MACHINE LEARNING PROJECT
(2020-21)

MOVIE RECOMMENDATION SYSTEM

PROJECT REPORT

Institute Of Engineering & Technology

Submitted By:

Yash Kumar Singh(Team Lead)
(181500824)
Sarvendra Kumar Singh
(181500624)
Sanal Mishra
(181500615)
Prabhat Singh
(181500458)

Supervised By:

Mr. Pawan Kumar Verma
Assistant Professor
Department Of Computer Engineering And Applications

DECLARATION BY THE STUDENT

The completion of this project work could only have been possible with continued & dedicated efforts & guidance of large number of faculty & staff members of the institute. I acknowledge our gratitude to all of them. The acknowledgement however will be incomplete without specific mention as follows. I wish to acknowledge my deep gratitude to Mr Pawan Verma, Associate Professor at GLAU for his cooperation and guidance. Finally, we would like to say that we are indebted to our parents for everything that they have done for us. All of this would have been impossible without their constant support. And I also thank to God for being kind to me and driving me through this journey.

Name of Candidates:
Yash Kumar Singh(Team Lead)
Sarvendra Kumar Singh
Sanal Mishra
Prabhat Singh

Roll. No. :
181500824
181500624
181500615
181500458

Course: B.Tech(CSE)

Year: 3rd

Semester: V

Synopsis

Group No: 2

Group Information :

MEMBERS	UNIVERSITY ROLL NO.
Yash Kumar Singh	*181500824*
Sarvendra Kumar Singh	*181500624*
Sanal Mishra	*181500615*
Prabhat Kumar Singh	*181500458*

Information About Project:

Title of Project:	**Movie Recommendation System**	
Technical Details:	**Hardware requirements:**	➢ 4GB or higher RAM ➢ Minimum i3 Processor ➢ A back up disk/drive
	Software Requirements:	➢ Jupyter Notebook ➢ Pycharm ➢ Github
	DataSet:	➢ Movie Lens

Aim and Scope of the project: This Application allows us to sort out some of our favourite movies from the movies highly rated by critics and recommend that movie to us. Demand of this app nowadays is very high because of variation of choices of individual in the type of movies they like.

Expected outcome:
This application will recommend us our favourite movies shortlisted by reviews of critics after knowing our choices.

A detailed description of the Project:
This application will firstly ask us to login in our account if we are an existing user or register for a new account in case we have not done it earlier. Then it will ask us to chose some of our movies we like and rate them according to us. Then it will match our records with records of any critics in a database and recommend other movies to us which we may like.

ACKNOWLEDGEMENT

It is with my immense gratitude that I acknowledge the support and help of our mentor ,Mr. Pawan Kumar Verma, who has always encouraged us for this project. Without his continuous guidance and persistent help, this project would not have been a success for us. We are grateful to the GLA University, Mathura and the department of Computer Science without which this project would have not been an achievement. I also thank our family and friends, for their endless love and support throughout our project.

TABLE OF CONTENT

Introduction

1.1 Introduction to Machine learning:

1.1.1 What is Machine Learning?[2]

Learning, like intelligence, covers such a broad range of processes that it is difficult to define precisely. A dictionary definition includes phrases such as "to gain knowledge, or understanding of, or skill in, by study, instruction, or experience," and "modification of a behavioral tendency by experience." Zoologists and psychologists study learning in animals and humans. In this book we focus on learning in machines. There are several parallels between animal and machine learning. Certainly, many techniques in machine learning derive from the efforts of psychologists to make more precise their theories of animal and human learning through computational models. It seems likely also that the concepts and techniques being explored by researchers in machine learning may illuminate certain aspects of biological learning.

As regards machines, we might say, very broadly, that a machine learns whenever it changes its structure, program, or data (based on its inputs or in response to external information) in such a manner that its expected future performance improves. Some of these changes, such as the addition of a record to a data base, fall comfortably within the province of other disciplines and are not necessarily better understood for being called learning. But, for example, when the performance of a speech-recognition machine improves after hearing several samples of a person's speech, we feel quite justified in that case to say that the machine has learned.

Machine learning usually refers to the changes in systems that perform tasks associated with artificial intelligence (AI). Such tasks involve recognition, diagnosis, planning, robot control, prediction, etc. The "changes" might be either enhancements to already performing systems or ab initio synthesis of new systems. To be slightly more specific, we show the architecture of a typical AI "agent" in Fig. 1.1. This agent perceives and models its environment and computes appropriate actions, perhaps by anticipating their effects. Changes made to any of the components shown in the figure might count as learning. Different learning mechanisms might be employed depending on which subsystem is being changed.

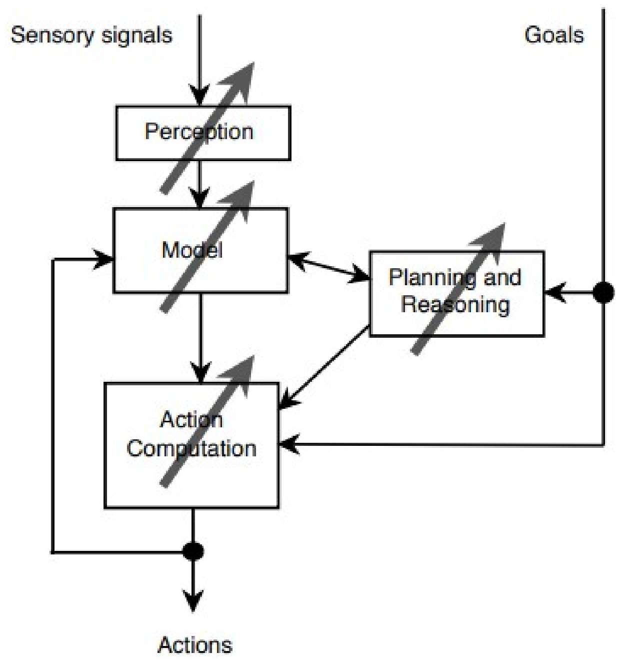


Figure 1.1: An AI System

1.1.2 Difference Between AI and ML[1]

AI is a bigger concept to create intelligent machines that can simulate human thinking capability and behavior, whereas, machine learning is an application or subset of AI that allows machines to learn from data without being programmed explicitly.

Artificial Intelligence

Artificial intelligence is a field of computer science which makes a computer system that can mimic human intelligence. It is comprised of two words "**Artificial**" and "**intelligence**", which means "a human-made thinking power." Hence we can define it as,

Artificial intelligence is a technology using which we can create intelligent systems that can simulate human intelligence.

The Artificial intelligence system does not require to be pre-programmed, instead of that, they use such algorithms which can work with their own intelligence. It involves machine learning algorithms such as Reinforcement learning algorithm and deep learning neural networks. AI is being used in multiple places such as Siri, Google?s AlphaGo, AI in Chess playing, etc.

Based on capabilities, AI can be classified into three types:

- **Weak AI**
- **General AI**
- **Strong AI**

Currently, we are working with weak AI and general AI. The future of AI is Strong AI for which it is said that it will be intelligent than humans.

Machine learning

Machine learning is about extracting knowledge from the data. It can be defined as,

Machine learning is a subfield of artificial intelligence, which enables machines to learn from past data or experiences without being explicitly programmed.

Machine learning enables a computer system to make predictions or take some decisions using historical data without being explicitly programmed. Machine learning uses a massive amount of structured and semi-structured data so that a machine learning model can generate accurate result or give predictions based on that data.

Machine learning works on algorithm which learn by it?s own using historical data. It works only for specific domains such as if we are creating a machine learning model to detect pictures of dogs, it will only give result for dog images, but if we provide a new data like cat image then it will become unresponsive. Machine learning is being used in various places such as for online recommender system, for Google search algorithms, Email spam filter, Facebook Auto friend tagging suggestion, etc.

It can be divided into three types:

- **Supervised learning**
- **Reinforcement learning**
- **Unsupervised learning**

Machine learning is one subfield of AI. The core principle here is that machines take data and "learn" for themselves. It's currently the most promising tool in the AI kit for businesses. ML systems can quickly apply knowledge and training from large data sets to excel at facial recognition, speech recognition, object recognition, translation, and many other tasks. Unlike hand-coding a software program with specific instructions to complete a task, ML allows a system to learn to recognize patterns on its own and make predictions.

1.2 OVERVIEW OF PROJECT

Movie Recommendation System:

This application will firstly ask us to login in our account if we are an existing user or register for a new account in case we have not done it earlier. Then it will ask us to chose some of our movies we like and rate them according to us. Then it will match our records with records of any critics in a database and recommend other movies to us which we may like. It will then recommend a movie to us.

What are Recommendation systems?[4]

Recommendation systems are becoming increasingly important in today's extremely busy world. People are always short on time with the myriad tasks they need to accomplish in the limited 24 hours. Therefore, the recommendation systems are important as they help them make the right choices, without having to expend their cognitive resources.

The purpose of a recommendation system basically is to search for content that would be interesting to an individual. Moreover, it involves a number of factors to create personalised lists of useful and interesting content specific to each user/individual. Recommendation systems are Artificial Intelligence based algorithms that skim through all possible options and create a customized list of items that are interesting and relevant to an individual. These results are based on their profile, search/browsing history, what other people with similar traits/demographics are watching, and how likely are you to watch those movies. This is achieved through predictive modeling and heuristics with the data available.

Use-cases of Recommendation systems

Recommendations are not a new concept. Even when e-commerce was not that prominent, the sales staff in retail stores recommended items to the customers for the purpose of upselling and cross-selling, and ultimately maximise profit. The aim of recommendation systems is just the same.
Another objective of the recommendation system is to achieve customer loyalty by providing relevant content and maximising the time spent by a user on your website or channel. This also helps in increasing customer engagement.

On the other hand, ad budgets can be optimized by showcasing products and services only to those who have a propensity to respond to them.

Why Recommendation systems?

– They help the user find items of their interest
– Helps the item provider to deliver their items to the right user
  – To identify the most relevant products for each user
  – Showcase personalised content to each user
  – Suggest top offers and discounts to the right user
– Websites can improve user-engagement
– It increases revenues for business through increased consumption

Real-World examples

Here are some of the examples of the pioneers in creating algorithms for recommendation systems and using them to serve their customers better in a personalized manner. These are:

GroupLens:
– Helped in developing initial recommender systems by pioneering collaborative filtering model
– It also provided many data-sets to train models including MovieLens and BookLens
Amazon:
– Implemented commercial recommender systems
– They also implemented a lot of computational improvements
Netflix Prize:
– Pioneered Latent Factor/ Matrix Factorization models
Google-Youtube:
– Hybrid Recommendation Systems
– Deep Learning based systems
– Social Network Recommendations

Various types of recommendation systems

– Popularity based recommendation systems

– Classification model based

– Content based recommendations

– Nearest neighbour collaborative filtering

1. User-based
2. Item-based

How to build a popularity based recommendation system in Python?

For this exercise, we will consider the MovieLens small dataset, and focus on two files, i.e., the movies.csv and ratings.csv.
Movies.csv has three fields namely:

1. MovieId – It has a unique id for every movie
2. Title – It is the name of the movie
3. Genre – The genre of the movie

The ratings.csv file has four fields namely:

1. Userid – The unique id for every user who has rated one or multiple movies
2. MovieId – The unique id for each movie
3. Rating – The rating given to a user to a movie
4. Timestamp – When was the rating given to a specific movie

SOFTWARE REQUIREMENT ANALYSIS

2.1 Pycharm:[5]

PyCharm is the most popular IDE used for Python scripting language. This chapter will give you an introduction to PyCharm and explains its features.
PyCharm offers some of the best features to its users and developers in the following aspects:

- ➢ Code completion and inspection
- ➢ Advanced debugging
- ➢ Support for web programming and frameworks such as Django and Flask

Features of PyCharm

Besides, a developer will find PyCharm comfortable to work with because of the features mentioned below:

Code Completion: PyCharm enables smoother code completion whether it is for built in or for an external package.

SQLAlchemy as Debugger: You can set a breakpoint, pause in the debugger and can see the SQL representation of the user expression for SQL Language code.

Git Visualization in Editor: When coding in Python, queries are normal for a developer. You can check the last commit easily in PyCharm as it has the blue sections that can define the difference between the last commit and the current one.

Code Coverage in Editor: You can run .py files outside PyCharm Editor as well marking it as code coverage details elsewhere in the project tree, in the summary section etc.

Package Management: All the installed packages are displayed with proper visual representation. This includes list of installed packages and the ability to search and add new packages.

Local History: Local History is always keeping track of the changes in a way that complements like Git. Local history in PyCharm gives complete details of what is needed to rollback and what is to be added.

Refactoring: Refactoring is the process of renaming one or more files at a time and PyCharm includes various shortcuts for a smooth refactoring process.

User Interface of PyCharm Editor:

The user interface of PyCharm editor is shown in the screenshot given below. Observe that the editor includes various features to create a new project or import from an existing project. From the screenshot shown below, you can see the newly created project Demo and the site-packages folder for package management along with various other folders.

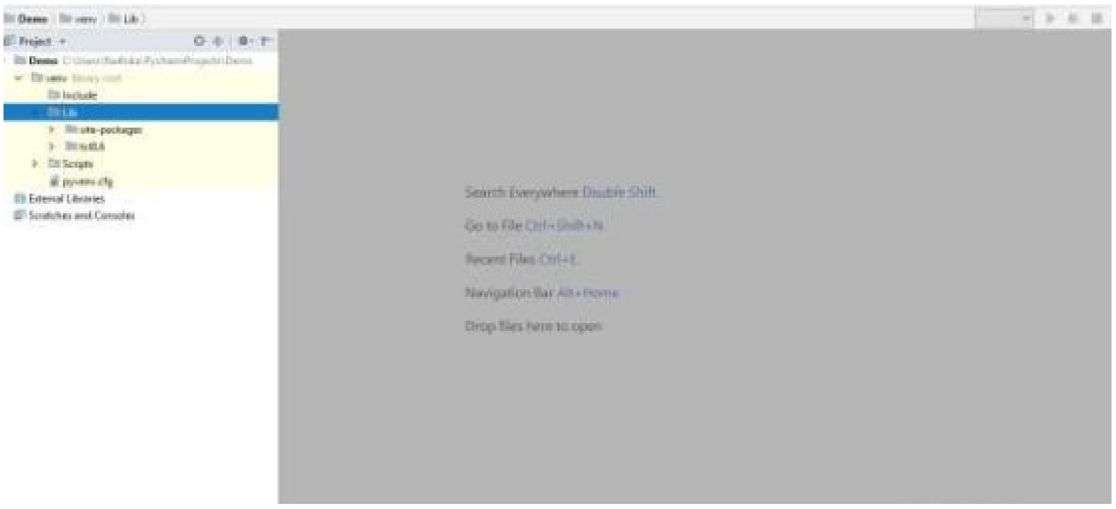


Fig 2.1 Pycharm Interface

2.2 JUPYTER NOTEBOOK[9][6]

The notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components:

A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output.

Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

Main features of the web application

- In-browser editing for code, with automatic syntax highlighting, indentation, and tab completion/introspection.
- The ability to execute code from the browser, with the results of computations attached to the code which generated them.
- Displaying the result of computation using rich media representations, such as HTML, LaTeX, PNG, SVG, etc. For example, publication-quality figures rendered by the matplotlib library, can be included inline.
- In-browser editing for rich text using the Markdown markup language, which can provide commentary for the code, is not limited to plain text.
- The ability to easily include mathematical notation within markdown cells using LaTeX, and rendered natively by MathJax.

Notebooks and privacy

Because you use Jupyter in a web browser, some people are understandably concerned about using it with sensitive data. However, if you followed the standard install instructions, Jupyter is actually running on your own computer. If the URL in the address bar starts with *http://localhost:* or *http://127.0.0.1:*, it's your computer acting as the server. Jupyter doesn't send your data anywhere else—and as it's open source, other people can check that we're being honest about this.

You can also use Jupyter remotely: your company or university might run the server for you, for instance. If you want to work with sensitive data in those cases, talk to your IT or data protection staff about it.

Notebook user interface

When you create a new notebook document, you will be presented with the notebook name, a menu bar, a toolbar and an empty code cell.

