VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI – 590 018.

A Internship Report On

"Movie Recommendation System Using Machine Learning "

Submitted in Partial Fulfilment of the Requirement for

The Award of the Degree Of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE & ENGINEERING

Submitted by,

Rahul Naik (2KD19CS068)

Under the Guidance of,
Prof. Mahantesh Laddi

Department of Computer Science & Engineering

K.L.E. College of Engineering and Technology, Chikodi – 591201

2022-2023

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI – 590 018.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE OF APPROVAL OF INTERNSHIP

This is to certify that **Mr. Rahul Naik** bearing **USN: 2KD19CS068** has satisfactorily completed the Internship entitled **"Movie Recommendation System Using Machine Learning in Python" in Zephyr technologies limited Manglore** for the partial fulfilment of Bachelor of Engineering in Computer Science andEngineering prescribed by the **Visvesvaraya Technological University, Belagavi** for the academic year 2022-23. The report has been approved as it satisfies the academic requirements in Internship prescribed for the Bachelor of Engineering Degree.

Guide	H.O.D.	Principal
Prof. Mahantesh Laddi	**Prof. Ashwini V Gavali**	**Dr. Prasad B. Rampure**

Examiner 1:

Examiner 2:

ACKNOWLEDGEMENT

We place on record and warmly acknowledge the encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our guide **Prof. Madhurani B S,** Department of Computer Science and Engineering, K.L.E College of Engineering and Technology, Chikodi, in bringing this internship to a successful completion.

We are grateful to **Prof. Ashwini V Gavali,** Head of the Department of Computer Science and Engineering, for permitting us to make use of the facilities available in the department to carry out the internship successfully.

We extend our gratefulness to our internship coordinator **Prof. Mahantesh Laddi, Prof. Vinod Biradar** Department of Computer Science and Engineering, for their great support in coordinating to the needs of us in our endeavour.

We express our sincere gratitude to **Dr. Prasad.B. Rampure**, Principal, K.L.E College of Engineering and Technology, Chikodi, for his support and encouragement.

Finally, we extend our thanks to the teaching and non-teaching staff of our department for their help.

Rahul Naik (2KD19CS068)

ABSTRACT

An internship is an opportunity to relate what has been covered in class and what is applicable in the field in an operational environment. The purpose of the program is to get a practical aspect of the theoretical work studied at the university and understand the operations in the IT sector and enable students to gain experience in different tasks. During my internship period, several approaches and exposure methods were used which included: hands-on, thorough reading of relevant materials, and also questions and answer approaches. This internship work investigates the feasibility of using Data Science with machine learning techniques in real-life problems. The heart disease prediction project is aimed at developing a machine learning model to predict the likelihood of an individual developing heart disease based on their medical history and demographic information. The project involves collecting and preprocessing a large dataset of medical records, including patient age, sex, bloodpressure, cholesterol levels, and other clinical variables. Feature engineering techniques are used to extract relevant features from the data, and various machine learning algorithms are trained and evaluated to select the best-performing model. The final model is then deployed and integrated into a user-friendly web application that allows users to input their personal health information and receive a risk assessment for heart disease. The project aims to improve early detection and prevention of heart disease, ultimately improving public health outcomes.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 What is internship?

An internship is a period of work experience offered by an organization for a limited period of time. Once confined to medical graduates, internship is used for a wide range of placements in businesses, non-profit organizations and government agencies. They are typically undertaken by students and graduates looking to gain relevant skills and experience in a particular field. Employers benefit from these placements because they often recruit employees from their best interns, who have known capabilities, thus saving time and money in the long run. Internships are usually arranged by third-party organizations that recruit interns on behalf of industry groups. Rules vary from country to country about when interns should be regarded as employees. The system can be open to exploitation by unscrupulous employers.

Typically, an internship consists of an exchange of services for experience between the intern and the organization. Internships are used to determine if the intern still has an interest in that field after the real-life experience. In addition, an internship can be used to create a professional network that can assist with letters of recommendation or lead to future employment opportunities. The benefit of bringing an intern into full-time employment is that they are already familiar with the company, their position, and they typically need little to no training. Internships provide current college students the ability to participate in a field of their choice to receive hands-on learning about a particular future career, preparing them for full-time work following graduation.

1.2 Company Profile

They are Developers, specializing in Web Development, Application Development, Networking, and Security Systems. As a developing company, they build our applications which reach all market standards. Zeel Code Labs has improved performance for clients.

By driving leads and working to find solutions for their clients, Zeel Code Labs has grown to become one of the largest and most successful code development firms. They are servicing many clients. Their experience in development is the basis for future success. They know where to find the leads and convert them to revenue streams.

ZEPHYR
Technologies & Solutions Pvt. Ltd.

Fig 1.2.1 Company Logo

1.2.1 Vision

To implement and measure e-business strategies to provide maximum exposure to consumers everywhere around the globe. Our strategy makes our clients thrive. Mobile Marketing with mobile search, your ads will display through search results performed on cell phones to expand your reach.

1.2.2 Mission

Evolution is part of their mission and culture. It is also the driving engine of our market. That means embracing new technologies and expanding your horizons. Because true innovation can only happen when you have the curiosity to explore new ideas and the courage to discover better solutions. They are always looking for a better way to deliver qualified prospects to our business.

1.2.3 Company Information

Corporate Name	Zephyr Technologies And Solutions Pvt Ltd
Main Office	5th floor,Oberele Tower Balmata,Manglore-575002

CEO	Abdulla Abid Samah
Email	mail@zephyrtechnologies.co
Established Year	December 2015
Company Category	IT Software/Embedded

Table 1.2.3.1 Company Information

1.3 Objectives

1. One of the main objectives of an internship is to expose you to a particular job and a profession or industry.
2. While you might have an idea about what a job is like, you won't know until you perform it if it's what you thought it was, if you have the training and skills to do it and if it's something you like.
3. The main objective of following this internship is the benefit of the experience.
4. It was a new challenge in the real-life working environment, which helped in learning a new set of skills.
5. The objective of this internship is to gain experience in the use of novel Machine Learning algorithms and the exploitation of large datasets.

CHAPTER 2

REQUIREMENTS AND INSTALLATION

The most convenient way of installing the Python requirements for this training is by using the Anaconda scientific Python distribution. Anaconda is a collection of the most commonly used Python packages preconfigured and ready to use.

Approximately 150 scientific packages are included in the Anaconda installation. To install Anaconda, we visited the site:https://www.continuum.io/downloads and install the version of Anaconda for our operating system. All Python software described here is available for Windows, Linux, and Macintosh. All code samples presented in this tutorial were tested under Ubuntu Linux 14.04 using Python 2.7. Some code examples may not work on Windows without slight modification (e.g. file paths in Windows use \ and not / as in UNIX type systems).The main software used in a typical Python machine learning pipeline can consist of almost any combination of the following tools:

1. NumPy, for matrix and vector manipulation.

2. Pandas for time series and R-like DataFrame data structures.

3. The 2D plotting library matplotlib.

4. SciKit-Learn as a source for many machine learning algorithms and utilities.

2.1 Jupyter

Fig 2.1.1 Jupyter Logo

Jupyter, previously known as IPython Notebook, is a web-based, interactive development environment. Originally developed for Python, it has since expanded to support over 40 other programming languages including Julia and R.

