

A  
Project Report  
on  
**MOVIE RECOMMENDATION SYSTEM USING COSINE  
SIMILARITY**

Submitted for partial fulfilment of the requirements for the award of the degree of

**BACHELOR OF ENGINEERING**

in

**COMPUTER SCIENCE AND ENGINEERING**

By

**SYED SHOAIB SHAH (2451-18-733-055)**

**B.ADHI SHAKTI(2451-18-733-012)**

**ADITHYA KARKATA(2451-18-733-055)**

Under the guidance of

**G.Madhu**

Assistant Professor

Department of CSE



**Maturi Venkata Subba Rao (MVSR) ENGINEERING COLLEGE**

Department of Computer Science and Engineering

(Affiliated to Osmania University & Recognized by AICTE)

Nadergul, Saroor Nagar Mandal, Hyderabad – 501 510

Academic Year: 2021-22



## CERTIFICATE

*This is to certify that the project work entitled “Movie Recommendation System using cosine similarity” is a bonafide work carried out by **Mr. Adithya Karkata (2451-18-733-055)**, **Mr. Syed Shoaib Shah Harooni(2451-18-733-006)**, **Mr.B.Adhi Shakthi(2451-18-733-012)** in partial fulfilment of the requirements for the award of degree of Bachelor of Engineering in Computer Science and Engineering from Maturi Venkata Subba Rao (MVSR) Engineering College, affiliated to OSMANIA UNIVERSITY, Hyderabad, during the Academic Year 2021-22 under our guidance and supervision.*

*The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma to the best of our knowledge and belief.*

### Internal Guide

G.Madhu  
Assistant Professor  
Department of CSE  
MVSREC.

### Head of the Department

J.Prasanna Kumar  
Professor & Head  
Department of CSE  
MVSREC.

## **DECLARATION**

This is to certify that the work reported in the present project entitled “Movie Recommendation System Using Cosine Similarity” is a record of bonafide work done by us in the Department of Computer Science and Engineering, Maturi Venkata Subba Rao (MVSR) Engineering College, Osmania University during the Academic Year 2021-22. The reports are based on the project work done entirely by us and not copied from any other source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

**Adithya Karkata (2451-18-733-055)**

**Syed Shoaib Shah Harooni (2451-18-733-006)**

**B.Adhi Shakti (2451-18-733-012)**

## **ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to our project guide **G.Madhu** for his valuable guidance and interest throughout the course of this project. We extend our sincere thanks to Head of the Department **Prof.J.Prasanna Kumar** for his encouragement and mentorship in the due course of the project. We acknowledge with thanks for the kind of patronage and motivation which we received from the principal **Dr. G. Kanaka Durga**. We would like to express our gratitude to the technical and non-technical staff members for their support and helping hand in carrying out the project. We would like to thank our project coordinator Sabitha Nagamala for his constant monitoring, guidance and support.

Finally, we would like to take this opportunity to thank our families for their support through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

**Adithya Karkata (2451-18-733-055)**

**Syed Shoaib Shah Harooni (2451-18-733-006)**

**B.Adhi Shakti (2451-18-733-012)**

## **VISION**

- To impart technical education of the highest standards, producing competent and confident engineers with an ability to use computer science knowledge to solve societal problems.

## **MISSION**

- To make the learning process exciting, stimulating and interesting.
- To impart adequate fundamental knowledge and soft skills to students.
- To expose students to advanced computer technologies in order to excel in engineering practices by bringing out the creativity in students.
- To develop economically feasible and socially acceptable software.

## **PEOs:**

**PEO-1:** Achieve recognition through demonstration of technical competence for successful execution of software projects to meet customer business objectives..

**PEO-2:** Practice life-long learning by pursuing professional certifications, higher education or research in the emerging areas of information processing and intelligent systems at a global level.

**PEO-3:** Contribute to society by understanding the impact of computing using a multidisciplinary and ethical approach.

## **PROGRAM OUTCOMES (POs)**

At the end of the program the students (Engineering Graduates) will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialisation for the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and the need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Lifelong learning:** Recognise the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

v

#### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

13. (PSO-1) Demonstrate competence to build effective solutions for computational real-world problems using software and hardware across multi-disciplinary domains.
14. (PSO-2) Adapt to current computing trends for meeting the industrial and societal needs through a holistic professional development leading to pioneering careers or entrepreneurship.

## **COURSE OBJECTIVES AND OUTCOMES**

### **Course Objectives:**

- To enhance practical and professional skills.
- To familiarize tools and techniques of systematic literature survey and documentation
- To expose the students to industry practices and teamwork.
- To encourage students to work with innovative and entrepreneurial ideas

### **Course Outcomes:**

**CO1:** Summarize the survey of the recent advancements to infer the problem statements with applications towards society

**CO2:** Design a software based solution within the scope of the project.

**CO3:** Implement test and deploy using contemporary technologies and tools

**CO4:** Demonstrate qualities necessary for working in a team.

**CO5:** Generate a suitable technical document for the project..



## **ABSTRACT**

Multimedia is considered as one of the best sources of entertainment. People of all age groups love to watch movies. Movie Recommender System is essential in our social lives as it enhances the field of entertainment. The proposed system on Movie Recommendation System caters the requirements of the user. The major aim is to provide crisp relevant content to the end-users out of semi-structured content on the internet. The main purpose is to generate accurate, efficient and personalized recommendations to the user. Various building blocks of the paper like Introduction, Literature Survey, Proposed System, Implementation & Result, Comparative Analysis, Conclusion and Future Work are discussed in detail. The proposed machine learning model is trained, tested, and a sentiment classifier is generated which classifies the sentiments as a good or a bad sentiment. The recommender system is generated by applying Cosine similarity and making API Calls. As a result, the live working of the system generates accurate and personalized recommendations along with the analysis of sentiments for the end users. It is also concluded that Cosine Similarity provides better and efficient results for a recommender system.

