Python Programming(15IT322E)

Movie Recommendation System

Submitted By:
Yash Goswami - RA1711003030227
Rishabh Goel - RA1711003030246

Submitted To:
Dr. Manikandan Rajagopal
Assistant Professor
Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
(SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, NCR CAMPUS)
MODINAGAR, GHAZIABAD - 201204

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY, NCR CAMPUS
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Bonafide Certificate

It is to be certified that the bonafide record submitted by Yash Goswami , Rishabh Goel having registration number RA1711003030227 and RA1711003030246 respectively, of the Sixth Semester (Third Year) for Bachelor of Technology in Computer Science and Engineering Degree in the Department of Computer Science and Engineering, SRM Institute of Science and Technology has been done for the course Python Programming (15IT322E) during the academic year 2019 - 2020.

Dr. Manikandan Rajagopal

Submitted for the Python Programming Project Assessment/Examination held on 30-04-2020.

HEAD OF DEPARTMENT
(Dr. R.P. Mahapatra)
(Head Of Department)

FACULTY IN-CHARGE
(Dr. Manikandan Rajagopal)
(Assistant Professor)

ACKNOWLEDGEMENT

With a deep sense of gratitude, we wish to express my sincere thanks to my guides, Prof. Dr. Manikandan Rajagopal, C.S.E. Department for giving us the opportunity to work under them on the project.

We truly appreciate and value their esteemed guidance and encouragement from the beginning to end of this project. We are extremely grateful to him. We want to thank all my teachers for providing a solid background for our studies and research thereafter. They have been a great source of inspiration to us and we thank them from the bottom of our hearts.

We also want to thank our parents, who taught us the value of hard work by their own example. We would like to share this moment of happiness with our parents. Finally, we would like to thank our department for giving us the opportunity and platform to make our effort a successful one.

…………………...
YASH GOSWAMI
RA1711003030227

RISHABH GOEL
RA1711003030246

CSE(B. Tech)
Department of Computer Science
Srm Institute of Science and Technology

DECLARATION

I hereby declare that the project work entitled “Movie Recommendation System” submitted, is a record of an original work done by us under the guidance of Dr. Manikandan Rajagopal, Assistant Prof. Dept Of Computer Science & Engineering, S.R.M Institute of Science & Technology, and this project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering. The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

…………………...
YASH GOSWAMI
RA1711003030227

RISHABH GOEL
RA1711003030246

CSE (B. Tech)
Department of Computer Science
SRM Institute of Science and Technology

ABSTRACT

A movie recommendation is important in our social life due to its strength in providing enhanced entertainment. Such a system can suggest a set of movies to users based on their interest, or the popularities of the movies. Although a set of movie recommendation systems have been proposed, most of these either cannot recommend a movie to the user efficiently. It mines movie databases to collect all the important information,such as, popularity and attractiveness, required for recommendation.

In this Project “Movie Recommendation System” the accurate movie recommendations are provided by the system efficiently and effectively.

TABLE OF CONTENTS

INTRODUCTION

The project on “Movie Recommendation System” provides a simple UI to provide the user with recommended movies. The system uses powerful machine learning algorithms to provide which movie suits the provided movies.

A recommendation system is a type of information filtering system which challenges to assume the priorities of a user, and make recommendations on the basis of the user's priorities. The popularity of the recommendation systems have gradually increased and are recently implemented on almost all online platforms that people use. The content of such a system differs from films, podcasts, books and videos, to colleagues and stories on social media, to commodities in e-commerce websites, to people commercial and dating websites. This system uses content based filtering to recommend movies.

Introduction to Content Based Filtering

Content Based Recommendation procedure checks for the adores and aversion of the user and creates a User-based Profile. For producing a user profile, we check for the item profiles and their equivalent user rating. The user profile is a combination of the sum of the item profiles where combination being the ratings customer or user has evaluated. After the profile of the user has been generated, we estimate the resemblance of the user profile with all the items in the database, which is considered using cosine resemblance between the user generated profile and item profile. Benefits of Content oriented procedure is that other user’s information or data is not essential, and the recommender system can commend new commodities or anything which are not evaluated presently, nevertheless the recommender system will not recommend the items outside the type of items the user has given ratings of.

REQUIREMENTS

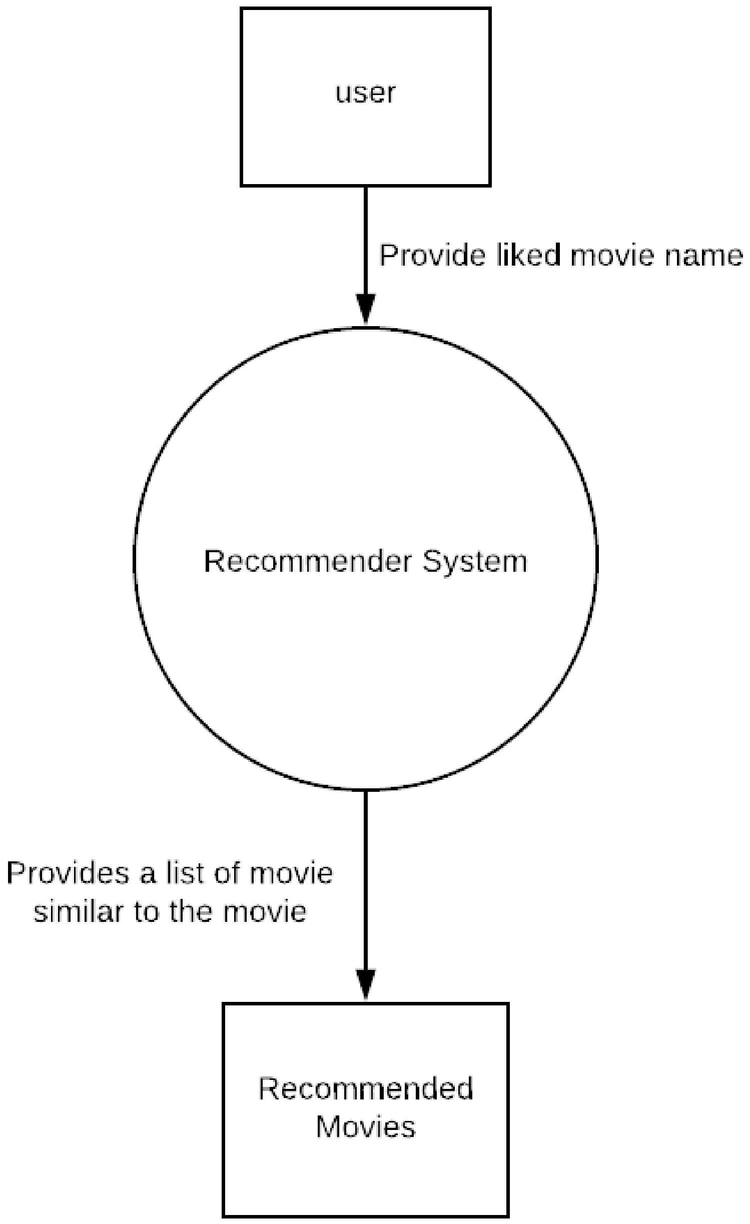
- **Software**
  - Python
  - Visual Studio Code
  - Google Docs
  - Mac OS

- **Hardware**
  - Processor: 1.4 Ghz Processor or more
  - RAM: 4 GB 2133 Mhz
  - Hard Disk: 50 MB
  - Monitor: 13.3” inch DISPLAY
  - Keyboard: 108 key normal

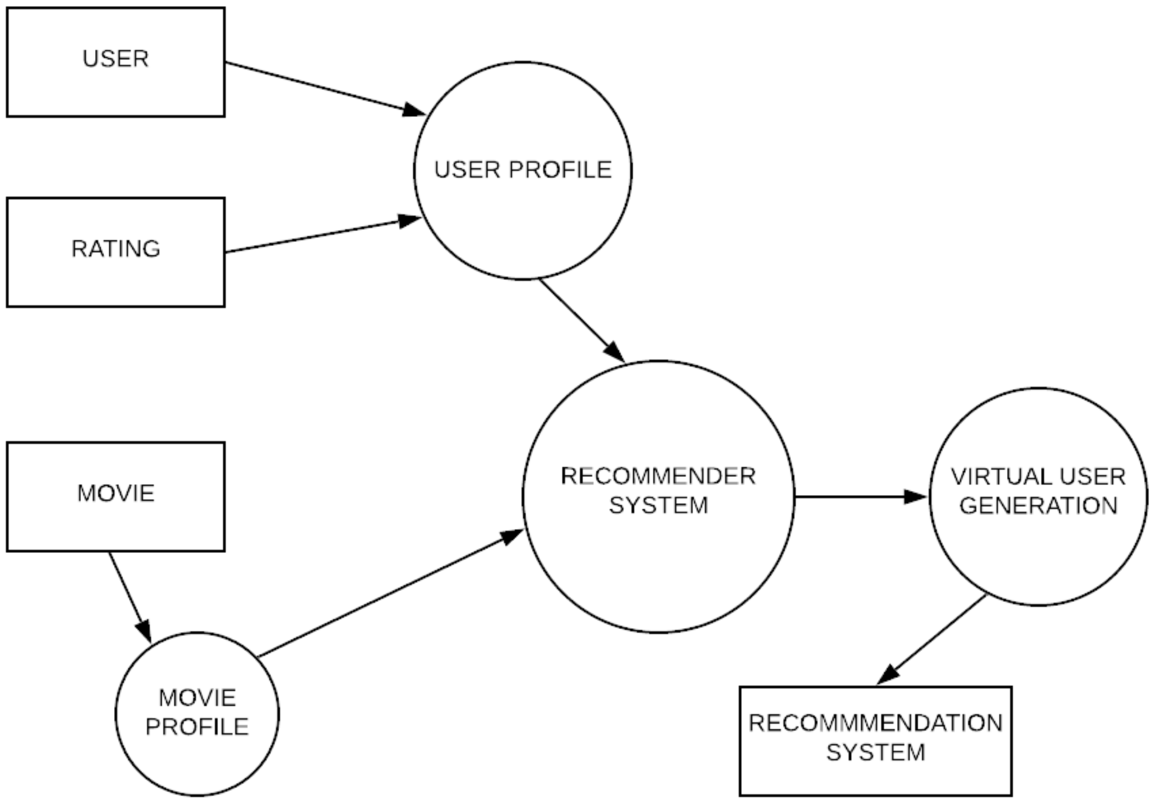
BLOCK DIAGRAM

DATA FLOW DIAGRAM (DFD):

