

MACHINE LEARNING
(Movie Recommender System)

Summer Internship Report Submitted in partial fulfillment

of the requirement for undergraduate degree of

Bachelor of Technology

In

Computer Science and Engineering

BY

B. SWAMI CHARAN RAO

221710309010

Under the Guidance of

Professor



GITAM
(DEEMED TO BE UNIVERSITY)

VISAKHAPATNAM • HYDERABAD • BENGALURU

Department Of Computer Science and Engineering GITAM

School of Technology

GITAM (Deemed to be University)

Hyderabad-502329

July 2020

DECLARATION

I submit this industrial training work entitled “**MOVIE RECOMMENDER SYSTEM**” to GITAM (Deemed To Be University), Hyderabad in partial fulfillment of the requirements for the award of the degree of “**Bachelor of Technology**” in “**Computer Science and Engineering**”. I declare that it was carried out independently by me under the guidance of Professor, GITAM (Deemed To Be University), Hyderabad, India.

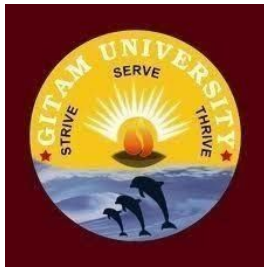
The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma.

Place: HYDERABAD

B. Swami Charan

Date:

221710309010



GITAM (DEEMED TO BE
UNIVERSITY)

Hyderabad-502329, India

Dated:

CERTIFICATE

ACKNOWLEDGEMENT

ABSTRACT

A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. Recommender systems have become ubiquitous in our lives. Yet, currently, they are far from optimal. In this project, we attempt to understand the different kinds of movie recommendation systems on the MovieLens dataset. We attempt to build a scalable model to perform this analysis. We start by preparing and comparing the various models on a smaller dataset. Three main approaches are used for our recommender systems. One is Popularity based i.e. They offer generalized recommendations to every user, based on movie popularity and/or genre. The System recommends the same movies to users with similar demographic features. The basic idea behind this system is that movies that are more popular will have a higher probability of being liked by the average audience. Second is content-based filtering, where we try to profile the users interests using information collected, and recommend items based on that profile. The other is collaborative filtering, where we try to group similar users together and use information about the group to make recommendations to the user.

Table of Contents:

Chapter1: Machine Learning

1.1 INTRODUCTION	8
1.2 IMPORTANCE OF MACHINE LEARNING	8
1.3 USES OF MACHINE LEARNING.....	9
1.4 TYPES OF LEARNING ALGORITHMS.....	10
1.4.1 Supervised Learning.....	10
1.4.2 Unsupervised Learning.....	11
1.4.3 Semi Supervised Learning.....	12
1.5 RELATION BETWEEN DATA MINING, MACHINE LEARNING AND DEEP LEARNING.....	12

CHAPTER 2: PYTHON

2.1 INTRODUCTION TO PYTHON.....	14
2.2 HISTORY OF PYTHON.....	14
2.3 FEATURES OF PYTHON.....	14
2.4 HOW TO SETUP PYTHON.....	15
2.4.1 Installation (using python IDLE).....	15
2.4.2 Installation (using Anaconda).....	16
2.5 PYTHON VARIABLE TYPES.....	18
2.5.1 Python Numbers.....	19
2.5.2 Python Strings.....	19
2.5.3 Python Lists.....	19
2.5.4 Python Tuples.....	20
2.5.5 Python Dictionary.....	20
2.6 PYTHON FUNCTION.....	21
2.6.1 Defining a Function.....	21
2.6.2 Calling a Function.....	21
2.7 PYTHON USING OOP'S CONCEPTS.....	22
2.7.1 class	22
2.7.2 __init__ method in class	23

CHAPTER 3: CASE STUDY.....24

3.1 PROBLEM STATEMENT.....	24
----------------------------	----

3.2 DATA SET.....	24
3.3 OBJECTIVE OF CASE STUDY.....	25
CHAPTER 4: MODEL BUILDING.....	26
4.1 PREPROCESSING OF THE DATA	26
4.1.1 GETTING THE DATASET.....	26
4.1.2 IMPORTING THE LIBRARIES.....	26
4.1.3 VERSIONS OF THE PACKAGES.....	27
4.1.4 IMPORTING THE DATA SET.....	27
4.1.5 HANDLING THE MISSING VALUES.....	30
4.1.6 CATEGORICAL DATA.....	31
CHAPTER 5: PREPROCESSING/FEATURE ENGINEERING AND EDA.....	33
5.1 STATISTICAL ANALYSIS.....	33
5.2 GENERATING PLOTS.....	35
5.3 DATA TYPE CONVERSIONS.....	36
CHAPTER 6: FEATURE SELECTION.....	37
6.1 SELECTING THE RELEVANT FEATURE FOR THE ANALYSIS..	37
6.2 DROPPING IRRELEVANT FEATURES.....	37
CHAPTER 7: MODEL BUILDING AND EVALUATION.....	38
7.1 POPULARITY BASED RECOMMENDATION SYSTEM..	38
7.1.1 BRIEF ABOUT THE ALGORITHM.....	38
7.1.2 APPROACH.....	38
7.1.3 VISUALIZING THE TREND.....	40
7.2 COLLABORATIVE FILTERING BASED RECOMMENDATION SYSTEM	42
7.2.1 INSTALLING THE SURPRISE PACKAGE.....	43
7.2.2 ESTIMATING THE MODEL.....	43
8. CONCLUSIONS.....	44
9. REFERENCES.....	45

LIST OF FIGURES:

