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

SATHUNURI DINESH

221710302059

Under the Guidance of

Assistant Professor

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 titled **"Movie Recommender System"** to GITAM (Deemed to Be University), Hyderabad in partial fulfillment of the requirement for the award of degree of "**Bachelor of Technology**" in "**Computer Science and Engineering**".

I declare that it was carried out independently by me under the guidance of **Mr.**, Asst. 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   Student Name: SATHUNURI DINESH

Date: Student Roll No: 221710302059

GITAM (DEEMED TO BE UNIVERSITY)

Hyderabad-502329, India

Dated:

CERTIFICATE

This is to certify that the Industrial Training Report titled **"Movie Recommender System"** is being submitted by SATHUNURI DINESH (221710302059) in partial fulfillment of the requirement for the award of **Bachelor of Technology in Computer Science and Engineering** at GITAM (Deemed to Be University), Hyderabad during the academic year 2020-21

It is the faithful record work carried out by him at the **Computer Science and Engineering Department**, GITAM University Hyderabad Campus under my guidance and supervision.

Dr. S.Phani Kumar

Assistant Professor   Professor and HOD

ACKNOWLEDGEMENT

Apart from my effort, the success of this internship largely depends on the encouragement and guidance of many others. I take this opportunity to express my gratitude to the people who have helped me in the successful competition of this internship.

I would like to thank respected **Dr. N. Siva Prasad,** Pro Vice Chancellor, GITAM Hyderabad and **Dr. N.Seetharamaiah,** Principal, GITAM Hyderabad

I would like to thank **Dr.S.Phani Kumar,** Head of the Department of Computer Science and Engineering for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a internship report. It helped me a lot to realize of what we study for.

I would like to thank the respected faculties **Mr.** who helped me to make this internship a successful accomplishment.

I would also like to thank my friends who helped me to make my work more organized and well-stacked till the end.

SATHUNURI DINESH

221710302059

ABSTRACT

A recommendation engine filters the data using different algorithms and recommends the most relevant items to users. It first captures the past behavior of a customer and based on that, recommends products which the users might be likely to buy. If a completely new user visits an e-commerce site, that site will not have any past history of that user. So how does the site go about recommending products to the user in such a scenario? One possible solution could be to recommend the best selling products, i.e. the products which are high in demand. Another possible solution could be to recommend the products which would bring the maximum profit to the business. Three main approaches are used for our recommender systems. One is Demographic Filtering 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. Since each user is different, this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience. Second is content-based filtering, where we try to profile the user's 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

List of Figures

1. MACHINE LEARNING

1.1 Introduction:

Over the past two decades Machine Learning has become one of the mainstays of information technology and with that, a rather central, albeit usually hidden, part of our life. With the ever increasing amounts of data becoming available there is good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress. Human designers often produce machines that do not work as well as desired in the environments in which they are used. In fact, certain characteristics of the working environment might not be completely known at design time. Machine learning methods can be used for on- the-job improvement of existing machine designs. The amount of knowledge available about certain tasks might be too large for explicit encoding by humans. Machines that learn this knowledge gradually might be able to capture more of it than humans would want to write down. Environments change over time. Machines that can adapt to a changing environment would reduce the need for constant redesign. New knowledge about tasks is constantly being discovered by humans. There is a constant stream of new events in the world. Continuing redesign of AI systems to conform to new knowledge is impractical, but machine learning methods might be able to track much of it.

1.2: Importance of Machine Learning:

Machine learning is a branch of artificial intelligence that aims at enabling machines to perform their jobs skillfully by using intelligent software. The statistical learning methods constitute the backbone of intelligent software that is used to develop machine intelligence. Because machine learning algorithms require data to learn, the discipline must have connection with the discipline of database. Similarly, there are familiar terms such as Knowledge Discovery from Data (KDD), data mining, and pattern recognition. One wonders how to view the big picture in which such connection is illustrated.
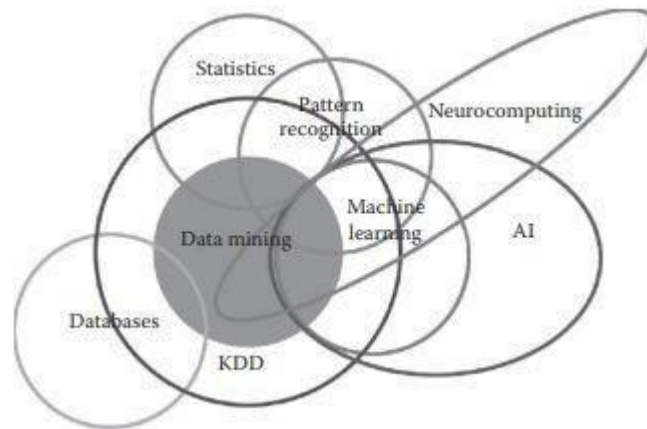
Fig 1.2.1 Usage of Machine learning in different fields

There are some tasks that humans perform effortlessly or with some efforts, but we are unable to explain how we perform them. For example, we can recognize the speech of our friends without much difficulty. If we are asked how we recognize the voices, the answer is very difficult for us to explain. Because of the lack of understanding of such phenomenon (speech recognition in this case), we cannot craft algorithms for such scenarios. Machine learning algorithms are helpful in bridging this gap of understanding.

The idea is very simple. We are not targeting to understand the underlying processes that help us learn. We write computer programs that will make machines learn and enable them to perform tasks, such as prediction. The goal of learning is to construct a model that takes the input and produces the desired result. Sometimes, we can understand the model, whereas, at other times, it can also be like a black box for us, the working of which cannot be intuitively

explained. The model can be considered as an approximation of the process we want machines to mimic. In such a situation, it is possible that we obtain errors for some input, but most of the time, the model provides correct answers. Hence, another measure of performance (besides performance of metrics of speed and memory usage) of a machine learning algorithm will be the accuracy of results.

1.3: Uses of Machine Learning:

Artificial Intelligence (AI) is everywhere. Possibility is that you are using it in one way or the other and you don't even know about it. One of the popular applications of AI is Machine Learning (ML), in which computers, software, and devices perform via cognition (very similar to human brain). Herein, we share few examples of machine learning that we use every day and perhaps have no idea that they are driven by ML. These are some the uses and applications of ML

i. Virtual Personal Assistants:

Siri, Alexa, Google Now are some of the popular examples of virtual personal assistants. As the name suggests, they assist in finding information, when asked over voice. All you need to do is activate them and ask "What is my schedule for today?", "What are the flights from Germany to London", or similar questions. For answering, your personal assistant looks out for the information, recalls your related queries, or send a command to other resources (like phone apps) to collect info. You can even instruct assistants for certain tasks like "Set an alarm for 6 AM next morning", "Remind me to visit Visa Office day after tomorrow".

Machine learning is an important part of these personal assistants as they collect and refine the information on the basis of your previous involvement with them. Later, this set of data utilized to render results that are tailored to your preferences.

Virtual Assistants are integrated to a variety of platforms. For example:

- Smart Speakers: Amazon Echo and Google Home
- Smart phones: Samsung Bixby on Samsung S8
- Mobile Apps: Google Allo

ii. Predictions while Commuting:

Traffic Predictions*:* We all have been using GPS navigation services. While we do that, our current locations and velocities are being saved at a central server for managing traffic. This data is then used to build a map of current traffic. While this helps in preventing the traffic and does congestion analysis, the underlying problem is that there are less number of cars that are

equipped with GPS. Machine learning in such scenarios helps to estimate the regions where congestion can be found on the basis of daily experiences.

Online Transportation Networks*:* When booking a cab, the app estimates the price of the ride. When sharing these services, how do they minimize the detours? The answer is machine learning. Jeff Schneider, the engineering lead at Uber ATC reveals in an interview that they use ML to define price surge hours by predicting the rider demand. In the entire cycle of the services, ML is playing a major role.

iii. Social Media Services:

From personalizing your news feed to better ads targeting, social media platforms are utilizing machine learning for their own and user benefits. Here are a few examples that you must be noticing, using, and loving in your social media accounts, without realizing that these wonderful features are nothing but the applications of ML.

People You May Know: Machine learning works on a simple concept: understanding with experiences. Face book continuously notices the friends that you connect with, the profiles that you visit very often, your interests, workplace, or a group that you share with someone etc. On the basis of continuous learning, a list of Face book users is suggested that you can become friends with.

Face Recognition: You upload a picture of you with a friend and Face book instantly recognizes that friend. Face book checks the poses and projections in the picture, notice the unique features, and then match them with the people in your friend list. The entire process at the backend is complicated and takes care of the precision factor but seems to be a simple application of ML at the front end.

Similar Pins: Machine learning is the core element of Computer Vision, which is a technique to extract useful information from images and videos. Pinterest uses computer vision to identify the objects (or pins) in the images and recommend similar pins accordingly.

iv. Search Engine Result Refining:

Google and other search engines use machine learning to improve the search results for you. Every time you execute a search, the algorithms at the backend keep a watch at how you respond to the results. If you open the top results and stay on the web page for long, the search engine assumes that the results it displayed were in accordance to the query. Similarly, if you reach the second or third page of the search results but do not open any of the results, the search engine estimates that the results served did not match requirement. This way, the algorithms working at the backend improve the search results.

v. Product Recommendations:

You shopped for a product online few days back and then you keep receiving emails for shopping suggestions. If not this, then you might have noticed that the shopping website or the app recommends you some items that somehow match with your taste. On the basis of your behavior with the website/app, past purchases, items liked or added to cart, brand preferences etc., the product recommendations are made.

vi. Online Fraud Detection:

Machine learning is proving its potential to make cyberspace a secure place and tracking monetary frauds online is one of its examples. For example: PayPal is using ML for protection against money laundering. The company uses a set of tools that helps them to compare millions of transactions taking place and distinguish between legitimate or illegitimate transactions taking place between the buyers and sellers.
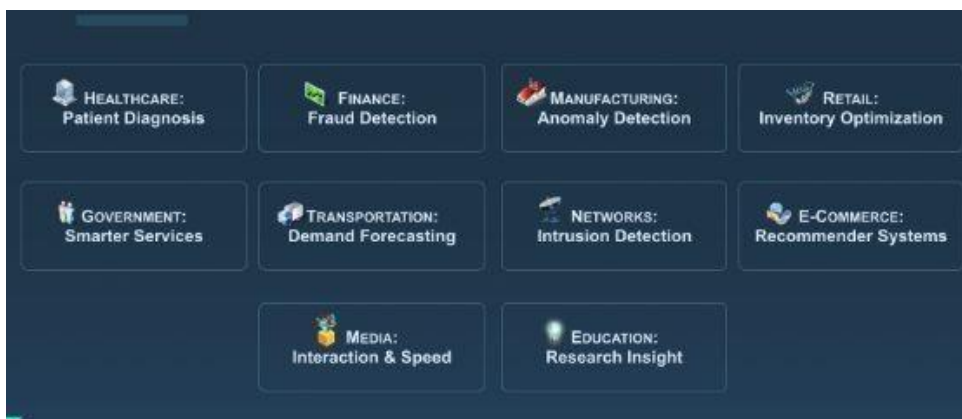
Fig 1.3.1 Uses of Machine learning

