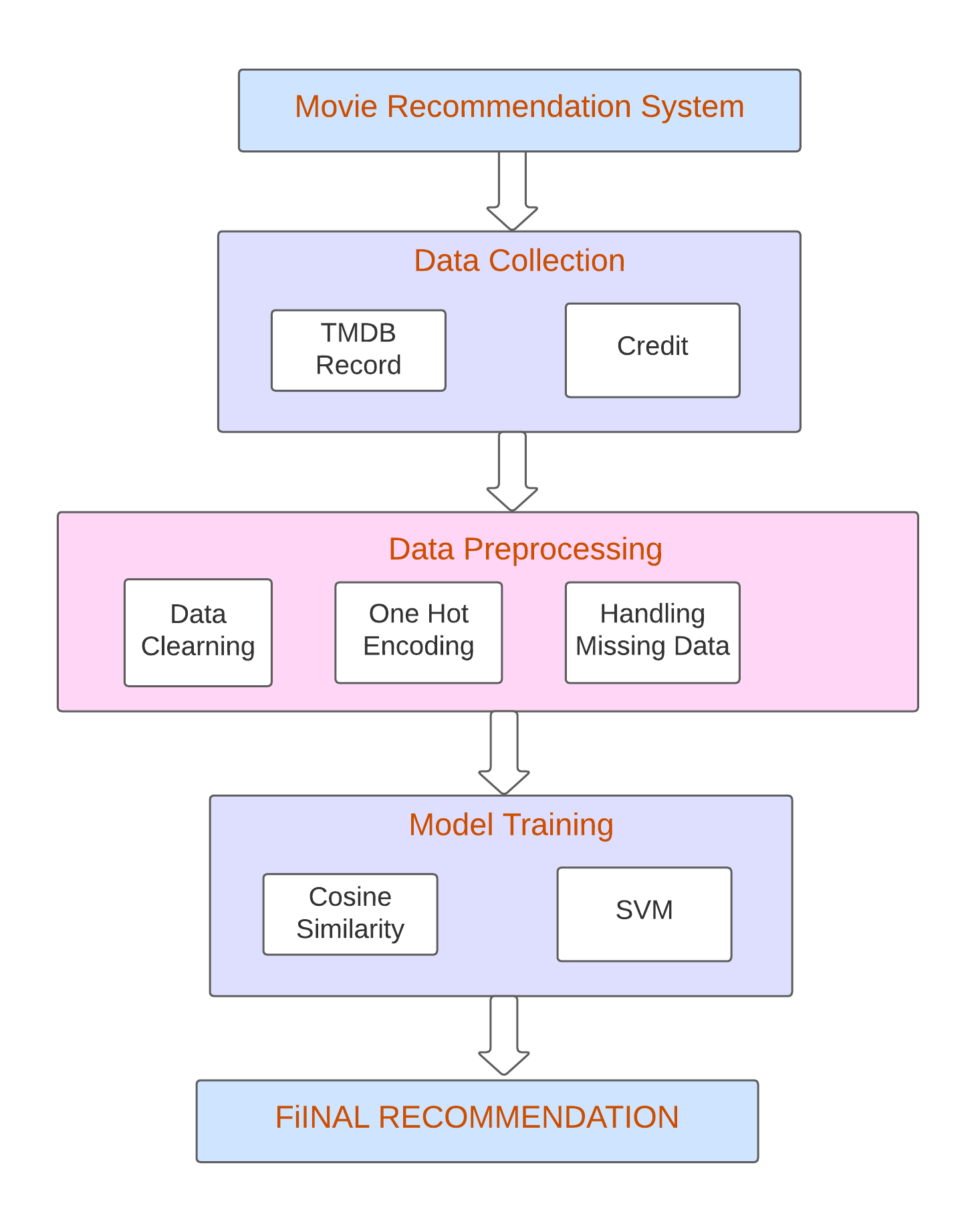
vectors = cv.fit\_transform(data['tag']).toarray()

cv.get\_feature\_names\_out()

from sklearn.metrics.pairwise import cosine\_similarity

similarity = cosine\_similarity(vectors)

**ARCHITECTURE**



**a**,+

(("(%-#!

./ !/

!.

'!./

("(%

**MODEL IMAGES:**

**A picture containing text, screenshot

Description automatically generated**

**A screenshot of a movie

Description automatically generated with low confidence**

**A screenshot of a computer

Description automatically generated with low confidence**

**CONCLUSION:**

In this project, a Hybrid approach is presented in the proposed methodology to improve the accuracy, quality, and scalability of movie recommendation systems by unifying content-based filtering and collaborative filtering; using Singular Value Decomposition (SVD) as a classifier and Cosine Similarity. Existing pure techniques and the suggested hybrid approach are tested on three different Movie datasets, and the results are compared. Furthermore, the proposed solution takes less time to compute than the other two pure alternatives.

**RESULT :**

In the proposed method, It has examined movie genres, but in the future, we may also include user age because movie tastes change with age, for example, during our youth, we favour cartoon movies over other movies. In the future, there is a need to work on the memory requirements of the proposed strategy. The proposed method has only been tested on several movie datasets. It may also be implemented on the Film Affinity and Netflix databases in the future, and the performance can be estimated.