Notebook name: The name displayed at the top of the page, next to the Jupyter logo, reflects the name of the . ipynb file. Clicking on the notebook name brings up a dialog which allows you to rename it. Thus, renaming a notebook from “Untitled0” to “My first notebook” in the browser, renames the *Untitled0.ipynb* file to *My first notebook.ipynb*.
Menu bar: The menu bar presents different options that may be used to manipulate the way the notebook functions.
Toolbar: The tool bar gives a quick way of performing the most-used operations within the notebook, by clicking on an icon.
Code cell: The default type of cell; read on for an explanation of cells.

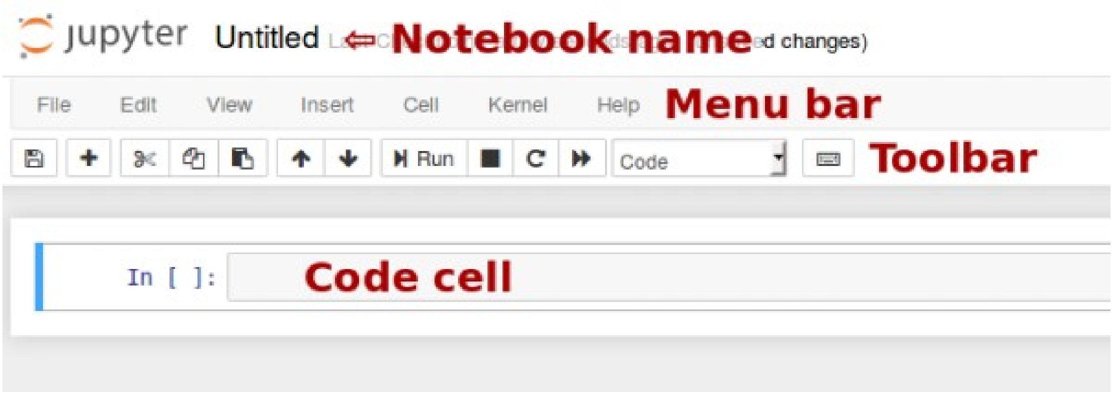


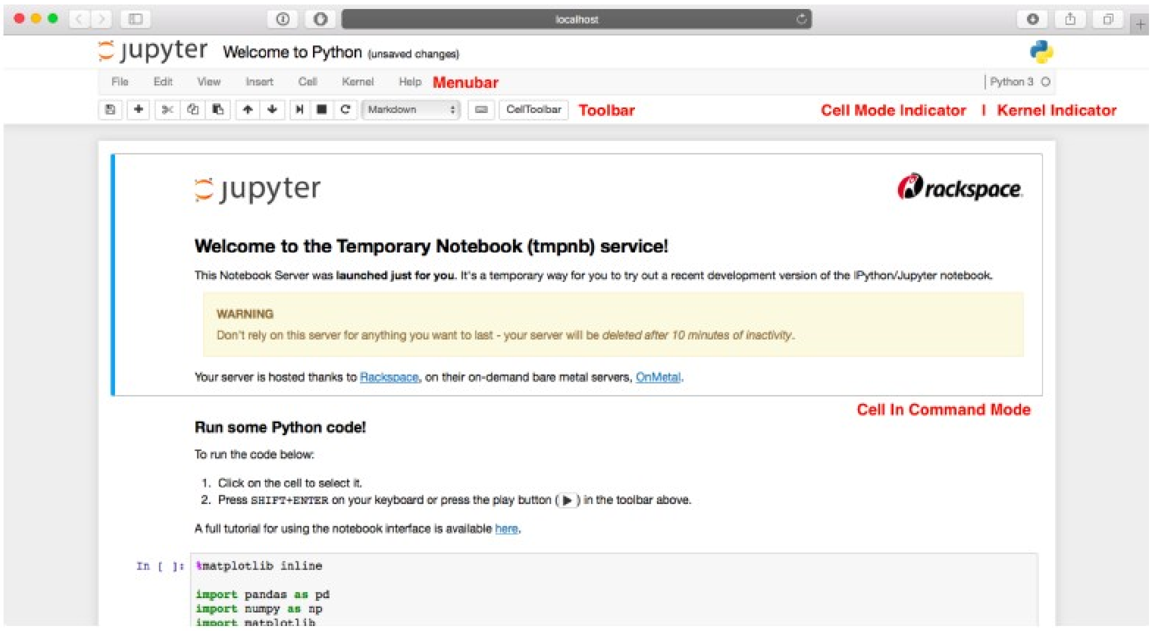
Fig 2.2 Jupyter Notebook Interface

The Jupyter Notebook is an interactive computing environment that enables users to author notebook documents that include: - Live code - Interactive widgets - Plots - Narrative text - Equations - Images - Video
These documents provide a complete and self-contained record of a computation that can be converted to various formats and shared with others using email, Dropbox, version control systems (like git/GitHub) or nbviewer.jupyter.org.

Components

The Jupyter Notebook combines three components:

- **The notebook web application:** An interactive web application for writing and running code interactively and authoring notebook documents.
- **Kernels:** Separate processes started by the notebook web application that runs users' code in a given language and returns output back to the notebook web application. The kernel also handles things like computations for interactive widgets, tab completion and introspection.
- **Notebook documents:** Self-contained documents that contain a representation of all content visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects. Each notebook document has its own kernel



jupyter

Welcome to the Temporary Notebook (tmpnb) service!

This Notebook Server was **launched just for you.** It's a temporary way for you to try out a recent development version of the IPython/Jupyter notebook.

WARNING
Don't rely on this server for anything you want to last - your server will be *deleted after 10 minutes of inactivity.*

Your server is hosted thanks to Rackspace, on their on-demand bare metal servers, OnMetal.

Run some Python code!

To run the code below:

1. Click on the cell to select it.
2. Press `SHIFT+ENTER` on your keyboard or press the play button (▶) in the toolbar above.

A full tutorial for using the notebook interface is available here.

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Fig 2.3 Jupyter Notebook Welcome Screen

Modules and Dependencies

3.1 Tkinter[7]

Tkinter ("Tk Interface")is python's standard cross-platform package for creating graphical user interfaces (GUIs). It provides access to an underlying Tcl interpreter with the Tk toolkit, which itself is a cross-platform, multilanguage graphical user interface library.

Tkinter isn't the only GUI library for python, but it is the one that comes standard. Additional GUI libraries that can be used with python include wxPython, PyQt, and kivy.

Tkinter's greatest strength is its ubiquity and simplicity. It works out of the box on most platforms (linux, OSX, Windows), and comes complete with a wide range of widgets necessary for most common tasks (buttons, labels, drawing canvas, multiline text, etc).

As a learning tool, tkinter has some features that are unique among GUI toolkits, such as named fonts, bind tags, and variable tracing.

Importing in python 3.x

Although functionality did not change much between python 2 and 3, the names of all of the tkinter modules have changed. The following is a typical set of import statements for python 3.x:

```
import tkinter as tk
from tkinter import filedialog
from tkinter import ttk
```

Installation or Setup

Tkinter comes pre-installed with the Python installer binaries for Mac OS X and the Windows platform. So if you install Python from the official binaries for Mac OS X or Windows platform, you are good to go with Tkinter.

For Debian versions of Linux you have to install it manually by using the following commands.

For Python 3

```
    sudo apt-get install python3-tk
```

For Python 2.7

```
sudo apt-get install python-tk
```

Linux distros with yum installer can install tkinter module using the command:

```
yum install tkinter
```

Putting it all together:

```
import tkinter as tk # Python 3.x Version
#import Tkinter as tk # Python 2.x Version

root = tk.Tk()

label = tk.Label(root, text="Hello World!") # Create a text label
label.pack(padx=20, pady=20) # Pack it into the window

root.mainloop()
```

And something like this should pop up:

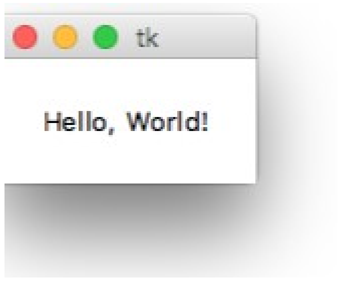


Fig3.1 Tkinter GUI Interface

3.2 PANDAS[8]

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. The name Pandas is derived from the word Panel Data – an Econometrics from Multidimensional data.

In 2008, developer Wes McKinney started developing pandas when in need of high performance, flexible tool for analysis of data.

Prior to Pandas, Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data — load, prepare, manipulate, model, and analyze.

Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Key features of pandas library

1. Fast and efficient DataFrame object with default and customized indexing.
2. Tools for loading data into in-memory data objects from different file formats.
3. Data alignment and integrated handling of missing data.
4. Reshaping and pivoting of date sets.
5. Label-based slicing, indexing and subsetting of large data sets.
6. Columns from a data structure can be deleted or inserted.
7. Group by data for aggregation and transformations.
8. High performance merging and joining of data.
9. Time Series functionality.

3.3 NUMPY[1]

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Numeric, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project.

Operations using NumPy library

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy - A Replacement to the MatLab

NumPy is often used along with packages like **SciPy** (Scientific Python) and **Mat−plotlib** (plotting library). This combination is widely used as a replacement for MatLab, a popular platform for technical computing. However, Python alternative to MatLab is now seen as a more modern and complete programming language.

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with the corresponding element in another sequence of the same length. If the data are stored in two Python lists, a and b, we could iterate over each element:

```
c = []
for i in range(len(a)):
        c.append(a[i]*b[i])
```

This produces the correct answer, but if a and b each contain millions of numbers, we will pay the price for the inefficiencies of looping in Python. We could accomplish the same task much more quickly in C by writing (for clarity we neglect variable declarations and initializations, memory allocation, etc.)

```
for (i = 0; i < rows; i++): {
        c[i] = a[i]*b[i];
}
```

3.4 SCIPY[1]

SciPy is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. With SciPy an interactive Python session becomes a data-processing and systemprototyping environment rivaling sytems such as MATLAB, IDL, Octave, R-Lab, and SciLab.

The additional benefit of basing SciPy on Python is that this also makes a powerful programming language available for use in developing sophisticated programs and specialized applications. Scientific applications using SciPy benefit from the development of additional modules in numerous niche's of the software landscape by developers across the world. Everything from parallel programming to web and data-base subroutines and classes have been made available to the Python programmer. All of this power is available in addition to the mathematical libraries in SciPy.

This tutorial will acquaint the first-time user of SciPy with some of its most important features. It assumes that the user has already installed the SciPy package. Some general Python facility is also assumed, such as could be acquired by working through the Python distribution's Tutorial. For further introductory help the user is directed to the Numpy documentation.

For brevity and convenience, we will often assume that the main packages (numpy, scipy, and matplotlib) have been imported as:

```
>>> import numpy as np
>>> import scipy as sp
>>> import matplotlib as mpl
>>> import matplotlib.pyplot as plt
```

These are the import conventions that our community has adopted after discussion on public mailing lists. You will see these conventions used throughout NumPy and SciPy source code and documentation. While we obviously don't require you to follow these conventions in your own code, it is highly recommended.

3.5 MATPLOTLIB[1]

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an objectoriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Matplotlib has a procedural interface named the Pylab, which is designed to resemble MATLAB, a proprietary programming language developed by MathWorks. Matplotlib along with NumPy can be considered as the open source equivalent of MATLAB.

Matplotlib was originally written by John D. Hunter in 2003. The current stable version is 2.2.0 released in January 2018.

Matplotlib and its dependency packages are available in the form of wheel packages on the standard Python package repositories and can be installed on Windows, Linux as well as MacOS systems using the pip package manager.

```
pip3 install matplotlib
```

Important features of Seaborn:

1. Built in themes for styling matplotlib graphics.
2. Visualizing univariate and bivariate data.
3. Fitting in and visualizing linear regression models.
4. Plotting statistical time series data.
5. Seaborn works well with NumPy and Pandas data structures.
6. It comes with built in themes for styling Matplotlib graphics.

3.6 SeabornPycharm[1]

Seaborn is complimentary to Matplotlib and it specifically targets statistical data visualization. But it goes even further than that: Seaborn extends Matplotlib and that's why it can address the two biggest frustrations of working with Matplotlib. One of these hard things or frustrations had to do with the default Matplotlib parameters. Seaborn works with different parameters, which undoubtedly speaks to those users that don't use the default looks of the Matplotlib plots.

3.7 GIT and GITHUB[10]

3.7.1 What is Git?

Git is a revision control system. It is designed to help you keep track of collections of files which stem from a common source and undergo modifications over time. The files tend to be human generated text. It is very good at managing source code repositories, but it can also be used to manage other things, such as configuration files and text documents. It is not, however, a file backup system.

Git encourages a distributed style of project development. Each contributor to a project has their own complete repository. Changes are shared between repositories by pushing changes to, or pulling changes from other repositories. Collaboration between developers is greatly enhanced by websites such as github, bitbucket and gitorious which provide convenient interfaces to managing multiple repositories.

There are many alternatives to git which each have their pros and cons. Two of the more popular alternatives are:

- Subversion is particularly suited to a centralised model of development.
- Mercurial is very similar to Git, but is sometimes considered more user friendly.

3.7.2 WHAT IS GITHUB?[5]

GitHub is a cloud-based hosting service that lets you manage Git repositories. If you have open-source projects that use Git, then GitHub is designed to help you better manage them.

Version control systems keep these revisions straight, storing the modifications in a central repository. This allows developers to easily collaborate, as they can download a new version of the software, make changes, and upload the newest revision. Every developer can see these new changes, download them, and contribute.

Similarly, people who have nothing to do with the development of a project can still download the files and use them. Most Linux users should be familiar with this process, as using Git, Subversion, or some other similar method is pretty common for downloading needed files—especially in preparation for compiling a program from source code (a rather common practice for Linux geeks).

We've established that Git is a version control system, similar but better than the many alternatives available. So, what makes GitHub so special? Git is a command-line tool, but the center around which all things involving Git revolve is the hub—GitHub.com—where developers store their projects and network with like minded people.

Repository

A repository (usually abbreviated to "repo") is a location where all the files for a particular project are stored. Each project has its own repo, and you can access it with a unique URL.

Forking a Repo

"Forking" is when you create a new project based off of another project that already exists. This is an amazing feature that vastly encourages the further development of programs and other projects. If you find a project on GitHub that you'd like to contribute to, you can fork the repo, make the changes you'd like, and release the revised project as a new repo. If the original repository that you forked to create your new project gets updated, you can easily add those updates to your current fork.

Pull Requests

You've forked a repository, made a great revision to the project, and want it to be recognized by the original developers—maybe even included in the official project/repository. You can do so by creating a pull request. The authors of the original repository can see your work, and then choose whether or not to accept it into the official project. Whenever you issue a pull request, GitHub provides a perfect medium for you and the main project's maintainer to communicate.

Social networking

The social networking aspect of GitHub is probably its most powerful feature, allowing projects to grow more than just about any of the other features offered. Each user on GitHub has their own profile that acts like a resume of sorts, showing your past work and contributions to other projects via pull requests.

Project revisions can be discussed publicly, so a mass of experts can contribute knowledge and collaborate to advance a project forward. Before the advent of GitHub, developers interested in contributing to a project would usually need to find some means of contacting the authors—probably by email—and then convince them that they can be trusted and their contribution is legit.