1.4: Types of Machine Learning:

There are 3 types of Machine learning which are widely used in today's world these are:
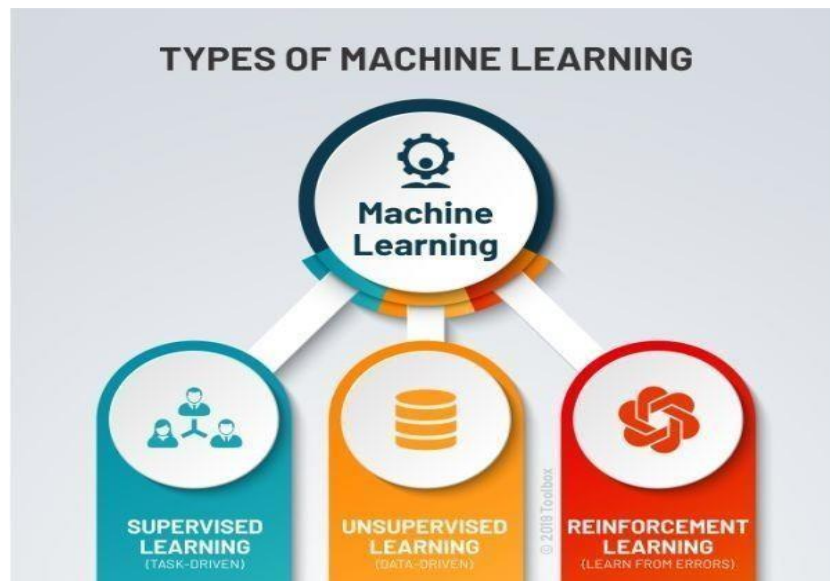
Fig 1.4.1 types of ML

1.4.1: Supervised Learning:

Supervised learning is one of the most basic types of machine learning. In this type, the machine learning algorithm is trained on labeled data. Even though the data needs to be labeled accurately for this method to work, supervised learning is extremely powerful when used in the right circumstances. In supervised learning, the ML algorithm is given a small training dataset to work with. This training dataset is a smaller part of the bigger dataset and serves to give the algorithm a basic idea of the problem, solution, and data points to be dealt with. The training dataset is also very similar to the final dataset in its characteristics and provides the algorithm with the labeled parameters required for the problem. The algorithm then finds relationships between the parameters given, essentially establishing a cause and effect relationship between the variables in the dataset. At the end of the training, the algorithm has an idea of how the data works and the relationship between the input and the output.

1.4.2: Unsupervised Learning:

Unsupervised machine learning holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine-readable, allowing much larger datasets to be worked on by the program. In supervised learning, the labels allow the algorithm to find the exact nature of the relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings. The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post-deployment development than supervised learning algorithms.

1.4.3: Reinforcement Learning:

It directly takes inspiration from how human beings learn from data in their lives. It features an algorithm that improves upon itself and learns from new situations using a trial- and-error method. Favorable outputs are encouraged or 'reinforced', and non-favorable outputs are discouraged or 'punished'. Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not. In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result. In typical reinforcement learning use-cases, such as finding the shortest route between two points on a map, the solution is not an absolute value. Instead, it takes on a score of effectiveness, expressed in a percentage value. The higher this percentage value is, the more reward is given to the algorithm. Thus, the program is trained to give the best possible solution for the best possible reward.
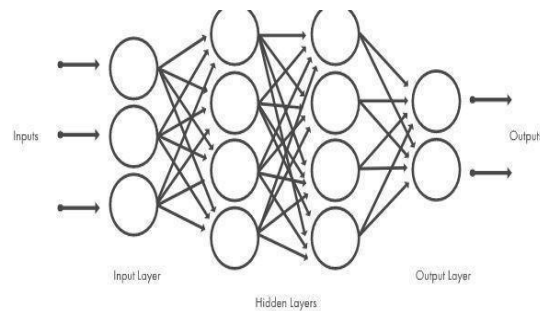
2. DEEP LEARNING

2.1: Deep Learning and its Importance:

Deep learning algorithms run data through several "layers" of neural network algorithms, each of which passes a simplified representation of the data to the next layer. Most machine learning algorithms work well on datasets that have up to a few hundred features, or columns.

Basically deep learning is itself a subset of machine learning but in this case the machine learns in a way in which humans are supposed to learn. The structure of deep learning model is highly similar to a human brain with large number of neurons and nodes like neurons in human brain thus resulting in artificial neural network. In applying traditional machine learning algorithms we have to manually select input features from complex data set and then train them which becomes a very tedious job for ML scientist but in neural networks we don't have to manually select useful input features, there are various layers of neural networks for handling complexity of the data set and algorithm as well. In my recent project on human activity recognition, when we applied traditional machine learning algorithm like K-NN then we have to separately detect human and its activity also had to select impactful input parameters manually which became a very tedious task as data set was way too complex but the complexity dramatically reduced on applying artificial neural network, such is the power of deep learning. Yes it's correct that deep learning algorithms take lots of time for training sometimes even weeks as well but its execution on new data is so fast that it's not even comparable with traditional ML algorithms. Deep learning has enabled Industrial Experts to overcome challenges which were impossible, decades ago like Speech and Image recognition and Natural Language Processing. Majority of the Industries are currently depending on it, be it Journalism, Entertainment, Online Retail Store, Automobile, Banking and Finance, Healthcare, Manufacturing or even Digital Sector. Video recommendation, Mail Services, Self-Driving cars, Intelligent Chat bots, Voice Assistants are just trending achievements of Deep Learning. Furthermore, Deep learning can most profoundly be considered as future of Artificial Intelligence due to constant rapid increase in amount of data as well as the gradual development in hardware field as well, resulting in better computational power.

Fig 2.1.1 Deep Neural network

2.2: Uses of Deep Learning:

Translations:

Although automatic machine translation isn't new, deep learning is helping enhance automatic translation of text by using stacked networks of neural networks and allowing translations from images.

Adding color to black-and-white images and videos:

It is used to be a very time-consuming process where humans had to add color to black- and-white images and videos by hand can now be automatically done with deep-learning models.

Language recognition:

Deep learning machines are beginning to differentiate dialects of a language. A machine decides that someone is speaking English and then engages an AI that is learning to tell the differences between dialects. Once the dialect is determined, another AI will step in that specializes in that particular dialect. All of this happens without involvement from a human.

Autonomous vehicles:

There's not just one AI model at work as an autonomous vehicle drives down the street. Some deep-learning models specialize in streets signs while others are trained to recognize pedestrians. As a car navigates down the road, it can be informed by up to millions of individual AI models that allow the car to act.

Computer vision:

Deep learning has delivered super-human accuracy for image classification, object detection, image restoration and image segmentation—even handwritten digits can be recognized. Deep learning using enormous neural networks is teaching machines to automate the tasks performed by human visual systems.

Text generation:

The machines learn the punctuation, grammar and style of a piece of text and can use the model it developed to automatically create entirely new text with the proper spelling, grammar and style of the example text. Everything from Shakespeare to Wikipedia entries has been created.

Deep-learning robots:

Deep-learning applications for robots are plentiful and powerful from an impressive deep-learning system that can teach a robot just by observing the actions of a human completing a task to a housekeeping robot that's provided with input from several other AIs in order to take action. Just like how a human brain processes input from past experiences, current input from senses and any additional data that is provided, deep-learning models will help robots execute tasks based on the input of many different AI opinions.
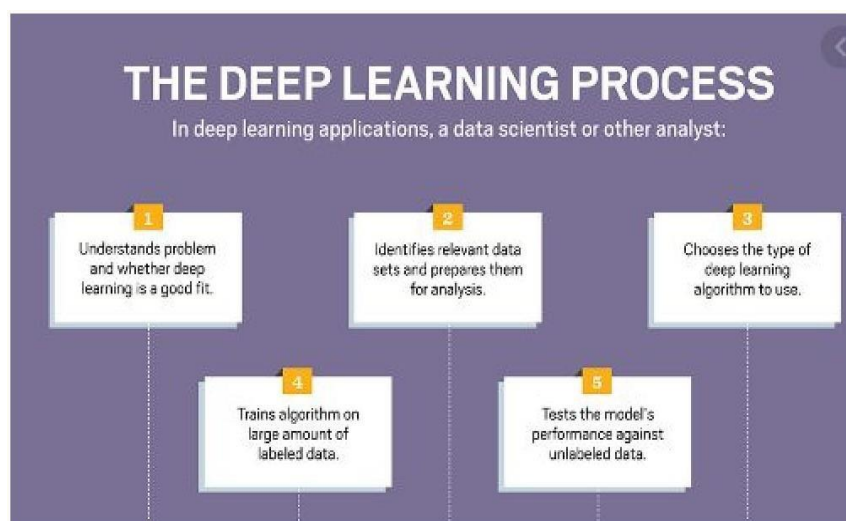
Fig 2.2.1 The deep learning process

2.3: Relation between Data Mining, Machine Learning and Deep Learning:
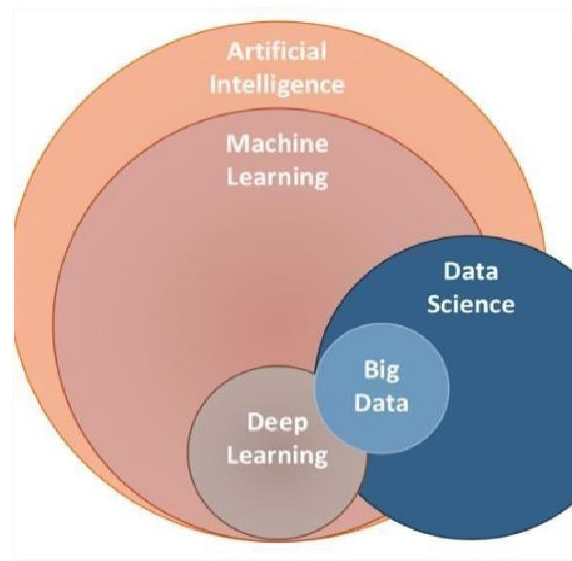
Fig 2.3 Relation between DM, ML, DL

The deep learning, data mining and machine learning share a foundation in data science, and there certainly is overlap between the two. Data mining can use machine learning algorithms to improve the accuracy and depth of analysis, and vice-versa; machine learning can use mined data as its foundation, refining the dataset to achieve better results.