Jupyter allows for notebooks to be written that contain text, live code, images, and equations. These notebooks can be shared, and can even be hosted on GitHub for free.

For each section of this tutorial, you can download a Juypter notebook that allows you to edit and experiment with the code and examples for each topic. Jupyter is part of the Anaconda distribution, it can be started from the command line using using the jupyter command:**$ jupiter notebook**

Upon typing this command, the Jupyter server will start, and you will briefly see some information messages, including, for example, the URL and port at which the server is running (by default http://localhost:8888/). Once the server has started, it will then open your default browser and point it to this address. This browser window will display the contents of the directory where you ran the command.

To create a notebook and begin writing, click the new button and select Python. A new notebook will appear in a new tab in the browser. A Jupyter notebook allows you to run code blocks and immediately see the output of these blocks of code, much like the IPython REPL

Jupyter has several short-cuts to make navigating the notebook and entering code or text quickly and easily. For a list of shortcuts, use the menu Help → Keyboard Shortcuts.

CHAPTER 3

INTERACTIVE DEVELOPMENT ENVIRONMENTS

3.1 Python

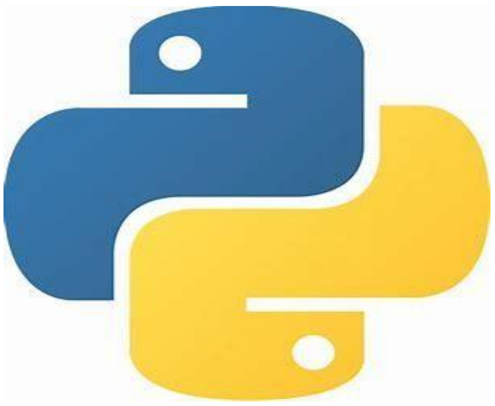


Fig 3.1.1 Python Logo

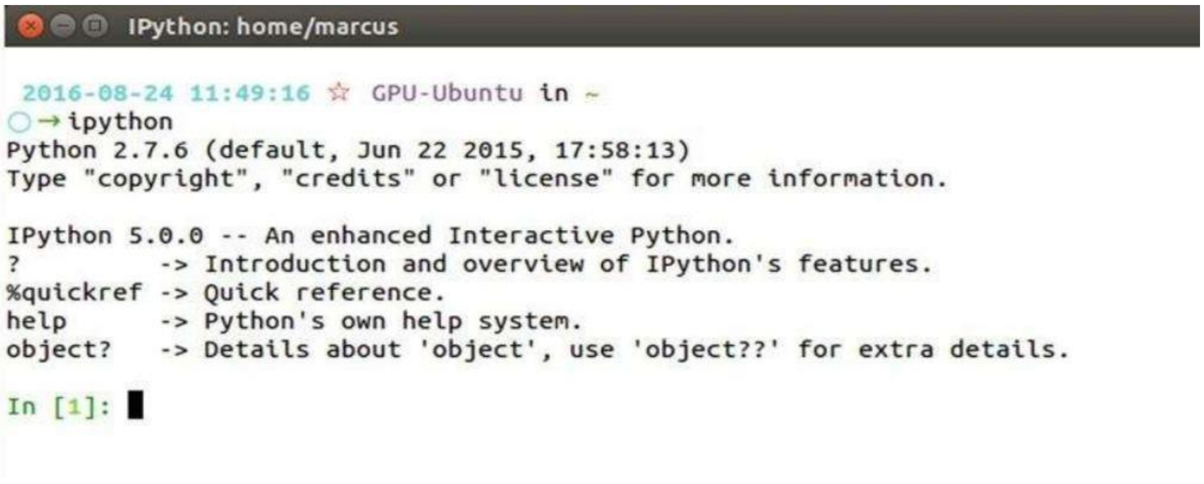
Python is a general-purpose programming language that is used for anything from web development to deep learning. According to several metrics, it is ranked as one of the topthree most popular languages. It is now the most frequently taught introductory language at top U.S. universities according to a recent ACM blog article. Due to its popularity, Pythonhas a thriving open-source community, and there are over 80,000 free software packages available for the language on the Python Package Index (PyPI).

Python is a widely-used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is a programming language that lets you work quickly and integrate systems more efficiently. There are two major Python versions **Python 2 and Python 3**. Both are quite different.

3.2 IPython

IPython is a REPL that is commonly used for Python development. It is **$ ipython** Some informational data will be displayed, similar to what is seen in, and you will then be presented with a command prompt.



```
IPython: home/marcus

 2016-08-24 11:49:16 ☆ GPU-Ubuntu in ~
○→ ipython
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
Type "copyright", "credits" or "license" for more information.

IPython 5.0.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: █
```

Fig 3.2.1 The IPython Shell

IPython is what is known as a REPL: a Read Evaluate Print Loop. The interpreter allows you to type in commands which are evaluated as soon as you press the Enter key. Any returned output is immediately shown in the console. For example, we may type the following:

```
1. In [1]: 1 + 1
   Out [1]: 2
2. In [2]: import math
   In [3]: math . radians (90)
   Out [3]: 1 . 5 7 0 7 9 6 3 2 6 7 9 4 8 9 6 6
```

After pressing return (Line 1 in Listing 2), Python immediately interprets the line and responds with the returned result (Line 2 in Listing 2). The interpreter then awaits the next command, hence Read Evaluate Print Loop. Using IPython to experiment with code allows you to test ideas without needing to create a file (e.g. fibonacci.py) and running this file from the com-mand line (by typing python fibonacci.py at the command prompt). Using the IPython REPL, this entire process can be made much easier. Of course, creating permanent files is essential for larger projects. A useful feature of IPython are the so-called magic functions. These com-mands are not interpreted as Python code by the REPL, instead they arespecial commands that IPython understands. For example, to run a Python script you can use the %run magic function:

- [In]: % run fibonacci. py 30
- [Out]: fibonacci number 30 is 832040.

3.3 Data

For the Introduction to Python, NumPy, and Pandas sections we will work with either generated data or with a toy dataset. Later in the chapter, we will move on to medical examples, including a breast cancer dataset, a diabetes dataset, and a high-dimensional gene expression dataset. All medical datasets used in this chapter are freely available and we will describe how to get the data in each relevant section. In earlier sections, generated data will suffice in order to demonstrate example usage, while later we will see that analysing more involved medical data using the same open-source tools is equally possible.

3.4 Variables

One of the most powerful features of a programming language is the ability to manipulate variables. A variable is a name that refers to a value. An assignment statement creates new variables and gives them values.

