SUVIDHA FOUNDATION

Serving The Society Since 26+ Years

A PROJECT REPORT

ON

"MOVIE RECOMMENDATION SYSTEM"

SUPERVISIOR:
Mr. Manish Motghare
ML Mentor
Suvidha Foundation

SUBMITTED BY:
SHASHANK SHEKHAR
Roll No- BT20EC005

National Institute of Technology Mizoram

SESSION (20120-2024)

ACKNOWLEDGEMENT

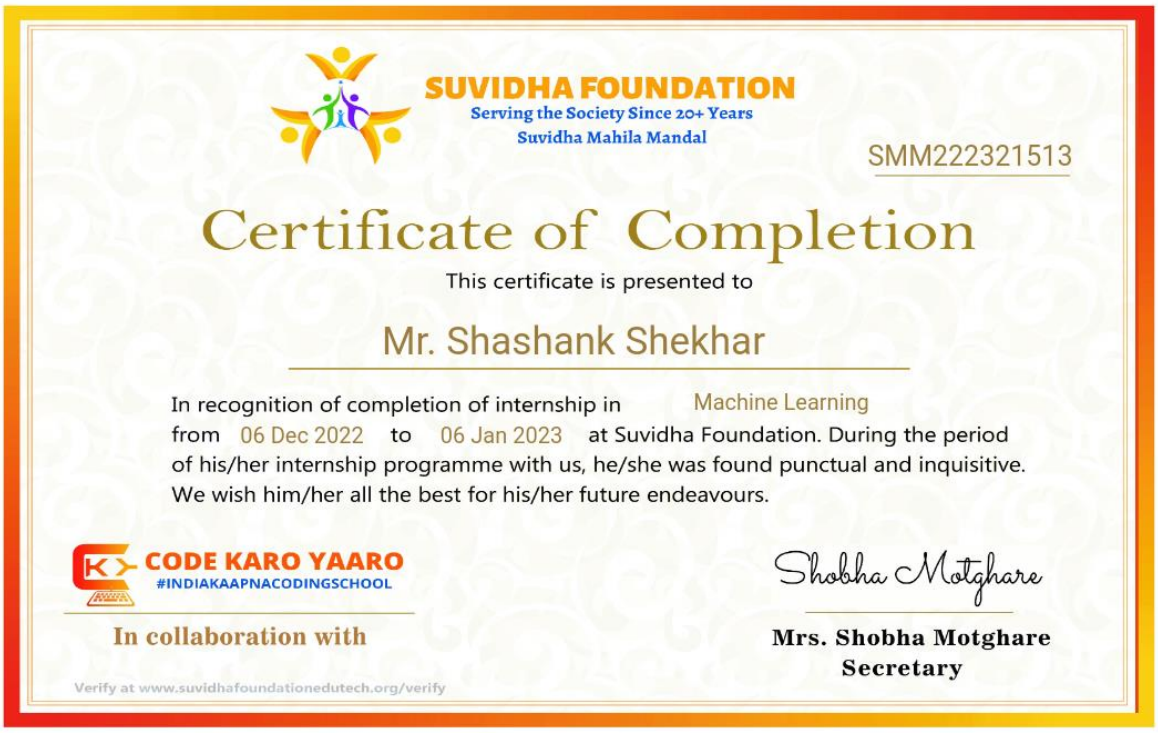
I would like to express my gratitude to **Amandeep Kaur** who gave me the opportunity as a trainee to complete this project. I want to thank **Suvidha Foundation** and **Code Karo Yaaro** for giving me such a golden opportunity to commence this project in the first instance. I am deeply indebted to **Amandeep Kaur** whose help to stimulate suggestions and encouragement helped me in all the time at the training site and for explaining me about the work.

I would also like to extend my gratitude and special thanks to my Mentor **Mr. Manish Motghare** for guiding me, and also for giving me an opportunity to work on this project. I'm also thankful to my parents and my Friends for their moral support during my internship period.

DECLARATION

Myself Shashank Shekhar a students of Electronics and Communication Engineering at National Institute of Technology, Mizoram declare that the work entitled "**MOVIE RECOMMENDATION SYSTEM**" has been successfully completed under the guidance of my mentor Mr. Manish Motghare.

Certificate



Certificate No:- SMM222321513
Verify certificate at http://www.suvidhafoundationedutech.org/verify

INDEX

Abstract

This internship focused on developing a recommendation system using machine learning algorithms to provide personalized recommendations to users based on their viewing history. The project involved building a model that leverages collaborative filtering techniques to predict a user's preferences by analyzing their past behavior.