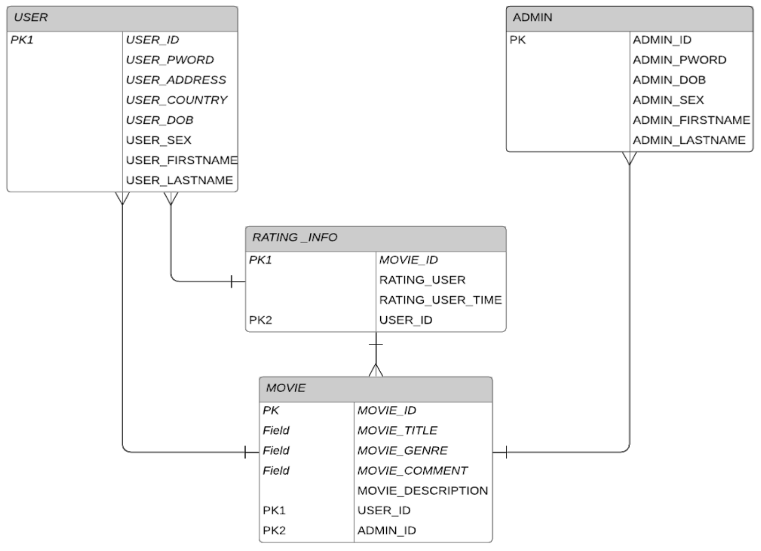
LEVEL 0 DFD:



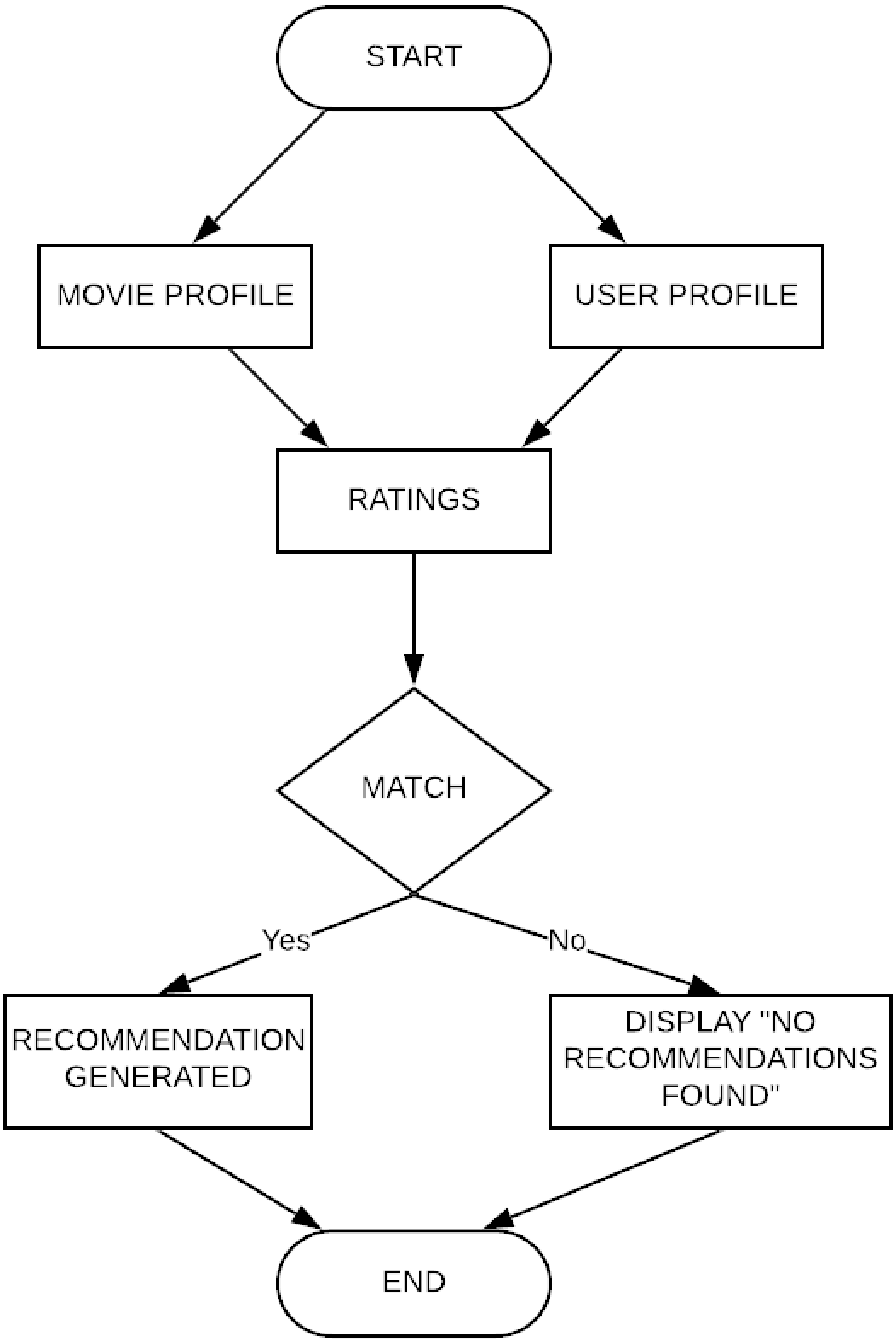
LEVEL 1 DFD:



ER DIAGRAM:



FLOWCHART:


START
MOVIE PROFILE
USER PROFILE
RATINGS
MATCH
Yes
No
RECOMMENDATION GENERATED
DISPLAY "NO RECOMMENDATIONS FOUND"
END

METHODOLOGY

Machine Learning :

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial **intelligence** based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.It has a major role in this project, as making the machine learn through repeated collection of data from previous observations and learning algorithm.This machine learning algorithm is very useful in evaluation of our problems as we human's can't match the working capacity of the machines.The only need is an algorithm which can guide the machine so that we can extract the computation to our problems more effectively.Because of new computing technologies, machine learning today is not like machine learning of the past. It was born from pattern recognition and the theory that computers can learn without being programmed to perform specific tasks; researchers interested in artificial intelligence wanted to see if computers could learn from data. The iterative aspect of machine learning is important because as models are exposed to new data, they are able to independently adapt.

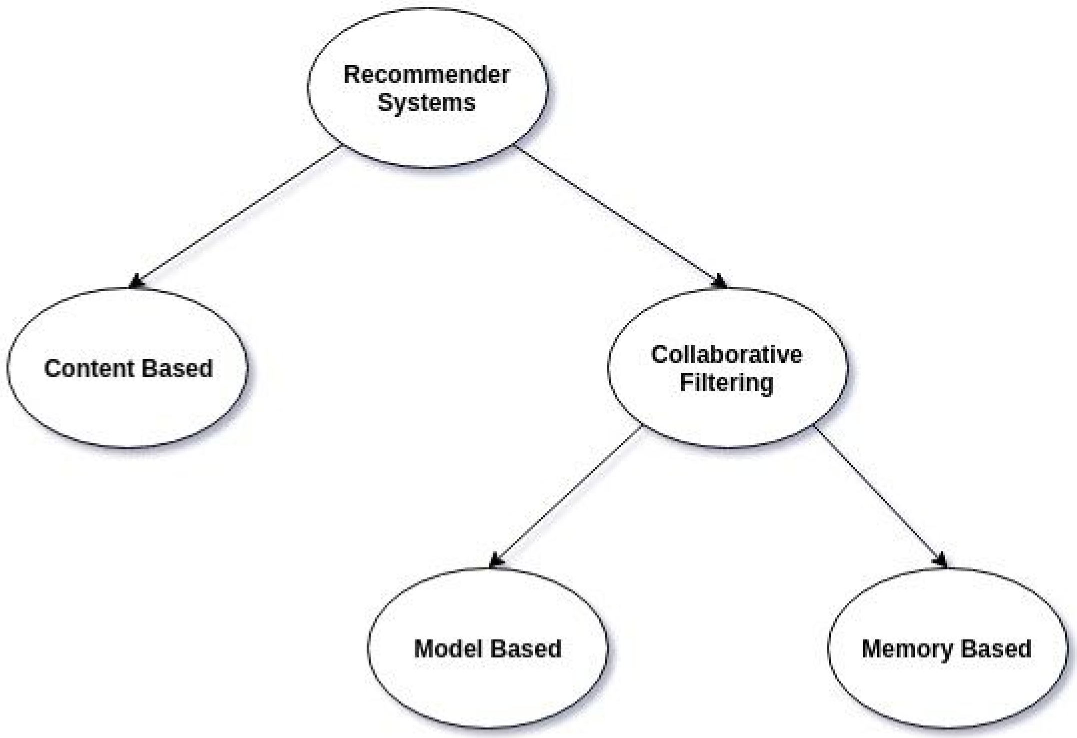
Applications of Machine Learning :

- Virtual Personal Assistants
- Predictions while Commuting
- Videos Surveillance
- Social Media Services
- Email Spam and Malware Filtering
- Recommender Systems

<u>Recommender Systems</u> :

Recommender systems are an important class of machine learning algorithms that offer "relevant" suggestions to users. Categorized as either collaborative filtering or a content-based system, check out how these approaches work along with implementations to follow from example code.
Practically, recommender systems encompass a class of techniques and algorithms which are able to suggest "relevant" items to users. Ideally, the suggested items are as relevant to the user as possible, so that the user can engage with those items: YouTube videos, news articles, online products, and so on.
Items are ranked according to their relevance, and the most relevant ones are shown to the user. The relevancy is something that the recommender system must determine and is mainly based on historical data. If you've recently watched YouTube videos about elephants, then YouTube is going to start showing you a lot of elephant videos with similar titles and themes.