**Adithya Karkata (2451-18-733-055)**

**Syed Shoaib Shah Harooni (2451-18-733-006)**

**B.Adhi Shakti (2451-18-733-012)**

## TABLE OF CONTENTS

CONTENTS	PAGE NO.s
Certificate	i
Declaration	ii
Acknowledgement	iii
Vision & Mission	iv
Course Objectives & Outcomes	vi
Abstract	vii
Index	vii
<b>Chapters:</b>	
<b>1. Introduction</b>	<b>1</b>
1.1 Problem statement	2
1.2 Objectives	2
1.3 Motivation	2
1.4 Scope	2
1.5 Software and Hardware requirements	3
<b>2. Literature Survey</b>	<b>4</b>
2.1 Broad Areas	4
2.2 Related work on Movie Recommendation System	6
2.3 Techniques and algorithms applicable	7
2.4 Applications	9
2.5 Summary	10
<b>3. System Design</b>	<b>11</b>
3.1 Block diagram	11

3.2 Description of each module	11
<b>4.Implementation</b>	<b>13</b>
4.1 Environmental setup	13
4.2 Implementation of each module	13
<b>5.Testing and results</b>	<b>21</b>
5.1 DataSet	21
5.2 Testing	21
5.3 Result	21
<b>6. Conclusion and Future Enhancements</b>	<b>23</b>
<b>References</b>	<b>24</b>
<b>Appendix</b>	<b>25</b>
<b>A: source/pseudo code</b>	

## LIST OF FIGURES

Figure No	Figure Name	Page No
2.1	Broad areas for building Machine Learning projects	4
2.2.1	Content Based Filtering & Collaborative Filtering	7
2.3.1	Cosine Similarity	8
3.1.1	System architecture	11
4.2.1	CSV loaded into a Dataframe	14
4.2.3	The Combined Features column in our data frame	15
4.2.5	Cosine Similarity Matrix	17
4.2.7	Similar Movies list	18
4.2.8	Sorted Similar Movies List with similarity score	19
5	Testing and results	21
5.1	Test Case	22

## **LIST OF ABBREVIATIONS**

NLP - Natural Language Processing

POS - Parts Of Speech

NLTK - Natural Language Toolkit

TF-IDF - Term Frequency Inverse Document Frequency



## **CHAPTER-1**

### **1.INTRODUCTION**

Every day, technological advancements reach new heights, resulting in massive increases of information. To deal with such vast amounts of data, we employ machine learning to automate the creation of analytical models. Machine learning was originally classified into three categories: supervised learning, unsupervised learning, and reinforcement learning. Using diverse computational statistics, we employ computers to create predictions to assist us attain better results. Tasks can be completed even if they are not explicitly programmed. Extracting the required data becomes a time-consuming operation. To some extent, search engines fix the problem, but they do not solve the problem of personalisation. Recommendation System framework is important in today's internet browsing, whether you're buying a new computer or just browsing the web. We rely on suggestions from others in our daily lives, whether through word of mouth or assessments of generic surveys. People frequently utilize web-based recommender systems to make purchasing selections for things linked to their preferences. Recommendation systems are software tools and approaches whose objective is to offer meaningful and sensible recommendations for items or products that may be of interest to a group of users. In other words, a recommender system (or a set of recommender systems) is a type of information filtering system that seeks to forecast an item's 'desire' or 'ranking.' Three approaches are commonly used in recommendation systems. Content-based filtering involves profiling a user based on the type of content he or she is interested in, and then recommending items based on the information gathered. Another is collaborative filtering, in which we group individuals who are similar and use that data to produce recommendations. Hybrid systems are those that combine both of the above ways to deal with operational data in a more concise manner. Our goal is to make accurate recommendations with a minimum of computing effort.

## 1.1 PROBLEM STATEMENT

The main purpose of this project is to build a movie recommendation system that recommends movies based on users choice of selection making it easier for the user to get the desired movies based on his/her preference.

## 1.2 OBJECTIVE

- The Movie Recommendation System is a tool that allows users to group together people who have similar interests. A recommendation system's primary goal is to find material that will be of interest to a person. Furthermore, it takes into account a variety of characteristics in order to build individualized lists of helpful and interesting material for each user/individual.
- The objective is **to create recommendations of the movies, based on their similarity scores with the best-rated movie**. Based on the similarity scores, the recommender system would be able to identify the top 30 movies having the highest similarity scores and present them as the top 30 recommendations

## 1.3 MOTIVATION

Similarly, a movie recommendation system provides a level of comfort and personalization that helps the user interact better with the system and watch movies that cater to his needs. **Providing this level of comfort to the user** was our primary motivation in opting for the movie recommendation system as our BE Project.

## 1.4 SCOPE

Recommender systems can be a very powerful tool in a company's arsenal, and future developments are going to increase business value even further. Some of the applications include being able to anticipate seasonal purchases based on recommendations, determine important purchases, and give better recommendations to customers which can increase retention and brand loyalty. Movie recommendation system scope is not limited to entertainment, but also information sharing.



## **1.5.SOFTWARE AND HARDWARE REQUIREMENTS**

### **1.5.1 Software Requirements**

- Jupyter NoteBook
- Language used: Python

### **1.5.2 Hardware Requirements**

- Operating System - Windows 7 or above or Linux
- 4GB RAM Minimum
- Processor - Intel i3 or above