You could also argue that data mining and machine learning are similar in that they both seek to address the question of how we can learn from data. However, the way in which they achieve this end, and their applications, form the basis of some significant differences.
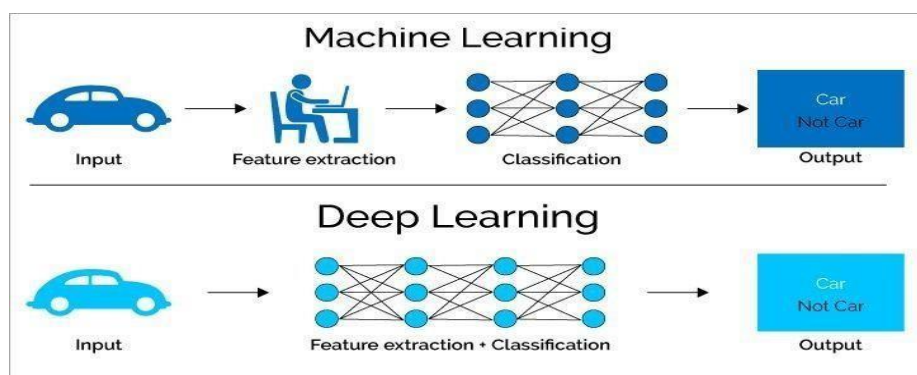
Fig 2.3.1 process in machine learning and deep learning

~ 12 ~

Machine Learning comprises of the ability of the machine to learn from trained data set and predict the outcome automatically. It is a subset of artificial intelligence.

Deep Learning is a subset of machine learning. It works in the same way on the machine just like how the human brain processes information. Like a brain can identify the patterns by comparing it with previously memorized patterns, deep learning also uses this concept.

Deep learning can automatically find out the attributes from raw data while machine learning selects these features manually which further needs processing. It also employs artificial neural networks with many hidden layers, big data, and high computer resources.

Data Mining is a process of discovering hidden patterns and rules from the existing data. It uses relatively simple rules such as association, correlation rules for the decision-making process, etc.

Deep Learning is used for complex problem processing such as voice recognition etc. It uses Artificial Neural Networks with many hidden layers for processing. At times data mining also uses deep learning algorithms for processing the data.

3. PYTHON

3.1: Introduction:

Python is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code. Python is a programming language that lets you work quickly and integrate systems more efficiently.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined dynamic binding make it very attractive for rapid application. Use as a scripting or glue language to connect existing components together. Python is simple, easy to learn syntax emphasizes readability and therefore reduces the maintenance. Python supports modules, packages and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

3.2: Setup of Python:

- Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

- The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python https://www.python.org/

3.2.1: Installation (using python IDLE):

- To start, go to python.org/downloads and then click on the button to download the latest version of Python

- We can download python IDLE in windows, Mac and Linux operating systems

Figure 3.2.1.1: Python download

- Run the .exe file that you just downloaded and start the installation of Python by clicking on Install Now

- We can give environmental variable i.e. path after completion of downloading

Fig 3.2.1.2 python installation

- When python is installed, a program called IDLE is also installed along with it. It provides a graphical user interface to work with python.

Fig 3.2.1.3 IDLE

3.2.2: Python Installation 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. Anaconda for Windows installation:
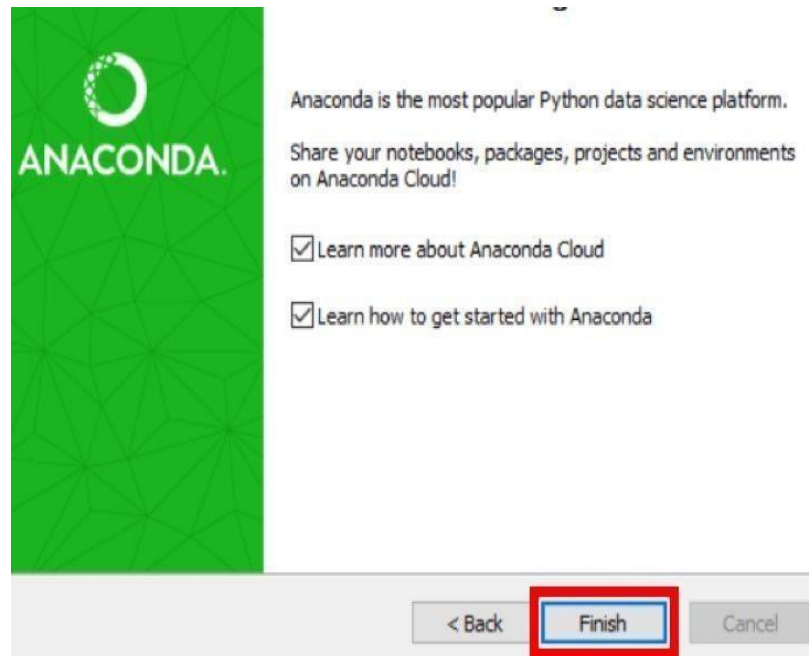
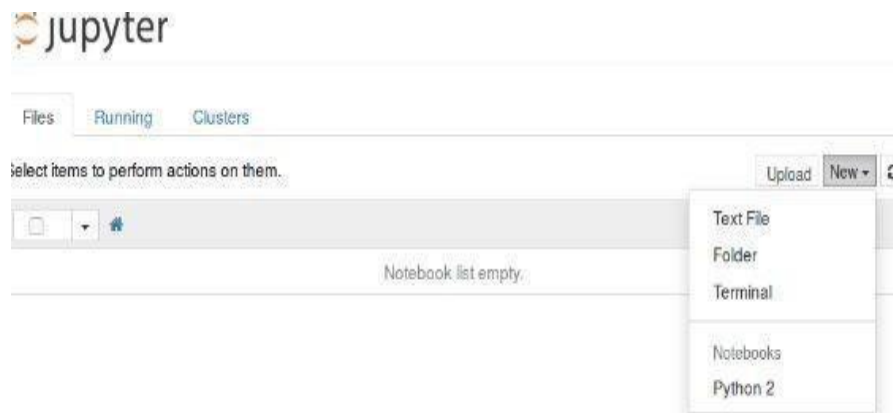vii.   Go to the following link: Anaconda.com/downloads

viii.   Download python 3.4 version for (32-bitgraphic installer/64 -bit graphic installer)

ix.   Select path(i.e. add anaconda to path & register anaconda as default python 3.4)

x.   Click finish

xi.   Open jupyter notebook

Fig 3.2.2.1 After installation

\

Fig 3.2.2.2 jupyter notebook

~ 17 ~

3.3: Features

i. **Readable:** Python is a very readable language.

ii. **Easy to Learn**: Learning python is easy as this is a expressive and high level programming language, which means it is easy to understand the language and thus easy to learn

iii. **Cross platform**: Python is available and can run on various operating systems such as Mac, Windows, Linux, Unix etc. This makes it a cross platform and portable language.

iv. **Open Source**: Python is a open source programming language.

v. **Large standard library**: Python comes with a large standard library that has some handy codes and functions which we can use while writing code in Python.

vi. **Free:** Python is free to download and use. This means you can download it for free and use it in your application. Python is an example of a FLOSS (Free/Libre Open Source Software), which means you can freely distribute copies of this software, read its source code and modify it.

vii. **Supports exception handling**: If you are new, you may wonder what is an exception? An exception is an event that can occur during program exception and can disrupt the normal flow of program. Python supports exception handling which means we can write less error prone code and can test various scenarios that can cause an exception later on.

viii. **Advanced features**: Supports generators and list comprehensions. We will cover these features later.

ix. **Automatic memory management**: Python supports automatic memory management which means the memory is cleared and freed automatically. You do not have to bother clearing the memory.

3.4: 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. Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Python has five standard data types –

- Numbers
- Strings
- Lists
- Tuples
- Dictionary

3.4.1: Python Numbers:

Number data types store numeric values. They are immutable data types, means that changing the value of a number data type results in a newly allocated object.

Python supports four different numerical types −

- **Int (signed integers)** − they are often called just integers or ints, are positive or negative whole numbers with no decimal point.
- **long (long integers)** − Also called longs, they are integers of unlimited size, written like integers and followed by an uppercase or lowercase L.
- **float (floating point real values)** − Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10 (2.5e2 = 2.5 x 102= 250).

3.4.2: Python Strings:

In Python, Strings can be created by simply enclosing characters in quotes. It does not support character types. These are treated as length-one strings, and are also considered as substrings. Substrings are immutable and can't be changed once created. Strings are the ordered blocks of text that are enclosed in single or double quotations. Thus, whatever is written in quotes is considered as string. Though it can be written in single or double quotations, double quotation marks allow the user to extend strings over multiple lines without backslashes, which is usually the signal of continuation of an expression, e.g., 'abc', "ABC".

3.4.3: Python lists:

List is a collection data type in python. It is ordered and allows duplicate entries as well. Lists in python need not be homogeneous, which means it can contain different data types like integers, strings and other collection data types. It is mutable in nature and allows indexing to access the members in a list. To declare a list, we use the square brackets. List is like any other array that we declare in other programming languages. Lists in python are often used to implement stacks and queues. The lists are mutable in nature. Therefore, the values can be changed even after a list is declared.

3.4.4: Python tuples:

A tuple is a collection of objects which ordered and immutable. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets. Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also

3.4.5: Python Dictionary:

It is a collection data type just like a list or a set, but there are certain features that make python dictionary unique. A dictionary in python is not ordered and is changeable as well. We can make changes in a dictionary unlike sets or strings which are immutable in nature. Dictionary contains key-value pairs like a map that we have in other programming languages. A dictionary has indexes.

3.4.6: Functions:

3.4.6.1: Defining a Function:

- Function blocks begin with the keyword def followed by the function name and parentheses (( )).

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

- The first statement of a function can be an optional statement - the documentation string of the function or docstring.

- 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.

3.4.6.2: 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

3.5: OOPs Concepts:

3.5.1 Class:

- Python is an object oriented programming language. Unlike procedure oriented programming, where the main emphasis is on functions, object oriented programming stresses on objects..