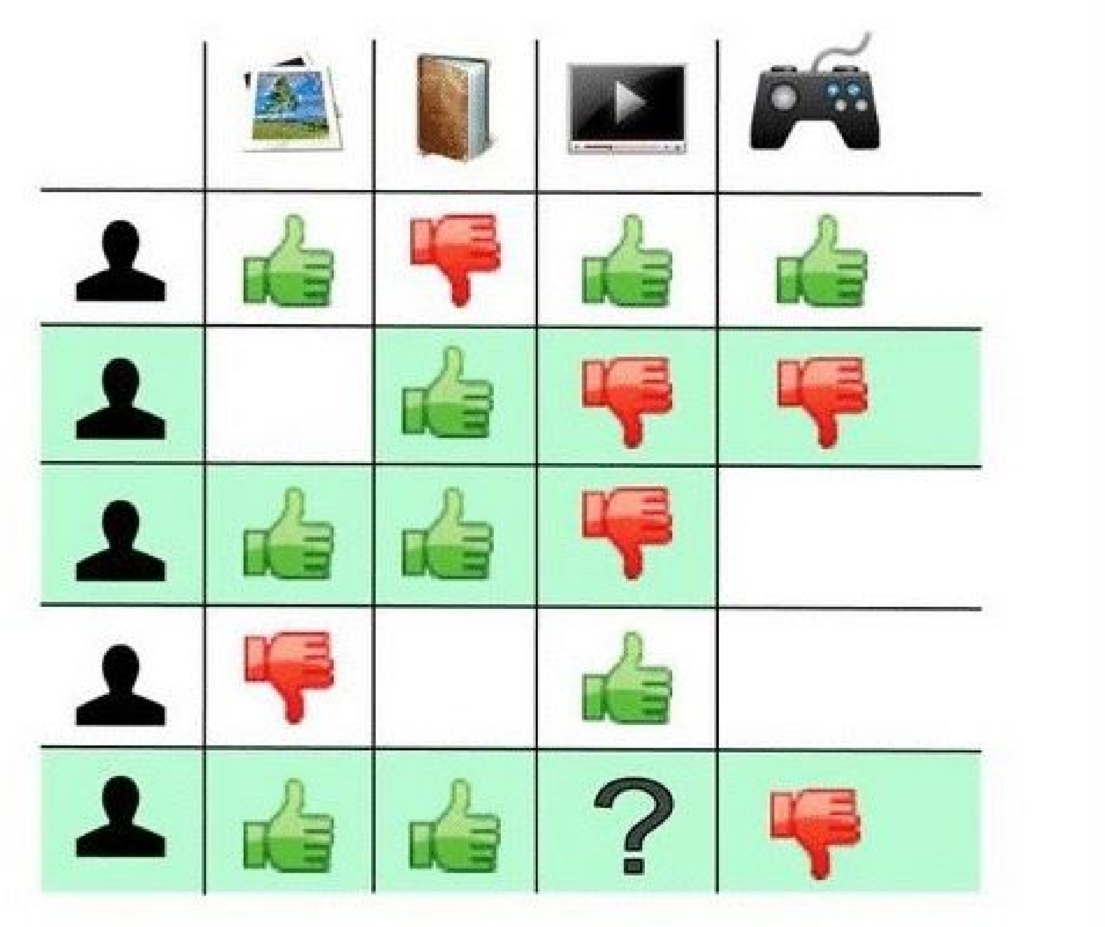
Recommender systems are generally divided into two main categories: collaborative filtering and content-based systems.



<u>Collaborative Filtering Systems:</u>

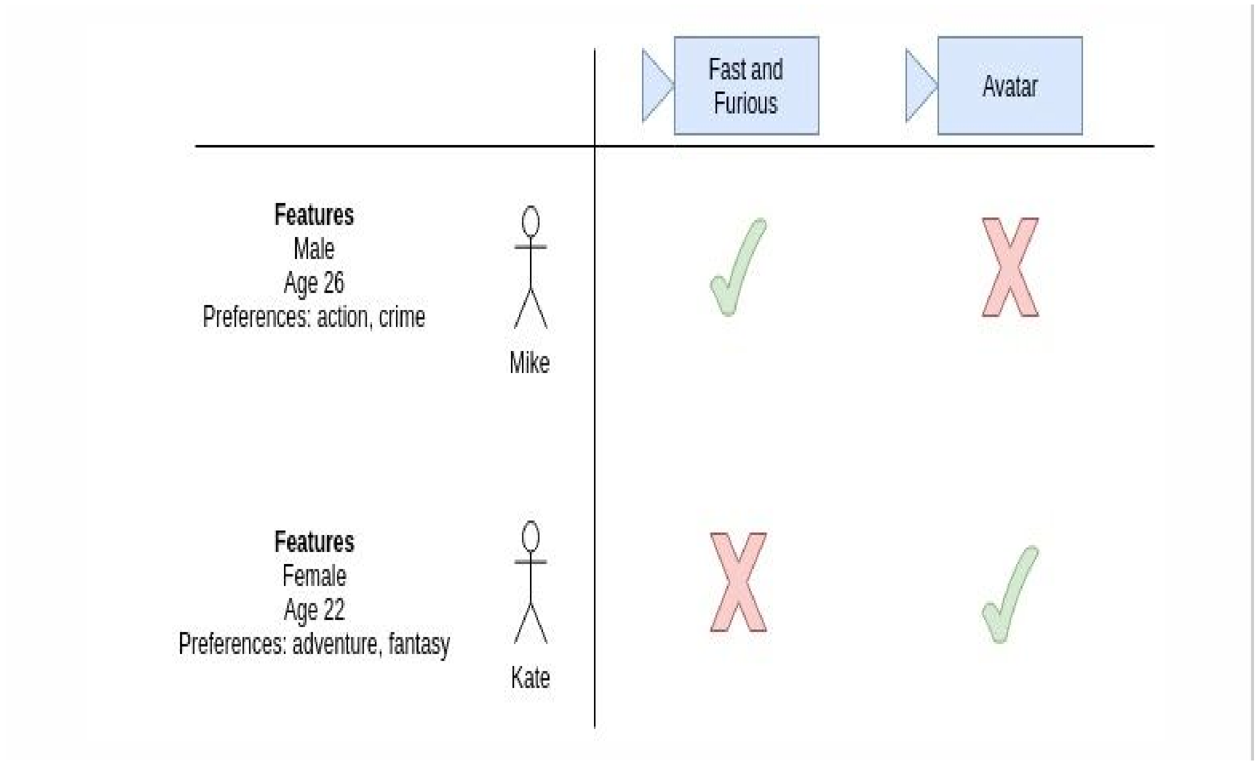
Collaborative Filtering Systems methods for recommender systems are methods that are solely based on the past interactions between users and the target items. Thus, the input to a collaborative filtering system will be all historical data of user interactions with target items. This data is typically stored in a matrix where the rows are the users, and the columns are the items.

The core idea behind such systems is that the historical data of the users should be enough to make a prediction. I.e we don't need anything more than that historical data, no extra push from the user, no presently trending information, etc.



Content-based Systems:

In contrast to collaborative filtering, content-based approaches will use additional information about the user and / or items to make predictions.
For example, in the gif we saw above, a content-based system might consider the age, sex, occupation, and other personal user factors when making the predictions. It's much easier to predict that the person wouldn't like the video if we knew it was about skateboarding, but the user's age is 87.
That's why when you sign up for many online websites and services, they ask you to (optionally) give your date of birth, gender, and ethnicity! It's just more data for their system to make better predictions.
Thus, content-based methods are more similar to classical machine learning, in the sense that we will build features based on user and item data and use that to help us make predictions. Our system input is then the features of the user and the features of the item. Our system output is the prediction of whether or not the user would like or dislike the item.



TECHNOLOGY USED

PYTHON :

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido Van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Why do we choose Python ?

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English words frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Characteristics of Python :

Following are important characteristics of Python Programming −

- It supports functional and structured programming methods as well as OOP.
- **It can be used as a scripting language or can be compiled to byte-code for building large applications**.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.

- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

MODULES USED

In this project we have used mainly two modules of Python. One is imported for adding the GUI (Graphical User Interface) component to our project and second is for manipulating that data of movie dataset using some predefined libraries of Python.

Tkinter :

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

- **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look at this option in this chapter.
- **wxPython** − This is an open-source Python interface for wxWindows.
- **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine.

It has a major role in our project and it has helped us to add the GUI component and create an interface for users to interact with.

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter is lightweight and relatively painless to use compared to other frameworks. This makes it a compelling choice for building GUI applications in Python, especially for applications where a modern sheen is unnecessary, and the top priority is to build something that's functional and cross-platform quickly.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Components of Tkinter :

Label :

This widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want.