3.5 Data Types

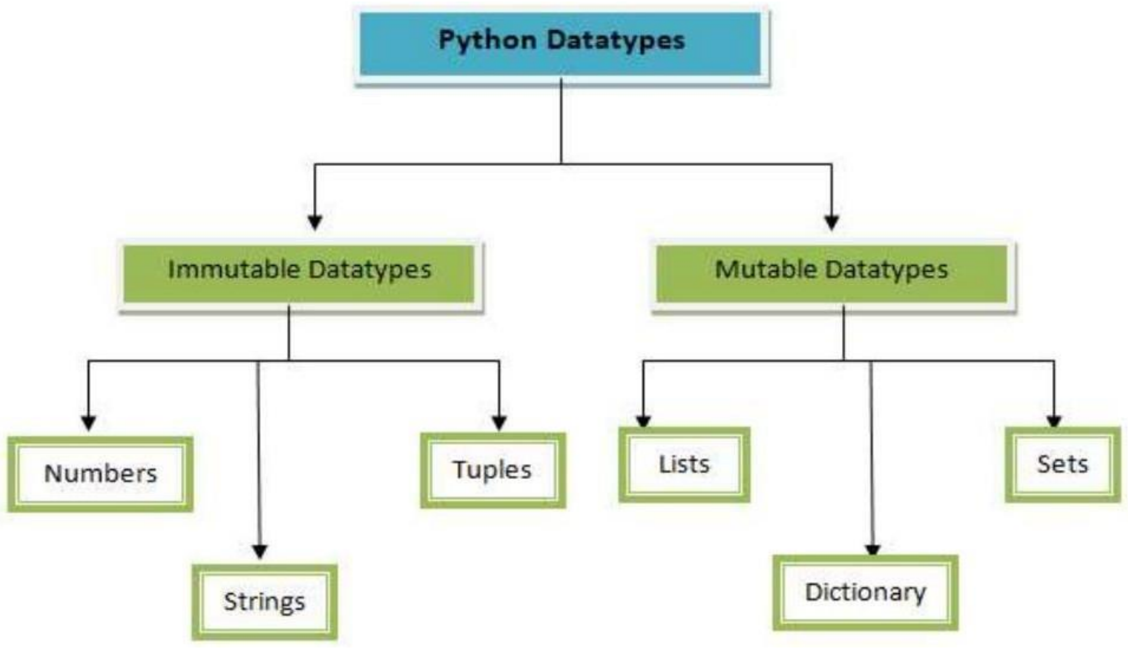


Fig 3.5.1 Data Types

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data.

3.5.1 Numbers

Number stores numeric values. The integer, float, and complex values belong to a Python Numbers data type. Python provides the **type ()** function to know the data type of the variable. Similarly, the is **instance ()** function is used to check an object belonging to a particular class.

Int - Integer value can be any length such as integers 10, 2, 29, -20, -150 etc. Python has no restriction on the length of an integer. Its value belongs to int.

Float - Float is used to store floating-point numbers like 1.9, 9.902, 15.2, etc. It is accurate up to 15 decimal points.

complex - A complex number contains an ordered pair, i.e., x + iy where x and y denote the real and imaginary parts, respectively. The complex numbers like 2.14j, 2.0 + 2.3j, etc.

3.5.2 Strings

- The string can be defined as the sequence of characters represented in the quotation marks. In Python, we can use single, double, or triple quotes to define a string.
- String handling in Python is a straightforward task since Python provides built-in functions and operators to perform operations in the string.
- In the case of string handling, the operator + is used to concatenate two strings as the operation "hello"+" python" returns "hello python".
- The operator is known as a repetition operator as the operation "Python" #2 returns * 'Python Python'.

3.5.3 Tuple

- A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of items of different data types. The items of the tuple are separated with a comma (.) and enclosed in parentheses ().
- A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

3.5.4 List

- Python Lists are similar to arrays in C. However, the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [ ].
- We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (*) work with the list in the same way as they were working with the strings.

3.5.4 Dictionary

Dictionary is an unordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type, whereas value is an arbitrary Python object. The items in the dictionary are separated with the comma (.) and enclosed in the curly braces { }.

3.5.5 Set

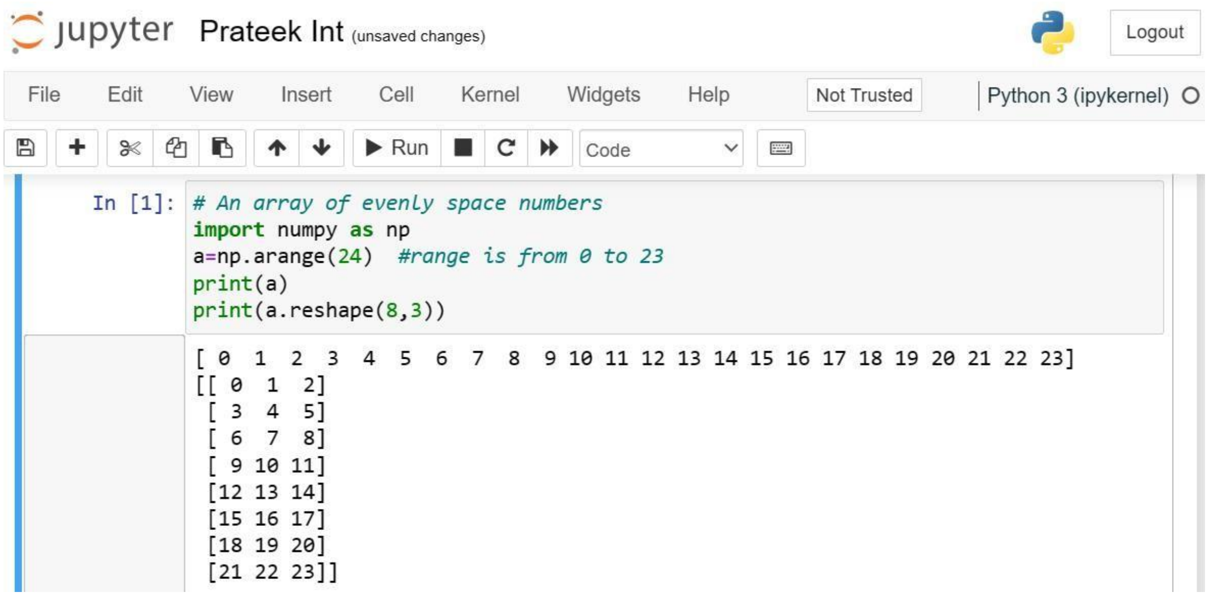
Python Set is the unordered collection of the data type. It is iterable, mutable (can modify after creation), and has unique elements. In set, the order of the elements is undefined; it may return the changed sequence of the element. The set is created by using a built-in function **set ( )**, or a sequence of elements is passed in the curly braces and separated by the comma. It can contain various types of values.

3.6 Libraries Used in Python

3.6.1 NumPy

NumPy is a general data structures, linear algebra, and matrix manipulation library for Python. Its syntax, and how it handles data structures and matrices are comparable to that of MATLAB.