Figure1.2.1 : The Process Flow
 Figure1.4.2.1 : Unsupervised Learning
 Figure1.4.3.1 : Semi Supervised Learning
 Figure2.4.1.1 : Python download
 Figure2.4.2.1: Anaconda download
 Figure2.4.2.2: Jupyter Notebook
 Figure2.7.1.1: Defining a class
 Figure4.1.2.1: Importing Libraries
 Figure4.1.3.1: Versions of Packages
 Figure4.1.4.1: Reading the movie dataset
 Figure4.1.4.2 : Reading the ratings dataset
 Figure4.1.4.3 : Reading the tags dataset
 Figure4.1.4.4: Reading the links dataset
 Figure4.1.4.5: Missing values in tmdbId
 Figure5.1.1 Statistical Analysis
 Figure5.1.2: Statistical Analysis
 Figure5.1.3: Describing the shape of the Files
 Figure5.2.1 : Missing values with heatmap (seaborn)
 Figure5.2.2: Visualizing with missing no
 Figure 5.3.1: Separating the release date of the movie and the title
 Figure 5.3.2: Splitting the genres with “[”
 Figure6.2.1: Removing the Irrelevant columns
 Figure 7.1.2.1: Merging on MovieId
 Figure7.1.2.2:Creating trend data frame for analysis
 Figure7.1.3.1: Plot Showing ratings with respect to the movies
 Figure7.1.3.2: Bar graph for number of movies for first 15 movies
 Figure7.1.3.3: Checking the Popularity
 Figure7.2.1.1: Installing Surprise package
 Figure7.2.1.2: Instantiating the Reader class
 Figure 7.2.1.3:Building the train set

Figure 7.2.1.4: Estimating the model

CHAPTER 1

MACHINE LEARNING

1.1 INTRODUCTION:

Machine Learning (ML) is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of Artificial Intelligence (AI).

1.2 IMPORTANCE OF MACHINE LEARNING:

Consider some of the instances where machine learning is applied: the self-driving Google car, Cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning. All these examples echo the vital role machine learning has begun to take in today’s data-rich world.

Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by involving automatic sets of generic methods that have replaced traditional statistical techniques.

The process flow depicted here represents how machine learning works

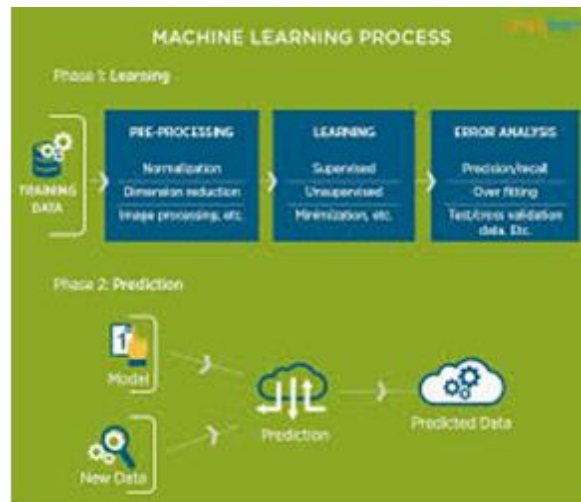


Figure 1.2.1: The Process Flow

1.3 USES OF MACHINE LEARNING:

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let's consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data.

By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning can produce accurate results and analysis.

1.4 TYPES OF LEARNING ALGORITHMS:

The types of machine learning algorithms differ in their approach, the type of data they input and output, and the type of task or problem that they are intended to solve.

1.4.1 Supervised Learning :

When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of supervised learning.

Supervised machine learning algorithms uncover insights, patterns, and relationships from a labeled training data set – that is, a data set that already contains a known value for the target variable for each record. Because you provide the machine learning algorithm with the correct answers for a problem during training, it is able to “learn” how the rest of the features relate to the target, enabling you to uncover insights and make predictions about future outcomes based on historical data.

Examples of Supervised Machine Learning Techniques are Regression, in which the algorithm returns a numerical target for each example, such as how much revenue will be generated from a new marketing campaign.

Classification, in which the algorithm attempts to label each example by choosing between two or more different classes. Choosing between two classes is called binary classification, such as determining whether or not someone will default on a loan. Choosing between more than two classes is referred to as multiclass classification.

1.4.2 Unsupervised Learning:

When an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of uncorrelated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

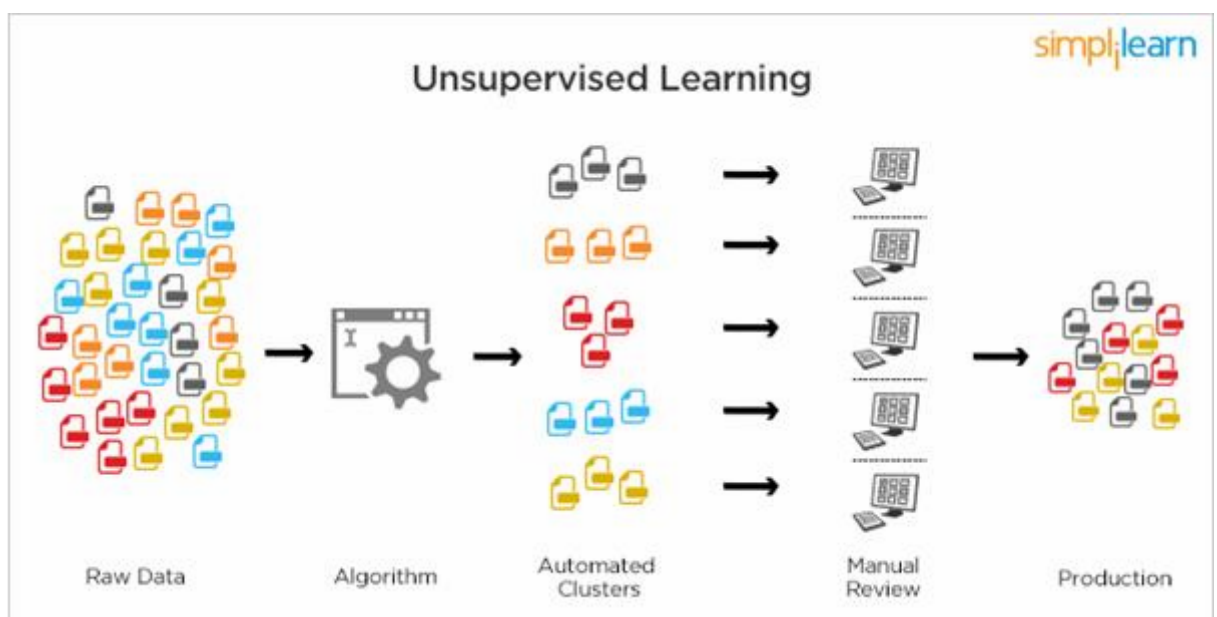


Figure 1.4.2.1 : Unsupervised Learning

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mapping, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

1.4.3 Semi Supervised Learning:

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.