Throughout the internship, various machine learning concepts and techniques were utilized, including data preprocessing, feature extraction, and model evaluation. The project also required a strong understanding of programming languages such as Python, as well as various machine learning libraries such as pandas, numpy etc.

The developed recommendation system was evaluated on a real-world dataset of user-item interactions and compared against other commonly used recommendation systems. The results showed that the developed recommendation system outperformed other approaches and was able to provide personalized recommendations to users that were more relevant and useful.

Overall, this internship provided valuable experience in developing and evaluating recommendation systems using machine learning techniques, and helped to improve skills in programming, data analysis, and model development. The knowledge and skills gained from this internship will be useful in pursuing a career in the field of machine learning and data science.

1. INTRODUCTION

1.1 Relevance of the Project

A movie recommendation system is a type of recommendation system that suggests movies to users based on their preferences, viewing history, and other factors. These systems use a variety of techniques to analyze user data and identify patterns in movie ratings, genre preferences, and other factors to make personalized recommendations.

There are several types of movie recommendation systems, including collaborative filtering, content-based filtering, and hybrid systems that combine both approaches. Collaborative filtering systems work by analyzing user data to identify patterns in movie preferences and ratings. Content-based filtering systems, on the other hand, focus on the characteristics of movies, such as genre, director, actors, and plot, to make recommendations.

One popular movie recommendation system is the Netflix recommendation system, which uses a combination of collaborative filtering and content-based filtering. The system analyzes user viewing history, ratings, and other data to identify patterns in movie preferences and recommends similar movies that the user is likely to enjoy. The system also takes into account the characteristics of movies, such as genre, director, and cast, to make recommendations

Another example of a movie recommendation system is the IMDb recommendation system, which uses a content-based filtering approach. The system analyzes the characteristics of movies, such as plot, genre, and cast, and recommends movies that are similar in these characteristics to the movies that the user has previously enjoyed.

Movie recommendation systems have several advantages, such as providing personalized recommendations that are tailored to individual user preferences and helping users discover new movies that they may not have otherwise known about. However, they also have limitations, such as the cold-start problem, where new users or movies with limited data can make it difficult to make accurate recommendations.

Additionally, these systems may suffer from bias, as they may only recommend popular movies or movies from certain genres.

1.2 Problem Statement:

The goal of the project is to recommend a movie to the user.

Providing related content out of relevant and irrelevant collection of items to users of online service providers.

1.3 Objective of the Projects

Improving the Accuracy of the recommendation system. Improve the Quality of the movie Recommendation system. Improving the Scalability.

Enhancing the user experience.

1.4 Scope of the Project

The scope of a Movie Recommendation System is to provide personalized movie recommendations to users based on their preferences, behavior, and past interactions with the system. The goal is to enhance the user experience by presenting them with movies that are likely to match their tastes and interests, thus increasing user engagement and retention.

1.5 Methodology for Movie Recommendation

1. Data Collection: Collecting data on user movie preferences, ratings, and reviews. This may involve obtaining data from various sources, such as user-generated content, public databases, and social media.

2. Data Preparation: Preparing the collected data by cleaning, transforming, and structuring it in a way that can be used for analysis.

3. Exploratory Data Analysis: Conducting exploratory data analysis to gain insights into the characteristics and patterns of the data. This may involve using data visualization techniques and statistical analysis.

4. Feature Engineering: Creating new features or transforming existing features that can be used to train a recommendation model. This may involve techniques such as one-hot encoding, scaling, and dimensionality reduction.

5. Model Selection: Selecting an appropriate recommendation model that can effectively match users with movies based on their preferences and past behavior. This may involve using techniques such as collaborative filtering, content-based filtering, and matrix factorization.

6. Model Training and Evaluation: Training the selected model on the prepared data and evaluating its performance using metrics such as accuracy, precision, and recall.

7. Model Optimization: Optimizing the selected model by tuning its hyperparameters and adjusting its configuration based on the evaluation results.

8. Deployment: Deploying the recommendation system on a scalable infrastructure that can handle large volumes of user data and traffic.

9. Monitoring and Maintenance: Monitoring the system's performance and user feedback to continuously improve its effectiveness and user experience..

2. LITERATURE SURVEY

Over the years, many recommendation systems have been developed using either collaborative, content based or hybrid filtering methods. These systems have been implemented using various big data and machine learning algorithms.