- An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object.

- We can think of class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows etc. Based on these descriptions we build the house. House is the object.

- As many houses can be made from a house's blueprint, we can create many objects from a class. An object is also called an instance of a class and the process of creating this object is called instantiation.

- Like function definitions begin with the def keyword in Python, class definitions begin with a class keyword.

- The first string inside the class is called docstring and has a brief description about the class

```
class MyNewClass:
    '''This is a docstring. I have created a new class'''
    pass
```

Fig 3.5.1.1 Class defining

- As soon as we define a class, a new class object is created with the same name. This class object allows us to access the different attributes as well as to instantiate new objects of that class.

```
class Person:
    "This is a person class"
    age = 10

    def greet(self):
        print('Hello')

# Output: 10
print(Person.age)

# Output: <function Person.greet>
print(Person.greet)

# Output: 'This is my second class'
print(Person.__doc__)
```

Fig 3.5.1.2 Example of class

4. MOVIE RECOMENDER SYSTEM

4.1: INTRODUCTION:

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google. Often, these systems are able to collect information about user's choice, and can use this information to improve their suggestions in the future. For example, Face book can monitor your interaction with various stories on your feed in order to learn what types of stories appeal to you. Sometimes, the recommender systems can make improvements based on the activities of a large number of people. For example, if Amazon observes that a large number of customers who buy the latest Apple Mac book also buy a USB-C-to USB Adapter; they can recommend the Adapter to a new user who has just added a Mac book to his cart. Due to the advances in recommender systems, users constantly expect good recommendations. They have a low threshold for services that are not able to make appropriate suggestions. If a music streaming app is not able to predict and play music that the user likes, then the user will simply stop using it. This has led to a high emphasis by tech companies on improving their recommendation systems. However, the problem is more complex than it seems. Every user has different preferences and likes. In addition, even the taste of a single user can vary depending on a large number of factors, such as mood, season, or type of activity the user is doing. For example, the type of music one would like to hear while exercising differs greatly from the type of music he'd listen to when cooking dinner. Another issue that recommendation systems have to solve is the exploration vs. exploitation problem. They must explore new domains to discover more about the user, while still making the most of what is already known about of the user. Three main

approaches are used for our recommender systems. One is Demographic Filtering i.e. they offer generalized recommendations to every user, based on movie popularity and/or 7 genres. The System recommends the same movies to users with similar demographic features. Since each user is different, this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience. Second is content-based filtering, where we try to profile the user's 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.

4.2: PREPROCESSING OF THE DATA:

Preprocessing of the data actually involves the following steps:

4.2.1: GETTING THE DATASET:

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

4.2.2: IMPORTING THE LIBRARIES:

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

4.2.2.1: Packages used:

Numpy: In Python we have lists that serve the purpose of arrays, but they are slow to process. Numpy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in Numpy is called ndata, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

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

Seaborn: Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Matplotlib: 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. Matplotlib is written in Python and makes use of Numpy, the numerical mathematics extension of Python. It provides an object-oriented 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 Math Works. Matplotlib along with Numpy can be considered as theopen source equivalent of MATLAB.

▼ Import Libraries

```
[1] import pandas as pd
    import numpy as np
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.metrics.pairwise import cosine_similarity
    # matplotlib for plotting
    import matplotlib.pyplot as plt
    import seaborn as sns
    # use ggplot style
    plt.style.use('ggplot')
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead
  import pandas.util.testing as tm
```

Fig 4.2.2.1.1 packages

4.2.2.2: Versions of the packages:

The versions of the packages are found by following command

▾ Version of Libraries

```
[3] print("numpy:",np.__version__)
    print("pandas:",pd.__version__)
```

```
numpy: 1.18.5
pandas: 1.0.5
```

Fig 4.2.2.2.1 version

4.3 Algorithms used:

For our project, we focused on two main algorithms for recommendations:

a) Collaborative filtering

b) Content-based filtering.

4.3.1: Collaborative Filtering:

Collaborative Filtering techniques make recommendations for a user based on ratings and preferences data of many users. The main underlying idea is that if two users have both liked certain common items, then the items that one user has liked that the other user has not yet tried can be recommended to him. We see collaborative filtering techniques in action on various Internet platforms such as Amazon.com, Netflix, and Face book. We are recommended items based on the ratings and purchase data that these platforms collect from their user base. We explore two algorithms for Collaborative filtering:

Nearest Neighbors Algorithm and the Latent Factors Algorithm.

4.3.1.1: Nearest Neighbors Collaborative Filtering:

This approach relies on the idea that users, who have similar rating behaviors so far, share the same tastes and will likely exhibit similar rating behaviors going forward. The algorithm first computes the similarity between users by using the row vector in the ratings matrix corresponding to a user as a representation for that user. The similarity is computed by using either cosine similarity or Pearson Correlation. In order to predict the rating for a particular user for a given movie j, we find the top k similar users to this particular user and then take a weighted average of the ratings of the k similar users with the weights being the similarity values.

4.3.1.2: Latent Factor Methods:

The latent factor algorithm looks to decompose the ratings matrix R into two tall and thin matrices Q and P, with matrix Q having dimensions num_users $\times$ k and P having the dimensions numitems $\times$ k where k is the number of latent factors. The decomposition of R into Q and P is such that R = Q.P T. Any rating rij in the ratings matrix can be computed by taking the dot product of row qi of matrix Q and pj of matrix P. The matrices Q and P are initialized randomly or by performing SVD on the ratings matrix. Then, the algorithm solves the problem of minimizing the error between the actual rating value rij and the value given by taking the dot product of rows qi and pj . The algorithm performs stochastic gradient descent to find the matrices Q and P with minimum error starting from the initial matrices.

\

.

4.4: Approach:

There are various types of recommender systems with different approaches and some of them are classified as below:

Demographic Filtering-

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. Since each user is different, this approach is considered to be too simple. The basic idea behind this system is that movies that are more popular and critically acclaimed will have a higher probability of being liked by the average audience.

Content-based Filtering Systems:

In content-based filtering, items are recommended based on comparisons between item profile and user profile. A user profile is content that is found to be relevant to the user in form of keywords (or features). A user profile might be seen as a set of assigned keywords (terms, features) collected by algorithm from items found relevant (or interesting) by the user. A set of keywords (or features) of an item is the Item profile. For example, consider a scenario in which a person goes to buy his favorite cake 'X' to a pastry. Unfortunately, cake 'X' has been sold out and as a result of this the shopkeeper recommends the person to buy cake 'Y' which is made up of ingredients similar to cake 'X'.

Collaborative filtering based systems:

Our content based engine suffers from some severe limitations. It is only capable of suggesting movies which are close to a certain movie. That is, it is not capable of capturing tastes and providing recommendations across genres. Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that movie, regardless of who she/he is.

In this section, we will use a technique called Collaborative Filtering to make

recommendations to Movie Watchers. It is basically of two types:-

a) **User based filtering**- These systems recommend products to a user that similar users have liked. For measuring the similarity between two users we can either use Pearson correlation or cosine similarity.

b) **Item Based Collaborative Filtering** - Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity. The major difference is that, with item-based collaborative filtering, we fill in the blank vertically, as oppose to the horizontal manner that user-based CF does

4.4.1: Advantages of collaborative filtering based systems:

a) It is dependent on the relation between users which implies that it is content-independent.CF recommender systems can suggest serendipitous items by observing similar-minded people's behavior.

b) They can make real quality assessment of items by considering other peoples experience. It is dependent on the relation between users which implies that it is content-independent.

c) CF recommender systems can suggest serendipitous items by observing similar-minded people's behavior.They can make real quality assessment of items by considering other peoples experience
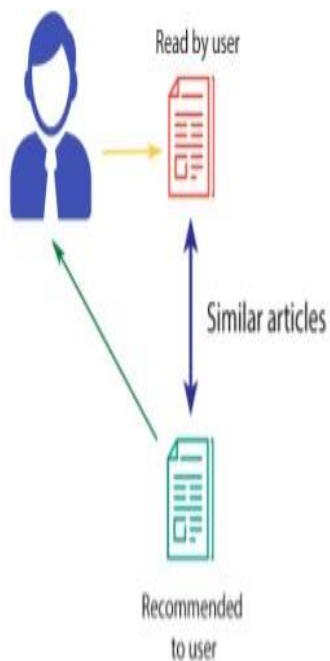
4.4.2: Disadvantages of collaborative filtering are:

a) **Early rater problem**: Collaborative filtering systems cannot provide recommendations for new items since there are no user ratings on which to base a prediction.

b) **Gray sheep**: In order for CF based system to work, group with similar characteristics are needed. Even if such groups exist, it will be very difficult to recommend users who do not consistently agree or disagree to these groups.

c) **Sparsity problem**: In most cases, the amount of items exceed the number of users by a great margin which makes it difficult to find items that are rated by enough
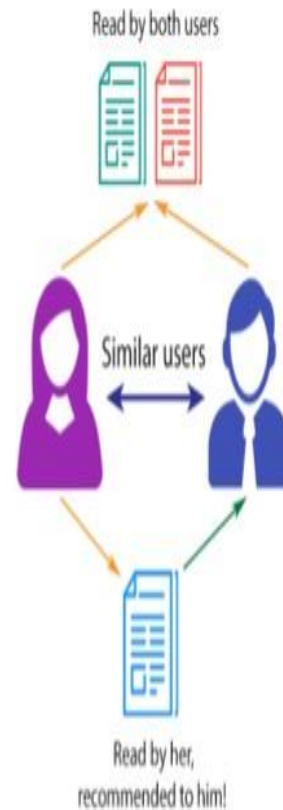
people.