It is also possible to underline part of the text (like to identify a keyboard shortcut) and span the text across multiple lines.

Syntax:

Here is the simple syntax to create this widget −

w = Label (master, option, ...)

Entry :

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

- If you want to display multiple lines of text that can be edited, then you should use the Text widget.
- If you want to display one or more lines of text that cannot be modified by the user, then you should use the Label widget.

Syntax:

Here is the simple syntax to create this widget −

w = Entry(master, option, ...)

Button :

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

Syntax :

Here is the simple syntax to create this widget −

w = Button (master, option=value, ...)

StringVar :

Tkinter supports some variables which are used to manipulate the values of Tkinter widgets. These variables work like normal variables.

set() and get() methods are used to set and retrieve the values of these variables.

The values of these variables can be set using set() method or by using constructor of these variables.

There are 4 tkinter variables.

- BooleanVar()
- StringVar()
- IntVar()
- DoubleVar()

Scikit-learn (sklearn) :

What is scikit-learn?

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.
It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.
The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis

Extensions or modules for SciPy care conventionally named Scikit. As such, the module provides learning algorithms and is named scikit-learn. The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance.

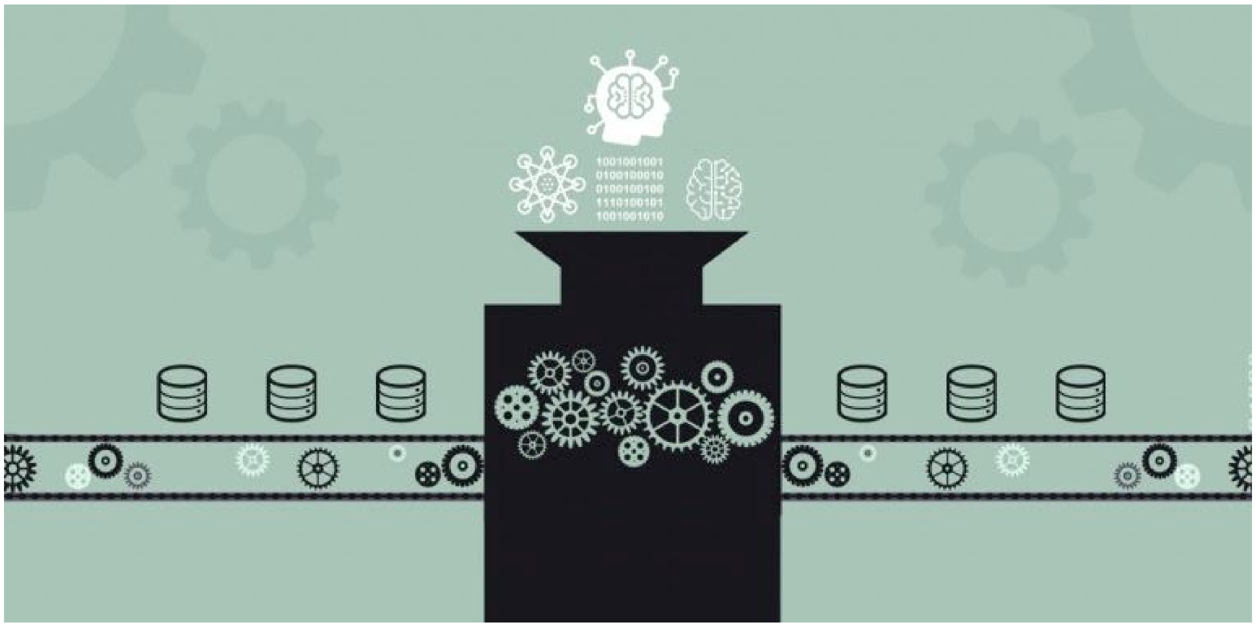
Some popular groups of models provided by scikit-learn include:

- **Clustering**: for grouping unlabeled data such as KMeans.
- **Cross Validation**: for estimating the performance of supervised models on unseen data.
- **Datasets**: for test datasets and for generating datasets with specific properties for investigating model behavior.
- **Feature extraction**: for defining attributes in image and text data.
- **Feature selection**: for identifying meaningful attributes from which to create supervised models.

Machine Learning pipeline using scikit-learn :

Introduction:

For building any machine learning model, it is important to have a sufficient amount of data to train the model. The data is often collected from various resources and might be available in different formats. Due to this reason, data cleaning and preprocessing become a crucial step in the machine learning project.
Whenever new data points are added to the existing data, we need to perform the same preprocessing steps again before we can use the machine learning model to make predictions. This becomes a tedious and time-consuming process.

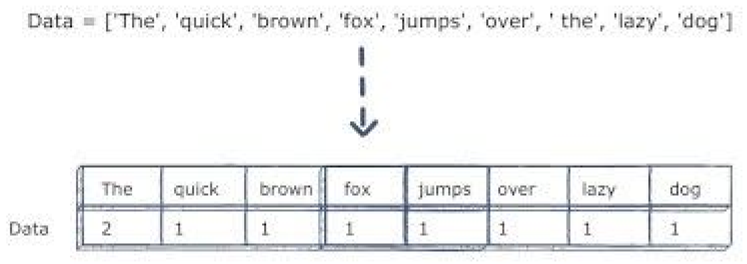


An alternate to this is creating a machine learning pipeline that remembers the complete set of preprocessing steps in the exact same order. So that whenever any new data point is introduced, the machine learning pipeline performs the steps as defined and uses the machine learning model to predict the target variable.

Count Vectorizer :

In order to use textual data for predictive modeling, the text must be parsed to remove certain words – this process is called tokenization. These words need to then be encoded as integers, or floating-point values, for use as inputs in machine learning algorithms. This process is called feature extraction (or vectorization).

Scikit-learn’s Count Vectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text.



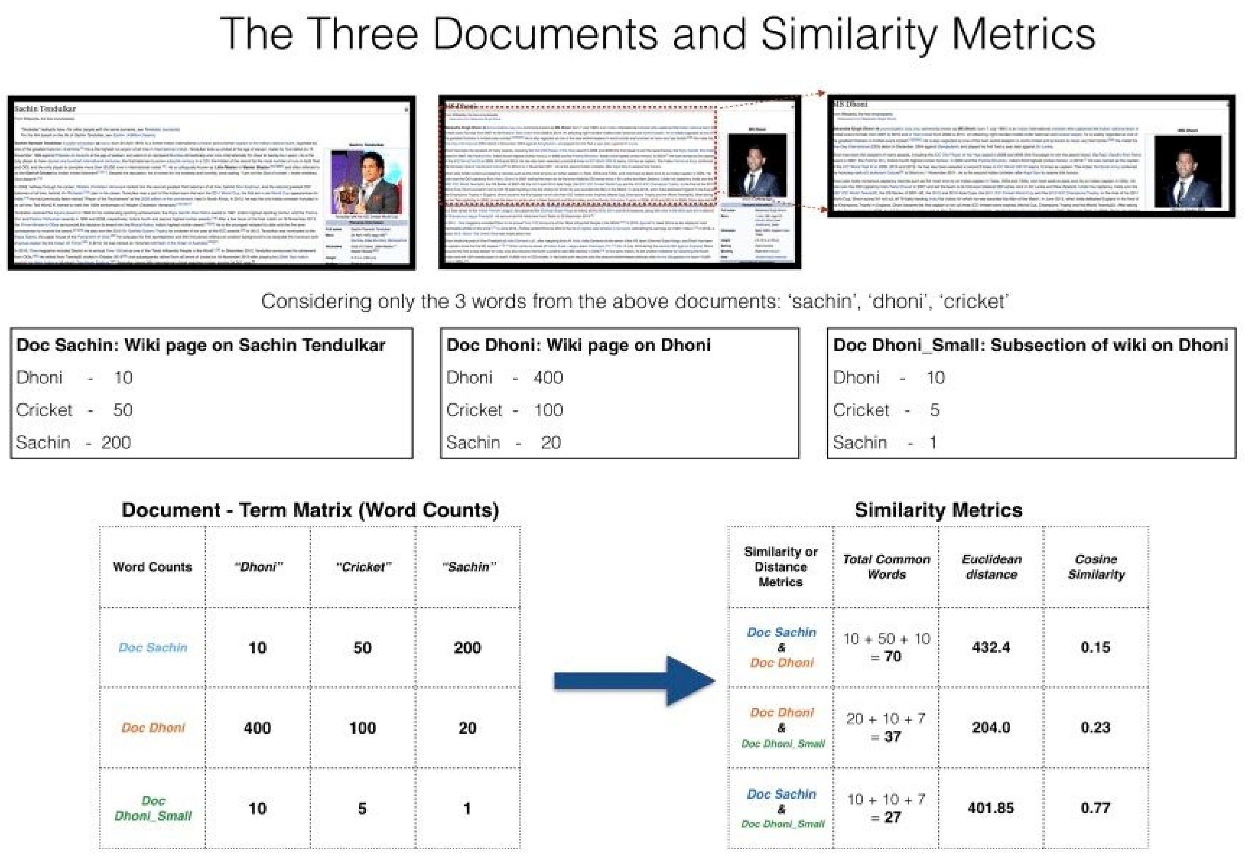
```
Data = ['The', 'quick', 'brown', 'fox', 'jumps', 'over', ' the', 'lazy', 'dog']
```

	The	quick	brown	fox	jumps	over	lazy	dog
Data	2	1	1	1	1	1	1	1

As we can see how a string is converted into a matrix and this is what count vectorizer does to string and convert into a form so that we can find the similarity index for that string with some other string.
Here we have a large data of movies in which a lot of categories
Are there any movies that can be compared with some other movie so using count vectorizer we will create a count of similarity of each movie.
Now from here on we have created a count vector of each movie.