To use NumPy, first import it (the convention is to import it as np, to avoid having to type out NumPy each time): import Numpy as np Rather than repeat this line for each code listing, we will assume you have imported NumPy. Any further imports that may be required for a code sample will be explicitly mentioned.



```
In [1]: # An array of evenly space numbers
        import numpy as np
        a=np.arange(24)  #range is from 0 to 23
        print(a)
        print(a.reshape(8,3))
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]
 [15 16 17]
 [18 19 20]
 [21 22 23]]
```

Fig 3.6.1.1 An example of NumPy

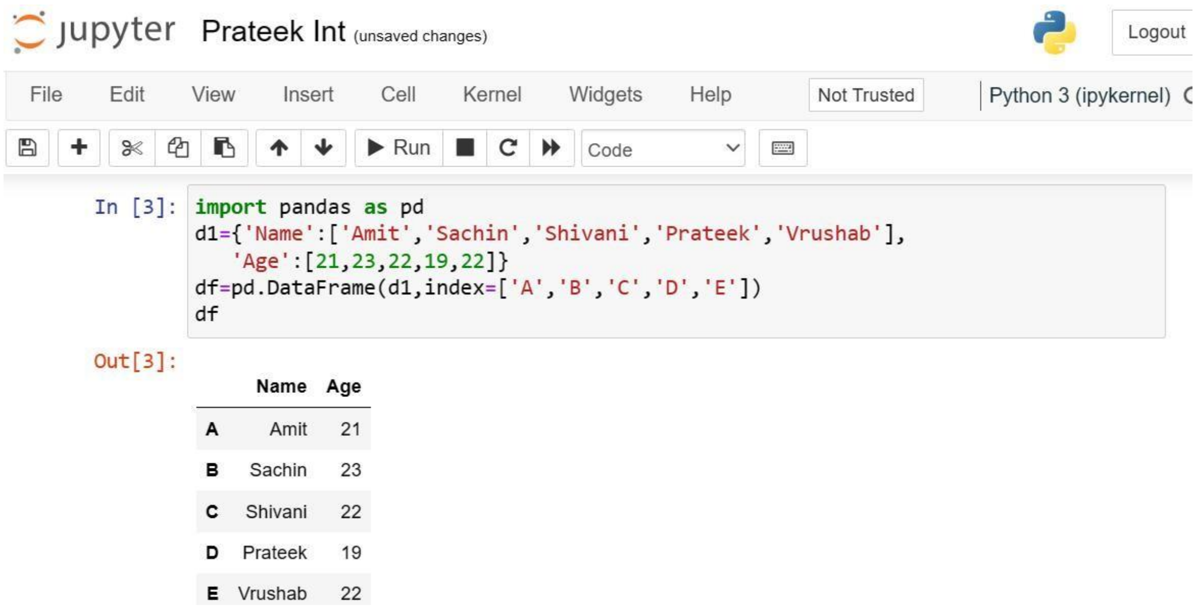
Fig.3.6.1.1 describes some basic usage of NumPy by first creating a NumPy array and then retrieving some of the elements of this array.

3.6.2 Pandas

Pandas is a software library for data analysis of tabular and time-series data. In many ways, it reproduces the functionality of R's DataFrame object. Also, many common features of Microsoft Excel can be performed using Pandas, such as "group by", table pivots, and easy column deletion and insertion.

Pandas DataFrame objects are label-based (as opposed to index-based as is the case with NumPy), so each column is typically given a name that can be called to perform operations. DataFrame objects are more similar to spreadsheets, and each column of a DataFrame can have a different type, such as Boolean, numeric, or text. Often, it should be stressed, that you will use NumPy and Pandas in conjunction. The two libraries complement each other and are not competing for frameworks, although there is an overlap in functionality between the two. First, we must get some data.

we will load some sample data into a Pandas DataFrame object, then rename the DataFrame object's columns, and lastly take a look at the rows contained in the DataFrame**.**



```
In [3]: import pandas as pd
        d1={'Name':['Amit','Sachin','Shivani','Prateek','Vrushab'],
           'Age':[21,23,22,19,22]}
        df=pd.DataFrame(d1,index=['A','B','C','D','E'])
        df
```

Out[3]:

```
      Name  Age
A     Amit   21
B   Sachin   23
C  Shivani   22
D  Prateek   19
E  Vrushab   22
```

Fig 3.6.2.1 Example of Pandas

CHAPTER 4

MACHINE LEARNING ALGORITHMS

There are many types of Machine Learning Algorithms specific to different use cases. As we work with datasets, a machine learning algorithm works in two stages. We usually split the data around 20%-80% between the testing and training stages. Under supervised learning. we spit a dataset into training data and test data in Python ML. Followings are the Algorithms of Python Machine Learning.

Content-Based Filtering

IT is a recommendation algorithm that suggests items to users based on the similarity between the content or attributes of the items. In the context of movie recommendation, it recommends movies to users based on the similarity of movie attributes, such as genre, director, actors, and plot keywords. The algorithm follows these general steps:

1. Item Representation: Each movie is represented by a set of relevant features or attributes. These attributes can include genre, director, actors, release year, plot keywords, and other descriptive information.
2. User Profile Creation: The algorithm creates a user profile based on the user's preferences, either explicitly (e.g., user ratings) or implicitly (e.g., movies the user has watched or liked). The user profile captures the user's preferred movie attributes.
3. Similarity Calculation: Calculate the similarity between movies based on their attributes. Various similarity metrics can be used, such as cosine similarity or Jaccard similarity, depending on the nature of the attributes.
4. Recommendation Generation: Identify the most similar movies to the ones that the user has already rated or interacted with. These similar movies are then recommended to the user.
5. Ranking and Filtering: Rank the recommended movies based on some criteria, such as similarity score or relevance. Apply filters to remove movies that may not be suitable for the user, such as movies the user has already seen or disliked.

The content-based filtering algorithm relies on the assumption that if a user has shown a preference for certain movie attributes in the past, they are likely to enjoy movies with similar attributes in the future. It enables personalized recommendations based on the user's taste, rather than relying solely on the preferences of other users.

Collaborative Filtering

IT is a popular machine learning technique used in recommender systems to make predictions or recommendations by leveraging the collective knowledge or preferences of a group of users. CF algorithms are based on the assumption that users who have similar preferences in the past will have similar preferences in the future.

There are two main approaches to Collaborative Filtering:

1. User-Based Collaborative Filtering: This approach identifies similar users based on their historical interactions with items. It finds users who have rated or interacted with items in a similar way to the target user and then recommends items that those similar users have liked but the target user hasn't yet experienced. The similarity between users is typically calculated using techniques like cosine similarity or Pearson correlation.
2. Item-Based Collaborative Filtering: This approach focuses on finding similarities between items themselves. It identifies items that are similar to the ones the target user has already rated or interacted with and recommends those similar items. The similarity between items is calculated using techniques such as cosine similarity or Jaccard similarity.

