MINI PROJECT

On

"Movie Recommendation System"

Submitted in partial fulfillment of

The requirement for the degree of

COMPUTER ENGINEERING

SUBMITTED BY:

Shanzey Adil Shaikh

UNDER THE GUIDENCE OF:

Prof. Bhosale Swati

DEPARTMENT OF COMPUTER ENGINEERING

HSBPVT's FACULTY OF ENGINEERING, KASHTI

(Approved by AICTE &Affiliated to Savitribai Phule Pune University)

Year of Submission:2023-2024

DEPARTMENT OF COMPUTER ENGINEERING
HSBPVT'S FACULTY OF ENGINEERING, KASHTI.
(Approved by AICTE &Affiliated to Savitribai Phule Pune University)

Year of Submission:2023-2024

CERTIFICATE

This is to certify that **Shaikh Shanzey Adil** from **Third Year Computer Engineering** has successfully completed her mini project report titled **on "Movie Recommendation System"** at Parikrama Faculty of Engineering, Kashti in the partial fulfillment of the Bachelor's Degree in Engineering of Savitribai Phule Pune University.

Prof. Bhosale Swati	Prof. Hiranwale S.B	Principal
Guide Name	Head of the Department	

ACKNOWLEDGEMENT:

As I write this acknowledgement, I must clarify that this is not just a formal acknowledgement but also a sincere note of thanks and regard from my side. I feel a deep sense of gratitude and affection for those who were associated with this course. Without theirs-operation and guidance this report could not have been conducted properly.

A student of a university will have to write the acknowledgement for their project or research paper stating that the submission is done and is not copied. The acknowledgement sample for the university project is mostly attached after the dedication page thanking all the faculty members of the department, HOD, the Dean and the mentor.

I would like to express my profound gratitude to Mr. Prof. Sachin Hiranwale (name of the HOD), of Computer (designation and department name) department, and Mr. Prof. Mahadik(Dean) of Savitri Bhai Phule Pune university for their contributions to the completion of my project titled Leadership and Personality development.

I would like to express my special thanks to our mentor Mrs. Prof. Bhosale Swati N. for her time and efforts she provided throughout the year. Your useful advice and suggestions were really helpful to me during the project's completion. In this aspect, I am eternally grateful to you.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

Signature

Your name

ABSTRACT

Movie recommender systems are meant to give suggestions to the users based on the features they love the most. A highly performing movie recommendation will suggest movies that match the similarities with the highest degree of performance. This study conducts a systematic literature review on movie recommender systems. It highlights the filtering criteria in the recommender systems, algorithms implemented in movie recommender systems, the performance measurement criteria, the challenges in implementation, and recommendations for future research.

Some of the most popular machine learning algorithms used in movie recommender systems such as *K*-means clustering, principal component analysis, and self-organizing maps with principal component analysis are discussed in detail. Special emphasis is given to research works performed using metaheuristic-based recommendation systems. The research aims to bring to light the advances made in developing the movie recommender systems, and what needs to be performed to reduce the current challenges in implementing the feasible solutions. The article will be helpful to researchers in the broad area of recommender systems as well as practicing data scientists involved in the implementation of such systems.

Keywords: movie recommender, filtering techniques, performance metrics, *K*-means, metaheuristic

Contents

Movie Recommendation System

1.Problem statement

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

2.Objective

The objective of this recommendation system is to provide satisfactory movie recommendations to users while keeping the system user friendly i.e. by taking minimum input from users. It recommends the movies based on metadata of the movies and past user ratings.

3.Technology used

Machine Learning Library:

•pandas

•numpy

•difflib

•AST

•scikit-learn

4.Requirements:

•Python 3.6

5.Theory

5.1.What is scikit-learn

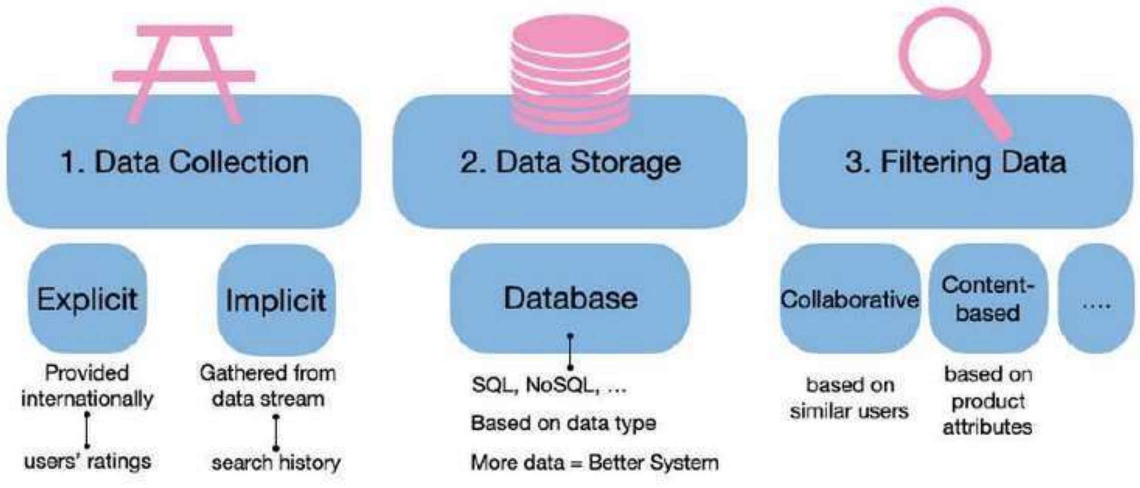
•Clustering: for grouping un labelled data such as K Means.
•Cross Validation: for estimating the performance of supervised models on unseen data.
•Datasets: for test datasets and for generating datasets with specific properties for investigating model behaviour.
•Dimensionality Reduction: for reducing the number of attributes in data for summarization, visualization and feature selection such as Principal component analysis.
•Ensemble methods: for combining the predictions of multiple supervised models.
•Feature extraction: for defining attributes in image and text data.
•Feature selection: for identifying meaningful attributes from which to create supervised models.
•Parameter Tuning: for getting the most out of supervised models.
•Manifold Learning: For summarizing and depicting complex multi-dimensional data.
•Supervised Models: a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees

5.2. What is a Recommendation System?

Simply put a Recommendation System is a filtration program whose prime goal is to predict the "rating" or "preference" of a user towards a domain-specific item or item. In our case, this domain-specific item is a movie, therefore the main focus of our recommendation system is to filter and predict only those movies which a user would prefer given some data about the user him or herself.

5.2.1. Recommendation System Mechanism:

The engine of the recommendation system filters the data via different machine learning algorithms, and based on that filtering, it can predicts the most relevant entities to be recommended. After studying the previous behaviours of the users, it recommends products/services that the used may be interested on



The engine's working of a recommendation is classified in these 3 steps

5.2.1.1. Data Collection

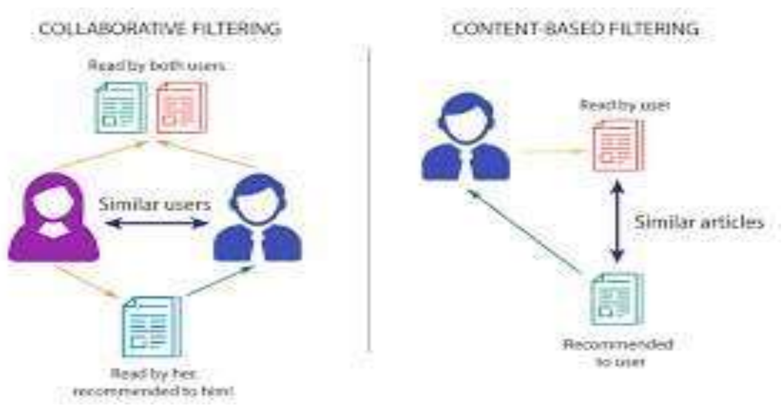
The techniques that can be used to collect data are:

1. Explicit, where data are provided intentionally as an information (e.g. user's input such as movies rating)
2. Implicit, where data are provided intentionally but gathered from available data stream (e.g. search history, clicks, order history, etc…)

5.2.1.2 Data Storage

It can be stored in a cloud storage such as SQL database, NoSQL database, or some other kind of object storage. However, it depends on the data type and amount as well. The more data that the storage can have for the model, the better recommendation system can be.