Fig:4.4.1 Implementing a recommender system

4.5: Problem Statement:

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google. Often, these systems are able to collect information about user's choices, and can use this information to improve their suggestions in the future. For example, Face book can monitor your interaction with various stories on your feed in order to learn what types of stories appeal to you. Sometimes, the recommender systems can make improvements based on the activities of a large number of people. For example, if Amazon observes that a large number of customers who buy the latest Apple Macbook also buy a USB-C-toUSB Adapter, they can recommend the Adapter to a new user who has just added a Macbook to his cart. Due to the advances in recommender systems, users constantly expect good recommendations. They have a low threshold for services that are not able to make appropriate suggestions. If a music streaming app is not able to predict and play music that the user likes, then the user will simply stop using it. This has led to a high emphasis by tech companies on improving their recommendation systems. However, the problem is more complex than it seems. Every user has different preferences and likes. In addition, even the taste of a single user can vary depending on a large number of factors, such as mood, season, or type of activity the user is doing. For example, the type of music one would like to hear while exercising differs greatly from the type of music he'd listen to when cooking dinner. Another issue that recommendation systems haveto solve is the exploration vs exploitation problem. They must explore new domains to discover more about the user, while still making the most of what is already known about of the user. Two main approaches are widely used for recommender systems. One is content-based filtering, where we try to profile the user's 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

4.6: Dataset:

It is a multivariate data set containing:

- Index
- Budget
- Genres
- Homepage
- Id
- Keywords
- Original_Language
- Original_Title
- Overview
- Popularity
- Production_Companies
- Production_Countries
- Release_Date
- Revenue
- Runtime
- Spoken_languages
- Status
- Tagline
- Title
- Vote_Average
- Vote_Count
- Cast
- Crew
- Director

4.7: IMPORTING THE DATA-SET:

Pandas in python provide an interesting method read_csv (). The read_csv function reads the entire dataset 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 has to be cleaned.

Fig 4.7.1 loading dataset

4.8: Objectives of the Case Study:

- Objective of problem to recommend movies to users based on user-movie (item) ratings.
- 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 and critically acclaimed will have a higher probability of being liked by the average audience. Second is content-based filtering, where we try to profile the user's 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.

5. DATA PREPROCESSING/FEATURE ENGINEERING

5.1: Statistical Analysis:

Pandas in python provide an interesting method read_csv (). The read_csv function reads the entire dataset 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 has to be cleaned.

Descriptive statistics include those that summarize the central tendency, dispersion and shape of a dataset's distribution, excluding nan values. Analyzes numeric and object series, as well as Data Frame column sets of mixed data types. The output will vary depending on what is provided.

For numeric data, the result's index will include count, mean, std, min, max as well as lower, 50 and upper percentiles. By default, the lower percentile is 25 and the upper percentile is 75. The 50 percentile is the same as the median.

For object data (e.g. strings or timestamps), the result's index will include count, unique, top, and freq. The top is the most common value. The freq is the most common value's frequency. Timestamps also include the first and last items.

If multiple object values have the highest count, then the count and top results will be arbitrarily chosen from among those with the highest count.

For mixed data types provided via a Data Frame, the default is to return only an analysis of numeric columns. If the data frame consists only of object and categorical data without any numeric columns, the default is to return an analysis of both the object and categorical columns. If include='all' is provided as an option, the result will include a union of attributes of each type.

Fig:5.1.1 Data Description

5.2: Data Type Conversions:

When doing data analysis, it is important to make sure you are using the correct data types; otherwise you may get unexpected results or errors. In the case of pandas, it will correctly infer data types in many cases and you can move on with your analysis without any further thought on the topic. Despite how well pandas work, at some point in your data analysis processes, you will likely need to explicitly convert data from one type to another. This article will discuss the basic pandas data types (aka dtypes), how they map to python and Numpy data types and the options for converting from one pandas type to another.

Fig 5.2.1 datatypes

5.3: Identifying and Handling missing values:

There are a number of schemes that have been developed to indicate the presence of missing data in a table or Data Frame. Generally, they revolve around one of two strategies: using a mask that globally indicates missing values, or choosing a sentinel value that indicates a missing entry. In the masking approach, the mask might be an entirely separate Boolean array, or it may involve appropriation of one bit in the data representation to locally indicate the null status of a value. In the sentinel approach, the sentinel value could be some data-specific convention, such as indicating a missing integer value with -9999 or some rare bit pattern, or it could be a more global convention, such as indicating a missing floating-point value with NaN (Not a Number), a special value which is part of the IEEE floating-point specification.

Identifying and handling the missing values

```
[39] df.isnull().sum()
```

```
index                    0
budget                   0
genres                  28
homepage              3091
id                       0
keywords               412
original_language        0
original_title           0
overview                 3
popularity               0
production_companies     0
production_countries     0
release_date             1
revenue                  0
runtime                  2
spoken_languages         0
status                   0
tagline                844
title                    0
vote_average             0
vote_count               0
cast                    43
crew                     0
director                30
dtype: int64
```

Fig 5.3.1: Identifying and Handling Missing Values

```
# Removing rows which are having null values
df1 = df.dropna()
df1.isnull().sum()
```

```
index                   0
budget                  0
genres                  0
homepage                0
id                      0
keywords                0
original_language       0
original_title          0
overview                0
popularity              0
production_companies    0
production_countries    0
release_date            0
revenue                 0
runtime                 0
spoken_languages        0
status                  0
tagline                 0
title                   0
vote_average            0
vote_count              0
cast                    0
crew                    0
director                0
dtype: int64
```

Fig 5.3.2: Removing null values from rows

5.4: Encoding Categorical Data:

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 color

● **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.

In the given dataset I have not used any encoding because dataset is numerical. We are going to build a recommender based on the following metadata: actors, the director, related genres and the movie plot keywords. From the cast, crew and keywords features, we need to extract the three most important actors, the director and the keywords associated with that movie

Fig 5.4.1 Extracting Features

5.5: Generating Plots:

5.5.1: Visualize the data between Target and the Features:

1. Correlation:

```
[ ] df1.corr()
```

	index	budget	id	popularity	revenue	runtime	vote_average	vote_count
index	1.000000	-0.770992	0.082707	-0.372255	-0.547246	-0.363512	-0.028606	-0.468143
budget	-0.770992	1.000000	-0.039062	0.445551	0.741173	0.363574	0.063392	0.597986
id	0.082707	-0.039062	1.000000	0.117441	-0.029608	-0.058912	-0.132695	0.028322
popularity	-0.372255	0.445551	0.117441	1.000000	0.587567	0.289795	0.297738	0.715044
revenue	-0.547246	0.741173	-0.029608	0.587567	1.000000	0.379620	0.247318	0.780843
runtime	-0.363512	0.363574	-0.058912	0.289795	0.379620	1.000000	0.380913	0.442273
vote_average	-0.028606	0.063392	-0.132695	0.297738	0.247318	0.380913	1.000000	0.431465
vote_count	-0.468143	0.597986	0.028322	0.715044	0.780843	0.442273	0.431465	1.000000

Fig 5.5.1.1 Correlation

2.Correlation graph:

```
fig = plt.subplots (figsize = (12, 12))
sns.heatmap(df1.corr (), square = True, cbar = True, annot = True, cmap="GnBu", annot_kws = {'size': 8})
plt.title('Correlations between Attributes')
plt.show ()
```
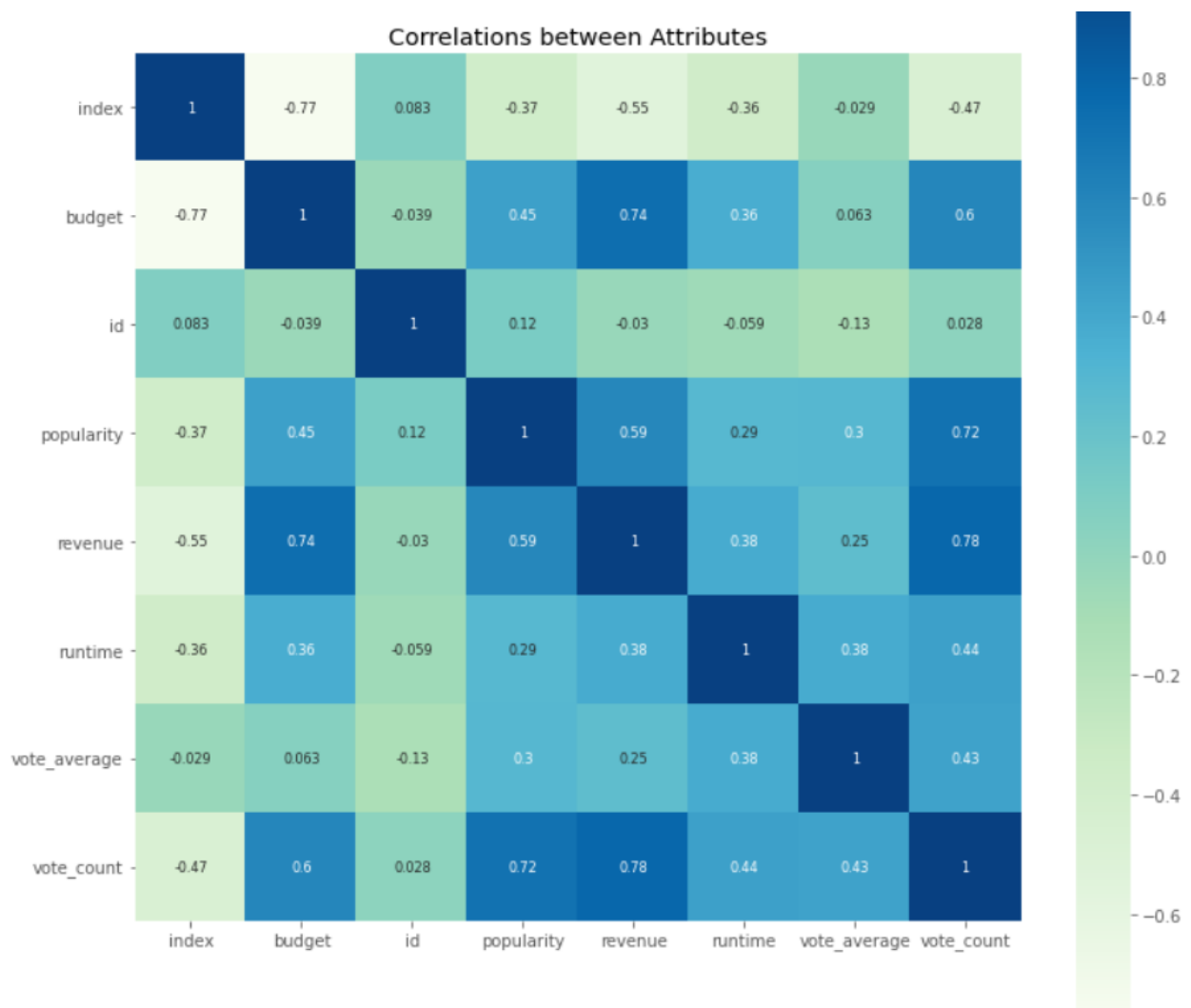