Changelogs

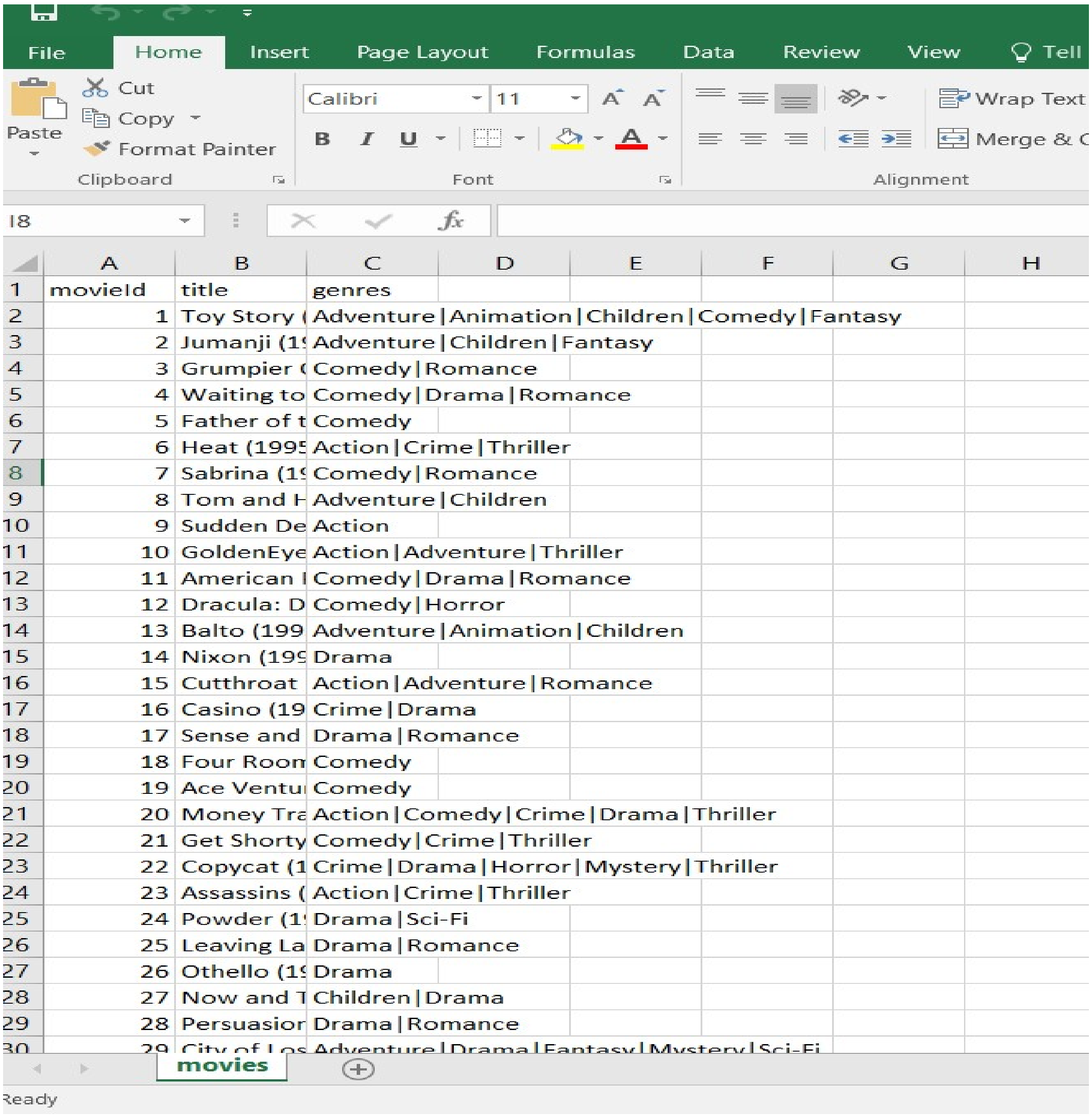
When multiple people collaborate on a project, it's hard to keep track revisions—who changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository.

IMPLEMENTATION DETAILS

4.1 DATA EXPLORATION

We are using the movies.csv and ratings.csv datasets for our movie recommender, here are the sample of the datasets as an image:

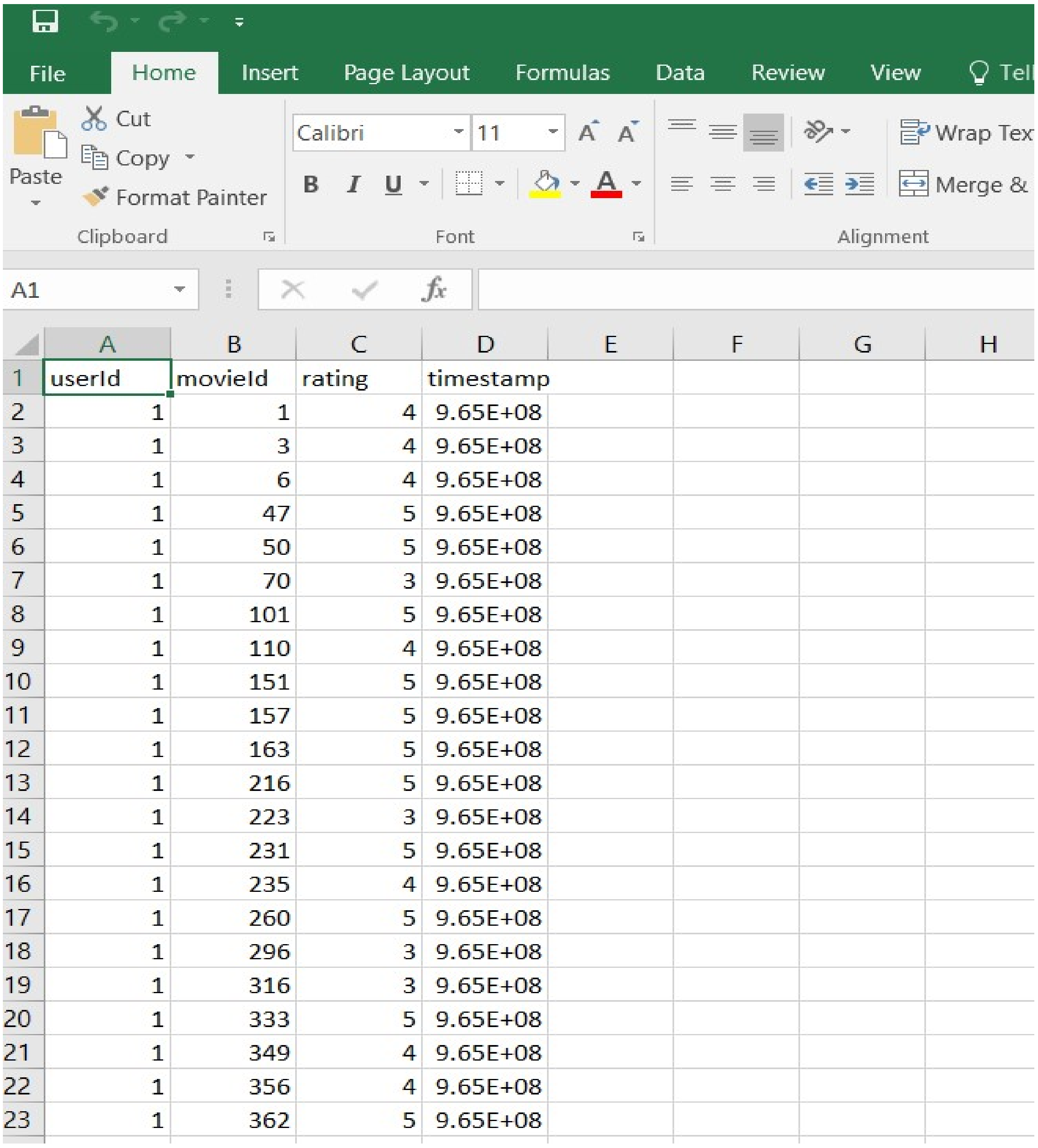
Movies.csv



	A	B	C	D	E	F	G	H
1	movieId	title	genres					
2	1	Toy Story (	Adventure|Animation|Children|Comedy|Fantasy					
3	2	Jumanji (1!	Adventure|Children|Fantasy					
4	3	Grumpier (	Comedy|Romance					
5	4	Waiting to	Comedy|Drama|Romance					
6	5	Father of t	Comedy					
7	6	Heat (1995	Action|Crime|Thriller					
8	7	Sabrina (1!	Comedy|Romance					
9	8	Tom and H	Adventure|Children					
10	9	Sudden De	Action					
11	10	GoldenEye	Action|Adventure|Thriller					
12	11	American I	Comedy|Drama|Romance					
13	12	Dracula: D	Comedy|Horror					
14	13	Balto (199	Adventure|Animation|Children					
15	14	Nixon (199	Drama					
16	15	Cutthroat	Action|Adventure|Romance					
17	16	Casino (19	Crime|Drama					
18	17	Sense and	Drama|Romance					
19	18	Four Room	Comedy					
20	19	Ace Ventu	Comedy					
21	20	Money Tra	Action|Comedy|Crime|Drama|Thriller					
22	21	Get Shorty	Comedy|Crime|Thriller					
23	22	Copycat (1	Crime|Drama|Horror|Mystery|Thriller					
24	23	Assassins (	Action|Crime|Thriller					
25	24	Powder (1!	Drama|Sci-Fi					
26	25	Leaving La	Drama|Romance					
27	26	Othello (1!	Drama					
28	27	Now and T	Children|Drama					
29	28	Persuasior	Drama|Romance					
30	29	City of Los	Adventure|Drama|Fantasy|Mystery|Sci-Fi					

Fig 4.1 Movies.csv Data Sample

Ratings.csv

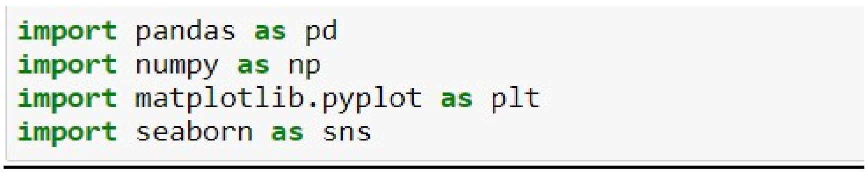


	A	B	C	D
1	userId	movieId	rating	timestamp
2	1	1	4	9.65E+08
3	1	3	4	9.65E+08
4	1	6	4	9.65E+08
5	1	47	5	9.65E+08
6	1	50	5	9.65E+08
7	1	70	3	9.65E+08
8	1	101	5	9.65E+08
9	1	110	4	9.65E+08
10	1	151	5	9.65E+08
11	1	157	5	9.65E+08
12	1	163	5	9.65E+08
13	1	216	5	9.65E+08
14	1	223	3	9.65E+08
15	1	231	5	9.65E+08
16	1	235	4	9.65E+08
17	1	260	5	9.65E+08
18	1	296	3	9.65E+08
19	1	316	3	9.65E+08
20	1	333	5	9.65E+08
21	1	349	4	9.65E+08
22	1	356	4	9.65E+08
23	1	362	5	9.65E+08

Fig 4.1 Ratings.csv Data Sample

Data Exploration and manipulation

Step1: Importing necessary python library.

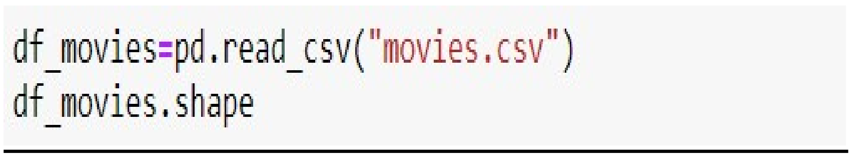


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Fig 4.3 Importing Panda and Numpy

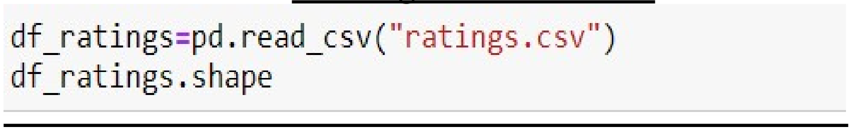
Step2: Reading the datasets .
Dataframe.read()

The first step is to read our dataset which is in the csv format. The Pandas library makes it very easy to read CSV files using the **read_csv()** function. pd.read_csv() opens, analyzes, and reads the CSV file provided, and stores the data in a DataFrame “**df_movies**”.



```
df_movies=pd.read_csv("movies.csv")
df_movies.shape
```

Reading movies.csv data



```
df_ratings=pd.read_csv("ratings.csv")
df_ratings.shape
```

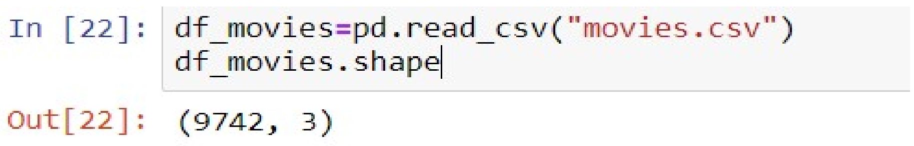
Reading ratings.csv data

Dataframe.Shape()

shape is a tuple that gives dimensions of the array.. shape is a tuple that gives you an indication of the number of dimensions in the array. So in your case, since the index value of Y. shape[0] is 0, your are working along the first dimension of your array.

Here by using “df_movies.shape()” we are displaying the number of rows and columns present in the dataset.

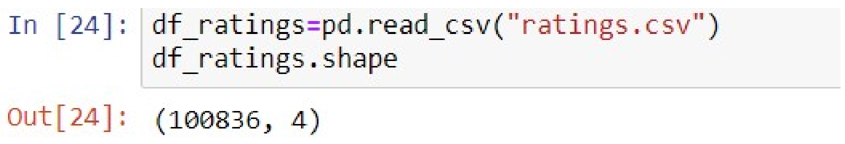
Output:



```
In [22]: df_movies=pd.read_csv("movies.csv")
         df_movies.shape

Out[22]: (9742, 3)
```

Implemented screenshot

The number of rows and column on the dataset is 9742,3 respectively in the movies.csv file.



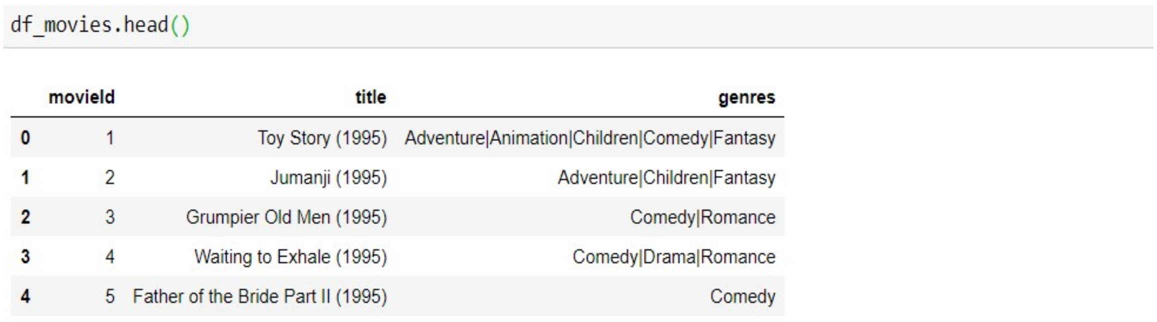
```
In [24]: df_ratings=pd.read_csv("ratings.csv")
         df_ratings.shape

Out[24]: (100836, 4)
```

Implemented screenshot

The number of rows and column on the dataset is 1000836 , 4 respectively in the ratings.csv file.