## CHAPTER-2

### LITERATURE SURVEY

#### 2.1 SURVEY OF MAJOR AREA RELEVANT TO PROJECT

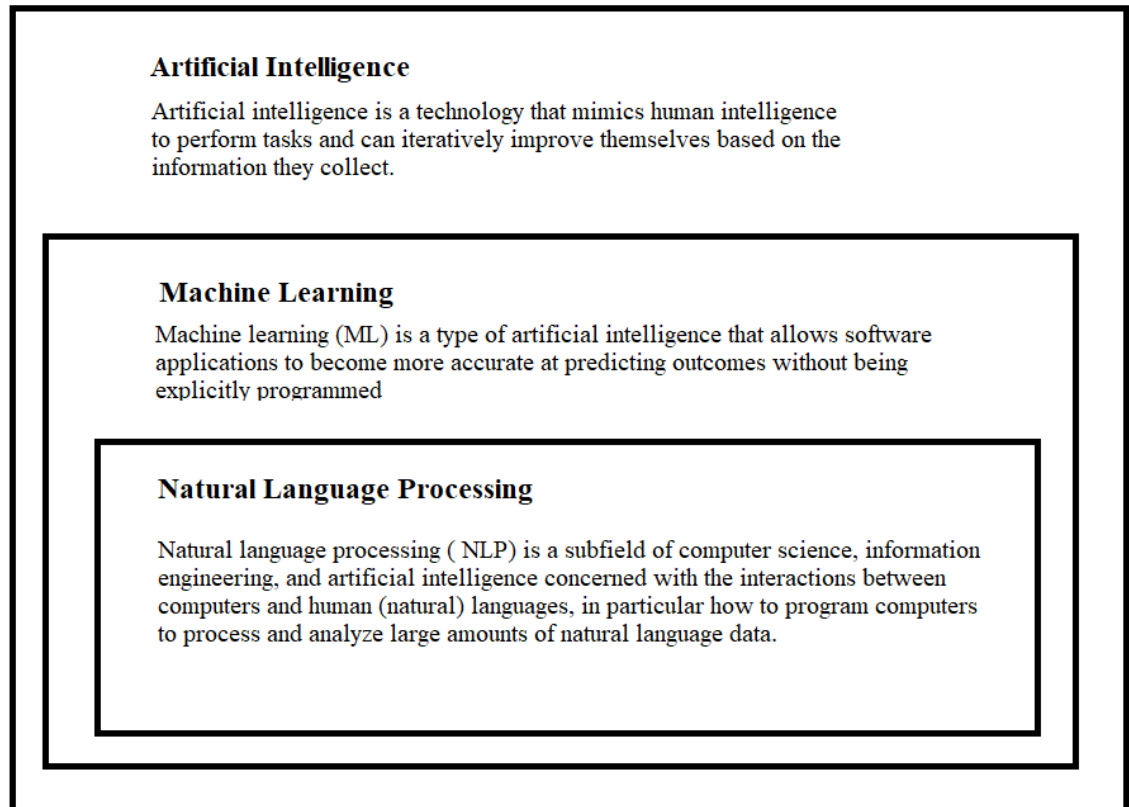


Fig.2.1. Broad areas for building Machine learning projects.

##### 2.1.1 Machine learning

Machine Learning is a field of study concerned with building systems or programs which have the ability to learn without being explicitly programmed. Machine learning systems take in huge amounts of data and learn patterns and labels from that, to basically predict information on never-seen-before data.

##### Machine learning process

###### Collection of Data:

First step is to collect the most appropriate dataset to solve the problem.

### **Data Exploration:**

Now the next step is to explore the data that is used for solving the problem.

Here the task is to understand:

- whether the data contains any missing values
- how to treat the missing values
- descriptive statistics
- data visualization of all the important features
- correlation
- understanding the relationship between the features and labels

### **Data Preprocessing**

The data preprocessing techniques in machine learning can be broadly segmented into two parts: Data Cleaning and Data Transformation.

The steps in data preprocessing in machine learning are:

- Consolidation after acquisition of the data
- Data Cleaning:
  - Convert the data types if any mismatch present in the data types of the variables
  - Change the format of the date variable to the required format
  - Replace the special characters and constants with the appropriate values
- Detection and treatment of missing values
- Treating for negative values, if any present depending on the data
- Outliers detection and treatment
- Transformation of variables
- Creation of new derived variables
- Scale the numerical variables
- Encode the categorical variables
- Split the data into training, validation, and test set

### **Model Selection**

Selecting the machine learning model and training the model to create a baseline model

### **Model Evaluation**

Model gets evaluated based on the test set and various performance metrics.

### **Model Tuning**

As per the evaluation, we tweak the model by changing some hyperparameters or even going back and adding more data

### **Model Predictions**

When the model starts performing well during tests, it is then saved and deployed to be consumed – making predictions on new incoming data

## **2.2 RELATED WORK ON MOVIE RECOMMENDATION SYSTEM USING COSINE SIMILARITY**

Advancement in technology is reaching new heights every day and due to which we can see enormous growth in information. To deal with such large data we use machine learning that automates analytical model building . The early classification of machine learning is divided into three broad categories: Supervised learning, Unsupervised learning and Reinforcement learning. We use computers to make predictions to help us achieve better results using various computational statistics. Tasks can be performed without being explicitly programmed to do so. It becomes a tedious task to extract the relevant information. Search engines solve the problem to some extent but it does not solve the personalization problem. Recommendation System framework plays a vital role in today's internet surfing, be it buying a product from an e-commerce site or watching a movie on some video-on-demand service. In our everyday life, we depend on recommendations given by other people either by word of mouth or reviews of general surveys. People often use recommender systems over the web to make decisions for the items related to their choice. Recommendation systems are software tools and techniques whose goal is to make useful and sensible recommendations to a collection of users for items or products that might interest them. In other words, the recommender system or recommendation systems belongs to a class of information filtering system that aims at predicting the 'preference' or 'rating' given to an item. Recommendation systems are primarily using three approaches [6]. In content-based filtering, we do profiling based on what type of content any user is interested in and using the collected information, it recommends items. Another one is collaborative filtering, where we make clusters of similar users

and use that information to make recommendations. Hybrid systems are the one which takes into account both above stated approaches to deal with operational data more concisely. Our goal is to provide accurate recommendations with less computational complexity.