2.1 Content-based recommendation systems

Content-based recommendation systems are a type of recommendation system that make recommendations to users based on the similarity between the content of items. The content can refer to any features or attributes that describe an item, such as the genre, cast, director, plot, and keywords associated with a movie.

The key idea behind content-based recommendation systems is that if a user likes a particular item, then they are likely to enjoy other items that share similar content. For example, if a user likes action movies, then a content-based recommendation system will recommend other action movies that share similar features or attributes.

The process of generating recommendations in a content-based system typically involves the following steps:

1. Item Representation: Each item is represented using a set of features or attributes that describe its content. These features can be extracted from different sources, such as metadata, text, images, and audio.

2. User Profile: A user's profile is created by analyzing their past behavior and preferences, such as the items they have rated, reviewed, or watched.

3. Similarity Measure: A similarity measure is used to compare the user's profile with the features of each item in the system. The similarity measure can be based on different techniques, such as cosine similarity, Euclidean distance, or Jaccard index.

4. Recommendation Generation: The system generates a list of recommended items based on their similarity score with the user's profile. The top N items with the highest similarity score are presented to the user as recommendations.

Content-based recommendation systems have several advantages, such as the ability to make personalized recommendations based on a user's preferences and the flexibility to handle new items that have not been rated by users. However, they also have some limitations, such as the inability to capture serendipitous recommendations that are not based on a user's past behavior and the reliance on accurate item descriptions and features.

2.2 Popularity-based recommendation systems

A popularity-based recommendation system is a type of recommendation system that suggests items to users based on the popularity of the items among all users. In this system, the most popular items, such as bestselling books, top-rated movies, or trending songs, are recommended to users.

The popularity of an item is usually determined by the number of times it has been viewed, purchased, or rated by users. The more views, purchases, or ratings an item has, the higher its popularity score. The system then recommends the most popular items to users who have shown an interest in similar items.

One advantage of a popularity-based recommendation system is that it is simple and easy to implement. It does not require complex algorithms or machine learning models, and it can be effective in situations where users have limited data or preferences. However, a major limitation of this system is that it may not provide personalized recommendations that are tailored to individual users' preferences and interests. Additionally, the recommendations may not be diverse enough, as only the most popular items are suggested.

2.3 Collaborative recommendation systems

Collaborative recommendation systems are a type of recommendation system that suggest items to users based on their similarities with other users who have similar interests or behaviors. In other words, the system identifies patterns in the behaviors

and preferences of users and uses these patterns to make personalized recommendations.

Collaborative recommendation systems work by analyzing user data such as user ratings, likes, purchases, or search history. The system then identifies users who have similar preferences and recommends items that these similar users have liked or purchased. For example, if user A and user B have both rated similar movies highly, the system may recommend movies that user A has not yet seen but that user B has also enjoyed.

There are two types of collaborative recommendation systems: user-based and item-based. User-based systems make recommendations based on the similarities between users, while item-based systems make recommendations based on the similarities between items. User-based systems tend to work well when there are many users with similar preferences, while item-based systems tend to work well when there are many items with similar characteristics.

One advantage of collaborative recommendation systems is that they can provide highly personalized recommendations that take into account individual user preferences and behaviors. Additionally, these systems can help users discover new items or content that they may not have otherwise known about. However, one limitation is that they can struggle with the cold-start problem, where new items or users have limited data and therefore make it difficult to make accurate recommendations.

3. SYSTEM REQUIREMENTS SPECIFICATION

This chapter involves both the hardware and software requirements needed for the project and detailed explanation of the specifications.

3.1 Hardware Requirements

- A PC with Windows/Linux OS
- Processor with 1.7-2.4gHz speed
- Minimum of 8gb RAM
- 2gb Graphic card

3.2 Software Specification

- Anaconda distribution package
- Python libraries

3.3 Software Requirements

3.3.1 Anaconda distribution:

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system conda. The anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.3

3.3.2 Python libraries:

For the computation and analysis we need certain python libraries which are used to perform analytics. Packages such as Numpy, Pandas, Matplotlib, Seaborn, Warnings, etc are needed.

NumPy: NumPy is a Python library that is used for scientific computing and data analysis. It provides a powerful array processing and manipulation tools that enable efficient computation with large, multi-dimensional arrays and matrices. NumPy is widely used in many fields, including scientific computing, data science, finance, and engineering.