Dataframe.head():

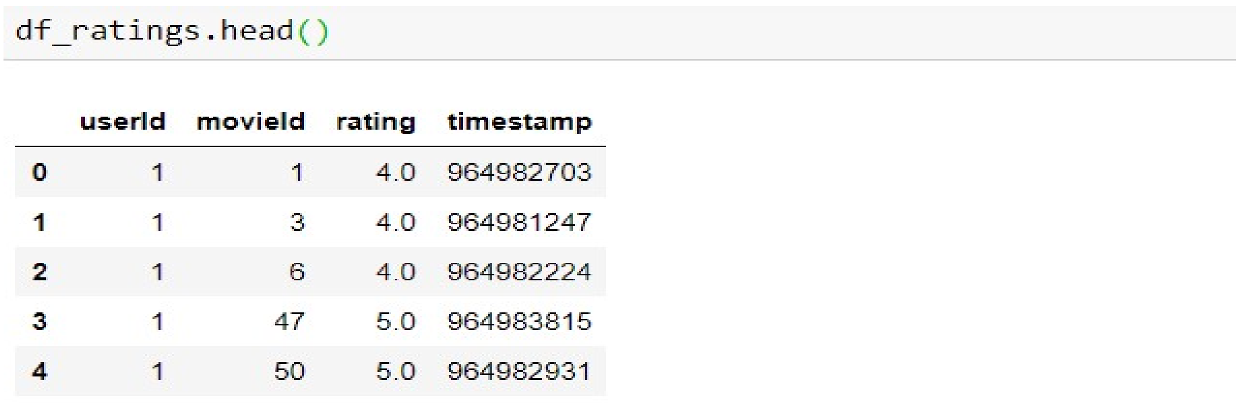
df. **head()** Returns the first 5 rows of the dataframe. To override the default, you may insert a value between the parenthesis to change the number of rows returned.



```
df_movies.head()
```

	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

The head fuction of the movies.csv file



```
df_ratings.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

The head function of the ratings.csv file

Dataframe.describe()

Pandas **describe()** is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.It takes percentile, include, exclude as the parameter.

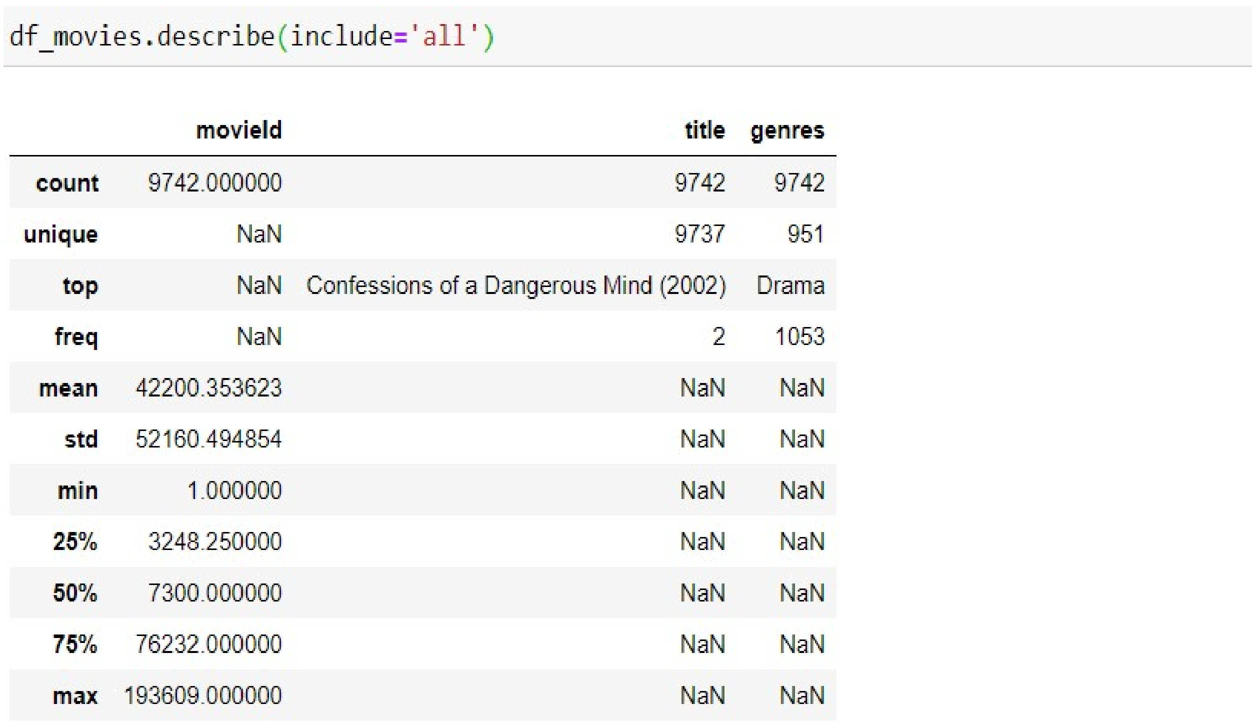
Syntax: *DataFrame.describe(percentiles=None, include=None, exclude=None)*

Parameters:

Percentiles: list like the datatypes of number between 0-1 to return the respective percentile.

Include: list of the data type included while describing the dataframe (default is none).

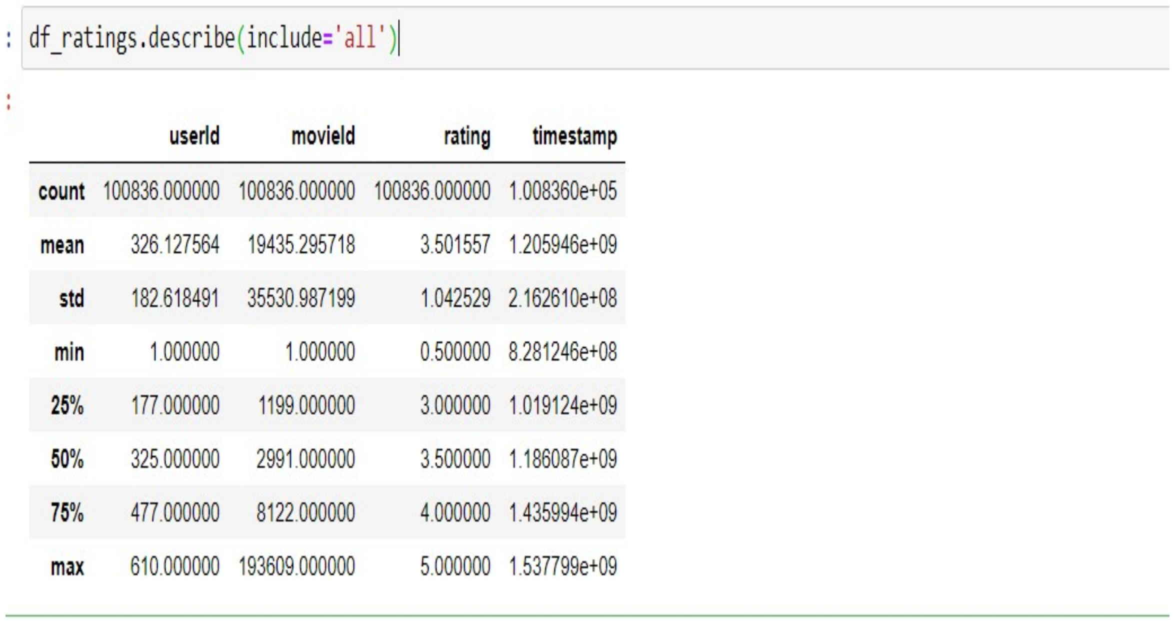
Exclude: list of the data types excluded while describing the dataframe(default is none).



```
df_movies.describe(include='all')
```

	movieId	title	genres
count	9742.000000	9742	9742
unique	NaN	9737	951
top	NaN	Confessions of a Dangerous Mind (2002)	Drama
freq	NaN	2	1053
mean	42200.353623	NaN	NaN
std	52160.494854	NaN	NaN
min	1.000000	NaN	NaN
25%	3248.250000	NaN	NaN
50%	7300.000000	NaN	NaN
75%	76232.000000	NaN	NaN
max	193609.000000	NaN	NaN

Implemented screenshot



```
: df_ratings.describe(include='all')
```

	userId	movieId	rating	timestamp
count	100836.000000	100836.000000	100836.000000	1.008360e+05
mean	326.127564	19435.295718	3.501557	1.205946e+09
std	182.618491	35530.987199	1.042529	2.162610e+08
min	1.000000	1.000000	0.500000	8.281246e+08
25%	177.000000	1199.000000	3.000000	1.019124e+09
50%	325.000000	2991.000000	3.500000	1.186087e+09
75%	477.000000	8122.000000	4.000000	1.435994e+09
max	610.000000	193609.000000	5.000000	1.537799e+09

Implemented screenshot

Dataframe.dtypes():

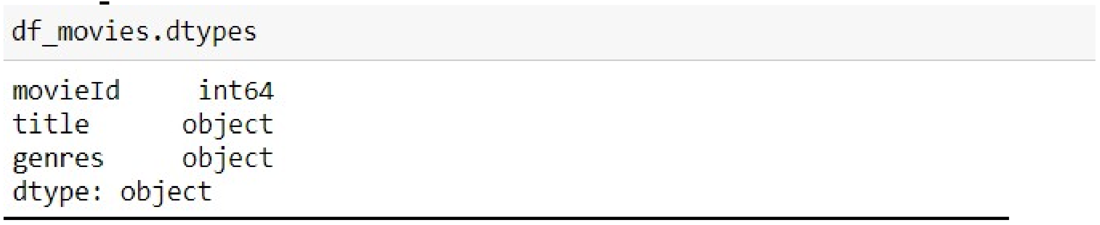
Pandas **DataFrame.dtypes** attribute return the dtypes in the DataFrame. It returns a Series with the data type of each column.

Syntax: dataframe.dtypes()

Parameter: none

Returns: data types of each column.

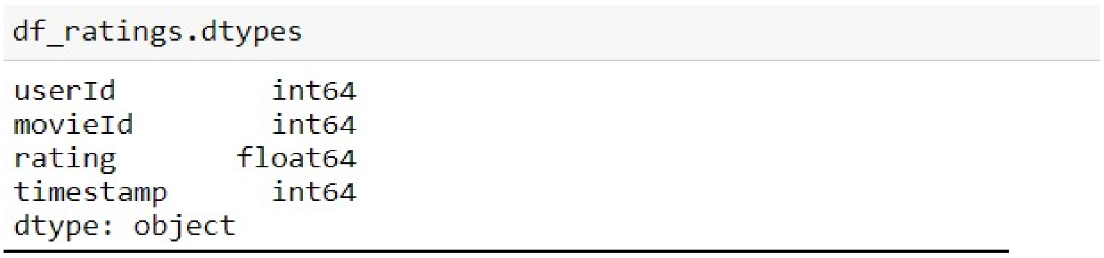
\



```
df_movies.dtypes
```

```
movieId     int64
title      object
genres     object
dtype: object
```

Implemented screenshot



```
df_ratings.dtypes
```

```
userId         int64
movieId        int64
rating       float64
timestamp      int64
dtype: object
```

Implemented screenshot

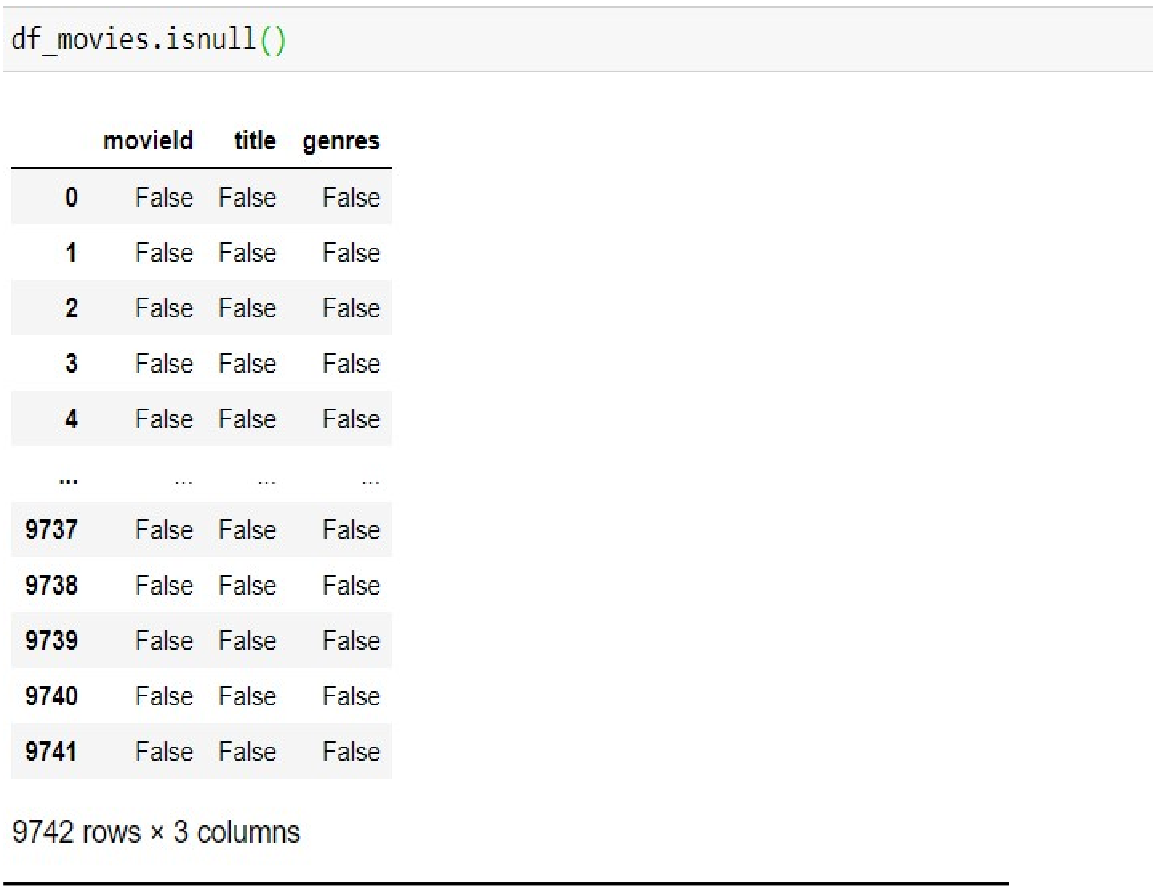
Dataframe.isnull():

While making a Data Frame from a csv file, many blank columns are imported as null value into the Data Frame which later creates problems while operating that data frame. Pandas isnull() and notnull() methods are used to check and manage NULL values in a data frame.

Syntax: pandas.isnull(“dataframe name”) OR dataframe.isnull().

Parameter: object of the dataframe.

Returns: dataframe of Boolean value which is true for NAN values.

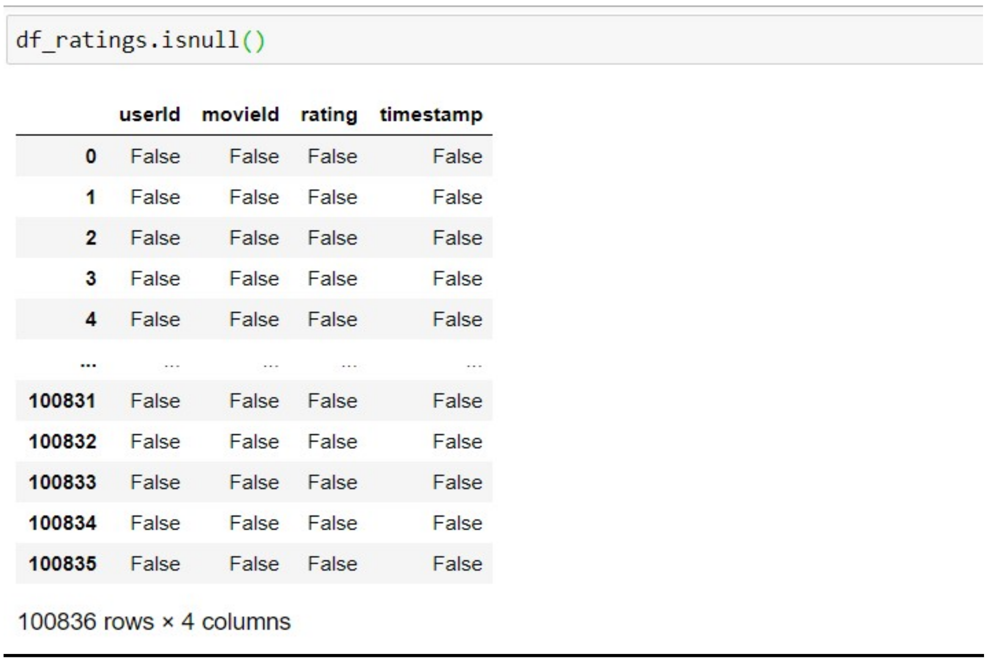


```
df_movies.isnull()
```

	movieId	title	genres
0	False	False	False
1	False	False	False
2	False	False	False
3	False	False	False
4	False	False	False
...	...	...	...
9737	False	False	False
9738	False	False	False
9739	False	False	False
9740	False	False	False
9741	False	False	False

9742 rows × 3 columns

Checking for the null value in the dataset movies.csv



```
df_ratings.isnull()
```

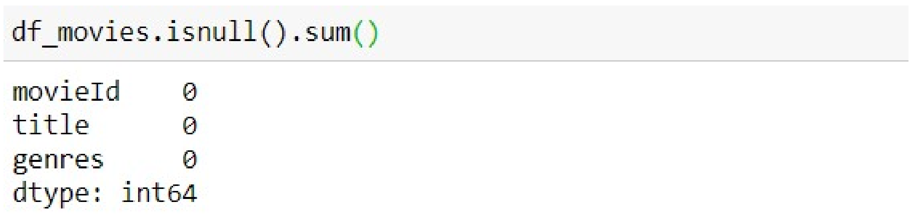
	userId	movieId	rating	timestamp
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
...	...	...	...	...
100831	False	False	False	False
100832	False	False	False	False
100833	False	False	False	False
100834	False	False	False	False
100835	False	False	False	False

100836 rows × 4 columns

Checking for the null value in the dataset ratings.csv

Dataframe.isnull().sum():

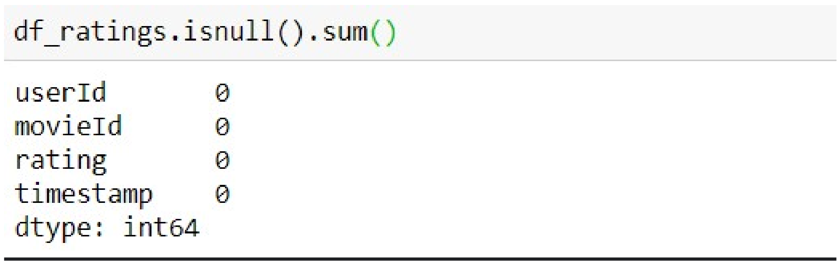
Dataframe.isnull(). **sum()**, It give the count of null values in a column. But the default axis for **sum()** is None, or 0 - which should be **summing** across the columns.