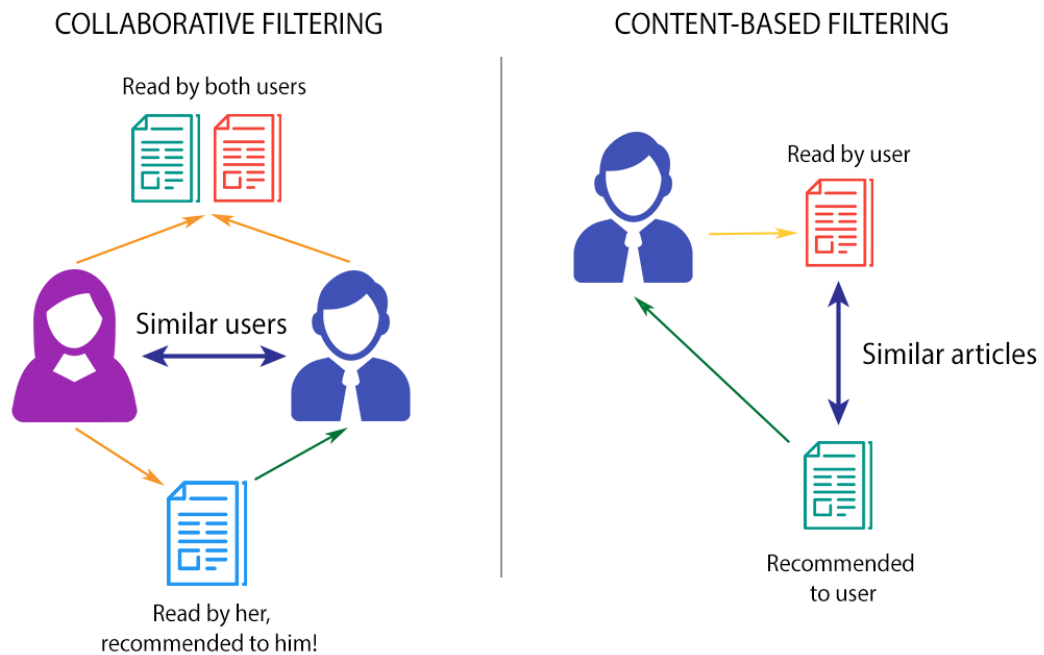


Fig.2.2.1 Content Based Filtering & Collaborative Filtering

## 2.3 TECHNIQUES AND ALGORITHMS APPLICABLE

### Models

#### Cosine Similarity

- Cosine similarity is a metric used to measure how similar two items are. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The output value ranges from 0–1.
- The Movie Recommendation System uses Cosine similarity to get the best possible results upon giving the inputs. Cosine similarity is a measurement that quantifies the similarity between two or more vectors. It is the cosine of the angle between vectors. The vectors are typically non-zero and are wit

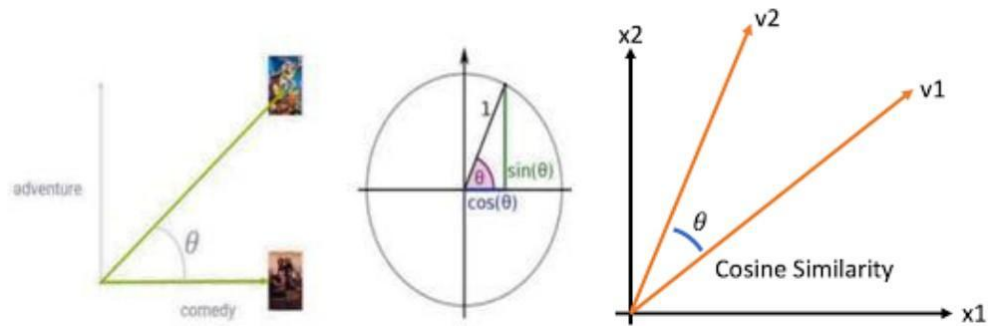


Fig.2.3.1 Cosine Similarity

## Vectorization

### TF-IDF

- Term frequency-inverse document frequency is a text vectorizer that transforms the text into a usable vector. It combines 2 concepts, Term Frequency (TF) and Document Frequency (DF).
- The term frequency is the number of occurrences of a specific term in a document. Term frequency indicates how important a specific term is in a document.
- Document frequency is the number of documents containing a specific term. Document frequency indicates how common the term is.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{ij}$  = number of occurrences of  $i$  in  $j$   
 $df_i$  = number of documents containing  $i$   
 $N$  = total number of documents

### Count Vectorization

- CountVectorizer is used to convert a collection of text documents to a vector of term/token counts.

- It denotes frequency of words occurring in the document.

### **Libraries Used**

**SciKit-Learn:** It is a useful and robust library for machine learning in python. It provides efficient tools and statistical modeling including classification, regression, clustering etc.

**Numpy:** As Mathematics is the foundation for Machine Learning ,there are many Mathematical tasks there to perform in our project .Numpy will be used to perform Mathematical operations .It will be used to perform data analysis

**Pandas:** We will use Pandas for Data Manipulation and Analysis . It provides many functions and methods to speed up the data analysis process. Pandas correlation method can be used to plot the correlation matrix to look for the presence of any strong correlations between any of the four personality features.

## **2.4 APPLICATIONS**

This project can be used in

### **Business**

Predicting Personality is a practical, real-world playbook for any individual or business whose success hinges on the ability to communicate effectively and build teams. Ability to predict a user's personality traits can help to build many customized services or products.

### **Friend recommendation in Social Media**

For making friends in social networks, there remains an underlying friend recommendation framework which suggests friends to the users. However, most of the existing friend recommendation frameworks consider only the number of mutual friends, geo-location, mutual interests etc. to recommend one person as a friend to another. But, in real life, people, who have similar personalities, tend to become friends with each other.Hence it will be useful in friend recommendations.

### **Movie/Music recommendation**

Day by Day Internet users are increasing and many people are watching movies and listening to music through OTT platforms . Through personality prediction we can recommend similar music/movies to persons having similar personality traits. If two persons have the same personality, previous searches of one person can be recommended to the other .

### **Marketing and Advertising**

Users prefer the interfaces which most closely resemble their own personality. This signifies the need to predict personality traits and in turn prepare a personality-oriented interface to make users more receptive. So if we can analyze and categorize the personality traits of the user, then based on it personalized features can be provided to the individual as an interface. This will help marketing agents in building good rapport with the customers.

### **2.5 SUMMARY**

A Movie Recommendation System has been built using cosine similarity, which on giving the inputs gives more accurate results and the complexity is low too. In the realm of the internet, recommendation systems have become the most important source of useful and trustworthy information. Simple ones take into account one or a few parameters, whilst more complicated ones employ multiple parameters to filter the results and make them more user-friendly. A good movie recommendation system can be constructed using advanced deep learning and various filtering approaches such as collaborative filtering and hybrid filtering. This might be a significant step forward in the development of this model, as it will not only become more efficient to use, but will also boost the business value.