5.3. What are the different filtration strategies?



5.3.1. Content-based Filtering:

This filtration strategy is based on the data provided about the items. The Algorithm recommends products that are similar to the ones that a user has liked in the past. This similarity (generally cosine similarity) is computed from the data we have about the items as well as the user's past preferences.
For example, if a user likes movies such as 'The Prestige' then we can recommend him the movies of 'Christian Bale' or movies with the genre 'Thriller' or maybe even movies directed by 'Christopher Nolan'. So what happens here the recommendation system checks the past preferences of the user and find the film "The Prestige", then tries to find similar movies to that using the information available in the database such as the lead actors, the director, genre of the film, production house, etc and based on this information find movies similar to "The Prestige".

Disadvantages:
1.Different products do not get much exposure to the user.
2.Businesses cannot be expanded as the user does not try different types of products.

3.2. Collaborative Filtering:

This filtration strategy is based on the combination of the user's behaviour and comparing and contrasting that with other users' behaviour in the database. The history of all users plays an important role in this algorithm. The main difference between content-based filtering and collaborative filtering that in the latter, the interaction of all users with the items influences the recommendation algorithm while for content-based filtering only the concerned user's data is taken into account. There are multiple ways to implement collaborative filtering but the main concept to be grasped is that in collaborative filtering multiple user's data influences the outcome of the recommendation. and doesn't depend on only one user's data for modelling.

Here's a step-by-step guide on how to build a basic movie recommendation model using collaborative filtering with scikit-learn:

1. **Data Preparation**: Load the dataset and preprocess it if needed.
2. **Split Data**: Split the data into training and testing sets.
3. **Model Training**: Train a collaborative filtering model using scikit-learn.
4. **Model Evaluation**: Evaluate the model's performance on the test set.
5. **Recommendation**: Use the trained model to make movie recommendations for a given user.

There are 2 types of collaborative filtering algorithms:

5.3.2.1. User-based Collaborative filtering:

The basic idea here is to find users that have similar past preference patterns as the user 'A' has had and then recommending him or her items liked by those similar users which 'A' has not encountered yet. This is achieved by making a matrix of items each user has rated/viewed/liked/clicked depending upon the task at hand, and then computing the similarity score between the users and finally recommending items that the concerned user isn’t aware of but users similar to him/her are and liked it. For example, if the user ‘A’ likes ‘Batman Begins’, ‘Justice League’ and ‘The Avengers’ while the user ‘B’ likes ‘Batman Begins’, ‘Justice League’ and ‘Thor’ then they have similar interests because we know that these movies belong to the super-hero genre. So, there is a high probability that the user ‘A’ would like ‘Thor’ and the user ‘B’ would like The Avengers’. **Disadvantages:**

1. People are fickle-minded i.e their taste change from time to time and as this algorithm is based on user similarity it may pick up initial similarity patterns between 2 users who after a while may have completely different preferences.
2. There are many more users than items therefore it becomes very difficult to maintain such large matrices and therefore needs to be recomputed very regularly.
3. This algorithm is very susceptible to shilling attacks where fake users profiles consisting of biased preference patterns are used to manipulate key decisions.

5.3.2.2. Item-based Collaborative Filtering:

The concept in this case is to find similar movies instead of similar users and then recommending similar movies to that ‘A’ has had in his/her past preferences. This is executed by finding every pair of items that were rated/viewed/liked/clicked by the same user, then measuring the similarity of those rated/viewed/liked/clicked across all user who rated/viewed/liked/clicked both, and finally recommending them based on similarity scores. Here, for example, we take 2 movies ‘A’ and ‘B’ and check their ratings by all users who have rated both the movies and based on the similarity of these ratings, and based on this rating similarity by users who have rated both we find similar movies. So if most common users have rated ‘A’ and ‘B’ both similarly and it is highly probable that ‘A’ and ‘B’ are similar, therefore if someone has watched and liked ‘A’ they should be recommended ‘B’ and vice versa.

Advantages over User-based Collaborative Filtering :

1. Unlike people’s taste, movies don’t change.
2. There are usually a lot fewer items than people, therefore easier to maintain and compute the matrices.
3. Shilling attacks are much harder because items cannot be faked.

5.4. Data Description:

A data set (or dataset) is a collection of data. In the case of tabular data, a data set corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the data set in question. The data set lists values for each of the variables, such as for example height and weight of an object, for each member of the data set. Data sets can also consist of a collection of documents or files. In the open data discipline, data set is the unit to measure the information released in a public open data repository. The European Open Data portal aggregates more than half a million data sets.Some other issues (real-time data sources,non-relational datasets, etc.) increases the difficulty to reach a consensus about it.