Fig 5.5.1.2 Correlation Graph

~ 41 ~

3.Genres Count:

```
threshold = 10
genres_value_counts= df1.genres.value_counts()
```

```
# genres value count above 10
to_remove = genres_value_counts[genres_value_counts<= threshold].index
df1['genres'].replace(to_remove, np.nan, inplace=True)
genres_value_counts = genres_value_counts[::-1]
genres_value_counts
```

```
/usr/local/lib/python3.6/dist-packages/pandas/core/generic.py:6746: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._update_inplace(new_data)
Action Thriller Crime              11
Animation Family                   11
Drama Comedy Romance               11
Adventure Action Science Fiction   11
Drama Crime                        12
Romance Comedy                     12
Comedy Family                      14
Documentary                        14
Crime Drama Thriller               15
Horror                             19
Drama Thriller                     23
Adventure Action Thriller          23
Horror Thriller                    27
Comedy Drama Romance               27
Comedy Drama                       31
Comedy Romance                     36
Drama Romance                      45
Comedy                             71
Drama                              89
Name: genres, dtype: int64
```

```
# genres value count above 10
genres_value_counts.plot.barh(figsize=(15, 15))
# add a suptitle
plt.title("Movie genresCount", fontsize=24)
# set xlabel to ""
plt.xlabel("Popularity", fontsize=20)
plt.ylabel("genres",fontsize=20)
# change xticks fontsize to 14
plt.xticks(fontsize=18)
plt.yticks(fontsize=16)
# finally show the plot
plt.show()
```
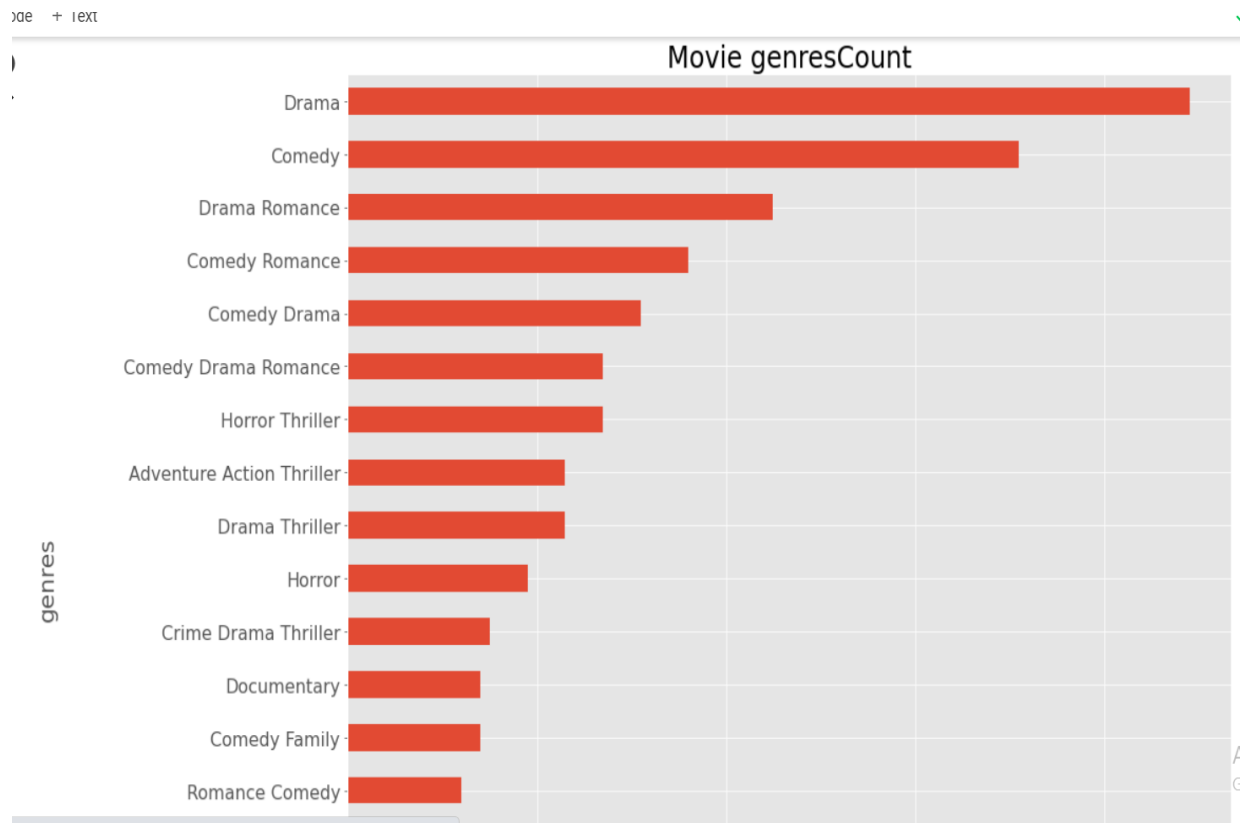
Fig 5.5.1.3 Movie Genre Count Graph

Here, we can see:

- Drama genres has more Movies
- Drama/comedy/Romance has less Movies

4. Director Movies:

```
[81] # director value count above 5
     threshold = 5
     director_value_counts = df1.director.value_counts()
     to_remove = director_value_counts[director_value_counts<= threshold].index
     df1['director'].replace(to_remove, np.nan, inplace=True)
     director_value_counts = director_value_counts[::-1]
```

```
/usr/local/lib/python3.6/dist-packages/pandas/core/generic.py:6746: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._update_inplace(new_data)
```

```
[82] # director value count above 5
     director_value_counts.plot.barh(figsize=(15, 15))
     # add a suptitle
     plt.title("director's movieCount", fontsize=24)
     # set xlabel to "count"
     plt.xlabel("Popularity", fontsize=20)
     plt.ylabel("director",fontsize=20)
     # change xticks fontsize to 14
     plt.xticks(fontsize=18)
     plt.yticks(fontsize=16)
     # finally show the plot
     plt.show()
```
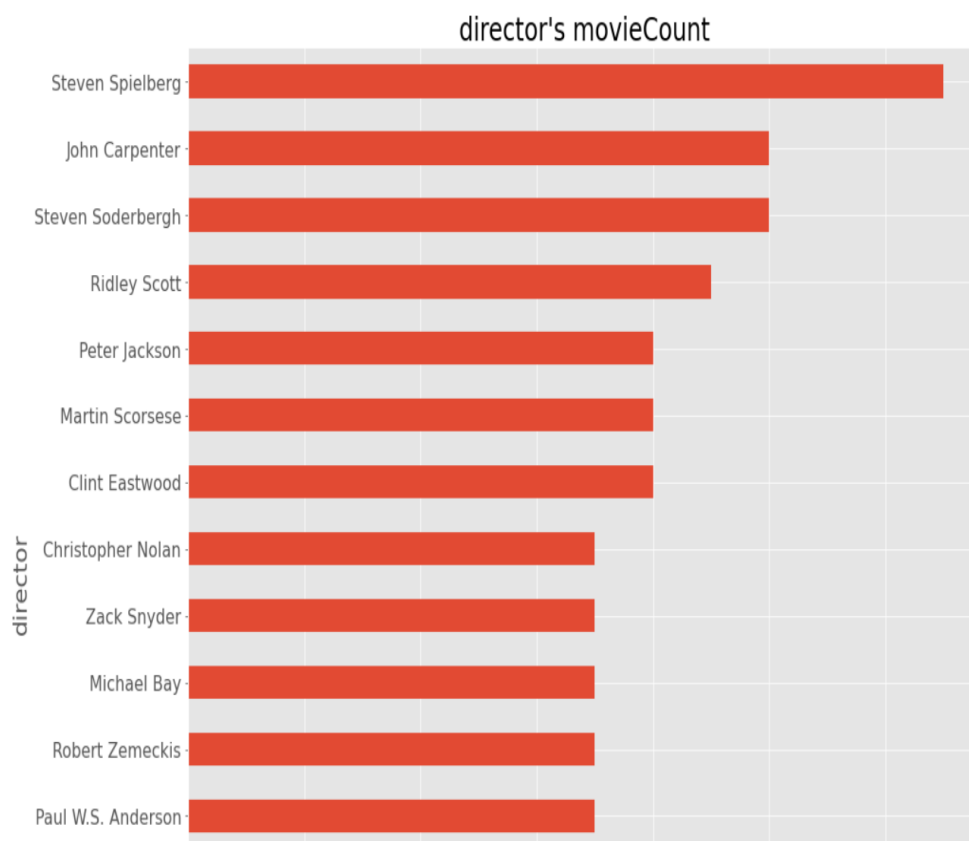

Fig 5.5.3.4 Directors Movie Count Graph

Observation-

Steven Spielberg director more movies in this data set.

~ 44 ~

5. Keywords

```
# keywords value count

keywords_value_counts= df1.keywords.value_counts()
keywords_value_counts = keywords_value_counts[::-1]
keywords_value_counts
```

```
skyscraper thanksgiving billionaire parade fbi agent        1
saving the world artificial intelligence rebel cyborg shotgun  1
ohio politics dirty tricks presidential campaign endorsement   1
based on novel job interview bad luck wish one day          1
riot protest musical music cultural difference             1
                                                          ..
independent film woman director                            2
based on novel revolution dystopia sequel dystopic future  2
independent film                                           4
duringcreditsstinger                                       7
woman director                                             8
Name: keywords, Length: 1412, dtype: int64
```

```
[100] # director value count above 5
      keywords_value_counts.plot.barh(figsize=(15, 15))
      # add a suptitle
      plt.title("keywords", fontsize=24)
      # set xlabel to "count"
      plt.xlabel("Popularity", fontsize=20)
      plt.ylabel("Keywords",fontsize=20)
      # change xticks fontsize to 14
      plt.xticks(fontsize=18)
      plt.yticks(fontsize=16)
```