<u>Vector Representation and Cosine Similarity</u> :

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.



The Three Documents and Similarity Metrics

Considering only the 3 words from the above documents: ‘sachin’, ‘dhoni’, ‘cricket’

Doc Sachin: Wiki page on Sachin Tendulkar	**Doc Dhoni: Wiki page on Dhoni**	**Doc Dhoni_Small: Subsection of wiki on Dhoni**
Dhoni - 10	Dhoni - 400	Dhoni - 10
Cricket - 50	Cricket - 100	Cricket - 5
Sachin - 200	Sachin - 20	Sachin - 1

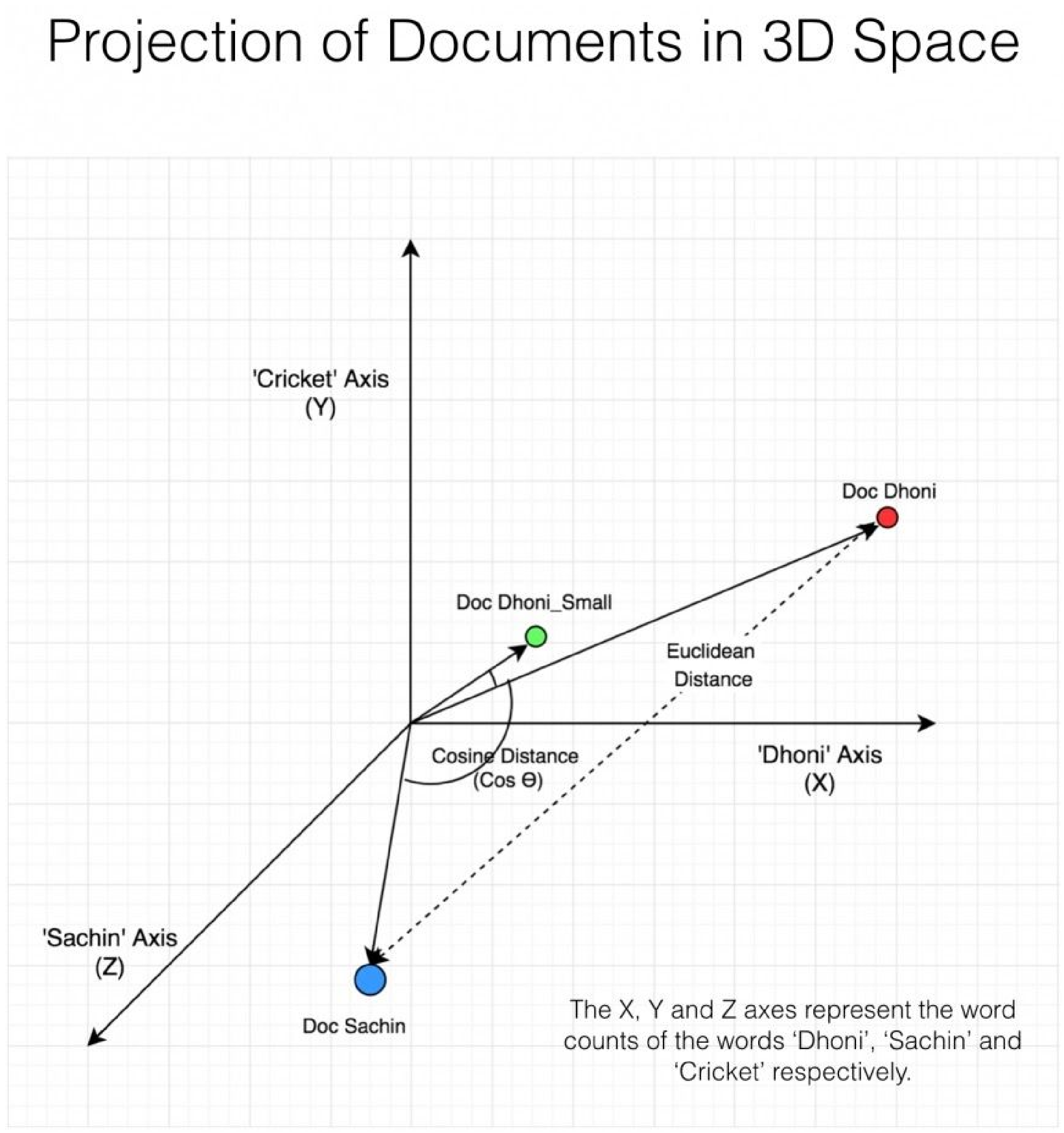
Document - Term Matrix (Word Counts)

Word Counts	*“Dhoni”*	*“Cricket”*	*“Sachin”*
Doc Sachin	10	50	200
Doc Dhoni	400	100	20
Doc Dhoni_Small	10	5	1

Similarity Metrics

Similarity or Distance Metrics	*Total Common Words*	*Euclidean distance*	*Cosine Similarity*
Doc Sachin & Doc Dhoni	10 + 50 + 10 = **70**	**432.4**	**0.15**
Doc Dhoni & Doc Dhoni_Small	20 + 10 + 7 = **37**	**204.0**	**0.23**
Doc Sachin & Doc Dhoni_Small	10 + 10 + 7 = **27**	**401.85**	**0.77**

As you can see, all three documents are connected by a common theme – the game of Cricket. Our objective is to quantitatively estimate the similarity between the documents.
For ease of understanding, let’s consider only the top 3 common words between the documents: ‘Dhoni’, ‘Sachin’ and ‘Cricket’.
You would expect Doc A and Doc C, that is the two documents on Dhoni would have a higher similarity over Doc A and Doc B, because Doc C is essentially a snippet from Doc A itself.
However, if we go by the number of common words, the two larger documents will have the most common words,


Projection of Documents in 3D Space
'Cricket' Axis (Y)
Doc Dhoni
Doc Dhoni_Small
Euclidean Distance
'Dhoni' Axis (X)
Cosine Distance (Cos Θ)
'Sachin' Axis (Z)
Doc Sachin
The X, Y and Z axes represent the word counts of the words 'Dhoni', 'Sachin' and 'Cricket' respectively.

and therefore will be judged as most similar, which is exactly what we want to avoid.
The results would be more congruent when we use the cosine similarity score to assess the similarity.

3d Projection:

As you can see, Doc Dhoni_Small and the main Doc Dhoni are oriented closer together in 3-D space, even though they are far apart by magnitude.
It turns out, the closer the documents are by angle, the higher is the Cosine Similarity (Cos theta).

Cosine Similarity Formula:

As you include more words from the document, it’s harder to visualize a higher dimensional space. But you can directly compute the cosine similarity using this math formula.

$$Cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \, \|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

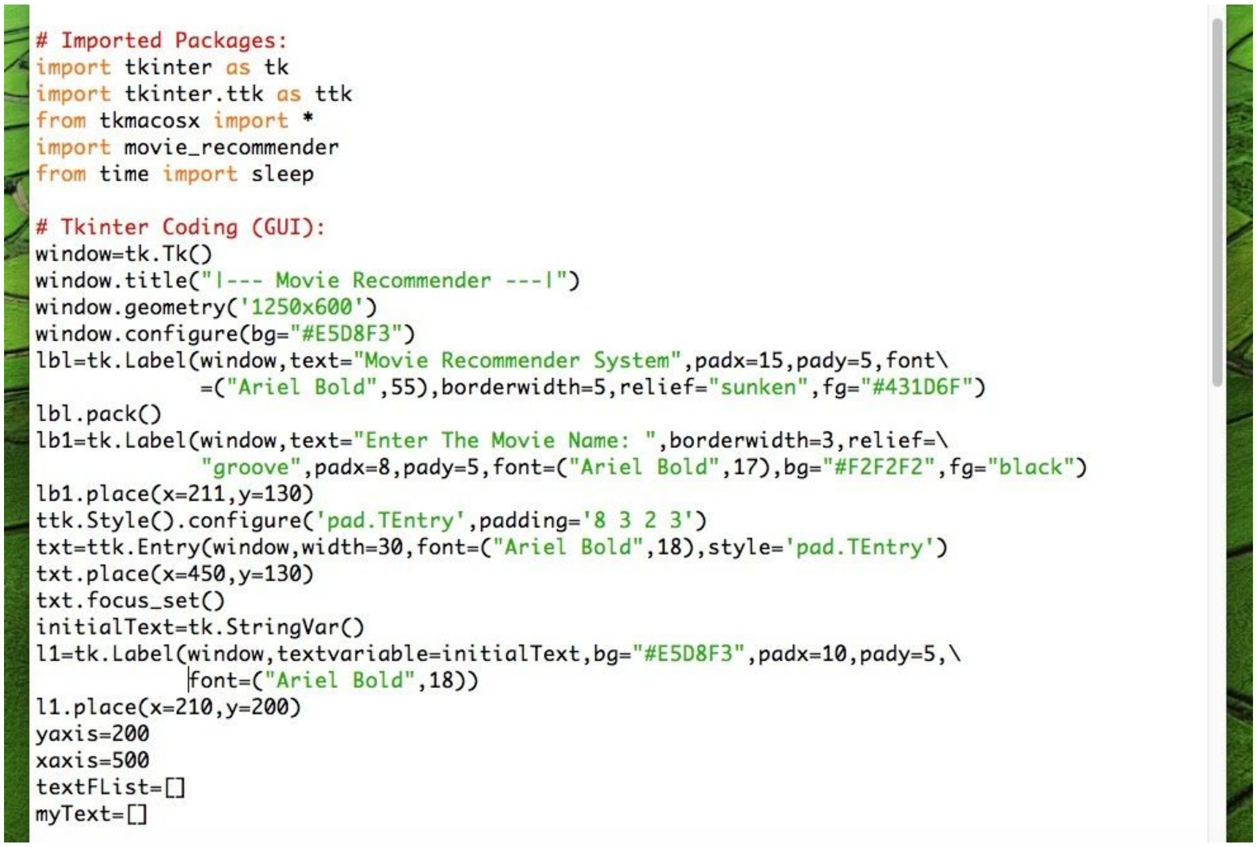
where, $\vec{a} \cdot \vec{b} = \sum_1^n a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$ is the dot product of the two vectors.