When building a movie recommendation system using the scikit-learn library in Python, the data description might be more focused on the specific data structures and format expected by scikit-learn algorithms. Here's a simplified data description tailored for such a system:

1. **User-Movie Ratings Matrix**:
   - This matrix represents the interactions between users and movies, where each cell contains the rating given by a user to a movie.
   - Rows correspond to users, and columns correspond to movies.
   - Values in the matrix represent user ratings (e.g., 0-5 stars), with zeros indicating that the user has not rated the movie.
   - This matrix is typically sparse since most users won't have rated most movies.
2. **User IDs**:
   - Unique identifiers for each user.
   - Used to index rows in the user-movie ratings matrix.
3. **Movie IDs**:
   - Unique identifiers for each movie.
   - Used to index columns in the user-movie ratings matrix.
4. **Ratings Data**:
   - A dataset containing user ratings for movies. o Each row represents a single rating given by a user to a movie. o Columns typically include user ID, movie ID, and rating.

5. **Feature Vectors** (Optional):
    - In some cases, additional features about users or movies may be used to enhance recommendations. o For example, features could include user demographics, movie genres, release year, etc.
    - These features are represented as vectors and can be used in content-based filtering or hybrid approaches.
6. **Train-Test Split**:
    - The ratings data is typically split into training and testing sets. o The training set is used to train the recommendation model, while the testing set is used to evaluate its performance.
7. **Cosine Similarity Matrix**:
    - In collaborative filtering, cosine similarity between users or items may be computed to identify similar users or movies.
    - This matrix contains pairwise cosine similarity scores between users or movies, indicating their similarity in terms of ratings.
8. **Recommendations**:
    - Once the model is trained, it can generate recommendations for users. o Recommendations are typically generated based on the predicted ratings for movies that the user has not yet rated. o These recommendations can be ranked based on predicted ratings or other relevancy scores.

Overall, the data description for a movie recommendation system using scikit-learn focuses on representing user interactions with movies in a structured format suitable for training scikitlearn algorithms, along with any additional features that may be used to improve recommendation quality.

6.Input

```
actual code  import pandas as pd from
sklearn.feature_extraction.text import CountVectorizer from
sklearn.metrics.pairwise import cosine_similarity

# Load the dataset df =
pd.read_csv("movie_dataset.csv")

# Define features to be used for similarity calculation features
= ['keywords', 'cast', 'genres', 'director']

# Function to combine features into a single string def
combine_features(row):
    return row['keywords'] + " " + row['cast'] + " " + row["genres"] + " " + row["director"]

# Fill missing values in features for
feature in features:
    df[feature] = df[feature].fillna('')

# Create a new feature 'combined_features' by combining all selected features
df["combined_features"] = df.apply(combine_features, axis=1)

# Initialize CountVectorizer to convert text data into matrix cv
= CountVectorizer()

# Fit and transform the 'combined_features' column count_matrix
= cv.fit_transform(df["combined_features"])

# Calculate cosine similarity matrix
cosine_sim = cosine_similarity(count_matrix)

# Function to get movie title from its index def
get_title_from_index(index):
    return df[df.index == index]["title"].values[0]

# Function to get movie index from its title def
get_index_from_title(title):
    return df[df.title == title]["index"].values[0]

# Movie to find similar movies for movie_user_likes =
"Avatar" movie_index =
get_index_from_title(movie_user_likes)

# Find similar movies based on cosine similarity similar_movies =
list(enumerate(cosine_sim[movie_index])) sorted_similar_movies =
sorted(similar_movies, key=lambda x: x[1], reverse=True)[1:]

# Print top 5 similar movies print("Top 5 similar movies to '" +
movie_user_likes + "' are:\n") for i, element in
enumerate(sorted_similar_movies[:5]):
    print(i+1, get_title_from_index(element[0]))
```

OUTPUT

```
Top 5 similar movies to Avatar are:
Guardians of the Galaxy
Aliens
Star Wars: Clone Wars: Volume 1
Star Trek Into Darkness
Star Trek Beyond
```

7.Conclusion

Recommendation systems have become an important part of everyone’s lives. With the enormous number of movies releasing worldwide every year, people often miss out on some amazing work of arts due to the lack of correct suggestion. Putting machine learning based Recommendation systems into work is thus very important to get the right recommendations. We saw content-based recommendation systems that although may not seem very effective on its own, but when combined with collaborative techniques can solve the cold start problems that collaborative filtering methods face when run independently.