Fig 5.5.1.5 Keyword Graph

6. Cast:

```
[85] # cast value count
     cast_value_counts= df1.cast.value_counts()
     cast_value_counts = cast_value_counts[::-1]
     cast_value_counts
```

```
⊟    Shannyn Sossamon Edward Burns Ana Claudia Talanc\u00f3n Ray Wise Azura Skye        1
     Leonardo DiCaprio Tom Hanks Christopher Walken Martin Sheen Nathalie Baye          1
     Meryl Streep Emily Blunt James Corden Anna Kendrick Chris Pine                     1
     Jason Bateman Mila Kunis Kristen Wiig Ben Affleck J.K. Simmons                     1
     Ryan Gosling Carey Mulligan Bryan Cranston Albert Brooks Oscar Isaac              1
                                                                                        ..
     Mike Myers Eddie Murphy Cameron Diaz Julie Andrews Antonio Banderas               2
     Ewan McGregor Natalie Portman Hayden Christensen Ian McDiarmid Samuel L. Jackson   2
     Mark Hamill Harrison Ford Carrie Fisher Billy Dee Williams Anthony Daniels         2
     Martin Freeman Ian McKellen Richard Armitage Ken Stott Graham McTavish             2
     Jennifer Lawrence Josh Hutcherson Liam Hemsworth Woody Harrelson Elizabeth Banks   3
     Name: cast, Length: 1425, dtype: int64
```
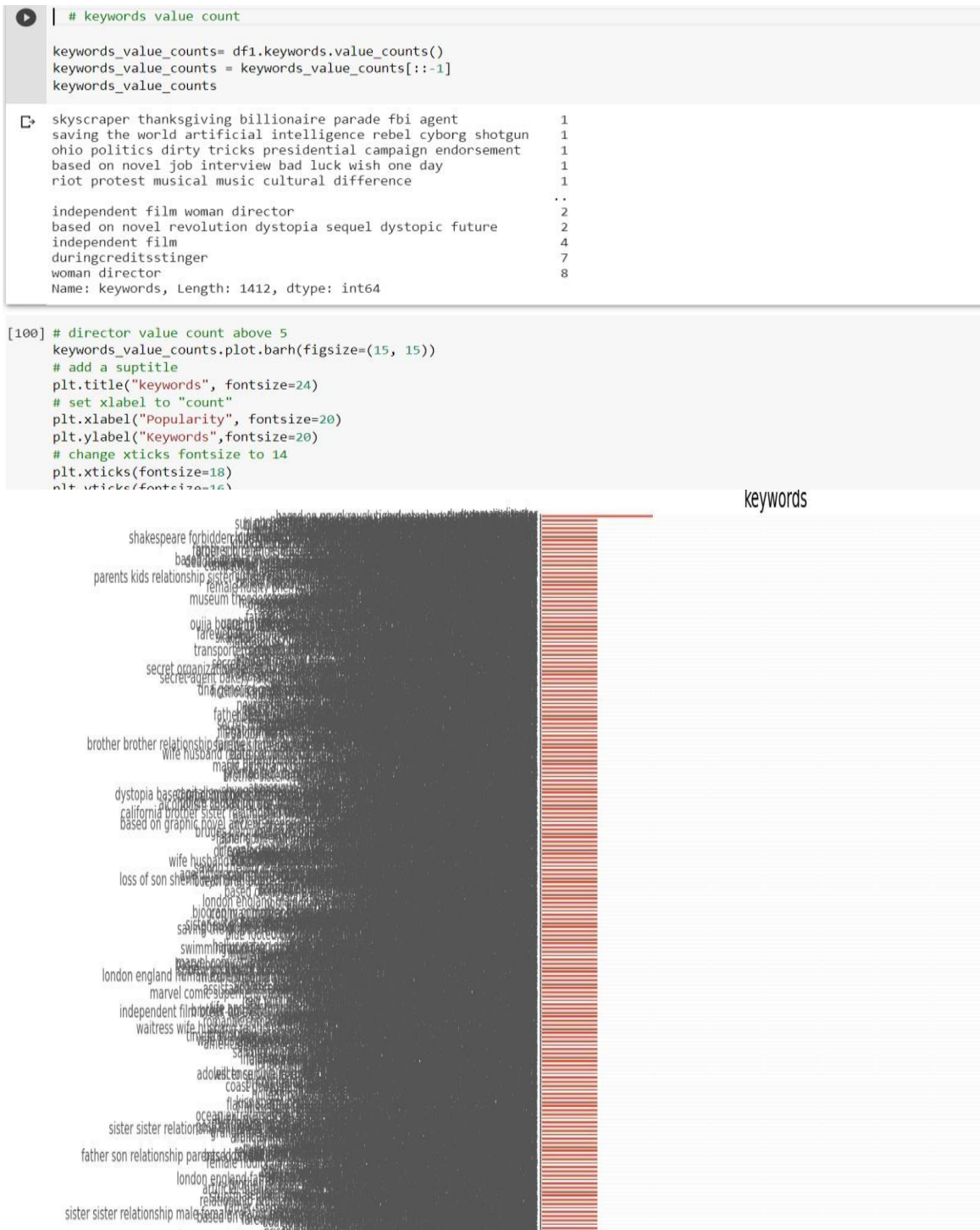
```
cv = CountVectorizer() #creating new CountVectorizer() object
count_matrix = cv.fit_transform(df1["combined_features"]) #feeding combined strings(movie contents) to CountVectorizer() object
print(count_matrix)
```

```
  (0, 1531)    1
  (0, 1274)    1
  (0, 2437)    1
  (0, 6083)    2
  (0, 6987)    1
  (0, 1339)    1
  (0, 6037)    1
  (0, 5657)    1
  (0, 7170)    1
  (0, 7278)    1
  (0, 5642)    1
  (0, 5954)    1
  (0, 7016)    1
  (0, 6175)    1
  (0, 3728)    1
  (0, 4355)    1
  (0, 5510)    1
  (0, 37)      1
  (0, 64)      1
  (0, 2197)    1
  (0, 5753)    1
  (0, 2263)    1
  (0, 3273)    1
  (0, 988)     1
  (1, 37)      1
  :    :
  (1430, 4909) 1
  (1430, 3661) 1
  (1430, 6383) 1
  (1430, 3508) 1
  (1431, 5753) 1
  (1431, 2263) 1
  (1431, 3121) 1
  (1431, 1871) 1
```

5.6: Detection of Outliers:

5.6.1 Outlier for genres:

```
plt.hist(genres_value_counts)
plt.show()
```
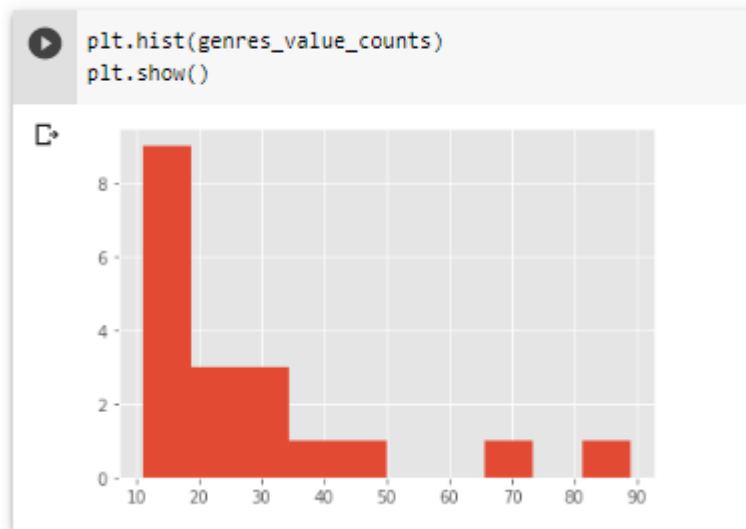

Fig 5.6.1.1 genres outliner

5.6.2 Outlier for directors:

```
plt.hist(director_value_counts)
plt.show()
```
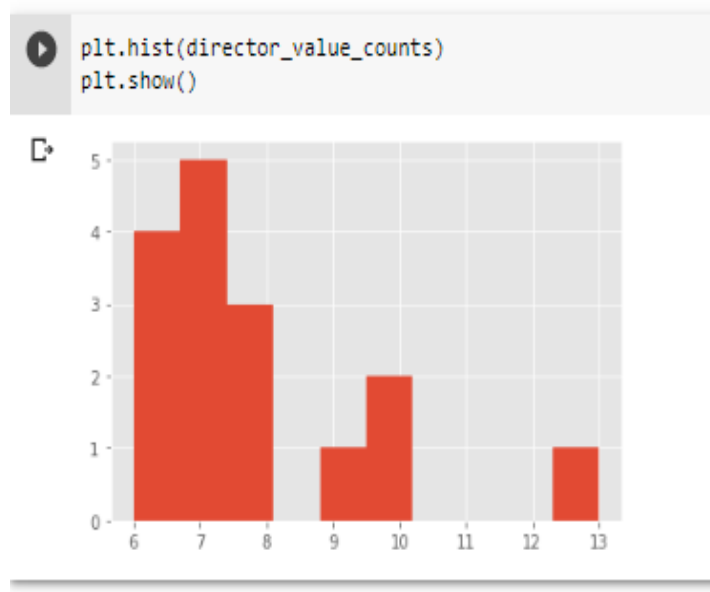

Fig 5.6.2.1 Directors outliner

~ 47 ~

5.6.3 Outlier for keyword:


```
plt.hist(keywords_value_counts)
plt.show()
```
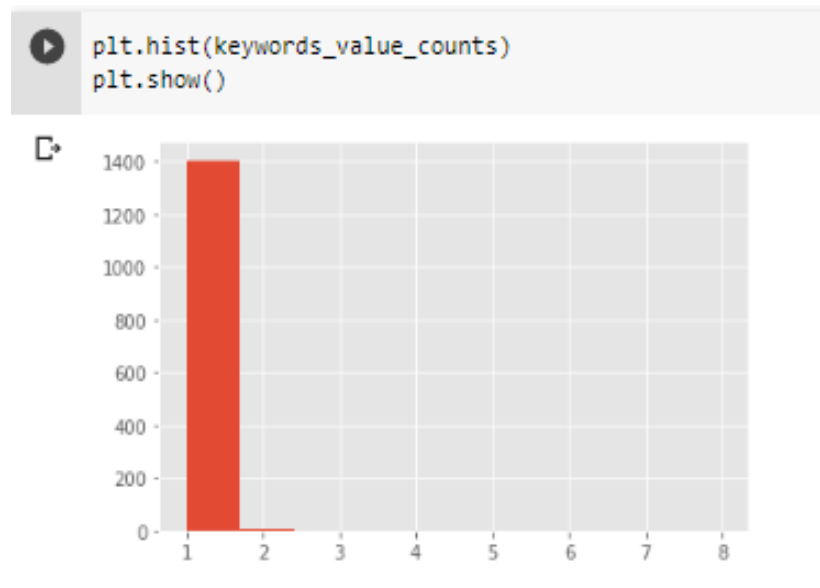
Fig 5.6.3.1 Keyword Outliner