Then, use cosine_similarity() to get the final output. It can take the document term matrix as a pandas dataframe as well as a sparse matrix as inputs. So this was all about cosine similarity.

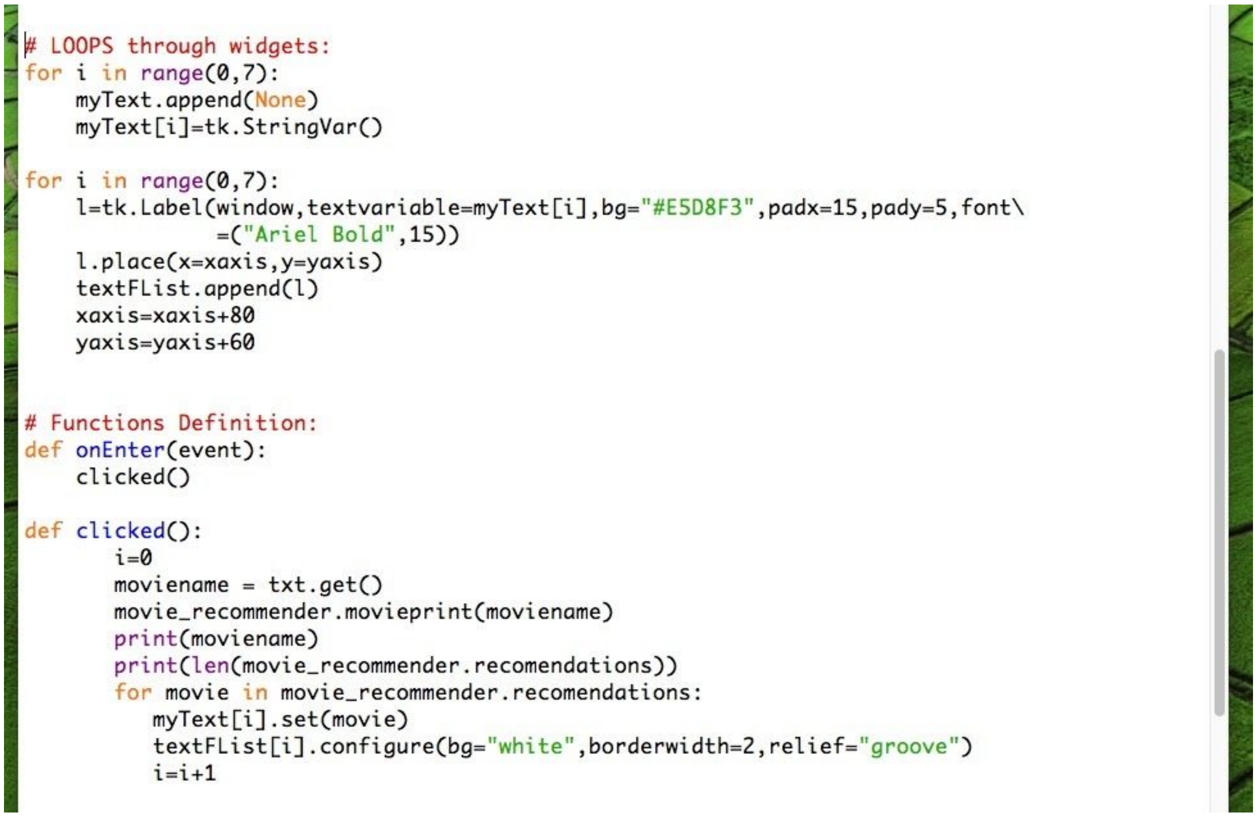
SOURCE CODE

The following shows the code for the GUI of the “Movie Recommendation System”:

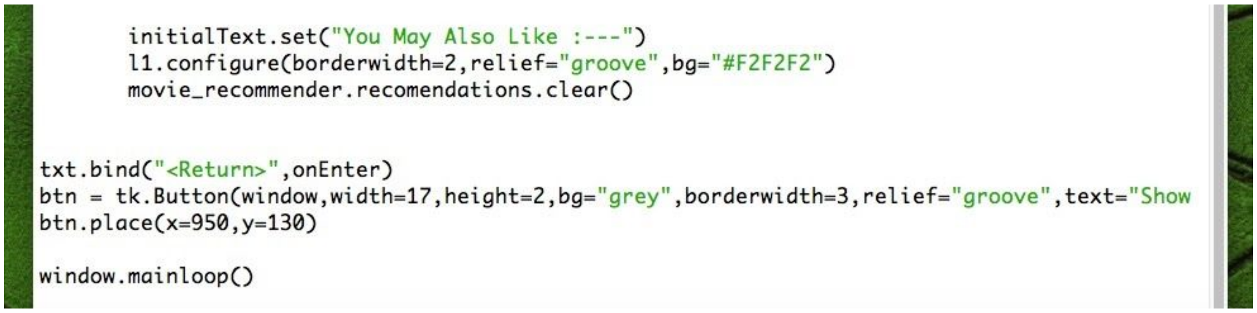
The code represents the GUI of the project.



```
# Imported Packages:
import tkinter as tk
import tkinter.ttk as ttk
from tkmacosx import *
import movie_recommender
from time import sleep

# Tkinter Coding (GUI):
window=tk.Tk()
window.title("|--- Movie Recommender ---|")
window.geometry('1250x600')
window.configure(bg="#E5D8F3")
lbl=tk.Label(window,text="Movie Recommender System",padx=15,pady=5,font\
             =("Ariel Bold",55),borderwidth=5,relief="sunken",fg="#431D6F")
lbl.pack()
lb1=tk.Label(window,text="Enter The Movie Name: ",borderwidth=3,relief=\
             "groove",padx=8,pady=5,font=("Ariel Bold",17),bg="#F2F2F2",fg="black")
lb1.place(x=211,y=130)
ttk.Style().configure('pad.TEntry',padding='8 3 2 3')
txt=ttk.Entry(window,width=30,font=("Ariel Bold",18),style='pad.TEntry')
txt.place(x=450,y=130)
txt.focus_set()
initialText=tk.StringVar()
l1=tk.Label(window,textvariable=initialText,bg="#E5D8F3",padx=10,pady=5,\
            font=("Ariel Bold",18))
l1.place(x=210,y=200)
yaxis=200
xaxis=500
textFList=[]
myText=[]
```

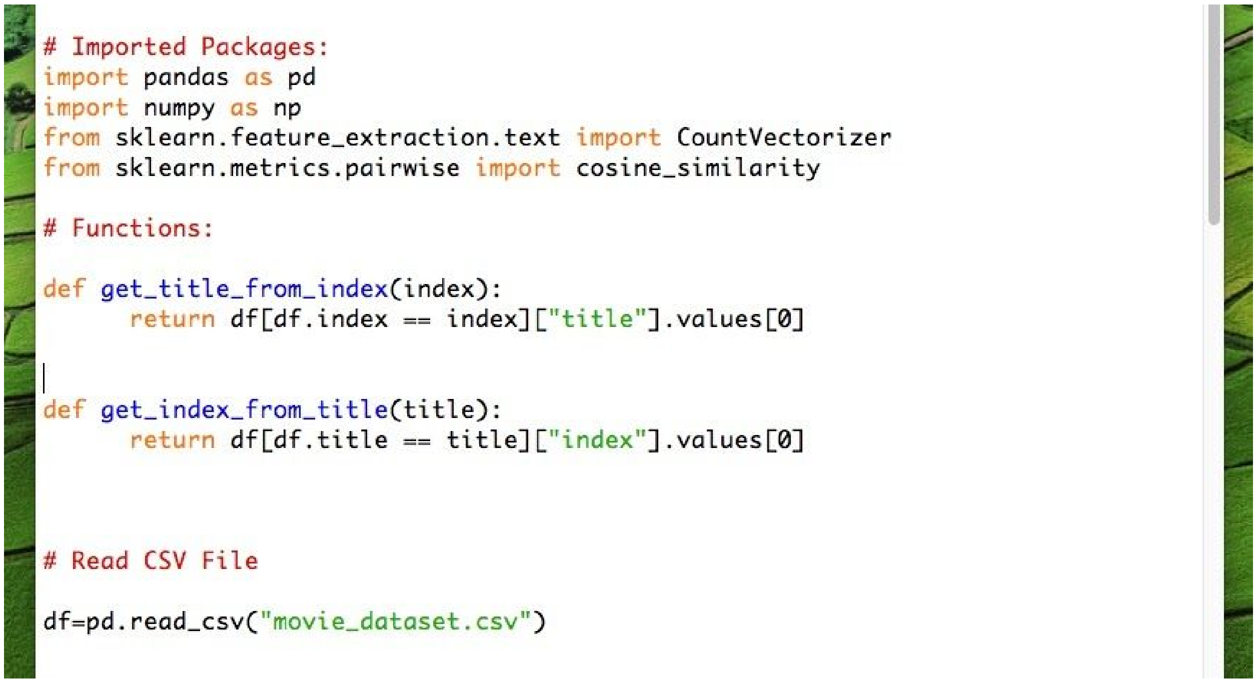


```
# LOOPS through widgets:
for i in range(0,7):
    myText.append(None)
    myText[i]=tk.StringVar()

for i in range(0,7):
    l=tk.Label(window,textvariable=myText[i],bg="#E5D8F3",padx=15,pady=5,font\
               =("Ariel Bold",15))
    l.place(x=xaxis,y=yaxis)
    textFList.append(l)
    xaxis=xaxis+80
    yaxis=yaxis+60


# Functions Definition:
def onEnter(event):
    clicked()

def clicked():
       i=0
       moviename = txt.get()
       movie_recommender.movieprint(moviename)
       print(moviename)
       print(len(movie_recommender.recomendations))
       for movie in movie_recommender.recomendations:
          myText[i].set(movie)
          textFList[i].configure(bg="white",borderwidth=2,relief="groove")
          i=i+1
```