```
df_movies.isnull().sum()
```

```
movieId    0
title      0
genres     0
dtype: int64
```

Counting the number of values in movies.csv file



```
df_ratings.isnull().sum()
```

```
userId       0
movieId      0
rating       0
timestamp    0
dtype: int64
```

Counting the number of null value in ratings.csv

Dataframe.plot():

Python has many popular plotting libraries that make visualization easy. Some of them are **matplotlib**, **seaborn**, and **plotly**. It has great integration with matplotlib. We can plot a dataframe using **the plot()** method. But we need a dataframe to plot.

There are a number of plots available to interpret the data. Each graph is used for a purpose. Some of the plots are BarPlots, ScatterPlots, and Histograms, etc.

1. **Bar plots**:

   The plot.bar() function is used to vertical bar plot.A bar plot is a plot that presents categorical data with rectangular bars with lengths proportional to the values that they represent. A bar plot shows comparisons among discrete categories. One axis of the plot shows the specific categories being compared, and the other axis represents a measured value.

Syntax:

Dataframe.plot.bar(self, x=None, y=None)

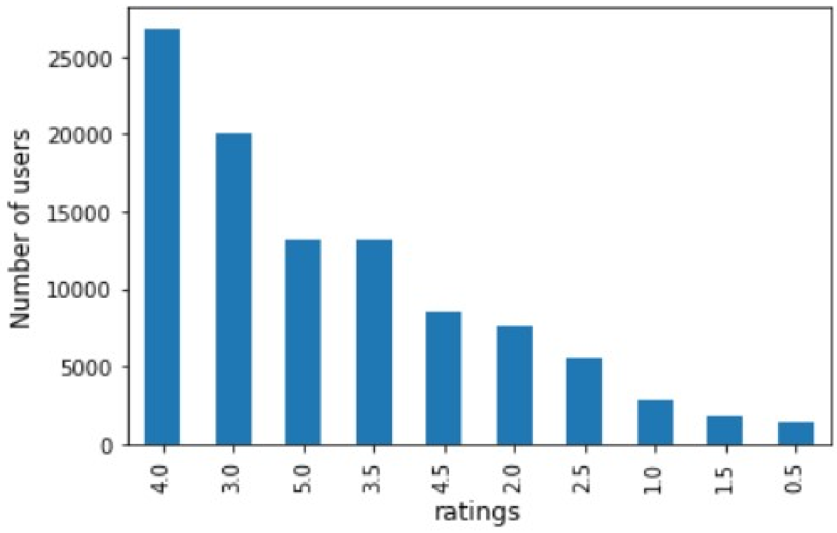
Parameters:

1. x: Allows plotting of one column versus another. If not specified, the index of the DataFrame is used.
2. y: Allows plotting of one column versus another. If not specified, all numerical columns are used.

Returns: Bar plot.

```
df_ratings['rating'].value_counts().plot(kind='bar')
plt.xlabel('ratings',fontsize=12)
plt.ylabel('Number of users',fontsize=12)
```

```
Text(0, 0.5, 'Number of users')
```



<u>Bar graph showing ratings given by the number of users.</u>

2. Hist plots:

The plot.hist() function is used to draw one histogram of the DataFrame's columns.

Syntax:

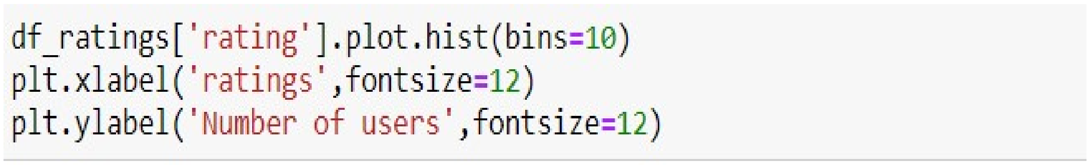
Dataframe.plot.hist(self, by=None, bins=10)

Parameters:

3. by: Column in the DataFrame to group by.
4. bins: Number of histogram bins to be used.

Returns:

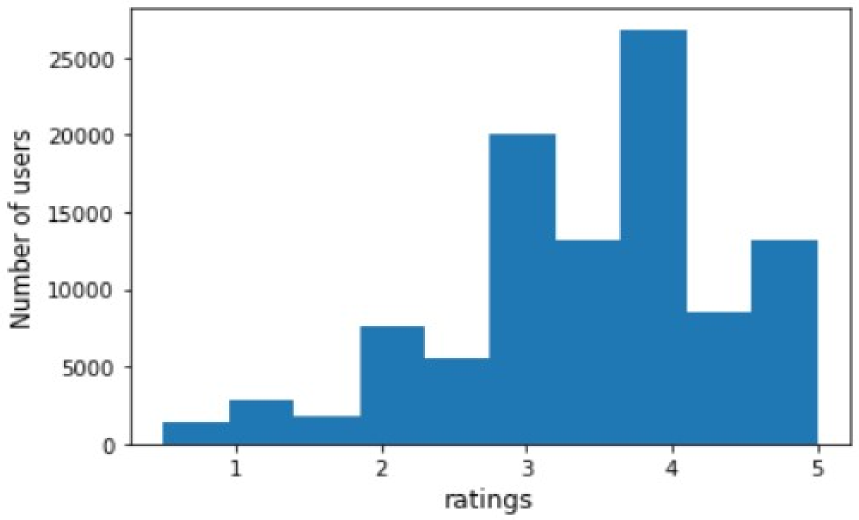
Returns a histogram plot.



```
df_ratings['rating'].plot.hist(bins=10)
plt.xlabel('ratings',fontsize=12)
plt.ylabel('Number of users',fontsize=12)
```

Implemented screenshot

Output:



Histogram showing rating given by the number of user in a particular range.

(the plotting shows that there is no presence of any outliers)

Dataframe.value_counts():

Dataframe.value_counts() function returns object containing counts of unique values. The resulting object will be in descending order so that the first element is the most frequently-occurring element. Excludes NA values by default.

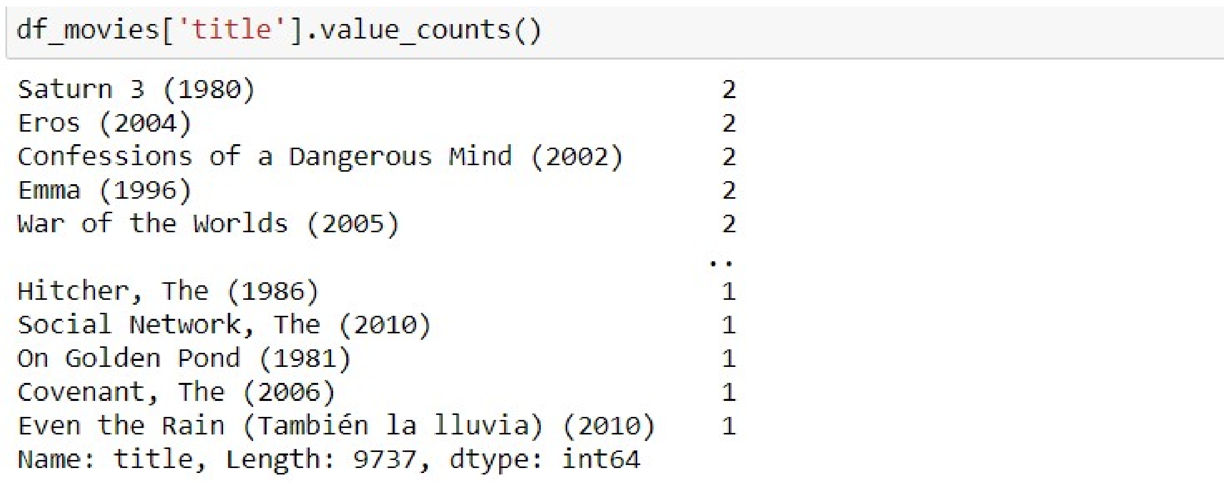
Syntax:

Dataframe.value_counts (self, normalize=False, sort=True, ascending=False, bins=None, dropna=True)

Parameters:

1. normalize: If True then the object returned will contain the relative frequencies of the unique values.
2. sort: Sort by frequencies.
3. ascending: Sort in ascending order.
4. bins: IRather than count values, group them into half-open bins, a convenience for pd.cut, only works with numeric data.
5. dropna: Don't include counts of NaN.

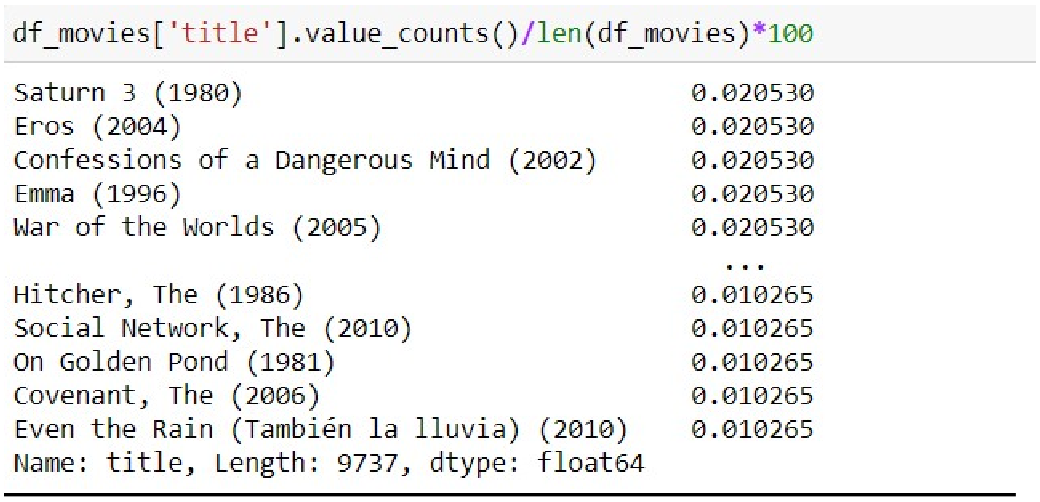
Return : Series.



```
df_movies['title'].value_counts()
```

```
Saturn 3 (1980)                                2
Eros (2004)                                    2
Confessions of a Dangerous Mind (2002)         2
Emma (1996)                                    2
War of the Worlds (2005)                       2
                                              ..
Hitcher, The (1986)                            1
Social Network, The (2010)                     1
On Golden Pond (1981)                          1
Covenant, The (2006)                           1
Even the Rain (También la lluvia) (2010)       1
Name: title, Length: 9737, dtype: int64
```

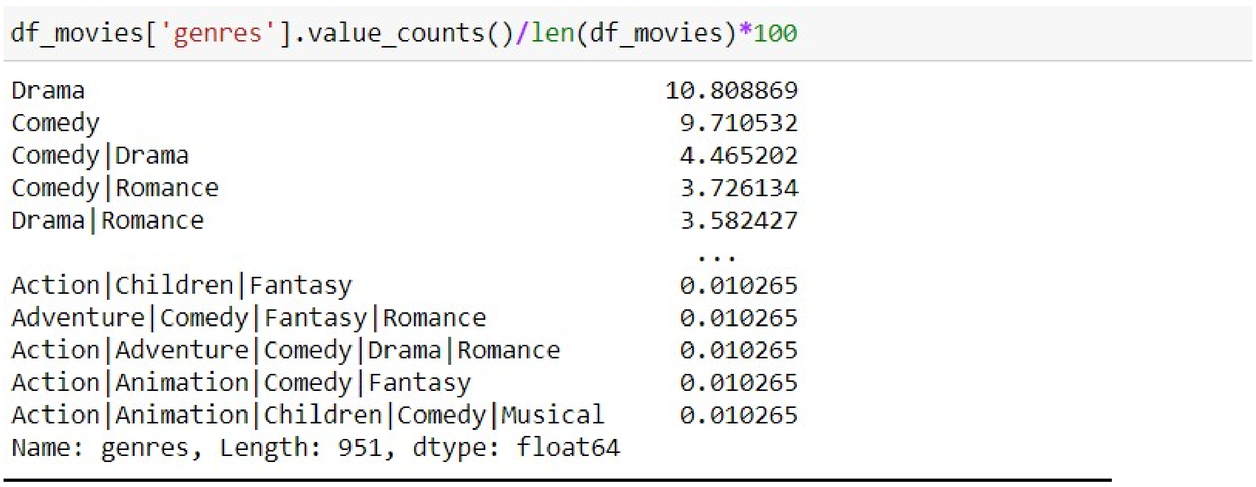
Counting the occurrence of the movies in the movies.csv dataset



```
df_movies['title'].value_counts()/len(df_movies)*100
```

```
Saturn 3 (1980)                              0.020530
Eros (2004)                                  0.020530
Confessions of a Dangerous Mind (2002)       0.020530
Emma (1996)                                  0.020530
War of the Worlds (2005)                     0.020530
                                               ...
Hitcher, The (1986)                          0.010265
Social Network, The (2010)                   0.010265
On Golden Pond (1981)                        0.010265
Covenant, The (2006)                         0.010265
Even the Rain (También la lluvia) (2010)     0.010265
Name: title, Length: 9737, dtype: float64
```

<u>Percentage of the movie in the movies.csv dataset</u>



```
df_movies['genres'].value_counts()/len(df_movies)*100
```

```
Drama                                        10.808869
Comedy                                        9.710532
Comedy|Drama                                  4.465202
Comedy|Romance                                3.726134
Drama|Romance                                 3.582427
                                                ...
Action|Children|Fantasy                       0.010265
Adventure|Comedy|Fantasy|Romance              0.010265
Action|Adventure|Comedy|Drama|Romance         0.010265
Action|Animation|Comedy|Fantasy               0.010265
Action|Animation|Children|Comedy|Musical      0.010265
Name: genres, Length: 951, dtype: float64
```

<u>Percentage of the genres in the movies.csv dataset</u>

Pandas.merge():

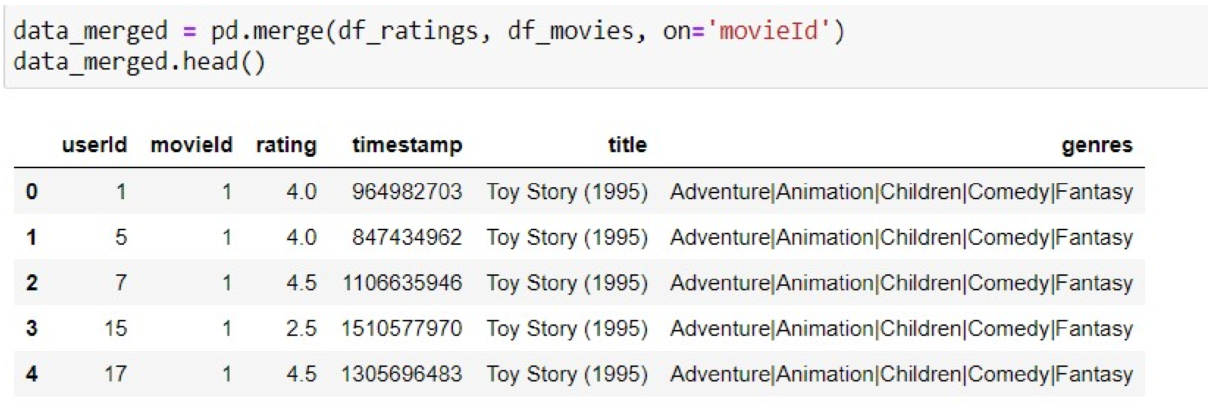
Pandas provides a single function, **merge**, as the entry point for all standard database join operations between DataFrame objects.

Syntax:

pd.merge(left,right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=True)

parameter:

1. Left: A DataFrame object.
2. Right: Another DataFrame object.
3. On: Columns (names) to join on. Must be found in both the left and right DataFrame objects.
4. left_on: Columns from the left DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.
5. right_on: Columns from the right DataFrame to use as keys. Can either be column names or arrays with length equal to the length of the DataFrame.
6. left_index: If **True,** use the index (row labels) from the left DataFrame as its join key(s). In case of a DataFrame with a MultiIndex (hierarchical), the number of levels must match the number of join keys from the right DataFrame.
7. right_index: Same usage as **left_index** for the right DataFrame.
8. how: One of 'left', 'right', 'outer', 'inner'. Defaults to inner. Each method has been described below.
9. sort: Sort the result DataFrame by the join keys in lexicographical order. Defaults to True, setting to False will improve the performance substantially in many cases.