## CHAPTER-3

### SYSTEM DESIGN

#### 3.1.BLOCK DIAGRAM

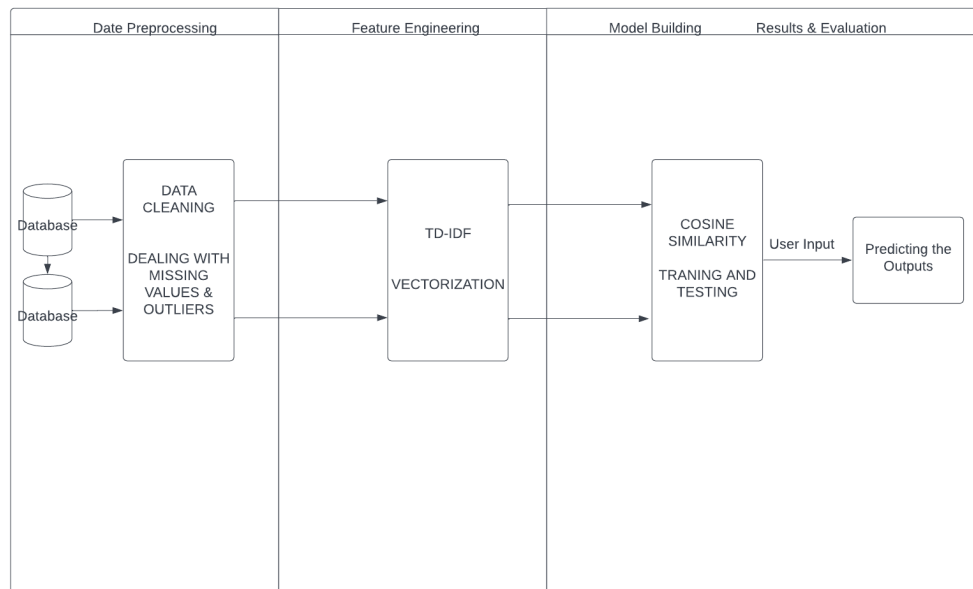


Fig.3.1.1. System Architecture

#### 3.2.DESRIPTION OF EACH MODULE

##### 3.2.1 Dataset

We have used the Movie Recommendation System Dataset from Kaggle. It provides various attributes like genre,budget,director,cast in a table of 4800 rows and 24 columns. This gives us total labels of 1,15,000 points.

##### 3.2.2 Data Analysis

The dataset is quite skewed and is not uniformly distributed among the 4800 movie types. For example, the most common label, en, occurs 4673 times whereas the least frequent,but homepage only occurs 1610 times. When training on the data in its original form, the model tends to overfit on the predominant type(s) while underperforming on the others.Since the classes are heavily imbalanced we have divided the single type feature into five features.

### 3.2.3 Data Preprocessing

Different pre-processing techniques have been exploited for more exploration of personality from the text.

The techniques include:

Cleaning the Data

Posts will be converted into lower case, and punctuations will be replaced by spaces.

Words with one to two character lengths will be dropped.

The cleaned data that has been generated after executing the above steps is Lemmitized using Vectorization .

### 3.2.4 Feature Extraction

**Term Frequency Inverse Document Frequency:** The TF-IDF score is useful in adjusting the weight between most regular or general words and less ordinarily utilized words. Term frequency figures the frequency of every token in the text; however, this frequency is balanced by frequency of that token in the entire dataset. TF-IDF value shows the significance of a token in a text of whole dataset

### 3.2.5 Cosine Similarity

We have built,trained and tested various models on the dataset.

Model is:

Cosine Similarity.

Cosine similarity **measures the similarity between two vectors of an inner product space**. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis.

## **CHAPTER-4**

### **IMPLEMENTATION**

#### **4.1 ENVIRONMENTAL SETUP**

##### **Jupyter Notebook**

Jupyter notebooks basically provide an interactive computational environment for developing Python based Data Science applications. We have used anaconda distribution to launch the Jupyter Notebook. The Anaconda can be used to simplify package management and deployment. It includes more than 300 data science packages that are suitable for Windows, Linux, and MacOS.

#### **4.2 IMPLEMENTATION OF EACH MODULE**

Our CSV file contains a total of 4802 movies and 24 columns: 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 and director (sigh!).

Among all these different features, the ones we are interested in to find the similarity for making the next recommendation are the following:

keywords, cast, genres and director

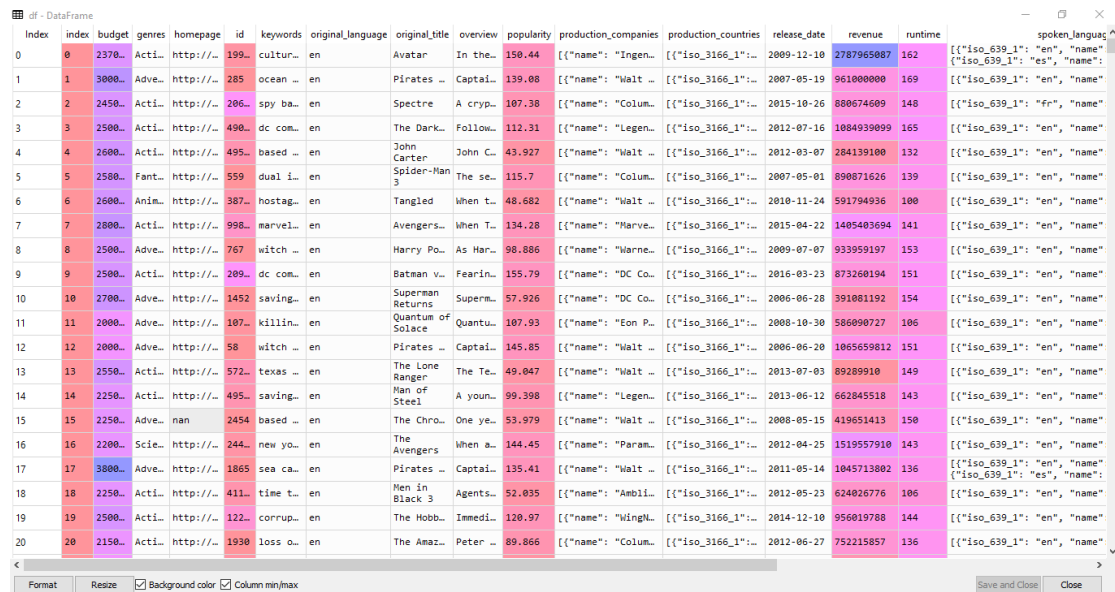
A user who likes a horror movie will most probably like another horror movie. Some users may like seeing their favorite actors in the cast of the movie. Others may love movies directed by a particular person. Combining all of these aspects, our shortlisted 4 features are sufficient to train our recommendation algorithm.

##### **4.2.1 Coding:**

We will import the two important libraries for data analysis and manipulation; pandas and numpy. We will also import Scikit-learn's CountVectorizer, used to convert a collection of text documents to a vector of term/token counts.

Lastly, we will import the `cosine_similarity` from `sklearn`, as the metric of our similarity matrix (which will be discussed in detail later).

We will read our CSV file into a dataframe `df`, which can then be accessed in the variable explorer of our Python IDE.



The screenshot shows a variable explorer window titled 'df - DataFrame'. It displays a table with 21 columns and 21 rows of movie data. The columns are: Index, index, budget, genres, homepage, id, keywords, original\_language, original\_title, overview, popularity, production\_companies, production\_countries, release\_date, revenue, runtime, and spoken\_language. The rows represent different movies, with columns like 'index', 'budget', 'genres', 'id', 'keywords', 'original\_language', 'original\_title', 'overview', 'popularity', 'production\_companies', 'production\_countries', 'release\_date', 'revenue', 'runtime', and 'spoken\_language' containing specific data for each movie.