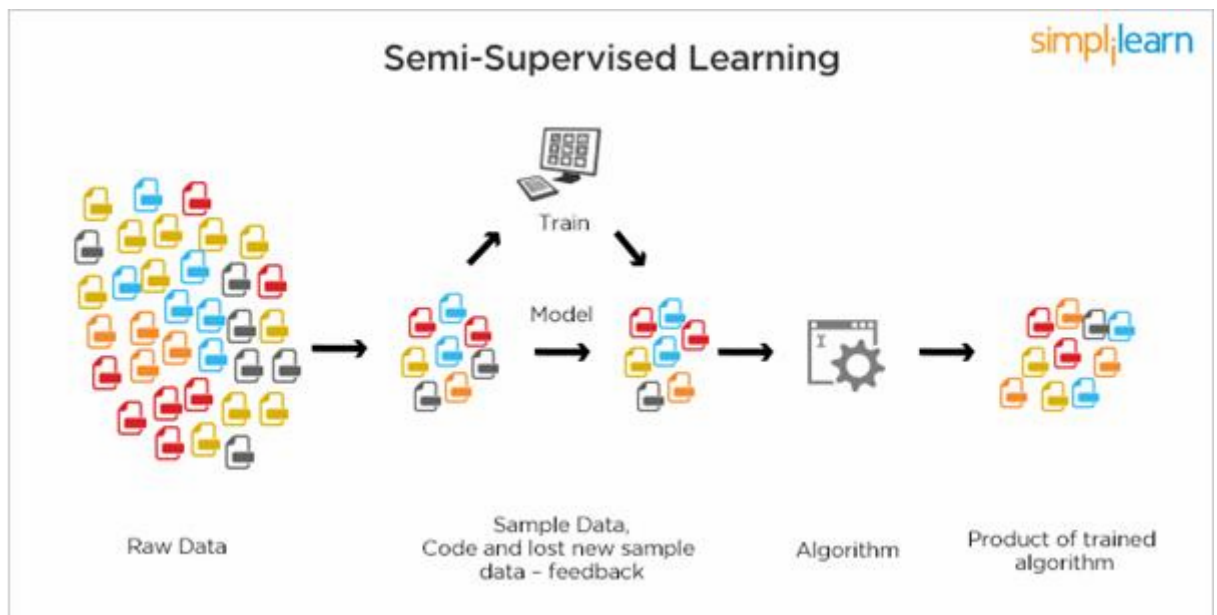


Figure 1.4.3.1 : Semi Supervised Learning

1.5 RELATION BETWEEN DATA MINING, MACHINE LEARNING AND DEEP LEARNING:

Machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special

types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

CHAPTER 2

PYTHON

Basic programming language used for machine learning is: PYTHON

2.1 INTRODUCTION TO PYTHON:

- Python is a high-level, interpreted, interactive and object-oriented scripting language.
- Python is a general purpose programming language that is often applied in scripting roles
- Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is like PERL and PHP.
- Python is Interactive: You can sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented: Python supports the Object-Oriented style or technique of programming that encapsulates code within objects.

2.2 HISTORY OF PYTHON:

- Python was developed by GUIDO VAN ROSSUM in early 1990's
- Its latest version is 3.7, it is generally called as python3

2.3 FEATURES OF PYTHON:

- Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax, This allows the student to pick up the language quickly.
- Easy-to-read: Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
- A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- Databases: Python provides interfaces to all major commercial databases.
- GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

2.4 HOW TO SETUP PYTHON:

- Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

2.4.1 Installation (using python IDLE):

- Installing python is generally easy, and nowadays many Linux and Mac OS distributions include a recent python.
- Download python from www.python.org
- When the download is completed, double click the file and follow the instructions to install it.
- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

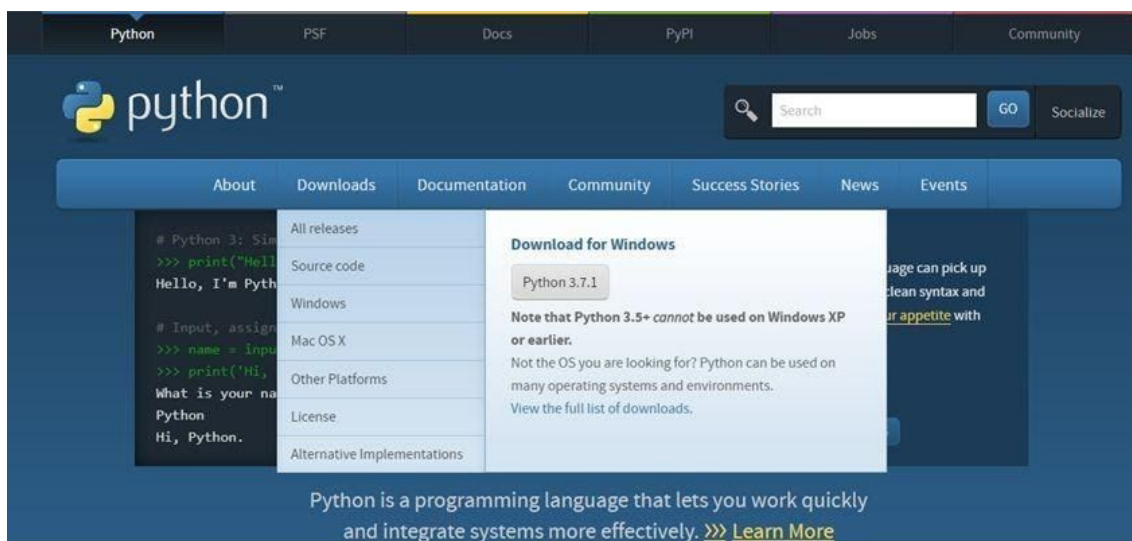


Figure 2.4.1.1 : Python download

2.4.2 Installation (using Anaconda):

- Python programs are also executed using Anaconda.
- Anaconda is a free open source distribution of python for large scale data processing, predictive analytics and scientific computing.
- Conda is a package manager quickly installs and manages packages.