```
data_merged = pd.merge(df_ratings, df_movies, on='movieId')
data_merged.head()
```

```
   userId  movieId  rating   timestamp             title                                        genres
0       1        1     4.0   964982703  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
1       5        1     4.0   847434962  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
2       7        1     4.5  1106635946  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
3      15        1     2.5  1510577970  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
4      17        1     4.5  1305696483  Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy
```

The above image shows the merging of the two datasets on the movieId column

Groupby():

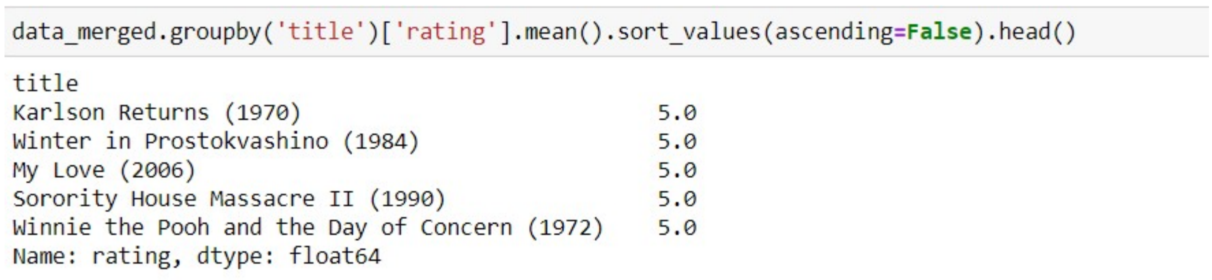
Pandas dataframe.groupby() **function is used to split the data into groups based on some criteria. pandas objects can be split on any of their axes. The**

abstract definition of grouping is to provide a mapping of labels to group names.

<u>Syntax:</u>
Dataframe.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=false)
Parameter:

1. by: mapping,function,str,or,iterable.
2. axis: int ,default0
3. level: if the axis is a MultiIndex(hierarchical),group by a particular level or levels.
4. as_index: For aggregate output, return object with group with group labels as the index.only relevant for Dataframe input,as_index=False is effectively"SQL-style" grouped output.
5. sort: sort group keys. Get better performance by turning this off. Note this does not influence the order of observations within each group. Groupby preserves the order of rows within each group.
6. group_keys: When calling apply, add group keys to index to identify pieces.
7. squeeze: Reduce the dimensionality of the return type if possible,otherwise return a consistence type.

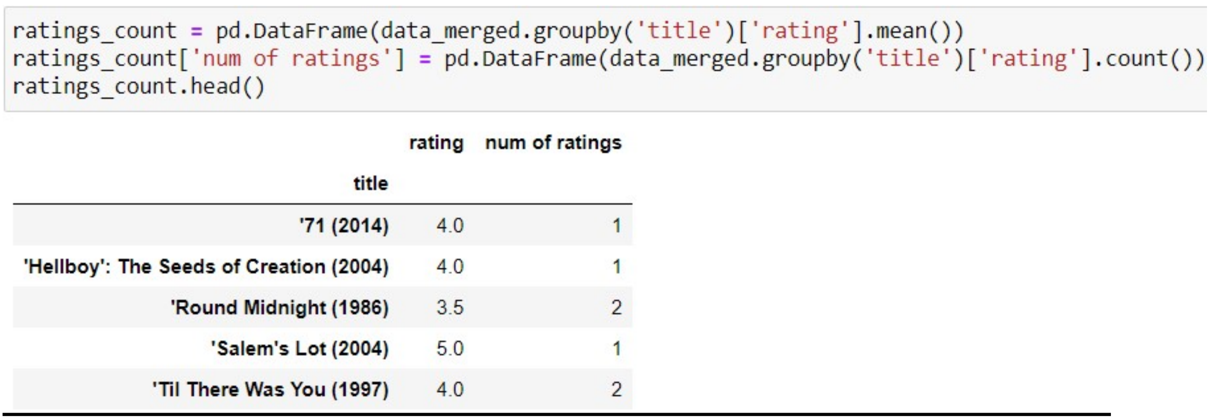
<u>Return</u>: GroupBy object



```
data_merged.groupby('title')['rating'].mean().sort_values(ascending=False).head()
```

```
title
Karlson Returns (1970)                             5.0
Winter in Prostokvashino (1984)                    5.0
My Love (2006)                                     5.0
Sorority House Massacre II (1990)                  5.0
Winnie the Pooh and the Day of Concern (1972)      5.0
Name: rating, dtype: float64
```

<u>Grouping of the movie based on the mean value of the rating in the sorted form</u>



```
ratings_count = pd.DataFrame(data_merged.groupby('title')['rating'].mean())
ratings_count['num of ratings'] = pd.DataFrame(data_merged.groupby('title')['rating'].count())
ratings_count.head()
```

title	rating	num of ratings
'71 (2014)	4.0	1
'Hellboy': The Seeds of Creation (2004)	4.0	1
'Round Midnight (1986)	3.5	2
'Salem's Lot (2004)	5.0	1
'Til There Was You (1997)	4.0	2

<u>Storing the above groupby function into a dataframe and displaying the results in the dataset form.</u>

plt.figure():
The **figure() function** in pyplot module of matplotlib library is used to create a new figure.
Syntax:

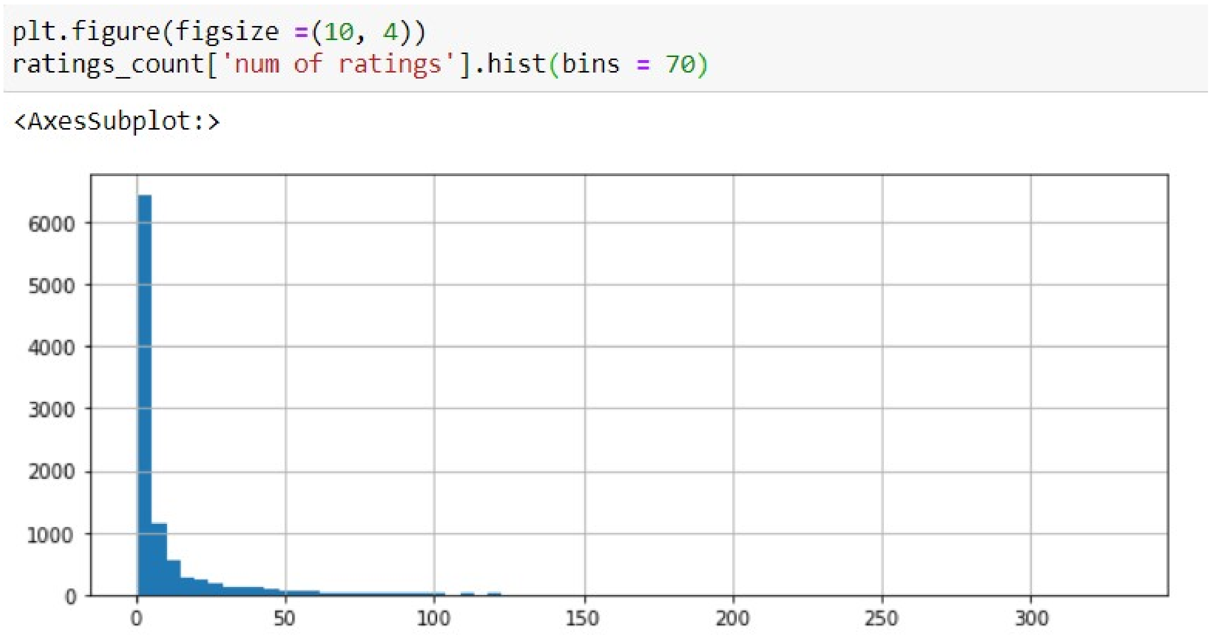
Matplotlib.pyplot.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, framon=True, FigureClass=, clear=False)

Parameters:

1. num: This parameter is provided,and a figure with this id already exists.
2. figsize: These parameter are the width,height in inches.
3. dpi: This parameter is the resolution of the figure.
4. facecolor: This parameter is the background color.
5. edgecolor: This parameter is the border color.
6. framon: This parameter suppress drawing the figure frame.
7. FigureClass: This parameter use a custom figure instance.
8. clear: This parameter if True and the figure already exists, then it is cleared.

Return value:

Figure: this returns the figure the Figure instance returned will also be passed t new_figure_manager in the backends.



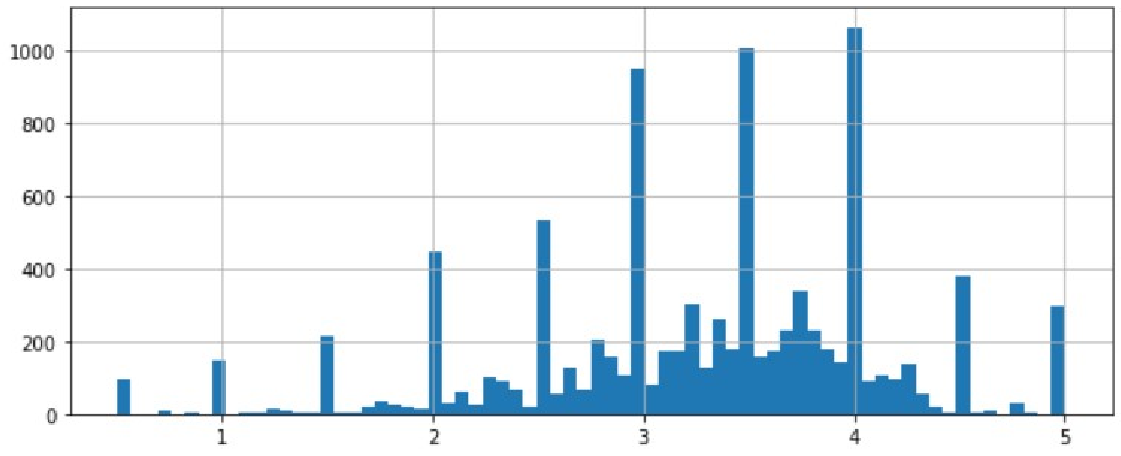
```
plt.figure(figsize =(10, 4))
ratings_count['num of ratings'].hist(bins = 70)
```

```
<AxesSubplot:>
```

Plotting the histogram by applying figure() on the column name ”num of ratings” in the ratings_count dataframe.

```
plt.figure(figsize =(10, 4))
ratings_count['rating'].hist(bins = 70)
```

```
<AxesSubplot:>
```



Plotting the histogram by applying figure() on the column name ”rating” in the ratings_count dataframe.

<u>Dataframe. pivot_table():</u>
The pivot_table() function is used to create a spreadsheet-style pivot table as a DataFrame. The levels in the pivot table will be stored in MultiIndex objects (hierarchical indexes) on the index and columns of the result DataFrame.
<u>Syntax:</u>

Dataframe.pivot_table(self, values=None, index=None,columns=None,aggfunc=’mean’,fill_value=None, margins=False, dropna=True, margins_name=’All’,observed=False)

1. index: If an array is passed, it must be the same length as the data. The list can contain any of the other types (except list). Keys to group by on the pivot table index. If an array is passed, it is being used as the same manner as column values.
2. columns: If an array is passed, it must be the same length as the data. The list can contain any of the other types (except list). Keys to group by on the pivot table column. If an array is passed, it is being used as the same manner as column values.
3. aggfunc: If list of functions passed, the resulting pivot table will have hierarchical columns whose top level are the function names (inferred from the function objects themselves) If dict is passed, the key is column to aggregate and value is function or list of functions
4. fill_value: Value to replace missing values with.
5. margins: Add all row / columns (e.g. for subtotal / grand totals).

6. dropna: Do not include columns whose entries are all NaN
7. margins_name: Name of the row / column that will contain the totals when margins is True
8. observed: This only applies if any of the groupers are Categoricals. If True: only show observed values for categorical groupers. If False: show all values for categorical groupers.

Returns: DataFrame

Dataframe.sort_value():

The sort_values() function is used to sort by the values along either axis.

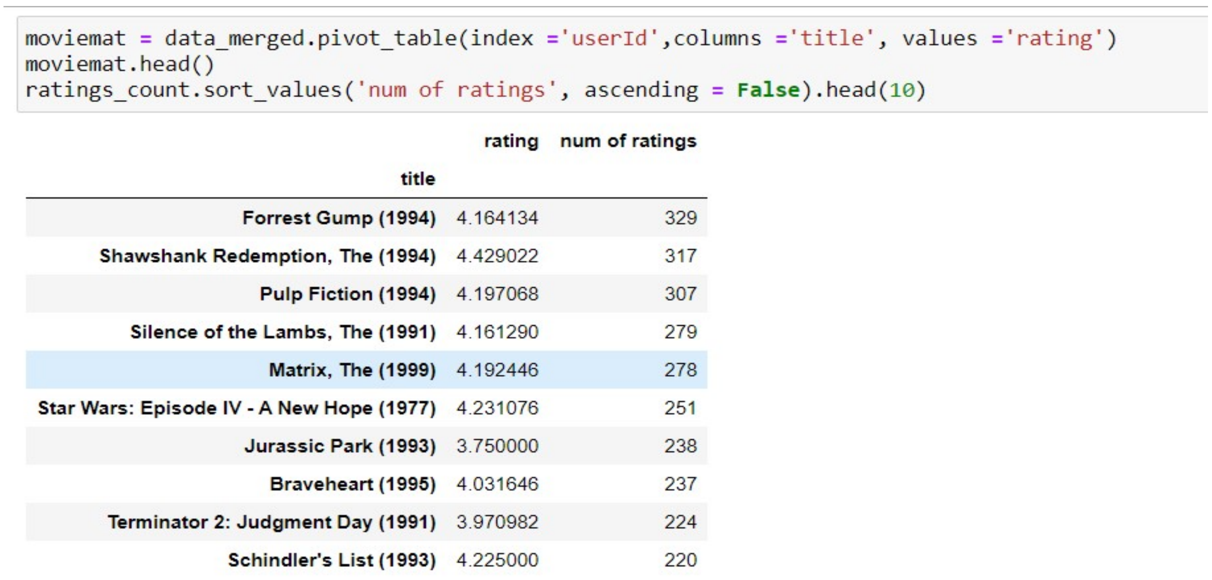
Syntax:

Dataframe.sort_value(self, by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')

Parameters:

1. by: Name or list of names to sort by.
    1. if axis is 0 or 'index' then by may contain index levels and/or column labels.
    2. if axis is 1 or 'columns' then by may contain column levels and/or index labels.
2. axis: Axis to be sorted.
3. ascending: Sort ascending vs. descending. Specify list for multiple sort orders. If this is a list of bools, must match the length of the by.
4. inplace: If True, perform operation in-place.
5. kind: Choice of sorting algorithm. See also ndarray.np.sort for more information. mergesort is the only stable algorithm. For DataFrames, this option is only applied when sorting on a single column or label
6. na_position: Puts NaNs at the beginning if first; last puts NaNs at the end.

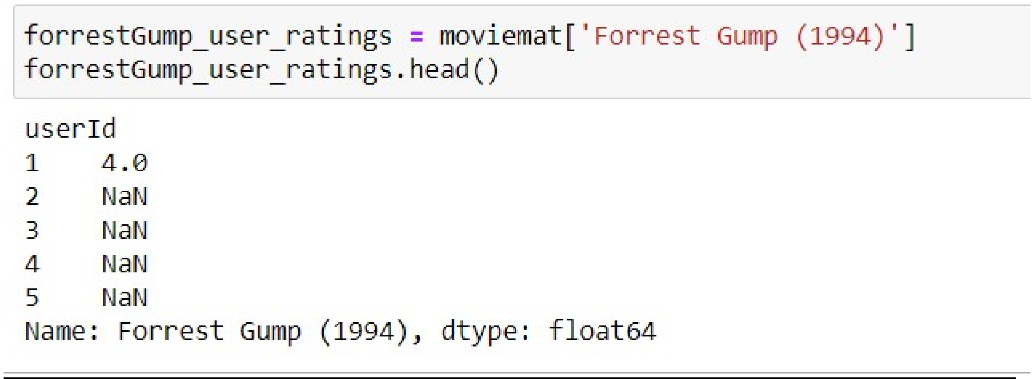
Returns: sorted_obj - DataFrame or None DataFrame with sorted values if inplace=False, None otherwise.