5.6.4 Outlier for cast:


```
plt.hist(cast_value_counts)
plt.show()
```

Fig 5.6.4.1 Cast Outliner

6. FEATURE SELECTION

We also implemented an additional feature that provides a distributed indexer on the movie synopsis as extracted from the movie website to allow searching for movies based on keyword searches. It is built on a Map Reduce framework to build an inverted index. The framework for the indexer is the same as the one developed earlier in the semester for the assignments. It has been modified to allow searching on the movie information dataset. First, the dataset is generated by scrapping data from the movie pages of the respective movies. The movie id is provided along with the movie lens dataset. After accessing the webpage, the metadata from the movie is collected and then pickled and stored in a dump. This same dump can also be use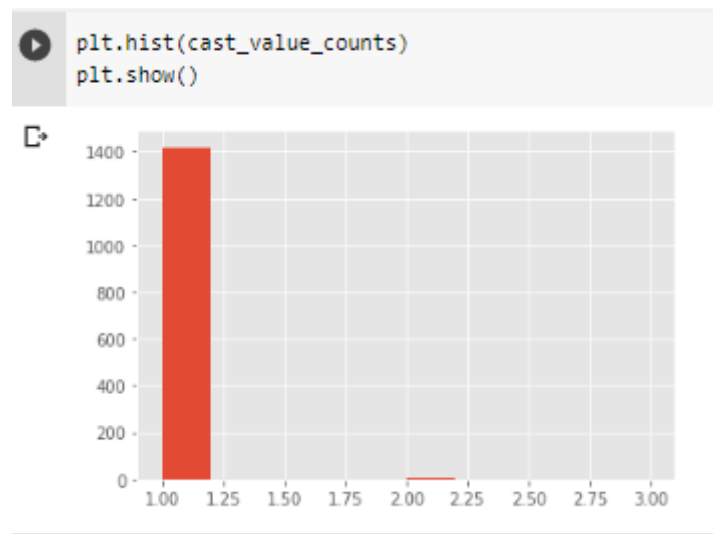d to implement and extend the content based model. In the current model, the synopsis refers to the first of the provided plot summaries. The summaries tend to be much shorter than the synopsis and can be extracted much more easily. Also, the size of the dump increased by a factor of almost 100 when using synopses instead of plot summaries. One more reason to select plot summaries is that detailed synopsis is not available for a large portion of the movies, which would result in a bias in the search. The rest of the search is similar to the one provided in the assignments. The reformatter has been adjusted to match the format of the dump. The rest of the search can be run in the identical manner as the assignment. More detailed instructions are provided in a readme file.

6.1: Select relevant features for the analysis:

If you visualize the dataset, you will see that it has many extra info about a movie. We don't need all of them. So, we choose keywords, cast, genres and director column to use as our feature set(the so called "content" of the movie).

```
[ ]  features = ['keywords','cast','genres','director']
```

6.2: Drop irrelevant features

6.2.1: Drop Manually:

Our next task is to create a function for combining the values of these columns into a single string.

```python
def combine_features(row):
    return row['keywords']+" "+row['cast']+" "+row['genres']+" "+row['director']
```

Now, we need to call this function over each row of our data frame. But, before doing that, we need to clean and preprocess the data for our use. We will fill all the NANvalues with blank string in the data frame.

```python
for feature in features:
    df[feature] = df[feature].fillna('') #filling all NaNs with blank
string
df["combined_features"] = df.apply(combine_features,axis=1) #applying
combined_features() method over each rows of dataframe and storing
the combined string in "combined features" column
```

7. MODELING:

In order to detect similarities between movies, I need to vectorize, as I mentioned above. I decided to use CountVectorizer rather than TfIdfVectorizer for one simple reason: I need a simple frequency counter for each word in my bag_of_words column. Tf-Idf tends to give less importance to the words that are more present in the entire corpus (our whole column, in this case) which is not what we want for this application, because every word is important to detect similarity! Once I have the matrix containing the count for each word, we can apply the cosine_similarity function

Now that we have obtained the combined strings, we can now feed these strings to a CountVectorizer() object for getting the count matrix.

```
[47] cv = CountVectorizer() #creating new CountVectorizer() object
     count_matrix = cv.fit_transform(df1["combined_features"]) #feeding combined strings(movie contents) to CountVectorizer() object
     print(count_matrix)

       (0, 1531)    1
       (0, 1274)    1
       (0, 2437)    1
       (0, 6083)    2
       (0, 6987)    1
       (0, 1339)    1
       (0, 6037)    1
       (0, 5657)    1
       (0, 7170)    1
       (0, 7278)    1
       (0, 5642)    1
       (0, 5954)    1
       (0, 7016)    1
       (0, 6175)    1
       (0, 3728)    1
       (0, 4355)    1
       (0, 5510)    1
       (0, 37)      1
       (0, 64)      1
       (0, 2197)    1
       (0, 5753)    1
       (0, 2263)    1
       (0, 3273)    1
       (0, 988)     1
       (1, 37)      1
       :      :
       (1430, 4909) 1
       (1430, 3661) 1
       (1430, 6383) 1
```

At this point, 60% work is done. Now, we need to obtain the cosine similarity matrix from the count matrix.

```
cosine_sim = cosine_similarity(count_matrix)
print(cosine_sim)

[[1.         0.10540926 0.12038585 ... 0.         0.         0.07027284]
 [0.10540926 1.         0.0761387  ... 0.         0.         0.        ]
 [0.12038585 0.0761387  1.         ... 0.         0.         0.        ]
 ...
 [0.         0.         0.         ... 1.         0.         0.        ]
 [0.         0.         0.         ... 0.         1.         0.        ]
 [0.07027284 0.         0.         ... 0.         0.         1.        ]]
```

The similarity matrix looks like this.

$$
\begin{array}{c|ccccc}
 & Movie_1 & Movie_2 & Movie_3 & \cdots & Movie_n \\
\hline
Movie_1 & 1 & 0.158 & 0.138 & \ldots & 0.056 \\
Movie_2 & 0.158 & 1 & 0.367 & \ldots & 0.056 \\
Movie_3 & 0.138 & 0.367 & 1 & \ldots & 0.049 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
Movie_n & 0.056 & 0.056 & 0.049 & \ldots & 1
\end{array}
$$

Similarity matrix

Let's have a brief look at it: all the numbers on the diagonal are 1 because, of course, every movie is identical to itself. The matrix is also symmetrical because the similarity between A and B is the same as the similarity between B and A.

At this point, I can write the actual function that takes a movie title as input, and returns the top 10 similar movies. In order to do this, I also created a simple series of movie titles with numerical indexes, in order to match the indexes from the similarity matrix to the actual movie titles. My function, in fact, once receives the input, detects the 10 highest numbers within the row corresponding to the movie entered, gets the correspondent indexes and matches them to the movie titles series, to return the list of recommended movies. When I say that the function considers the 10 highest similarity values, I am discarding the unit value (which is easily the highest), so that the function does not return the same movie title I entered.

· Sorting the list of similar_movies according to similarity scores in descending order.

```
[90] sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)[1:]
```

Now, we will define two helper functions to get movie title from movie index and vice-versa.

```
def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]
def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]
```

Our next step is to get the title of the movie that the user currently likes. Then we will find the index of that movie. After that, we will access the row corresponding to this movie in the similarity matrix. Thus, we will get the similarity scores of all other movies from the current movie. Then we will enumerate through all the similarity scores of that movie to make a tuple of movie index and similarity score. This will convert a row of similarity scores like this-
`[1 0.5 0.2 0.9]` to this- `[(0, 1) (1, 0.5) (2, 0.2) (3, 0.9)]`.
Here, each item is in this form- (movie index, similarity score).

```python
movie_user_likes = "Avatar"
movie_index = get_index_from_title(movie_user_likes)
similar_movies = list(enumerate(cosine_sim[movie_index])) #accessing the row corresponding to given movie to find all the similarity scores for that movie and then enumerating over it
```

Now comes the most vital point. We will sort the list similar movies according to similarity scores in descending order. Since the most similar movie to a given movie will be itself, we will discard the first element after sorting the movies.

```python
sorted_similar_movies = sorted(similar_movies,key=lambda x:x[1],reverse=True)[1:]
```

Now, we will run a loop to print first 5 entries from sorted_similar_movies list.

```python
i=0
print("Top 5 similar movies to "+movie_user_likes+" are:\n")
for element in sorted_similar_movies:
    print(get_title_from_index(element[0]))
    i=i+1
    if i>5:
        break
```
```
Top 5 similar movies to Avatar are:

Edge of Tomorrow
The Mechanic
Superman II
Austin Powers in Goldmember
Happy Feet Two
Pacific Rim
```

So, the whole combined code of our movie recommendation engine is:

```python
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity


df = pd.read_csv("movie_dataset.csv")

features = ['keywords','cast','genres','director']

def combine_features(row):
    return row['keywords'] +" "+row['cast']+" "+row["genres"]+"
"+row["director"]

for feature in features:
    df[feature] = df[feature].fillna('')

df["combined_features"] = df.apply(combine_features,axis=1)

cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])

cosine_sim = cosine_similarity(count_matrix)

def get_title_from_index(index):
    return df[df.index == index]["title"].values[0]

def get_index_from_title(title):
    return df[df.title == title]["index"].values[0]

movie_user_likes = "Avatar"
movie_index = get_index_from_title(movie_user_likes)
similar_movies =  list(enumerate(cosine_sim[movie_index]))

sorted_similar_movies = sorted(similar_movies,key=lambda
x:x[1],reverse=True)[1:]

i=0
print("Top 5 similar movies to "+movie_user_likes+" are:\n")
for element in sorted_similar_movies:
    print(get_title_from_index(element[0]))
    i=i+1
    if i>=5:
        break
```

8. Conclusion

A hybrid approach is taken between context based filtering and collaborative filtering to implement the system. This approach overcomes drawbacks of each individual algorithm and improves the performance of the system. Techniques like Clustering, Similarity and Classification are used to get better recommendations thus reducing MAE and increasing precision and accuracy. In future we can work on hybrid recommender using clustering and similarity for better performance. Our approach can be further extended to other domains to recommend songs, video, venue, news, books, tourism and e-commerce sites, etc..

Now, it's time to run our code and see the output. If you run the above code, you will see this output-

```
Top 5 similar movies to Avatar are:


Guardians of the Galaxy
Aliens
Star Wars: Clone Wars: Volume 1
Star Trek Into Darkness
Star Trek Beyond
```

After seeing the output, I went one step further to compare it to other recommendation engines.

9.REFERENCES

- https://medium.com/code-heroku/building-a-movie-recommendation-engine-in-python-using-scikit-learn- c7489d7cb145

- https://www.kaggle.com/rounakbanik/movie-recommender-systems

- https://www.geeksforgeeks.org/python-implementation-of-movie-recommender-system/

- https://data-flair.training/blogs/data-science-r-movie-recommendation/

- https://github.com/dinesh-sathunuri/AIML_project