- In WINDOWS:
- In windows
 - Step 1: Open Anaconda.com/downloads in web browser.
 - Step 2: Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)
 - Step 3: select installation type(all users)
 - Step 4: Select path(i.e. add anaconda to path & register anaconda as default python 3.4) next click install and next click finish
 - Step 5: Open jupyter notebook (it opens in default browser)

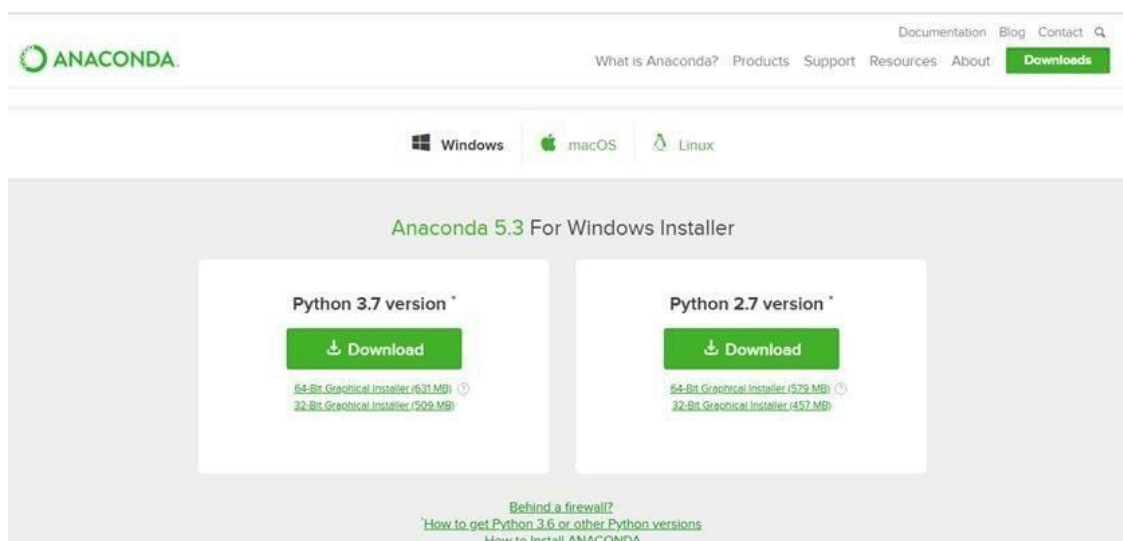


Figure 2.4.2.1: Anaconda download



Figure 2.4.2.2: Jupyter notebook

2.5 PYTHON VARIABLE TYPES:

- Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- Variables are nothing but reserved memory locations to store values.
- Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory.
- Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.
- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types –
 - Numbers
 - Strings
 - Lists

- Tuples
- Dictionary

2.5.1 Python Numbers:

- Number data types store numeric values. Number objects are created when you assign a value to them.
- Python supports four different numerical types – int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

2.5.2 Python Strings:

- Strings in Python are identified as a contiguous set of characters represented in the quotation marks.
- Python allows for either pairs of single or double quotes.
- Subsets of strings can be taken using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.
- The plus (+) sign is the string concatenation operator and the asterisk (*) is the repetition operator.

2.5.3 Python Lists:

- Lists are the most versatile of Python's compound data types.
- A list contains items separated by commas and enclosed within square brackets

([]).

- To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
- The values stored in a list can be accessed using the slice operator ([] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
- The plus (+) sign is the list concatenation operator, and the asterisk (*) is the repetition operator.

2.5.4 Python Tuples:

- A tuple is another sequence data type that is similar to the list.
- A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
- The main differences between lists and tuples are: Lists are enclosed in brackets ([]) and their elements and size can be changed, while tuples are enclosed in parentheses (()) and cannot be updated.
- Tuples can be thought of as read-only lists.
- For example – Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

2.5.5 Python Dictionary:

- Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
- Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
- You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
- What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

2.6 PYTHON FUNCTION:

2.6.1 Defining a Function:

You can define functions to provide the required functionality. Here are simple rules to define a function in Python. Function blocks begin with the keyword `def` followed by the function name and parentheses (i.e.()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses

The code block within every function starts with a colon (:) and is indented. The statement `return [expression]` exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as `return None`.

2.6.2 Calling a Function:

Defining a function only gives it a name, specifies the parameters that

are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

2.7 PYTHON USING OOP's CONCEPTS:

2.7.1 Class:

- **Class:** A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
- **Class variable:** A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
- **Data member:** A class variable or instance variable that holds data associated with a class and its objects.
- **Instance variable:** A variable that is defined inside a method and belongs only to the current instance of a class.
- **Defining a Class:**
 - We define a class in a very similar way how we define a function.
 - Just like a function, we use parentheses and a colon after the class name (i.e. `():`) when we define a class. Similarly, the body of our class is indented like a function's body is.

```
def my_function():  
    # the details of the  
    # function go here
```

```
class MyClass():  
    # the details of the  
    # class go here
```

Figure 2.7.1.1 : Defining a Class

2.7.2 `__init__` method in Class:

- The `__init__` method — also called a constructor — is a special method that runs when an instance is created so we can perform any tasks to set up the instance.
- The `__init__` method has a special name that starts and ends with two underscores: `__init__()`.

CHAPTER 3

CASE STUDY

3.1 PROBLEM STATEMENT:

The main aim of the Recommender system is to provide recommendations based on recorded information on the users' preferences. These systems use information filtering techniques to process information and provide the user with potentially more relevant items.