```
moviemat = data_merged.pivot_table(index ='userId',columns ='title', values ='rating')
moviemat.head()
ratings_count.sort_values('num of ratings', ascending = False).head(10)
```

title	rating	num of ratings
Forrest Gump (1994)	4.164134	329
Shawshank Redemption, The (1994)	4.429022	317
Pulp Fiction (1994)	4.197068	307
Silence of the Lambs, The (1991)	4.161290	279
Matrix, The (1999)	4.192446	278
Star Wars: Episode IV - A New Hope (1977)	4.231076	251
Jurassic Park (1993)	3.750000	238
Braveheart (1995)	4.031646	237
Terminator 2: Judgment Day (1991)	3.970982	224
Schindler's List (1993)	4.225000	220

Applying pivot_table() on the dataframe data_merged

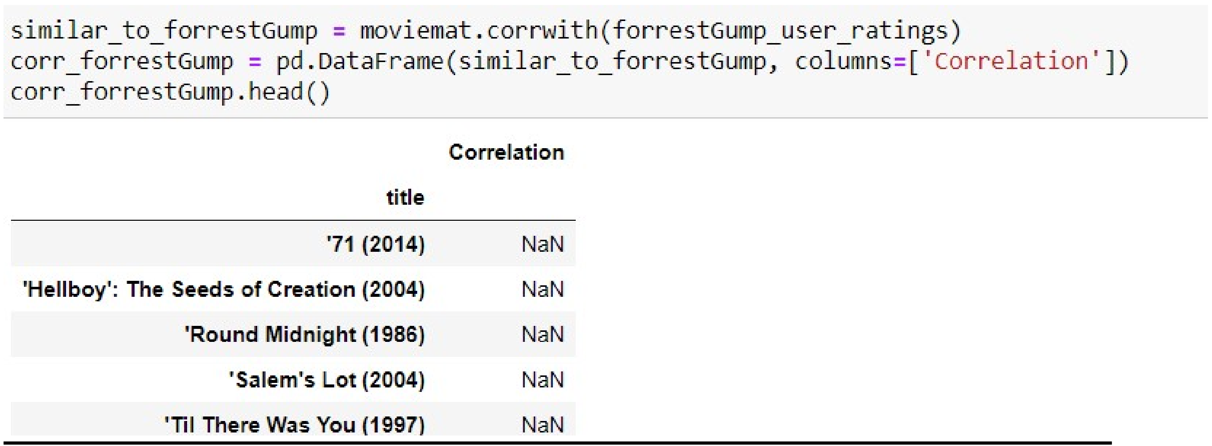


```
forrestGump_user_ratings = moviemat['Forrest Gump (1994)']
forrestGump_user_ratings.head()
```

```
userId
1    4.0
2    NaN
3    NaN
4    NaN
5    NaN
Name: Forrest Gump (1994), dtype: float64
```

Passing the movie names in the the dataframe "moviemat" to get the ratings of that movie

Dataframe.corrwith():
Pandas **dataframe.corrwith()** is used to compute pairwise correlation between rows or columns of two DataFrame objects. If the shape of two dataframe object is not same then the corresponding correlation value will be a NaN value.
Syntax: DataFrame.count(axis=0, level=None, numeric_only=False)
Parameter:
Other: DataFrame
axis : 0 or 'index' to compute column-wise, 1 or 'columns' for row-wise
drop : Drop missing indices from result, default returns union of all
Returns: correls : Series
Note: The correlation of a variable with itself is 1.



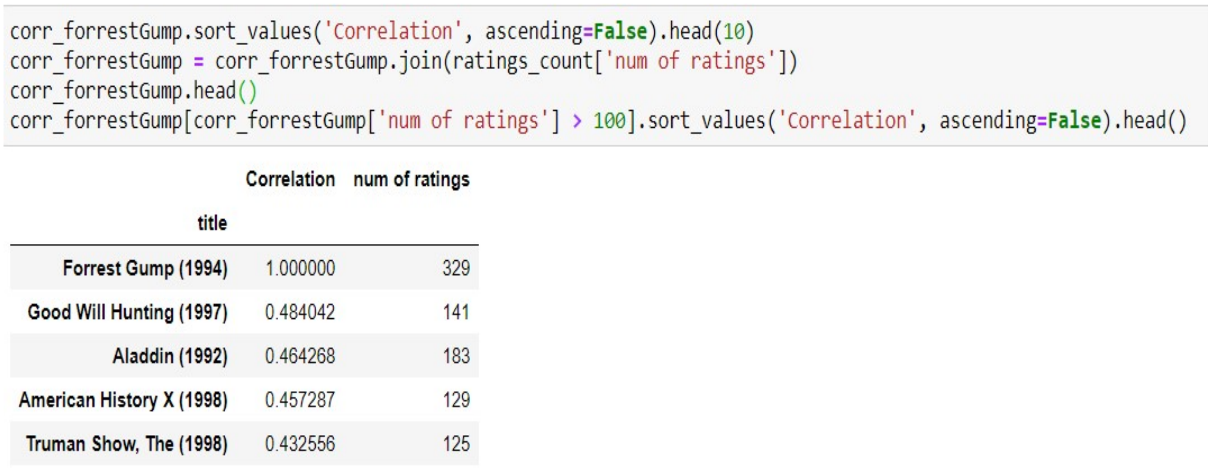
```
similar_to_forrestGump = moviemat.corrwith(forrestGump_user_ratings)
corr_forrestGump = pd.DataFrame(similar_to_forrestGump, columns=['Correlation'])
corr_forrestGump.head()
```

title	Correlation
'71 (2014)	NaN
'Hellboy': The Seeds of Creation (2004)	NaN
'Round Midnight (1986)	NaN
'Salem's Lot (2004)	NaN
'Til There Was You (1997)	NaN

Implemented screenshot

Dataframe.join():

The join() method is often useful when one DataFrame is a lookup table that contains additional data added into the other DataFrame. It is a convenient method that can combine the columns of two differently-indexed DataFrames into a single DataFrame.



```
corr_forrestGump.sort_values('Correlation', ascending=False).head(10)
corr_forrestGump = corr_forrestGump.join(ratings_count['num of ratings'])
corr_forrestGump.head()
corr_forrestGump[corr_forrestGump['num of ratings'] > 100].sort_values('Correlation', ascending=False).head()
```

title	Correlation	num of ratings
Forrest Gump (1994)	1.000000	329
Good Will Hunting (1997)	0.484042	141
Aladdin (1992)	0.464268	183
American History X (1998)	0.457287	129
Truman Show, The (1998)	0.432556	125

Implemented screenshot

Dataframe.pivot():

The pivot() function is used to reshaped a given DataFrame organized by given index / column values. This function does not support data aggregation, multiple values will result in a MultiIndex in the columns.

Syntax:

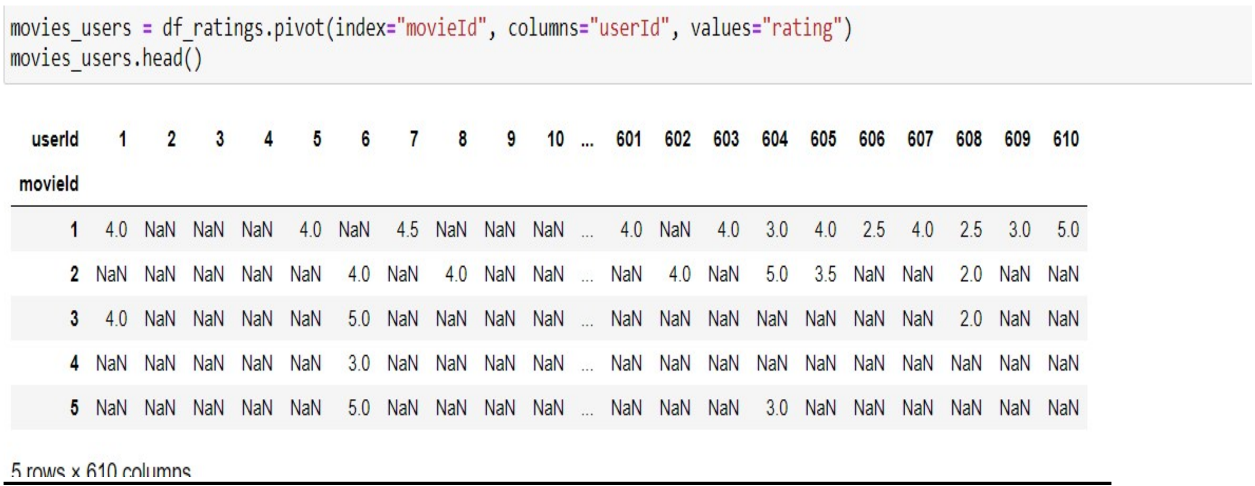
Dataframe.pivot(self, index=None, columns=None, values=None)

Parameters:

1. Index: Column to use to make new frame's index. If None, uses existing index.
2. Columns: Column to use to make new frame's columns.
3. Values: Column(s) to use for populating new frame's values. If not specified, all remaining columns will be used and the result will have hierarchically indexed columns.

Returns: DataFrame

Returns reshaped DataFrame.



```
movies_users = df_ratings.pivot(index="movieId", columns="userId", values="rating")
movies_users.head()
```

```
userId    1    2    3    4    5    6    7    8    9   10  ...  601  602  603  604  605  606  607  608  609  610
movieId
      1  4.0  NaN  NaN  NaN  4.0  NaN  4.5  NaN  NaN  NaN  ...  4.0  NaN  4.0  3.0  4.0  2.5  4.0  2.5  3.0  5.0
      2  NaN  NaN  NaN  NaN  NaN  4.0  NaN  4.0  NaN  NaN  ...  NaN  4.0  NaN  5.0  3.5  NaN  NaN  2.0  NaN  NaN
      3  4.0  NaN  NaN  NaN  NaN  5.0  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  2.0  NaN  NaN
      4  NaN  NaN  NaN  NaN  NaN  3.0  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN
      5  NaN  NaN  NaN  NaN  NaN  5.0  NaN  NaN  NaN  NaN  ...  NaN  NaN  NaN  3.0  NaN  NaN  NaN  NaN  NaN  NaN

5 rows x 610 columns
```

Implemented screenshot

Dataframe.fillna():

The fillna() function is used to fill NA/NaN values using the specified method.

Syntax:

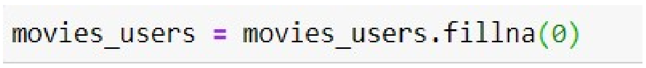
Series.fillna(self, value=None, method=None, axis=None, inplace=False, limit=None, downcast=None)

Parameter:

1. **Value:** Value to use to fill holes (e.g. 0), alternately a dict/Series/DataFrame of values specifying which value to use for each index (for a Series) or column (for a DataFrame). Values not in the dict/Series/DataFrame will not be filled. This value cannot be a list.
2. **Method:** Method to use for filling holes in reindexed Series pad / ffill: propagate last valid observation forward to next valid backfill / bfill: use next valid observation to fill gap

3. **Axis:** Axis along which to fill missing values.
4. **Inplace:** If True, fill in-place. Note: this will modify any other views on this object (e.g., a no-copy slice for a column in a DataFrame).
5. **Limit:** If method is specified, this is the maximum number of consecutive NaN values to forward/backward fill. In other words, if there is a gap with more than this number of consecutive NaNs, it will only be partially filled. If method is not specified, this is the maximum number of entries along the entire axis where NaNs will be filled. Must be greater than 0 if not None.
6. **Downcast:**
   A dict of item->dtype of what to downcast if possible, or the string ‘infer’ which will try to downcast to an appropriate equal type (e.g. float64 to int64 if possible).

Returns: Series-object with missing values filled.



```
movies_users = movies_users.fillna(0)
```

Scipy Sparse data:

Sparse data is data that has mostly unused elements (elements that don't carry any information).

SciPy has a module, scipy.sparse that provides functions to deal with sparse data.

There are primarily two types of sparse matrices that we use:

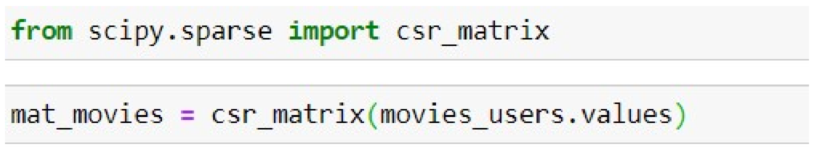
CSC - Compressed Sparse Column. For efficient arithmetic, fast column slicing.

CSR - Compressed Sparse Row. For fast row slicing, faster matrix vector products

We will use the **CSR** matrix in this tutorial.

CSR MATRIX:

We can create CSR matrix by passing an arrray into function scipy.sparse.csr_matrix().



```
from scipy.sparse import csr_matrix
```

```
mat_movies = csr_matrix(movies_users.values)
```

Implemented screenshot

Sklearn():[1]

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**.

4.2 Knn algorithm

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well.

1. Lazy-learning: KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.

2. Non-parametric learning algorithm: KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

Working of the knn- algorithm:

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps.

Step 1 − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

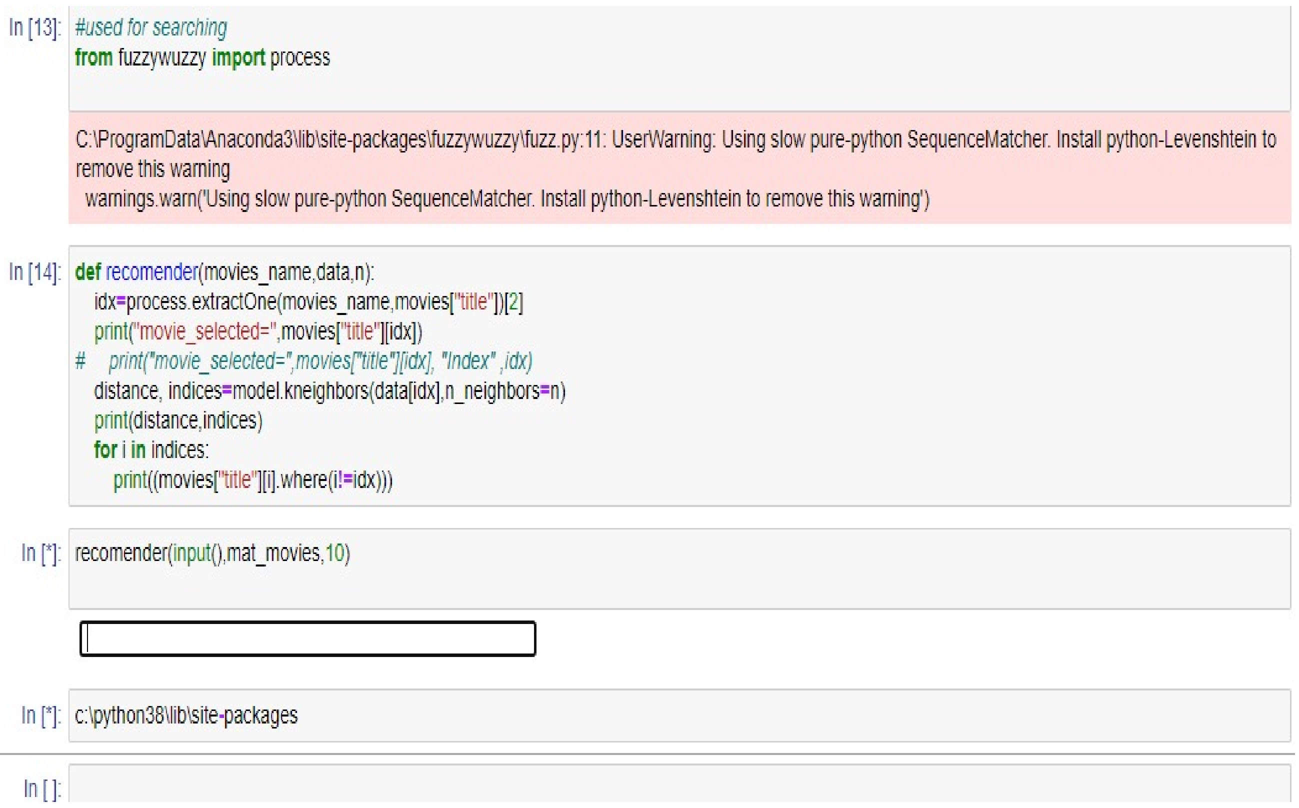
Step 3 − For each point in the test data do the following −

- **3.1** − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
- **3.2** − Now, based on the distance value, sort them in ascending order.
- **3.3** − Next, it will choose the top K rows from the sorted array.
- **3.4** − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 – End

FUZZYWUZZY:

Fuzzy string matching like a boss. It uses levenshtein distance to calculate the differences between sequences in a simple-to-use package.