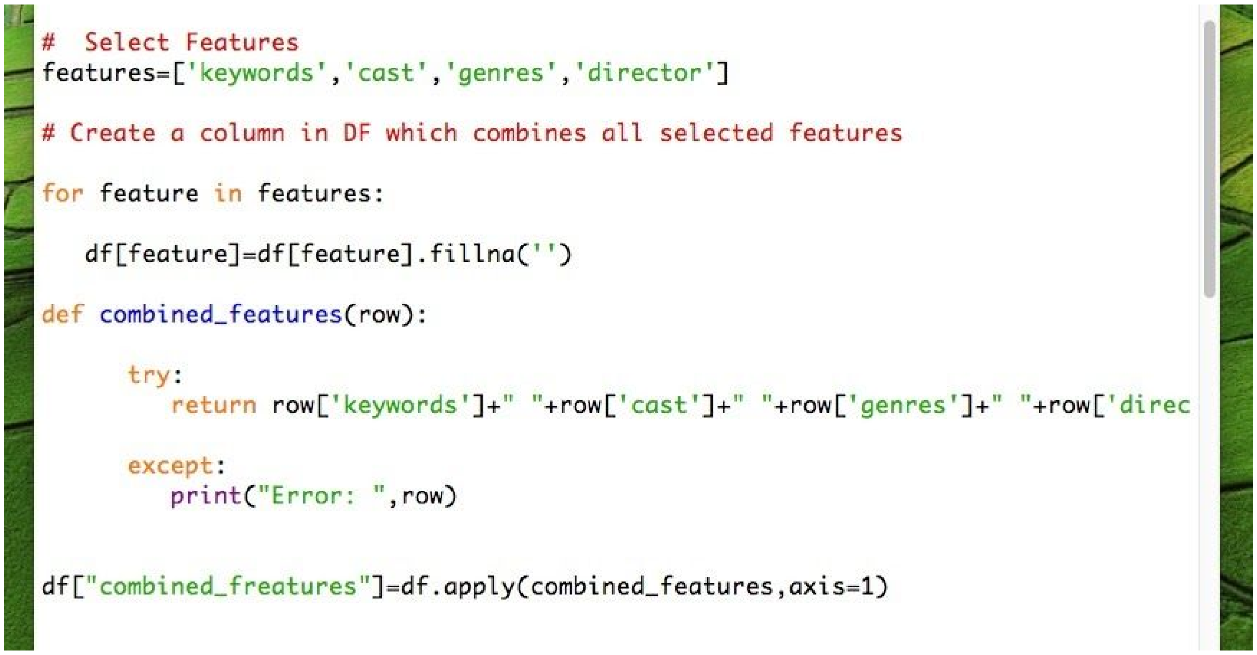


```
       initialText.set("You May Also Like :---")
       l1.configure(borderwidth=2,relief="groove",bg="#F2F2F2")
       movie_recommender.recomendations.clear()


txt.bind("<Return>",onEnter)
btn = tk.Button(window,width=17,height=2,bg="grey",borderwidth=3,relief="groove",text="Show
btn.place(x=950,y=130)

window.mainloop()
```

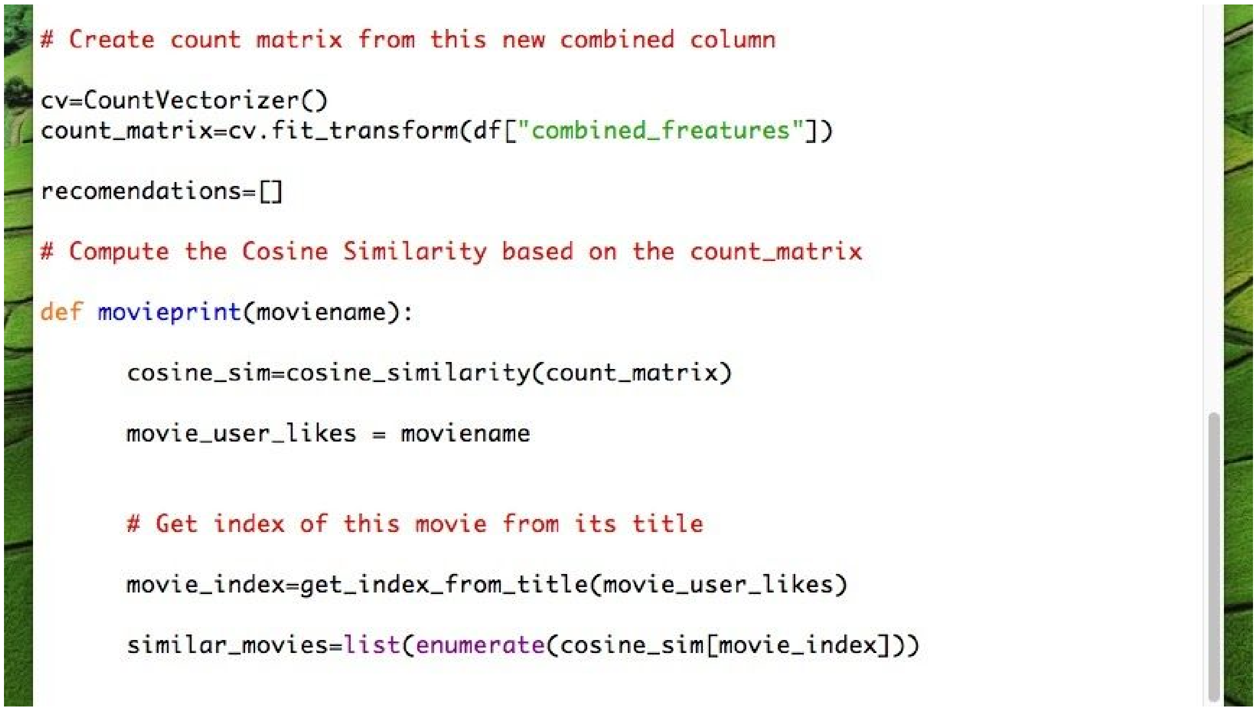
The following will show the Machine Learning code used to recommend the movies according to the user choice.



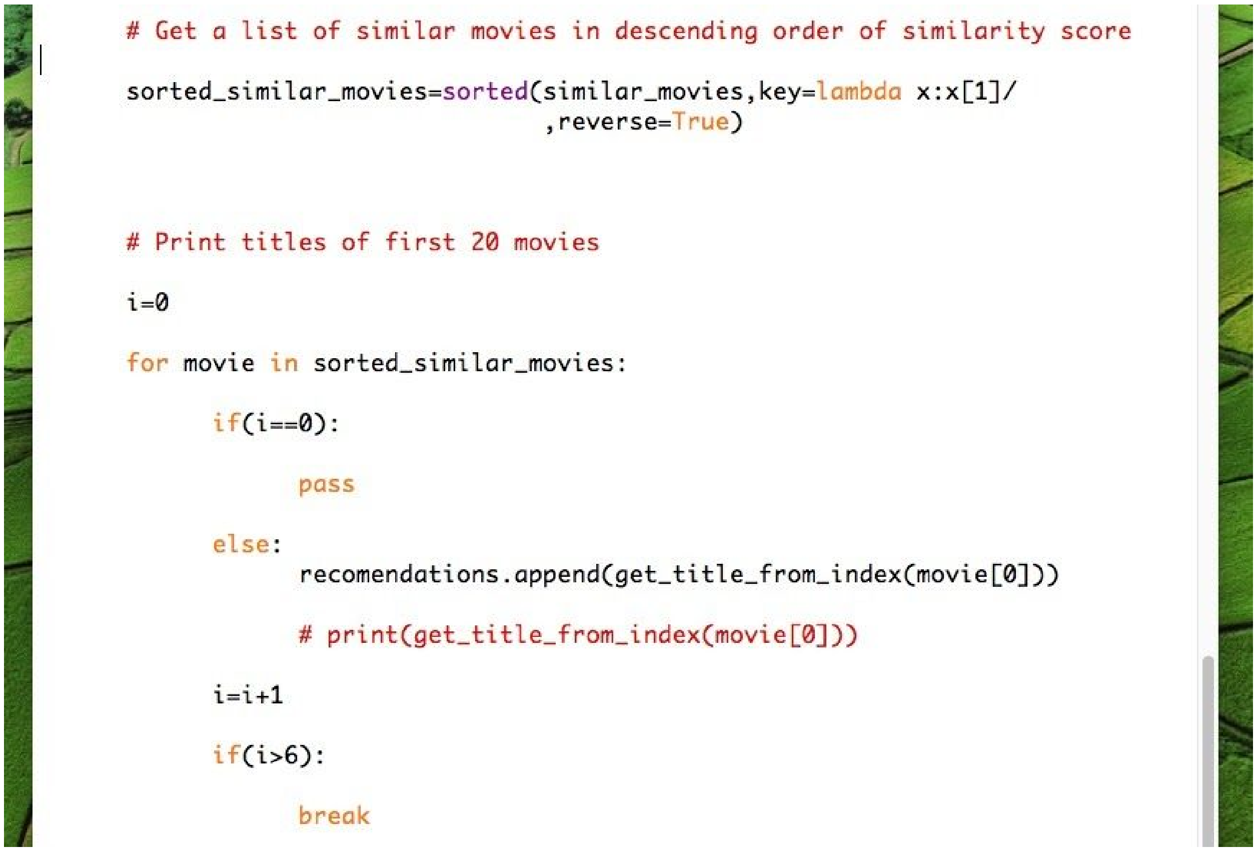
```
# Imported Packages:
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Functions:

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]



# Read CSV File

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