3.2 DATA SET:

The given data set consists of the following parameters:

File -movies.csv

Columns:

- movie id
- title
- genre

File-ratings.csv

Columns:

- user id
- movie id
- rating
- timestamp

File-tags.csv

Columns:

- user id
- movie id
- tags
- timestamp

File-links.csv

Columns:

- movieId
- imdbId
- tmdbId

3.3 OBJECTIVE OF THE CASE STUDY:

To get a better understanding and chalking out a plan of action for recommending a particular movie to the user, we have considered movie titles, ratings, id's of the movies and the users and reasoned out the factors of choice of the movies that a particular user may wish to see. It is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item. They are primarily used in commercial applications.

CHAPTER 4

MODEL BUILDING

4.1 PREPROCESSING OF THE DATA:

Preprocessing of the data actually involves the following steps:

4.1.1 GETTING THE DATASET:

We can get the data set from the database or we can get the data from client.

4.1.2 IMPORTING THE LIBRARIES:

We have to import the libraries as per the requirement of the algorithm.

Importing the Libraries

```
[94] # Importing the required libraries
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np
import ast
import os
from scipy import stats
plt.style.use('seaborn-bright')
%matplotlib inline
from ast import literal_eval
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.metrics.pairwise import linear_kernel, cosine_similarity
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import wordnet
#from surprise import Reader, Dataset, SVD
import warnings; warnings.simplefilter('ignore')
```

Figure 4.1.2.1 : Importing Libraries

4.1.3 VERSIONS OF THE PACKAGES

TO CHECK THE VERSION OF THE PACKAGES IMPORTED



```
print(pd.__version__)  
print(np.__version__)  
print(sns.__version__)
```



```
1.0.5  
1.18.5  
0.10.1
```

Figure 4.1.3.1: Versions of the packages

4.1.4 IMPORTING THE DATA-SET:

Pandas in python provide an interesting method `read_csv()`. The `read_csv` function reads the entire data set from a comma separated values file and we can assign it to a Data Frame to which all the operations can be performed. It helps us to access each and every row as well as columns and each and every value can be access using the data frame. Any missing value or NaN value have to be cleaned.

READING THE DATA SET

```
#Reading the movie data
```

```
movie_names=pd.read_csv("drive/My Drive/project /movies.csv")
```

```
movie_names.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

Figure 4.1.4.1: Reading the movie-dataset

```
#Reading the ratings data
```

```
ratings_data=pd.read_csv("drive/My Drive/project /ratings.csv")
```

```
ratings_data.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

Figure 4.1.4.2 : Reading the ratings-dataset

```
#Reading the tags file
tags_data=pd.read_csv("drive/My Drive/project /tags.csv")
tags_data.head()
```

	userId	movieId	tag	timestamp
0	2	60756	funny	1445714994
1	2	60756	Highly quotable	1445714996
2	2	60756	will ferrell	1445714992
3	2	89774	Boxing story	1445715207
4	2	89774	MMA	1445715200

Figure 4.1.4.3: Reading the tags-dataset

```
#Reading the links file
links_data=pd.read_csv("drive/My Drive/project /links.csv")
links_data.head()
```

	movieId	imdbId	tmdbId
0	1	114709	862.0
1	2	113497	8844.0
2	3	113228	15602.0
3	4	114885	31357.0
4	5	113041	11862.0

Figure 4.1.4.4: Reading the links-dataset

4.1.5 HANDLING MISSING VALUES:

Missing values can be handled in many ways using some inbuilt methods:

- (a) `dropna()`
- (b) `fillna()`
- (c) `interpolate()`
- (d) mean imputation and median imputation

(a) `dropna()`:

`dropna()` is a function which drops all the rows and columns which are having the missing values(i.e. NaN)

- `dropna()` function has a parameter called `how` which works as follows
- if `how = 'all'` is passed then it drops the rows where all the columns of the particular row are missing
- if `how = 'any'` is passed then it drops the rows where all the columns of the particular row are missing

(b) `fillna()`:

`fillna()` is a function which replaces all the missing values using different ways.

- `fillna()` also have parameters called `method` and `axis` .
- if we use `method = 'ffill'` where `ffill` is a method called forward fill, which carry forwards the previous row's value.
- if we use `method = 'bfill'` where `bfill` is a method called backward fill, which carry backward the next row's value.

- if we use `method = 'ffill'` , `axis = 'columns'` then it carry forwards the previous column's value.
- if we use `method = 'bfill'` , `axis = 'columns'` then it carry backward the next column's value.

(c) `interpolate()`:

- `interpolate()` is a function which comes up with a guess value based on the other

values in the dataset and fills those guess values in the place of missing values .

(d)mean and median imputation

- mean and median imputation can be performed by using fillna().
- mean imputation calculates the mean for the entire column and replaces the missing values in that column with the calculated mean.
- median imputation calculates the median for the entire column and replaces the missing values in that column with the calculated median.

MISSING VALUES

```
####Checking for the null or missing values in data
links_data.isnull().sum()

movieId    0
imdbId     0
tmdbId     8
dtype: int64
```

Figure 4.1.5.1 Missing values in tmdbId

tmdbId has 8 null values, they can be discarded or filled with suitable values, but since they are the id columns we are supposed to fill them.

4.1.6 CATEGORICAL DATA:

- Machine Learning models are based on equations, we need to replace the text by numbers. So that we can include the numbers in the equations.
- Categorical Variables are of two types: Nominal and Ordinal
- **Nominal**: The categories do not have any numeric ordering in between them. They don't have any ordered relationship between each of them. Examples: Male or Female, any colour .
- **Ordinal**: The categories have a numerical ordering in between them. Example: Graduate is less than Post Graduate, Post Graduate is less than Ph.D. customer satisfaction survey, high low medium .
- Categorical data can be handled by using dummy variables, which are also called as indicator variables.
- Handling categorical data using dummies:

In pandas library we have a method called `get_dummies()` which creates dummy variables for those categorical data in the form of 0's and 1's.

Once these dummies got created we have to concat this dummy set to our Data frame or we can add that dummy set to the data frame.


NOTE: There are no categorical columns in these datasets.