The collaborative filtering process typically involves the following steps:

1. Data Collection: Gather data on user-item interactions, such as ratings, reviews, or purchase histories.
2. User-Item Matrix: Represent the data in the form of a matrix, where rows correspond to users, columns correspond to items, and the matrix elements represent user-item interactions (e.g., ratings).
3. Similarity Calculation: Compute the similarity between users or items using a similarity measure. Common measures include cosine similarity, Pearson correlation, or Jaccard similarity.
4. Neighborhood Selection: Select a subset of similar users or items based on a predefined threshold or by considering the top-k most similar neighbors.
5. Prediction/Recommendation Generation: Generate predictions or recommendations based on the user-item matrix and the selected neighborhood. For user-based CF, predictions can be made by aggregating the ratings or preferences of the selected similar users. For item-based CF, recommendations can be generated by considering the ratings of similar items.
6. Ranking and Filtering: Rank the predicted or recommended items based on some criteria, such as predicted ratings or relevance, and filter out items that may not be suitable for the target user (e.g., items the user has already seen or disliked).

CHAPTER 5

ADVANTAGES AND DISADVANTAGES

5.1 Advantages

1. Personalized Recommendations: Movie recommendation systems analyze user preferences, viewing history, and behavior to generate personalized movie suggestions. By understanding individual tastes and preferences, these systems can offer tailored recommendations that align with a user's interests, leading to a more enjoyable and satisfying movie-watching experience.
2. Discovery of New Content: Movie recommendation systems can introduce users to a wide range of movies they may not have discovered otherwise. By analyzing user data and comparing it to similar users' preferences, these systems can suggest movies that align with a user's interests but may be lesser-known or from different genres, allowing users to explore and discover new content.
3. Time-Saving and Convenience: With the abundance of movies available today, finding something to watch can be overwhelming. Movie recommendation systems save users time by curating a list of relevant suggestions based on their preferences. Users don't have to spend hours searching for movies manually; instead, they can rely on the system to provide them with personalized recommendations quickly and conveniently.
4. Enhanced User Engagement and Satisfaction: By offering personalized recommendations, movie recommendation systems can increase user engagement and satisfaction. When users receive relevant suggestions that match their preferences, they are more likely to find movies they enjoy, resulting in a positive viewing experience. This can lead to increased user loyalty and retention for movie platforms.
5. Discovering Niche or Independent Films: Movie recommendation systems can help promote and bring attention to niche or independent films that may not have the same marketing budgets or widespread visibility as mainstream movies. By recommending these films to users who have shown an interest in similar content, the systems can contribute to the exposure and success of lesser-known movies.
6. Improved Movie Catalog Management: For movie platforms, recommendation systems can assist in managing their movie catalog efficiently. By analyzing user data and preferences, platforms can gain insights into the popularity of different movies, identify trends, and make informed decisions about acquiring or promoting specific titles.

7. Continuous Learning and Adaptation: Movie recommendation systems employ machine learning algorithms that continuously learn and adapt based on user feedback and behavior. As users interact with the system, providing ratings or feedback on suggested movies, the algorithms can refine their recommendations over time, improving their accuracy and relevance.

5.2 Disadvantages

1. Limited Personalization: Movie recommendation systems often rely on algorithms that analyze user preferences and behavior to make recommendations. However, these algorithms may not always accurately capture the complexities of individual taste. As a result, the recommendations may not be truly personalized and might not reflect the unique preferences of the user.
2. Narrow Focus: Movie recommendation systems typically prioritize popular or mainstream content, which may lead to a lack of diversity in recommendations. These systems often recommend movies that align with existing user preferences or movies that have been widely viewed. As a result, users may miss out on discovering niche or lesser-known films that could potentially interest them.
3. Overemphasizing Popularity: Some movie recommendation systems tend to prioritize popular movies, which can lead to a "herd mentality" effect. Users may end up being recommended the same popular movies over and over again, without exposure to alternative or independent films. This can limit users' exposure to diverse cinematic experiences.
4. Limited Exploration: While movie recommendation systems aim to provide users with relevant suggestions, they can sometimes restrict the exploration of new genres, directors, or styles. These systems often rely on past viewing habits to generate recommendations, which can create a filter bubble and prevent users from expanding their cinematic horizons.
5. Lack of Contextual Understanding: Recommendation systems primarily rely on user ratings, viewing history, or explicit feedback to generate recommendations. However, they often lack a deeper understanding of the context behind the movie choices. They may not consider factors such as mood, social setting, or specific occasions, which can result in inaccurate or irrelevant recommendations.
6. Privacy Concerns: Movie recommendation systems typically gather and analyze user data, including viewing history and personal preferences. This data collection raises privacy

concerns for users, as their personal information is being used to generate recommendations. There is a risk of potential misuse or unauthorized access to this sensitive data.

7. Ignoring Serendipity: Movie recommendation systems aim to provide users with relevant suggestions based on their preferences. However, this personalized approach may overlook the element of serendipity, which is the joy of stumbling upon unexpected movies or discovering new favorites through chance encounters. Recommendation systems may unintentionally limit the spontaneous exploration of films.

CHAPTER 6

APPLICATIONS

Music Recommendations: Content-based filtering can be used to recommend music to users based on their preferences. The algorithm analyzes the audio features of songs, such as genre, tempo, instrumentation, and lyrical content, to identify similarities and recommend songs with similar attributes.

News and Article Recommendations: Content-based filtering can be employed to recommend news articles or blog posts to users based on their interests. The algorithm analyzes the textual content, keywords, and topics of articles to identify similar articles and provide personalized recommendations.

Book Recommendations: Content-based filtering algorithms can analyze the attributes of books, such as genre, author, language, and plot summaries, to recommend books that align with the user's preferences. Users who enjoyed certain attributes in books are likely to receive recommendations for books with similar attributes.

E-commerce Product Recommendations: In e-commerce platforms, content-based filtering can suggest products to users based on their past interactions or preferences. The algorithm analyzes product attributes, such as category, brand, price, and description, to recommend products that match the user's preferences.

Job Recommendations: Content-based filtering can be applied in job recommendation systems, where the algorithm analyzes job attributes, such as job title, industry, required skills, and job description, to suggest job opportunities that align with the user's skills and preferences.

Travel Recommendations: Content-based filtering can be used in travel recommendation systems to suggest destinations, hotels, or travel packages based on user preferences. The algorithm analyzes attributes such as location, amenities, price, and user reviews to identify similar travel options.

CHAPTER 7

INTRODUCTON TO PROJECT

7.1 Introduction

A movie recommendation system is a technology that suggests movies to users based on their preferences, interests, and past movie-watching behavior. It leverages algorithms and data analysis techniques to provide personalized movie recommendations, enhancing the user's movie discovery and viewing experience. The goal of a movie recommendation system is to address the challenge of information overload in the vast landscape of available movies. With thousands of movies released each year across various genres, it can be overwhelming for users to navigate and select movies that align with their preferences. A recommendation system aims to alleviate this problem by offering tailored suggestions that match the user's taste, leading to increased user satisfaction and engagement.