CSV loaded into a Dataframe

#### 4.2.2 Features list:-

We'll build a list of the features we're going to use. As previously said, we will only employ the characteristics that are most relevant to our problem. As a result, we'll focus on keywords, cast, genres, and director.

Furthermore, we will perform some data preprocessing and replace any rows with NaN values with a space/empty string, ensuring that the function does not generate an error. The for loop was used to perform the pre-processing.

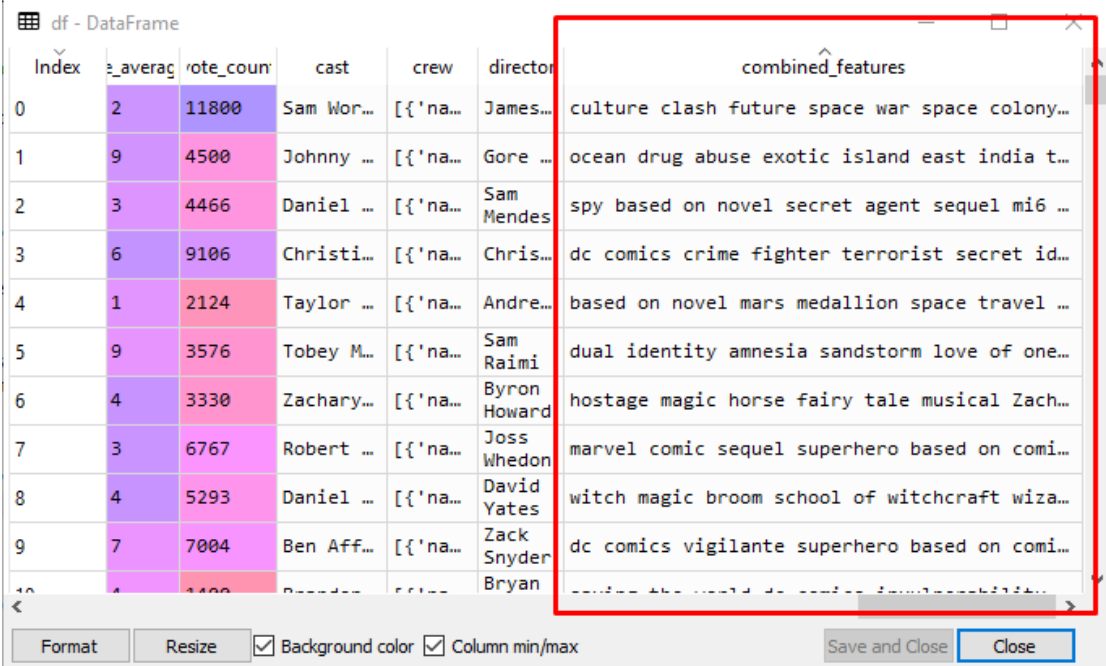
```
features = ['keywords', 'cast', 'genres', 'director']
for feature in features:
    df[feature] = df[feature].fillna('')
```

#### 4.2.3 Combining relevant features into a single feature:-

The function combined features will be defined next. The function will take all of our useful features (keywords, cast, genres, and director) from their respective rows and combine them into a single string.

```
def combined_features(row):
    return row['keywords']+" "+row['cast']+"
"+row['genres']+" "+row['director']
df["combined_features"] = df.apply(combined_features,
axis =1)
```

To our existing dataframe (df), we'll add a new column called combined features and apply the above function to each row (axis = 1). At the conclusion of the dataframe, there will be an extra column with rows of the combined features.



Index	average	vote_count	cast	crew	director	combined_features
0	2	11800	Sam Wor...	[{'na...	James...	culture clash future space war space colony...
1	9	4500	Johnny ...	[{'na...	Gore ...	ocean drug abuse exotic island east india t...
2	3	4466	Daniel ...	[{'na...	Sam Mendes	spy based on novel secret agent sequel mi6 ...
3	6	9106	Christi...	[{'na...	Chris...	dc comics crime fighter terrorist secret id...
4	1	2124	Taylor ...	[{'na...	Andre...	based on novel mars medallion space travel ...
5	9	3576	Tobey M...	[{'na...	Sam Raimi	dual identity amnesia sandstorm love of one...
6	4	3330	Zachary...	[{'na...	Byron Howard	hostage magic horse fairy tale musical Zach...
7	3	6767	Robert ...	[{'na...	Joss Whedon	marvel comic sequel superhero based on comi...
8	4	5293	Daniel ...	[{'na...	David Yates	witch magic broom school of witchcraft wiza...
9	7	7004	Ben Aff...	[{'na...	Zack Snyder	dc comics vigilante superhero based on comi...
10	4	1100	Ben Aff...	[{'na...	Bryan	...the world de...

The Combined Features column in our dataframe

#### 4.2.4 Extracting Features:-

The sklearn.feature\_extraction module can be used to extract features in a format supported by machine learning algorithms from datasets consisting of formats such as text and image. We will use CountVectorizer's fit.transform to count the

number of texts and we will print the transformed matrix `count_matrix` into an array for better understanding.

```
cv = CountVectorizer()
count_matrix = cv.fit_transform(df["combined_features"])
print("Count Matrix:", count_matrix.toarray())
```

#### 4.2.5 Using Cosine similarity:-

Cosine similarity is a metric used to measure how similar two items are.

Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The output value ranges from 0–1.

0 means no similarity, whereas 1 means that both the items are 100% similar.