CHAPTER 5

PREPROCESSING/FEATURE ENGINEERING AND EDA

5.1 Statistical Analysis

Statistical Analysis

 `movie_names.describe(include="all")`

	movieId	title	genres
count	9742.000000	9742	9742
unique	NaN	9737	951
top	NaN	War of the Worlds (2005)	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

Figure 5.1.1: Statistical Analysis

```
ratings_data.describe(include="all")
```

```
tags_data.describe(include="all")
```

```
links_data.describe()
```

Figure 5.1.2: Statistical Analysis

```
movie_names.shape
```

```
(9742, 3)
```

[+ Code](#)[+ Text](#)

There are 9742 rows and 3 columns

```
ratings_data.shape
```

```
(100836, 3)
```

There are 100836 rows and 3 columns

```
tags_data.shape
```

```
(3683, 3)
```

There are 3683 rows and 3 columns

```
links_data.shape
```

```
(9742, 3)
```

There are 9742 rows and 3 columns

Figure 5.1.3: Describing the shape of the Files

5.2 Generating Plots

Data Visualization

```
[ ] #Visualizing the missing values with heatmap
sns.heatmap(movie_names.isna())
```

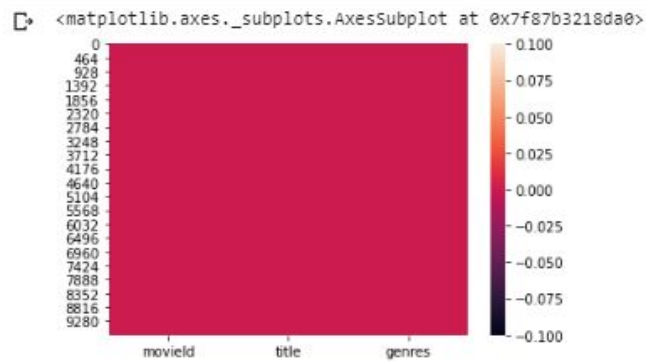


Figure 5.2.1 Missing values with heatmap (seaborn)

missingno □ It is specially used for checking (visualizing) the missing values.

```
import missingno as msno
msno.matrix(movie_names)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f87af0fab70>

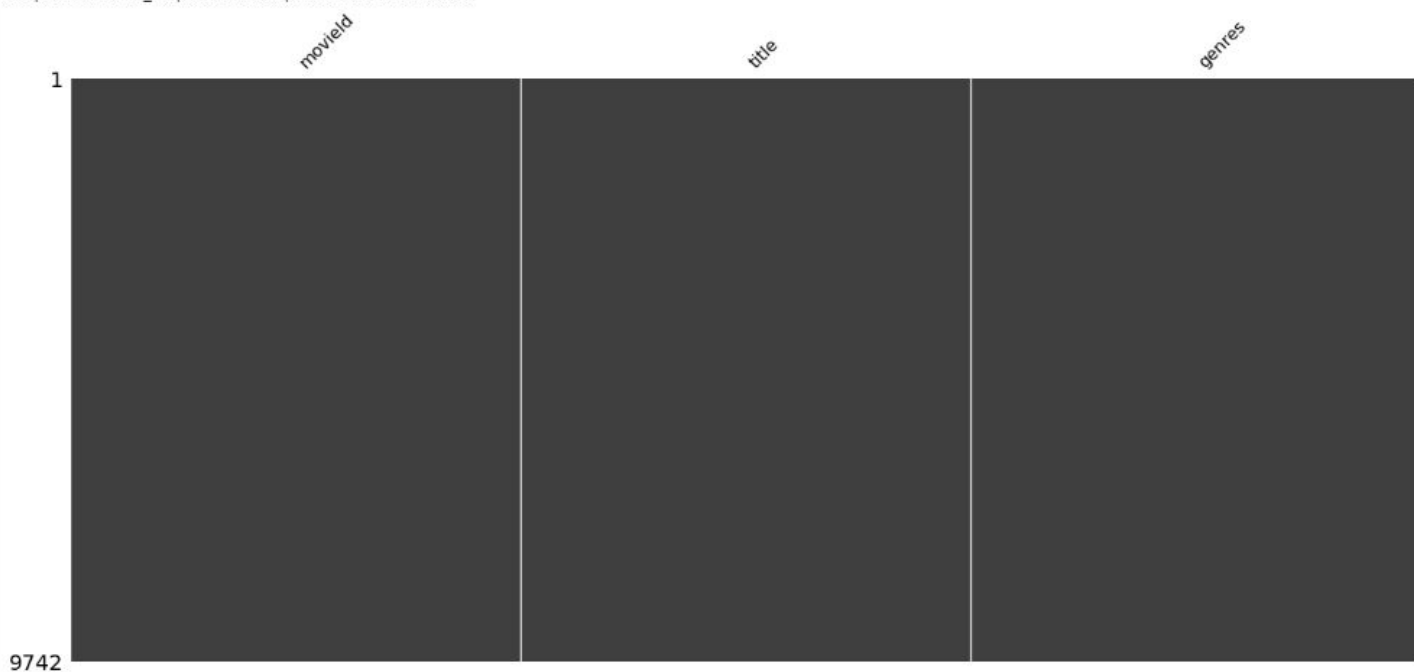


Figure 5.2.2: Visualizing with missing no

5.3 Data Type Conversions

```
#We specify the parantheses so we don't conflict with movies that have years in their titles
movie_names['year'] = movie_names.title.str.extract('(\d\d\d\d)',expand=False)
##Removing the parentheses
movie_names['year'] = movie_names.year.str.extract('(\d\d\d\d)',expand=False)
###Removing the years from the 'title' column
movie_names['title'] = movie_names.title.str.replace('(\d\d\d\d)', '')
##Applying the strip function to get rid of any ending whitespace characters that may have appeared
movie_names['title'] = movie_names['title'].apply(lambda x: x.strip())
movie_names.shape
```

(9742, 4)

movie_names.genres

```
0      Adventure|Animation|Children|Comedy|Fantasy
1      Adventure|Children|Fantasy
2      Comedy|Romance
3      Comedy|Drama|Romance
4      Comedy
...
9737    Action|Animation|Comedy|Fantasy
9738    Animation|Comedy|Fantasy
9739      Drama
9740    Action|Animation
9741    Comedy
```

Figure 5.3.1: Separating the release date of the movie and the title

```
movie_names['genres'] = movie_names.genres.str.split('|')
movie_names.head()
```

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

Figure 5.3.2: Splitting the genres with “|”

CHAPTER 6

FEATURE SELECTION:

6.1 Select relevant features for the analysis:

Select only relevant and necessary columns.

6.2 Drop irrelevant features:

Removing unnecessary and irrelevant columns.