Movie recommendation systems typically employ various algorithms and techniques to generate recommendations. These algorithms can be categorized into collaborative filtering, content-based filtering, matrix factorization, deep learning, and hybrid approaches. Collaborative filtering analyzes user behavior and preferences to identify similar users or movies, while content-based filtering focuses on the attributes and content of movies to suggest similar ones. Matrix factorization techniques decompose user-movie interaction data to uncover underlying patterns, and deep learning models process extensive data to generate personalized recommendations. The movie recommendation process involves several steps. It begins with data collection, where information about movies, user preferences, and interactions are gathered. This data is then processed and transformed into a suitable format for analysis. Algorithms analyze this data, calculating similarities between movies or users, generating predictions or recommendations, and ranking the suggested movies based on relevance. The final recommendations are presented to the user, who can explore and choose movies based on their interests.

Movie recommendation systems have become integral components of various platforms, including streaming services, online movie databases, and e-commerce websites. These systems enhance user engagement, increase user retention, and facilitate personalized movie discovery. By harnessing the power of algorithms and data analysis, movie recommendation systems help users find movies they are likely to enjoy, leading to a more satisfying and tailored movie-watching experience.

7.2 Objectives

- Personalized Movie Recommendations
- Improved Movie Discovery
- Increased User
- Adaptability and Learning
- Increased Revenue and Customer Satisfaction
- Diversity in Recommendations

7.3 Process

Data Collection: Gather data on movies, user preferences, and user interactions. This data can include movie attributes (genre, director, actors, etc.), user ratings or reviews, movie-watching history, and other relevant information.

Data Preprocessing: Clean and preprocess the collected data to make it suitable for analysis. This step involves handling missing values, removing duplicates, and transforming data into a structured format.

Feature Extraction: Extract relevant features from the movie data. This can include attributes such as genre, director, actors, release year, and plot keywords. Feature extraction helps represent movies in a format that can be used for comparison and similarity calculations.

User Profiling: Create user profiles based on their preferences and interactions. User profiles capture the movies they have rated, liked, or watched, as well as their demographic information or explicit preferences provided by the user.

Similarity Calculation: Calculate the similarity between movies or users based on their attributes or past interactions. Various similarity measures can be used, such as cosine similarity, Euclidean distance, or Pearson correlation. The goal is to identify movies or users that are similar to each other.

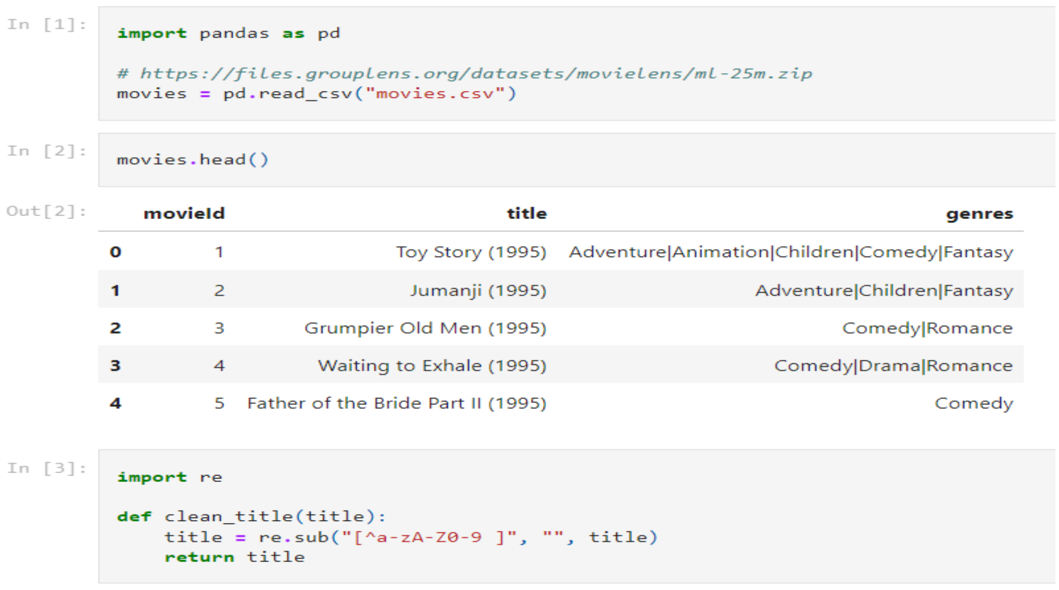
Recommendation Generation: Generate movie recommendations based on the calculated similarity scores. For a given user, identify movies that are similar to the ones the user has already rated or interacted with. This can be done using user-based collaborative filtering, item-based collaborative filtering, or content-based filtering approaches.

Ranking and Filtering: Rank the recommended movies based on some criteria, such as similarity scores, user preferences, or popularity. Apply filters to remove movies that may not be suitable for the user, such as movies the user has already seen or disliked.

Presentation and Delivery: Present the recommended movies to the user through a user interface or platform. The recommendations can be displayed as a list, grid, or personalized movie suggestions on the user's home page or recommendation section.

Feedback and Adaptation: Gather user feedback on the recommended movies, such as user ratings or explicit feedback. Incorporate this feedback into the recommendation system to refine and improve future recommendations. The system adapts and learns from user interactions over time to provide more accurate and relevant suggestions.
Evaluation and Performance Monitoring: Evaluate the performance of the recommendation system using appropriate metrics, such as precision, recall, or accuracy. Continuously monitor and analyze the system's performance to identify areas for improvement and optimize the recommendation algorithms.

7.4 Screenshots and Results



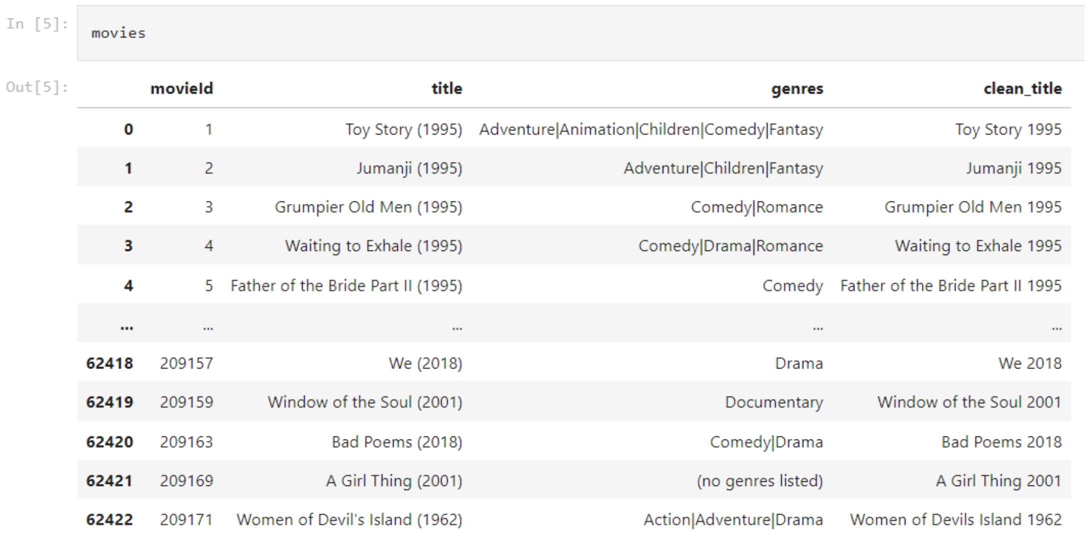
```
In [1]: import pandas as pd

        # https://files.grouplens.org/datasets/movielens/ml-25m.zip
        movies = pd.read_csv("movies.csv")

In [2]: movies.head()
```