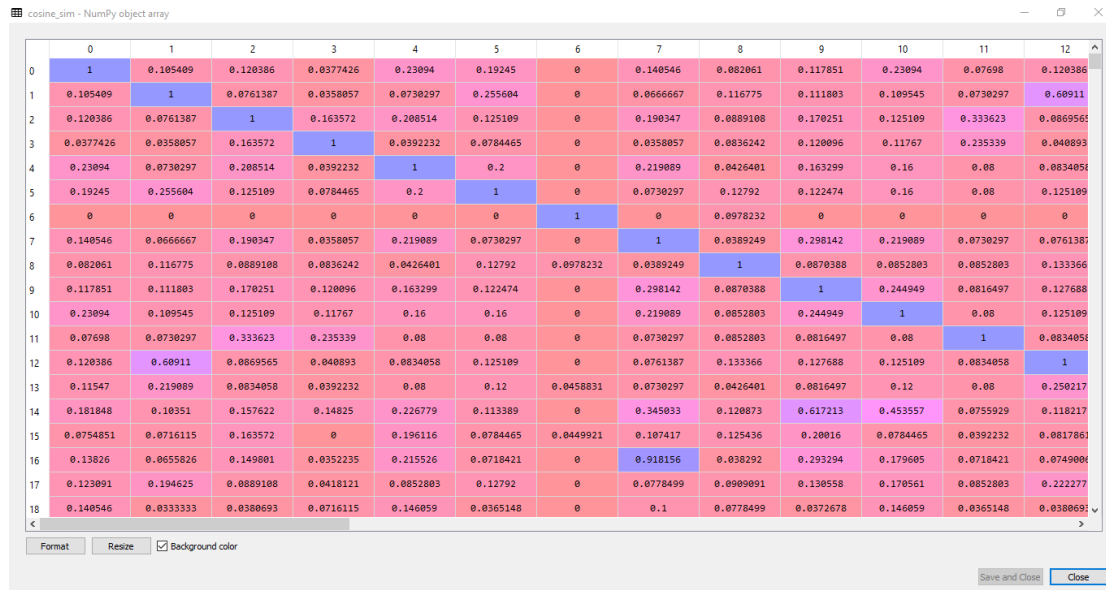
The python Cosine Similarity or cosine kernel, computes similarity as the normalized dot product of input samples X and Y. We will use the sklearn `cosine_similarity` to find the  $\cos \theta$  for the two vectors in the count matrix.

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

The `cosine_sim` matrix is a numpy array with calculated cosine similarity between each movie. As you can see in the image below, the cosine similarity of movie 0 with movie 0 is 1; they are 100% similar (as should be).

Similarly the cosine similarity between movie 0 and movie 1 is 0.105409 (the same score between movie 1 and movie 0 — order does not matter).

Movies 0 and 4 are more similar to each other (with a similarity score of 0.23094) than movies 0 and 3 (score = 0.0377426). The diagonal with 1s suggests what the case is, each movie 'x' is 100% similar to itself!



Cosine Similarity Matrix

#### 4.2.6 Content Users Like:-

The next step is to take as input a movie that the user likes in the `movie_user_likes` variable. Since we are building a content based filtering system, we need to know the users' likes in order to predict a similar item.

```
movie_user_likes = "Dead Poets Society"

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

movie_index = get_index_from_title(movie_user_likes)
```

Next, I will build a function to get the index from the name of this movie. The index will be saved in the `movie_index` variable.

Name	Type	Size	Value
cosine_sim	Array of float64	(4803, 4803)	Min: 0.0 Max: 1.0000000000000007
count_matrix	sparse.csr.csr...	1	csr_matrix object of scipy.sparse.csr module
cv	feature_extrac...	1	CountVectorizer object of sklearn...
df	DataFrame	(4803, 25)	Column names: index, budget, genre...
feature	str	1	director
features	list	4	['keywords', 'cast', 'genres', 'director']
i	int	1	51
movie	tuple	2	(2799, 0.14673479641335552)
movie_index	int64	1	2453
movie_user_likes	str	1	Dead Poets Society
similar_movies	list	4803	[(0, 0.0), (1, 0.0), (2, 0.0), (3, ...
sorted_similar_movies	list	4803	[(2453, 0.9999999999999993), (3250...

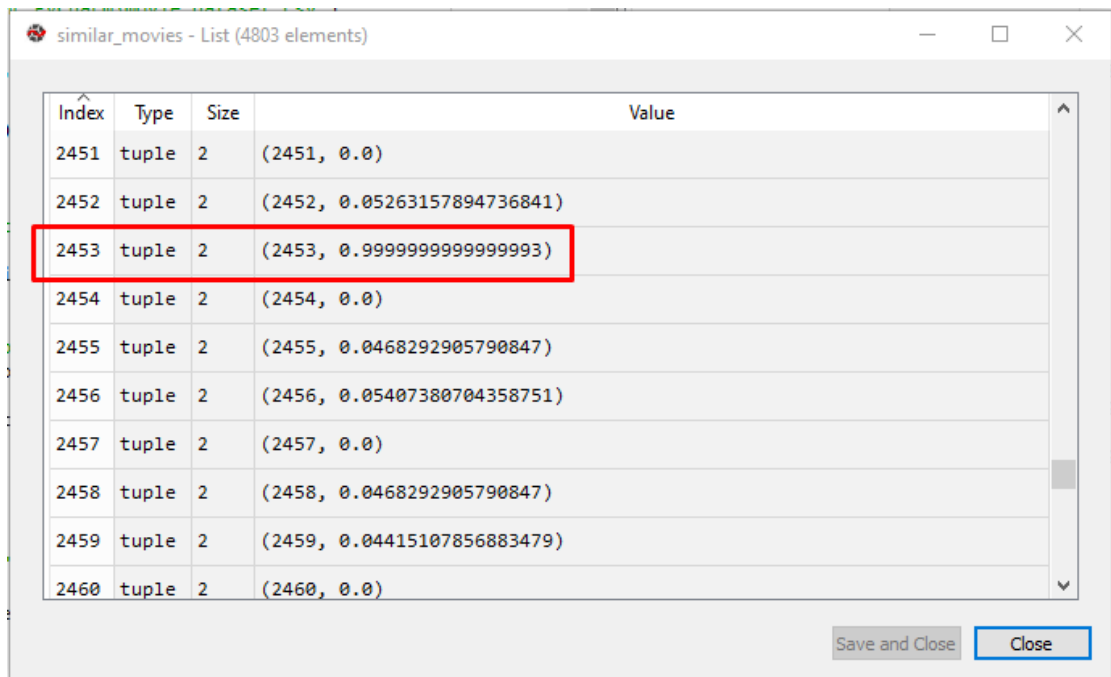
Movie Index variable of the movie User likes

#### 4.2.7 Generating the similar Movie Matrix:-

Next we will generate a list of similar movies. We will use the movie\_index of the movie we have given as input movie\_user\_likes. The enumerate() method will add a counter to the iterable list cosine\_sim and return it in the form of a list similar\_movies with the similarity score of each index.