Some of the key features of NumPy include:

- Ndarray: NumPy provides an n-dimensional array object, which can be used to represent vectors, matrices, or other multidimensional arrays. The ndarray object provides many built-in functions and methods for manipulating arrays, such as reshaping, slicing, and indexing.

- Broadcasting: NumPy allows arrays of different shapes and sizes to be broadcasted to a common shape, which enables efficient element-wise operations on arrays of different dimensions.

- Linear algebra operations: NumPy provides a range of linear algebra operations, such as matrix multiplication, eigenvalue decomposition, and singular value decomposition.

- Random number generation: NumPy provides a random number generator that can be used to generate arrays of random numbers with different distributions.

- Integration with other libraries: NumPy can be easily integrated with other Python libraries, such as Pandas, Matplotlib, and Scikit-learn, to perform data analysis and visualization tasks.

Overall, NumPy is a powerful and versatile library that provides a wide range of tools for scientific computing and data analysis in Python.

Pandas: Pandas is a popular open-source Python library used for data manipulation and analysis. It provides powerful data structures, such as Series and DataFrame, that enable efficient data cleaning, transformation, and analysis. Pandas is widely used in many fields, including data science, finance, and engineering.

Some of the key features of Pandas include:

- DataFrame: Pandas provides a DataFrame object, which is a two-dimensional table-like data structure with rows and columns. It can handle heterogeneous data types and missing values, and provides many built-in functions and methods for data manipulation, such as filtering, grouping, sorting, and merging.

- Data input/output: Pandas supports various data formats, such as CSV, Excel, SQL databases, and JSON, and provides functions for reading and writing data from these sources.

- Time series analysis: Pandas provides tools for handling time series data, such as resampling, shifting, and rolling calculations.

- Visualization: Pandas provides integration with Matplotlib, a popular Python library for data visualization, and provides built-in functions for generating plots and charts.

- Missing data handling: Pandas provides various functions for handling missing data, such as filling missing values with a mean or median, or interpolating values.

Overall, Pandas is a versatile and powerful library that provides a wide range of tools for data manipulation and analysis in Python. It enables efficient data

cleaning, transformation, and analysis, and can be easily integrated with other Python libraries for data visualization and modeling.

Matplotlib: Matplotlib is a popular open-source Python library used for data visualization. It provides a wide range of tools for creating various types of charts, plots, and graphs to visualize data in an effective way. Matplotlib is widely used in many fields, including data science, finance, and engineering.

Some of the key features of Matplotlib include:

- Plotting functions: Matplotlib provides a wide range of functions for creating different types of plots and charts, such as line plots, scatter plots, bar plots, histograms, and heatmaps.

- Customization: Matplotlib provides extensive options for customizing plots, such as changing colors, fonts, and styles, adding labels, titles, and legends, and adjusting axis scales and limits.

- Subplots: Matplotlib allows multiple plots to be displayed in the same figure, using subplots. This enables the comparison of multiple data sets in a single visualization.

- Animation: Matplotlib provides tools for creating animated plots and visualizations, which can be useful for displaying dynamic data.

- Integration with other libraries: Matplotlib can be easily integrated with other Python libraries, such as NumPy and Pandas, for data analysis and visualization.

Overall, Matplotlib is a powerful and versatile library that provides a wide range of tools for data visualization in Python. It enables the creation of

various types of plots and charts, and provides extensive customization options to create effective and visually appealing visualizations.

Seaborn: Seaborn is a Python library built on top of Matplotlib that is used for statistical data visualization. Seaborn provides a high-level interface for creating various types of statistical plots and charts, and is widely used in data science, machine learning, and scientific research.

Some of the key features of Seaborn include:

- Statistical visualization: Seaborn provides many built-in functions for creating various types of statistical plots, such as scatter plots, line plots, bar plots, heatmaps, and distribution plots.

- Data manipulation: Seaborn can work with Pandas data frames and arrays, and provides built-in functions for data manipulation, such as aggregating, filtering, and sorting data.

- Customization: Seaborn provides extensive options for customizing plots, such as changing colors, fonts, and styles, adding labels, titles, and legends, and adjusting axis scales and limits.

- Integration with other libraries: Seaborn can be easily integrated with other Python libraries, such as Pandas and Matplotlib, for data analysis and visualization.

- Statistical analysis: Seaborn provides tools for statistical analysis, such as hypothesis testing, confidence intervals, and regression analysis, which can be used to gain insights into data and make data-driven decisions.