Out[2]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

```
In [3]: import re

        def clean_title(title):
            title = re.sub("[^a-zA-Z0-9 ]", "", title)
            return title
```

Fig 7.5.1 Importing all the necessary libraries

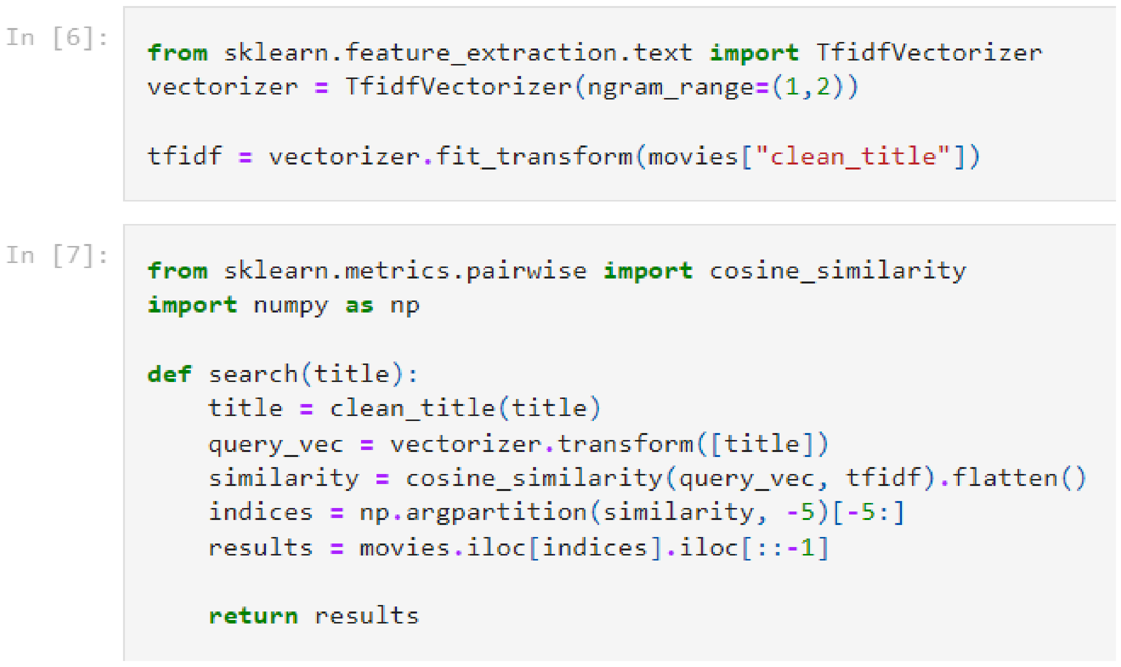


```
In [5]: movies
```

Out[5]:

	movieId	title	genres	clean_title
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	Toy Story 1995
1	2	Jumanji (1995)	Adventure|Children|Fantasy	Jumanji 1995
2	3	Grumpier Old Men (1995)	Comedy|Romance	Grumpier Old Men 1995
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	Waiting to Exhale 1995
4	5	Father of the Bride Part II (1995)	Comedy	Father of the Bride Part II 1995
...	...	...	...	...
62418	209157	We (2018)	Drama	We 2018
62419	209159	Window of the Soul (2001)	Documentary	Window of the Soul 2001
62420	209163	Bad Poems (2018)	Comedy|Drama	Bad Poems 2018
62421	209169	A Girl Thing (2001)	(no genres listed)	A Girl Thing 2001
62422	209171	Women of Devil's Island (1962)	Action|Adventure|Drama	Women of Devils Island 1962

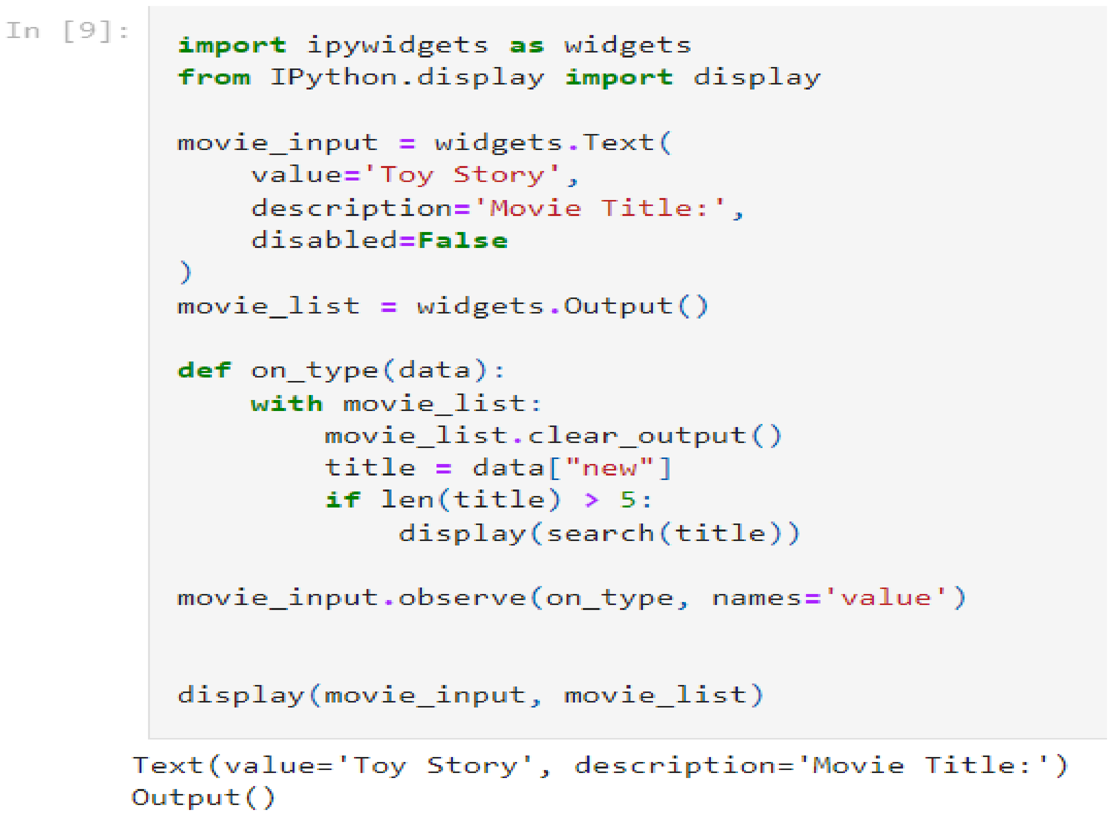
Fig 7.5.2 Information Of the Dataset



```
In [6]:
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(ngram_range=(1,2))

tfidf = vectorizer.fit_transform(movies["clean_title"])
```

```
In [7]:
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

def search(title):
    title = clean_title(title)
    query_vec = vectorizer.transform([title])
    similarity = cosine_similarity(query_vec, tfidf).flatten()
    indices = np.argpartition(similarity, -5)[-5:]
    results = movies.iloc[indices].iloc[::-1]

    return results
```