```
#  Select Features
features=['keywords','cast','genres','director']

# Create a column in DF which combines all selected features

for feature in features:

   df[feature]=df[feature].fillna('')

def combined_features(row):

      try:
         return row['keywords']+" "+row['cast']+" "+row['genres']+" "+row['direc

      except:
         print("Error: ",row)


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



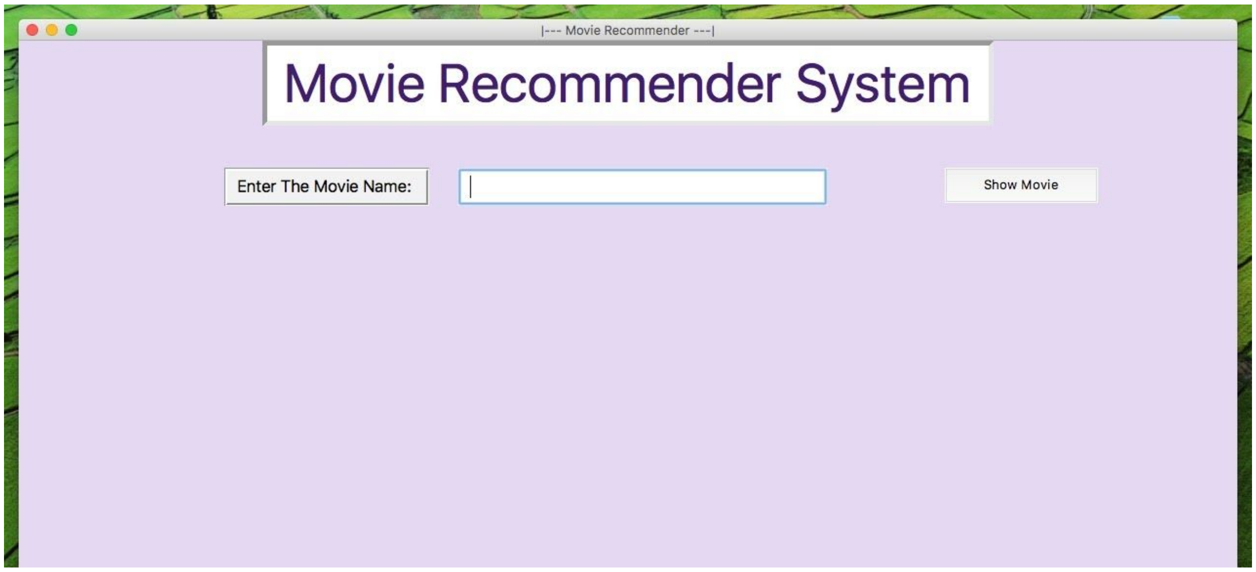
```
# Create count matrix from this new combined column

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

recomendations=[]

# Compute the Cosine Similarity based on the count_matrix

def movieprint(moviename):

      cosine_sim=cosine_similarity(count_matrix)

      movie_user_likes = moviename


      # Get index of this movie from its title

      movie_index=get_index_from_title(movie_user_likes)

      similar_movies=list(enumerate(cosine_sim[movie_index]))
```



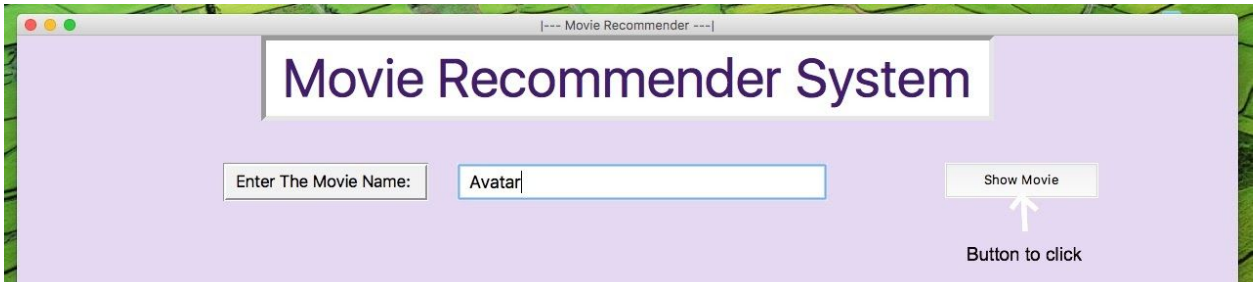
```
      # Get a list of similar movies in descending order of similarity score

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


      # Print titles of first 20 movies

      i=0

      for movie in sorted_similar_movies:

            if(i==0):

                  pass

            else:
                  recomendations.append(get_title_from_index(movie[0]))

                  # print(get_title_from_index(movie[0]))

            i=i+1

            if(i>6):

                  break
```

OUTPUT AND SCREENSHOT

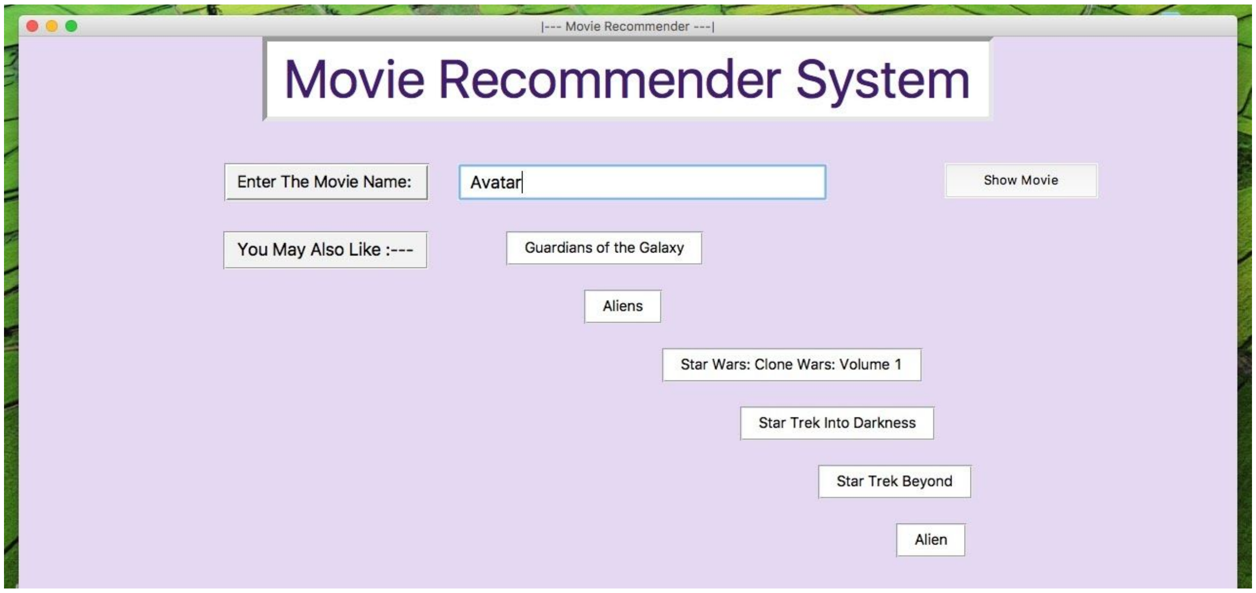
This is the screenshot of the application on its startup.



The screenshot here will represent how the movie name will be entered in the text field and then the show movie button is clicked to implement the recommendation of the movie and thus obtain movies similar to the one provided.



The screenshot provided under shows how the output of similar movies is provided back to the user.



CONCLUSION

- Recommender systems have a great value in recommending relevant resources to the users. It can be quite useful in finding novel and serendipitous recommendations.
- The effectiveness of the recommender system relies on the algorithm it uses to find interesting resources.
- Here machine learning algorithms have been used in the project which help us in understanding the problem of recommendation systems.

Our project of “Movie Recommendation System” uses content based Filtering to predict the movies to users. This in turn makes them better than the ones that work on the bases of movie ratings provided by the user.

Content based algorithms together with data mining can be used not only to provide the user with better movie recommendations but also deliver the user with additionally advanced and sophisticated endorsements as movies which have a poor rating score in any of the movie features produced based on data mining will be refined out the significant allocation platform.

REFERENCES

Throughout our project we have taken help through a number of websites to gain the knowledge and using it appropriately to implement our project . The following are the websites we have gone through :

Website References:

1. **https://www.machinelearningplus.com/nlp/cosine-similarity**
2. **https://www.educative.io/edpresso/countvectorizer-in-python**
3. **https://www.geeksforgeeks.org**
4. **https://www.python.org**
5. **https://www.analyticsvidhya.com/blog/2020/01/build-your-first-machine-learning-pipeline-using-scikit-learn/**

Data Set for movies:

- **https://www.imdb.com**