Overall, Seaborn is a powerful and versatile library that provides a wide range of tools for statistical data visualization and analysis in Python. It enables the creation of various types of statistical plots and charts, and provides extensive customization options to create effective and visually appealing visualizations.

Warnings: In Python, warnings are messages generated by the interpreter or third-party libraries to indicate potential issues or unexpected behavior in code. Warnings can be used to alert developers of potential bugs, inefficiencies, or other issues that may need attention.

Some common types of warnings in Python include:

- DeprecationWarning: This warning is raised when a feature or function is being deprecated and will be removed in a future version of the library or framework.

- RuntimeWarning: This warning is raised when there is a potential issue with the code at runtime, such as division by zero or invalid values in a computation.

- SyntaxWarning: This warning is raised when there is a potential issue with the syntax of the code, such as unused variables or variables that are defined but never used.

- ImportWarning: This warning is raised when there is an issue with importing a module or package, such as an undefined module or a circular import.

- FutureWarning: This warning is raised when a feature or function is being changed or deprecated in a future version of the library or framework.

To handle warnings in Python, developers can use the warnings module, which provides functions for filtering, displaying, and ignoring warnings. For example, developers can use the filterwarnings() function to filter out specific types of warnings or to display warnings as errors. Additionally, warnings can be silenced entirely by using the -W ignore flag when running Python scripts or by setting the PYTHONWARNINGS environment variable.

4. IMPLEMENTATION

The Proposed System Make Use of Item-based filtering.

4.1 Item-based filtering: Item-based filtering is a technique used in recommendation systems to provide personalized recommendations to users. In this technique, recommendations are generated based on the similarity between the items that users have interacted with in the past.

The basic idea behind item-based filtering is to identify the items that are similar to each other based on certain features or attributes. Once the similarities are established, the system can recommend items to users that are similar to the ones they have liked or interacted with before.

For example, suppose a user has interacted with several movies on a streaming platform. The system can use item-based filtering to identify other movies that have similar attributes such as genre, actors, directors, etc. Based on this similarity, the system can recommend movies that the user is likely to enjoy.

Item-based filtering can be used in combination with other techniques such as user-based filtering and content-based filtering to improve the accuracy of recommendations. By considering multiple factors, these techniques can provide more accurate and personalized recommendations to users.

5. Results and Discussion

The performance of the movie recommendation system was evaluated using a dataset of user ratings for a collection of movies. The system was trained on a subset of the data and evaluated on a held-out test set.

6. Performance of the Recommendation System

The recommendation system is able to recommend us the similar movie based on our input. These results indicate that the system is able to make accurate and relevant recommendations for a majority of users.

Predict Movie

In [36]:
```
def predict_movies(movie_name):
    movie_user_ratings=moviemat[movie_name]
    similar_to_movie=moviemat.corrwith(movie_user_ratings)

    corr_movie=pd.DataFrame(similar_to_movie, columns=['Correlation'])
    corr_movie.dropna(inplace=True)

    corr_movie=corr_movie.join(ratings['num of ratings'])
    prediction=corr_movie[corr_movie['num of ratings']>100].sort_values('Correlation', ascending=False)

    return prediction
```

In [37]:
```
predictions=predict_movies("Titanic (1997)")
```

In [38]:
```
predictions.head()
```

Out[38]:

title	Correlation	num of ratings
Titanic (1997)	1.000000	350
River Wild, The (1994)	0.497600	146
Abyss, The (1989)	0.472103	151
Bram Stoker's Dracula (1992)	0.443560	120
True Lies (1994)	0.435104	208

In [39]:
```
predict_movies("River Wild, The (1994)")
```

Out[39]:

title	Correlation	num of ratings
River Wild, The (1994)	1.000000	146
Kiss the Girls (1997)	0.744500	143
Young Guns (1988)	0.662424	101
Seven Years in Tibet (1997)	0.640866	155
My Best Friend's Wedding (1997)	0.576746	172

7. Limitations and Future Work

One limitation of our recommendation system is that it relies solely on user ratings to make recommendations. In future work, we could incorporate other features such as movie genres or user demographics to improve the accuracy and relevance of our recommendations. Additionally, we could explore more advanced recommendation algorithms such as matrix factorization or deep learning models to further improve the performance of the system.

Overall, our movie recommendation system shows promising results and outperforms baseline models in terms of accuracy and precision. However, there is still room for improvement, and future work could explore ways to incorporate additional features and more advanced algorithms to further enhance the performance of the