```
# Removing Unnecessary or irrelevant columns
ratings_data=ratings_data.drop("timestamp",axis=1)
ratings_data.head()
```

	userId	movieId	rating
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0

```
# Removing Unnecessary or irrelevant columns
tags_data=tags_data.drop("timestamp",axis=1)
tags_data.head()
```

	userId	movieId	tag
0	2	60756	funny
1	2	60756	Highly quotable
2	2	60756	will ferrell
3	2	89774	Boxing story
4	2	89774	MMA

Figure 6.2.1: Removing the Irrelevant columns

CHAPTER 7

MODEL BUILDING AND EVALUATION:

Build recommendation system

7.1 Popularity based Recommendation System:

7.1.1 Brief about the algorithms used:

The Simple Recommender offers generalized recommendations to every user based on movie popularity and (sometimes) genre.

7.1.2 Approach:

The basic idea behind this recommender is that movies that are more popular and more critically acclaimed will have a higher probability of being liked by the average audience.

Top review websites like IMDb and Rotten Tomatoes maintain a database of movies and their popularity in terms of reviews and ratings. Utilizing this data to recommend the most popular movies to users based on their star ratings, could increase their content consumption.

The popularity-based recommendation system eliminates the need for knowing other factors like user browsing history, user preferences, the star cast of the movie, genre, and other factors. Hence, the single-most factor considered is the star rating

What we are actually doing?

The implementation of this model is extremely trivial. All we have to do is sort our movies based on ratings and popularity and display the top movies of our list. As an added step, we can pass in a genre argument to get the top movies of a particular genre.

Since, both ratings data and movie names have movieId, we merge on movieId.

Merging ratings_data and movie_names data on *movieId*

```
[ ] #movie_data is the new DataFrame for the merged data
movie_data=pd.merge(ratings_data,movie_names,on='movieId')
movie_data.head()
```

	userId	movieId	rating	title	genres	year
0	1	1	4.0	Toy Story	[Adventure, Animation, Children, Comedy, Fantasy]	1995
1	5	1	4.0	Toy Story	[Adventure, Animation, Children, Comedy, Fantasy]	1995
2	7	1	4.5	Toy Story	[Adventure, Animation, Children, Comedy, Fantasy]	1995
3	15	1	2.5	Toy Story	[Adventure, Animation, Children, Comedy, Fantasy]	1995
4	17	1	4.5	Toy Story	[Adventure, Animation, Children, Comedy, Fantasy]	1995

Figure 7.1.2.1: Merging on MovieId

Now, we are checking how many viewers have reviewed for a particular movie.

We are creating a data frame for analysis called trend.

We are grouping on the title name and rating of the particular movie.

```
#Creating a dataframe for analysis
#We are grouping on the title name and rating of the particular movie
trend=pd.DataFrame(movie_data.groupby('title')['rating'].mean())
trend['Total number of ratings'] = pd.DataFrame(movie_data.groupby('title')['rating'].count())
trend
```

	rating	Total number of ratings
title		
'71	4.000000	1
'Hellboy': The Seeds of Creation	4.000000	1
'Round Midnight	3.500000	2
'Salem's Lot	5.000000	1
'Til There Was You	4.000000	2
...
eXistenZ	3.863636	22
xXx	2.770833	24
xXx: State of the Union	2.000000	5
¡Three Amigos!	3.134615	26
À nous la liberté (Freedom for Us)	1.000000	1

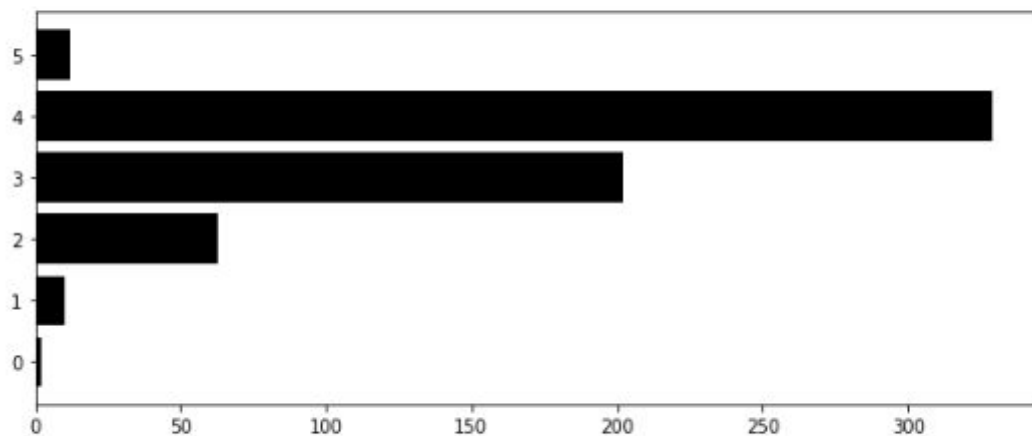
9446 rows × 2 columns

Figure 7.1.2.2: Creating trend data frame for analysis

7.1.3 Visualizing the trend :

Plotting the ratings with number of movies:

```
#Visualizing the trend
#plot rounded-up ratings with number of movies
plt.figure(figsize =(10, 4))
ax=plt.barh(trend['rating'].round(),trend['Total number of ratings'],color="black")
plt.show()
```

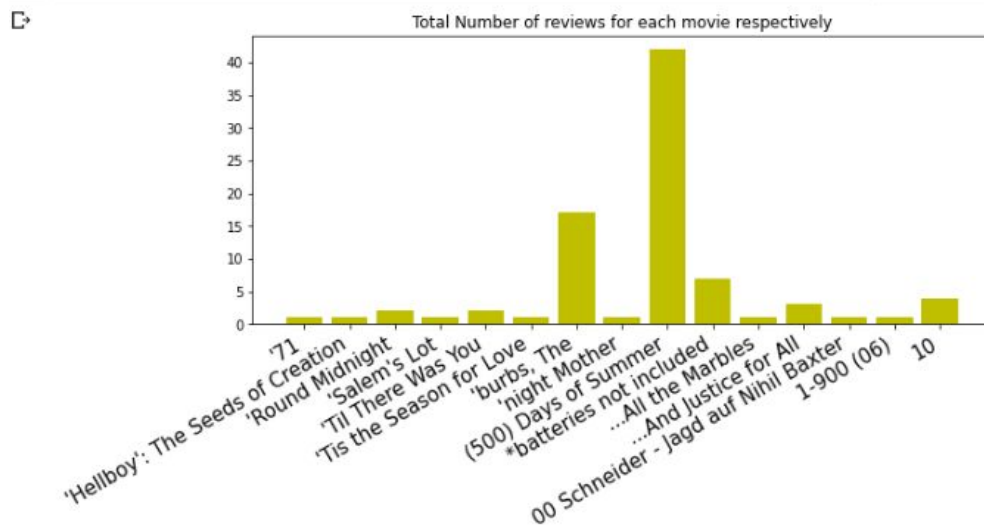


- We can observe that more than 300 movies were reviewed a rating of 4 on a scale of 0-5

Figure 7.1.3.1: Plot Showing ratings with respect to the movies

Plotting the Bar Graph describing the number of reviews for first 15 movies:

```
#Bar graph describing the number of reviews for first 15 movies
plt.figure(figsize=(10, 4))
ax=plt.subplot()
ax.bar(trend.head(15).index,trend['Total number of ratings'].head(15),color='y')
ax.set_xticklabels(trend.index,rotation=30,fontsize='15',horizontalalignment="right")
ax.set_title("Total Number of reviews for each movie respectively")
plt.show()
```



- We can observe from the bar graph that among first 15 movies (500)Days of Summer has highest number of reviews

Figure 7.1.3.2: Bar graph for number of movies for first 15 movies

Finally, we are checking for the Popularity of the movies given in the Data set.

- We do this by calculating the mean ratings of all movies.

Checking the Popularity

```
[ ] # Calculate mean rating of all movies and check the popular high rating movies
movie_data.groupby('title')['rating'].mean().sort_values(ascending=False)
```

```
title
Sorority House Massacre          5.0
Entertaining Angels: The Dorothy Day Story  5.0
Passenger, The (Professione: reporter)      5.0
Little Dieter Needs to Fly            5.0
Human Condition III, The (Ningen no joken III)  5.0
...
Wizards of the Lost Kingdom II        0.5
Reptilicus                          0.5
Brothers Solomon, The                0.5
Giant Spider Invasion, The           0.5
Anaconda: The Offspring              0.5
Name: rating, Length: 9446, dtype: float64
```

- Therefore, the Recommender System based on Popularity of the movie is observed

Figure 7.1.3.3: Checking the Popularity

Therefore, a Recommender System based on Popularity of the movie is observed.

The movies with higher popularity are generally recommended to the users.

7.2 Collaborative Filtering based Recommendation system

Collaborative filtering (CF) is a technique used by recommender systems.

Collaborative Filtering is based on the idea that users similar to a me can be used to predict how much I will like a particular product or service those users have used/experienced but I have not.

There are two types of collaborative filtering, namely:

- User – User collaborative filtering
- Item – Item collaborative filtering

One stating feature of this Recommendation System is that it doesn't care what the movie is (or what it contains). It works purely on the basis of assigned movie Id and tries to predict ratings based on how the other users have perceive the movie.

Instead of implementing CF from scratch, we can use the Surprise library that is used for extremely powerful algorithms like Singular Value Decomposition (SVD) to minimise RMSE (Root Mean Square Error) and give great recommendations.

7.2.1 Install surprise package

To install Surprise package ,we can use the following command:

```
pip install surprise
```

```
Requirement already satisfied: surprise in /usr/local/lib/python3.6/dist-packages (0.1)
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.6/dist-packages (from surprise) (1.1.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise->surprise) (0.16.0)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise->surprise) (1.4.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise->surprise) (1.15.0)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise->surprise) (1.18.5)
```

↑ ↓ G

```
from surprise import Reader, Dataset, SVD
```

Figure 7.2.1.1: Installing Surprise package

Instantiating the Reader class:

```
] # surprise reader API to read the dataset
reader = Reader()
```

Figure 7.2.1.3: Instantiating the Reader class

We use Ratings data for this approach

```
We use ratings_data for this approach
```

```
] data = Dataset.load_from_df(ratings_data[['userId', 'movieId', 'rating']], reader)
```

```
] trainset= data.build_full_trainset()
```

Figure 7.2.1.2: Building the trainset

Instantiating SVD class:

Fitting the trainset

:

```
#Instantiating SVD Class
svd = SVD()
#Fitting the trainset data
svd.fit(trainset)
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7f3cbcd3780>
```

Figure 7.2.1.3: Instantiating the SVD class and fitting the trainset data

7.2.2 Estimating the model:

```
#Estimating the model
svd.estimate(1, 105)
```

```
3.8302740879576684
```

Figure 7.2.1.4: Estimating the model

CONCLUSION:

It is concluded after performing thorough Exploratory Data analysis which include Stats models and also Heat maps which are computed to get a clear understanding of the data sets (which parameter has most abundant effect on the study case) and its come to point of getting the solution for the problem statement being , that recommending an appropriate movie to the user to watch based on his/her's choices.

References:

[1] <https://grouplens.org/datasets/movielens/latest/>

[2] https://en.wikipedia.org/wiki/Machine_learning