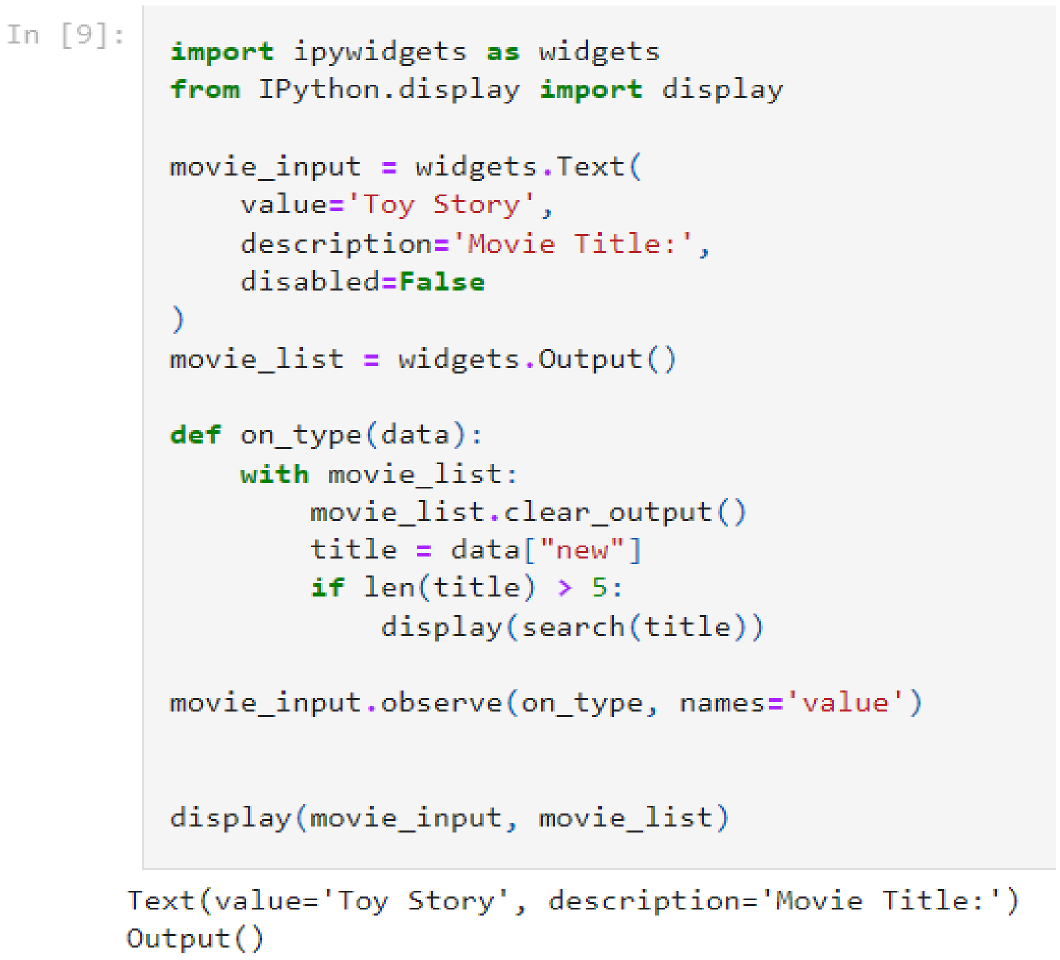
Fig 7.5.3 Applying Cosaine Similarlity



```
In [9]:
import ipywidgets as widgets
from IPython.display import display

movie_input = widgets.Text(
    value='Toy Story',
    description='Movie Title:',
    disabled=False
)
movie_list = widgets.Output()

def on_type(data):
    with movie_list:
        movie_list.clear_output()
        title = data["new"]
        if len(title) > 5:
            display(search(title))

movie_input.observe(on_type, names='value')


display(movie_input, movie_list)
```

```
Text(value='Toy Story', description='Movie Title:')
Output()
```

Fig 7.5.4 Creating a Search Button



```
In [9]: import ipywidgets as widgets
        from IPython.display import display

        movie_input = widgets.Text(
            value='Toy Story',
            description='Movie Title:',
            disabled=False
        )
        movie_list = widgets.Output()

        def on_type(data):
            with movie_list:
                movie_list.clear_output()
                title = data["new"]
                if len(title) > 5:
                    display(search(title))

        movie_input.observe(on_type, names='value')


        display(movie_input, movie_list)

        Text(value='Toy Story', description='Movie Title:')
        Output()
```

Fig 7.5.5 Applying Similarity Function

CONCLUSION

In conclusion, movie recommendation systems play a significant role in enhancing the movie-watching experience for users. By leveraging algorithms and data analysis techniques, these systems provide personalized and relevant movie suggestions to users based on their preferences, interests, and past interactions. The objectives of a movie recommendation system include personalized recommendations, improved movie discovery, increased user engagement, diversity in recommendations, adaptability, and learning. The process of a movie recommendation system involves steps such as data collection, preprocessing, feature extraction, user profiling, similarity calculation, recommendation generation, ranking and filtering, presentation, feedback and adaptation, and evaluation. These steps collectively aim to analyze user preferences, identify similar movies or users, and generate accurate and tailored movie recommendations.

Movie recommendation systems are utilized in various platforms, including movie streaming services, online databases, and e-commerce websites. They help users navigate through vast movie collections, discover new movies, and make informed choices based on their interests. By providing personalized recommendations, these systems drive user engagement, increase customer satisfaction, and contribute to revenue generation. As technology advances and more data becomes available, movie recommendation systems continue to evolve, incorporating advanced algorithms, machine learning techniques, and contextual information to provide even more accurate and personalized recommendations. By continuously improving the accuracy and relevance of suggestions, movie recommendation systems contribute to a more enjoyable and personalized movie-watching experience for users.

REFERENCES

1. Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2015). Recommender Systems Handbook. Springer.
2. Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer. Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (2004). An algorithmic framework for performing collaborative filtering. ACM Transactions on Information Systems (TOIS), 22(1), 5-53.
3. Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in Artificial Intelligence, 2009.
4. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295).
5. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
6. Zhang, Y., & Gong, M. (2019). Deep learning for recommender systems: A concise survey. ACM Transactions on Information Systems (TOIS), 37(4), 1-39.
7. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 734-749.
8. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370.