```
In [13]: #used for searching
         from fuzzywuzzy import process

C:\ProgramData\Anaconda3\lib\site-packages\fuzzywuzzy\fuzz.py:11: UserWarning: Using slow pure-python SequenceMatcher. Install python-Levenshtein to
remove this warning
  warnings.warn('Using slow pure-python SequenceMatcher. Install python-Levenshtein to remove this warning')

In [14]: def recomender(movies_name,data,n):
           idx=process.extractOne(movies_name,movies["title"])[2]
           print("movie_selected=",movies["title"][idx])
         #    print("movie_selected=",movies["title"][idx], "Index" ,idx)
           distance, indices=model.kneighbors(data[idx],n_neighbors=n)
           print(distance,indices)
           for i in indices:
             print((movies["title"][i].where(i!=idx)))

In [*]: recomender(input(),mat_movies,10)

In [*]: c:\python38\lib\site-packages

In [ ]:
```

Implemented screenshot

Finally, we get a box to input our choice of movie such that we can search for our recommended movies as per their ratings.

4.3 Tkinter[7][11]

Tkinter is a Python binding to the Tk GUI toolkit. Tk is the original GUI library for the Tcl language. Tkinter is implemented as a Python wrapper around a complete Tcl interpreter embedded in the Python interpreter. There are several other popular Python GUI toolkits. Most popular are wxPython, PyQt, and PyGTK.

def close_window():

This function will closing the whole after completion of the work.

window.destroy():

The destroy() method in Tkinter destroys a widget. It is useful in controlling the behavior of various widgets which depend on each other. Also when a process is complete by some user action we need to destroy the GUI components to free the memory as well as clear the screen. The destroy() method achieves all this.
syntax:

widget_object = widget(parent, command = widget_class_object.destroy)

def fu():

this function will connect the GUI from the backend for the complete functioning of our project.

window=tkinter,tk():

The root window is created. The root window is a main application window in our programs. It has a title bar and borders. These are provided by the window manager. It must be created before any other widgets

window.geometry():

Tkinter provides many methods; one of them is the geometry() method. This method is used to set the dimensions of the tkinter window and is used to set the position of the main window on the user’s desktop **.**

window.title():

This method is used to give the title to the tkinter main root window on the user's desktop.

recomender(tobesearched.get(), mat_movies, 10):

This function is used to recommend the movie similar to the movie typed in the search bar by the user.

tkinter.label():

This widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want.
It is also possible to underline part of the text (like to identify a keyboard shortcut) and span the text across multiple lines.

Syntax:

w=label(master, option,…)

Parameters:

1. **anchor**

This options controls where the text is positioned if the widget has more space than the text needs. The default is anchor=CENTER, which centers the text in the available space.

2. **bg**

The normal background color displayed behind the label and indicator.

3. **bitmap**

Set this option equal to a bitmap or image object and the label will display that graphic.

4. **bd**

The size of the border around the indicator. Default is 2 pixels.

5. **cursor**

If you set this option to a cursor name (*arrow, dot etc.*), the mouse cursor will change to that pattern when it is over the checkbutton.

6. **font**

If you are displaying text in this label (with the text or textvariable option, the font option specifies in what font that text will be displayed.

7. **fg**

If you are displaying text or a bitmap in this label, this option specifies the color of the text. If you are displaying a bitmap, this is the color that will appear at the position of the 1-bits in the bitmap.

8. **height**

The vertical dimension of the new frame

9. **image**

To display a static image in the label widget, set this option to an image object

10. **justify**

Specifies how multiple lines of text will be aligned with respect to each other: LEFT for flush left, CENTER for centered (the default), or RIGHT for right-justified

11. **padx**

Extra space added to the left and right of the text within the widget. Default is 1.

12. **pady**

Extra space added above and below the text within the widget. Default is 1.

13. **relief**

Specifies the appearance of a decorative border around the label. The default is FLAT; for other values.

14. **textvariable**

To slave the text displayed in a label widget to a control variable of class *StringVar*, set this option to that variable.

15. **underline**

You can display an underline (_) below the nth letter of the text, counting from 0, by setting this option to n. The default is underline=-1, which means no underlining.

tkinter.button():

The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

Tkinter.PhotoImage():

Several Tkinter widgets, including **Label** and **Button**, can take an **image** argument, which allows them to display an image. We can't simply put a path to an image file in those cases; instead, we have to create a PhotoImage object.

tkinter.Entry()
The Entry widget is used to accept single-line text strings from a user.

Syntax:
w = Entry (master, option)

Parameters

1) parent: The Parent window or frame in which the widget to display.
2) Options: The various options provided by the entry widget are:

- **bg :** The normal background color displayed behind the label and indicator.
- **bd :** The size of the border around the indicator. Default is 2 pixels.
- **font :** The font used for the text.
- **fg :** The color used to render the text.
- **justify :** If the text contains multiple lines, this option controls how the text is justified: CENTER, LEFT, or RIGHT.
- **relief :** With the default value, relief=FLAT. You may set this option to any of the other styles like : SUNKEN, RIGID, RAISED, GROOVE
- **show :** Normally, the characters that the user types appear in the entry. To make a .password. entry that echoes each character as an asterisk, set show=”*”.
- **textvariable :** In order to be able to retrieve the current text from your entry widget, you must set this option to an instance of the StringVar class.

Methods: The various methods provided by the entry widget are:

- **get() :** Returns the entry’s current text as a string.
- **delete() :** Deletes characters from the widget
- **insert (index, ‘name’) :** Inserts string ‘name’ before the character at the given index.

Return type:
It doesn’t return any value to a variable

toBesearched.grid():
The Grid geometry manager puts the widgets in a 2-dimensional table. The master widget is split into a number of rows and columns, and each “cell” in the resulting table can hold a widget. The grid manager is the most flexible of the geometry managers in tkinter.

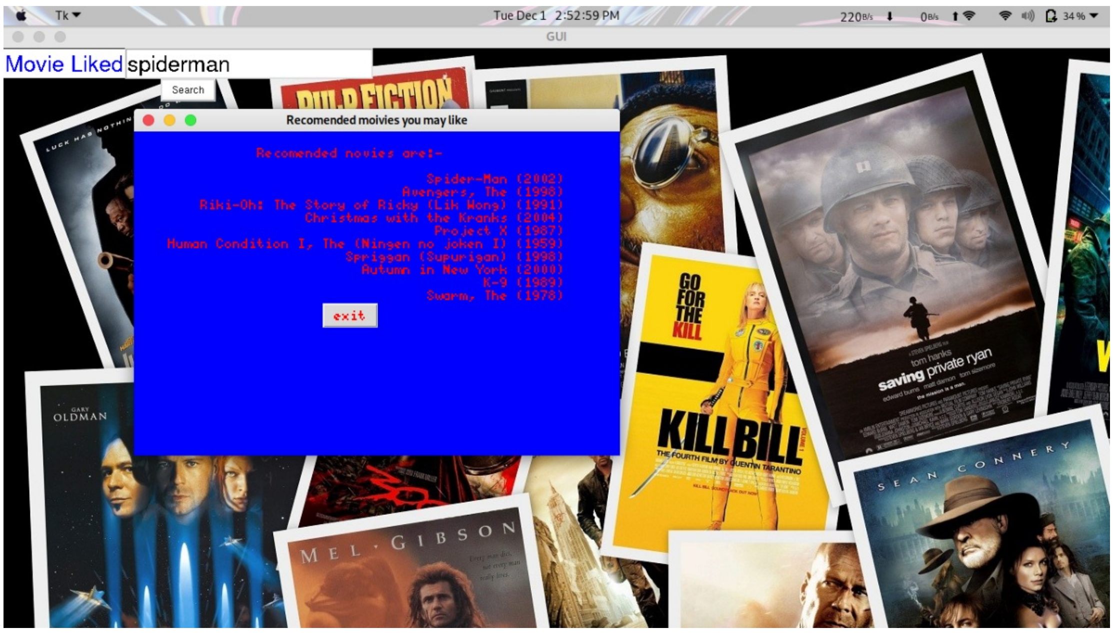
Syntax:
widget.grid(grid_options)

parameters:
the following possible options for parameters in grid are:

- **column** − The column to put widget in; default 0 (leftmost column).

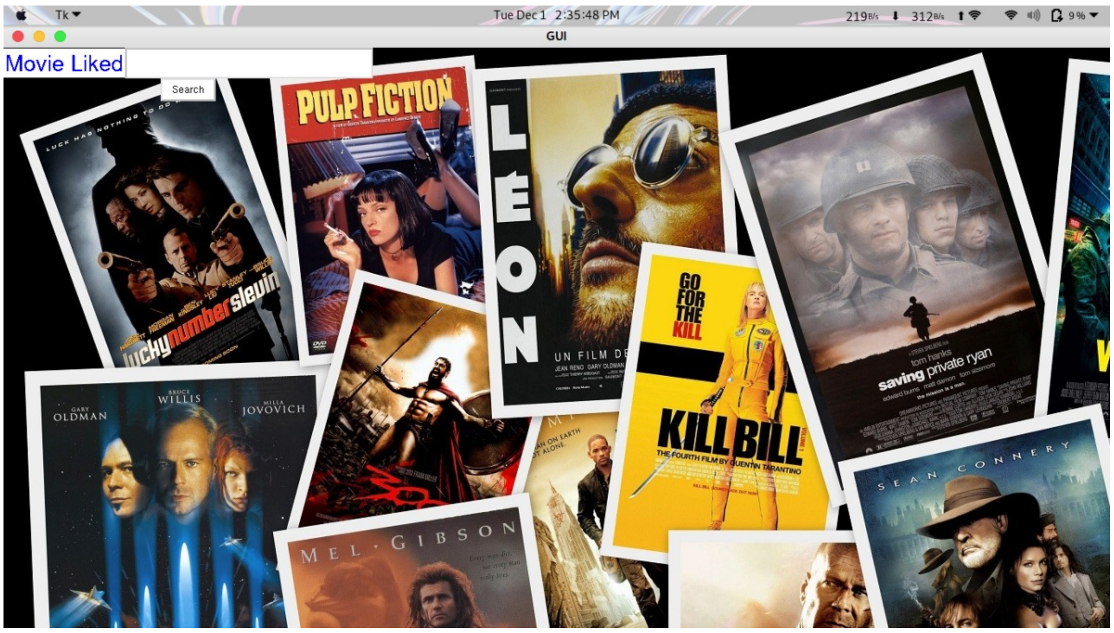
- **columnspan** − How many columns widgetoccupies; default 1.
- **ipadx, ipady** − How many pixels to pad widget, horizontally and vertically, inside widget's borders.
- **padx, pady** − How many pixels to pad widget, horizontally and vertically, outside v's borders.
- **row** − The row to put widget in; default the first row that is still empty.
- **rowspan** − How many rowswidget occupies; default 1.
- **sticky** − What to do if the cell is larger than widget. By default, with sticky='', widget is centered in its cell. sticky may be the string concatenation of zero or more of N, E, S, W, NE, NW, SE, and SW, compass directions indicating the sides and corners of the cell to which widget sticks.

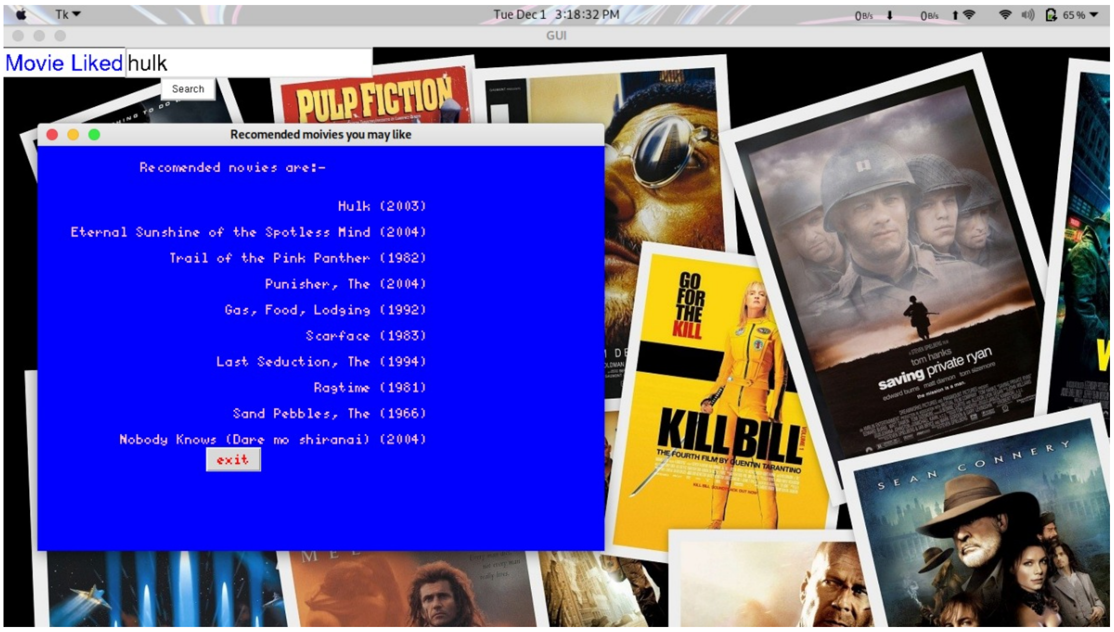
OUTPUT



```
Recomended movies are:-

                                        Spider-Man (2002)
                                      Avengers, The (1998)
              Riki-Oh: The Story of Ricky (Lik Wong) (1991)
                         Christmas with the Kranks (2004)
                                         Project X (1987)
     Human Condition I, The (Ningen no joken I) (1959)
                            Spriggan (Supurigan) (1998)
                              Autumn in New York (2000)
                                              K-9 (1989)
                                      Swarm, The (1978)
```

This is the home page of our project. In this figure we are seeing a search bar in which movie liked is written...that means in the box we have to type the input of the movie....and after clicking on the search button we will get our output as the recommended movies.After clicking on the search button we will go the new window containing the output as the recommended movies and also a exit button. After clicking on the exit button we will get out of the window.





```
Recomended movies are:-

Hulk (2003)
Eternal Sunshine of the Spotless Mind (2004)
Trail of the Pink Panther (1982)
Punisher, The (2004)
Gas, Food, Lodging (1992)
Scarface (1983)
Last Seduction, The (1994)
Ragtime (1981)
Sand Pebbles, The (1966)
Nobody Knows (Dare mo shiranai) (2004)
```

In the following output, the user is asked to rate the movies which user has already seen then these ratings are applied to recommend other movies to the user that user has not perceived by utilizing collaborative filtering that is based on similar ratings.

REFERENCES

This project is about the designing of online Movie Recommendation system. Present system of Movie Recommendation System is having some shortcomings on which we have tried to work on that to eliminate the disadvantages.

We have used tkinter module for designing of our GUI in Pycharm. We have used Python 3 for GUI implentation.

.

We preferred "movielens" website for our Dataset.

We would also like to credit the following for help in our Project Report:

1. Geeks for geeks
2. https://alex.smola.org
3. https://www.techrepublic.com/
4. https://www.mygreatlearning.com/
5. TutorialPoint
6. https://buildmedia.readthedocs.org/
7. Riptutorial.com
8. Pandas.pydata.org
9. readthedocs.org
10. Github
11. Gigapromo