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





Index	Type	Size	Value
2451	tuple	2	(2451, 0.0)
2452	tuple	2	(2452, 0.05263157894736841)
2453	tuple	2	(2453, 0.9999999999999993)
2454	tuple	2	(2454, 0.0)
2455	tuple	2	(2455, 0.0468292905790847)
2456	tuple	2	(2456, 0.05407380704358751)
2457	tuple	2	(2457, 0.0)
2458	tuple	2	(2458, 0.0468292905790847)
2459	tuple	2	(2459, 0.04415107856883479)
2460	tuple	2	(2460, 0.0)

Similar Movies list

#### 4.2.8 Sorting the Similar Movies in Descending order:-

Next step is to sort the movies in the list `similar_movies`. We have used the parameter `reverse=True` since we want the list in the descending order, with the most similar item at the top.

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

The `sorted_similar_movies` will be a list of all the movies sorted in descending order with respect to their similarity score with the input movie `movie_user_likes`.

As can be seen in the image below, the most similar one with a similarity score of 0.9999999999999993 is at the top most, with its index number 2453 (the movie is 'Dead Poets Society' which we gave as input, makes sense, right?).

Index	Type	Size	Value
0	tuple	2	(2453, 0.9999999999999999)
1	tuple	2	(3250, 0.21677749238102995)
2	tuple	2	(905, 0.21052631578947364)
3	tuple	2	(2975, 0.2051956704170308)
4	tuple	2	(825, 0.19564639521780736)
5	tuple	2	(1507, 0.19134594929397594)
6	tuple	2	(4273, 0.1908854288927333)
7	tuple	2	(1774, 0.1873171623163388)
8	tuple	2	(4488, 0.18394180184548975)
9	tuple	2	(3526, 0.1835325870964494)

Sorted Similar Movies List with the Similarity Score

#### 4.2.9 Printing Similar Movies:

Now, here comes the last part of the project, which is to print the names of the movies similar to the one we have given as input to the system through the `movie_user_likes` variable.

As seen in the `sorted_similar_movies` list, the movies are sorted by their index number. Printing the index number will be of no use to us, so we will define a simple function that takes the index number and converts it into the movie title as in the dataframe.

Index Number → Movie Title

## CHAPTER-5

### TESTING AND RESULTS

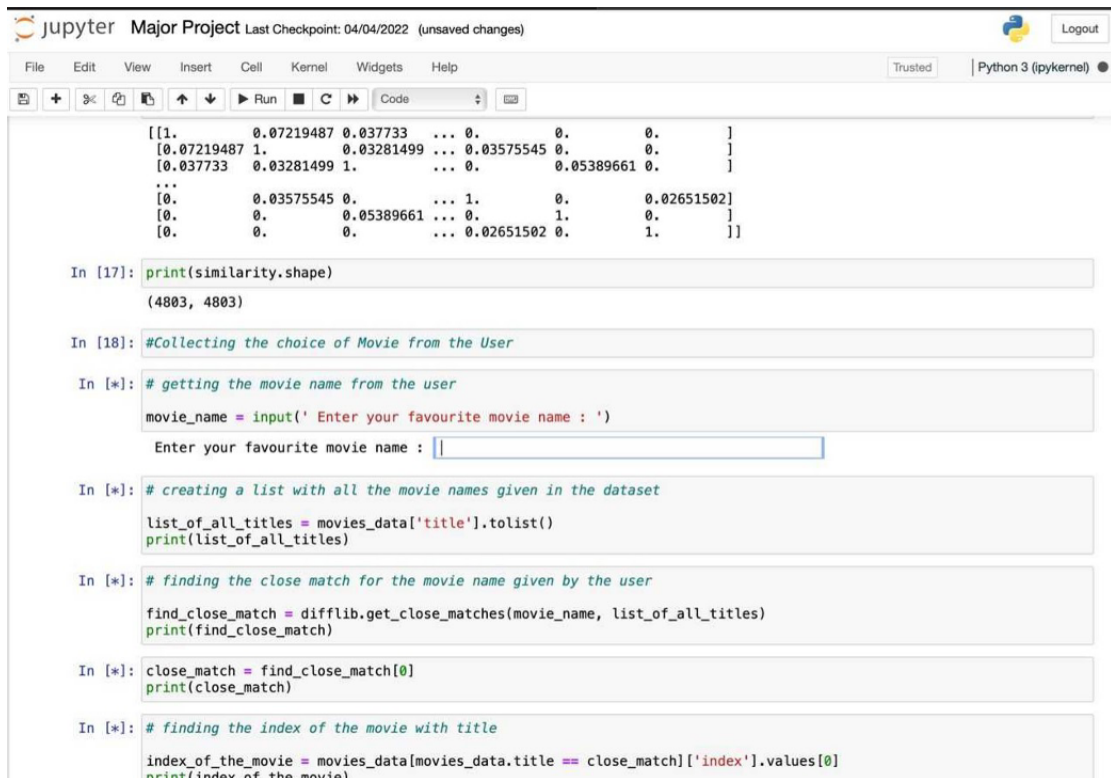
#### DATASET

We have used the Movie Recommendation System Dataset from Kaggle. It provides various attributes like genre,budget,director,cast in a table of 4800 rows and 24 columns. This gives us total labels of 1,15,000 points.The dataset is quite skewed and is not uniformly distributed among the 4800 movie types. For example, the most common label, en, occurs 4673 times whereas the least frequent,but homepage only occurs 1610 times. When training on the data in its original form, the model tends to overfit on the predominant type(s) while underperforming on the others.Since the classes are heavily imbalanced we have divided the single type feature into five features.

#### Testing and Result:-

Now comes the application. Use the steps above to code your own recommender systems and run the code by giving a movie you like to the movie\_user\_likes.

We have given “Iron Man”, and it prints me the following similar movies:



```

jupyter Major Project Last Checkpoint: 04/04/2022 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

[[1. 0.07219487 0.037733 ... 0. 0. 0.]
 [0.07219487 1. 0.03281499 ... 0.03575545 0. 0.]
 [0.037733 0.03281499 1. ... 0. 0.05389661 0.]
 ...
 [0. 0.03575545 0. ... 1. 0. 0.02651502]
 [0. 0. 0.05389661 ... 0. 1. 0.]
 [0. 0. 0. ... 0.02651502 0. 1.]]

In [17]: print(similarity.shape)
(4803, 4803)

In [18]: #Collecting the choice of Movie from the User

In [*]: # getting the movie name from the user
movie_name = input(' Enter your favourite movie name : ')
Enter your favourite movie name : 

In [*]: # creating a list with all the movie names given in the dataset
list_of_all_titles = movies_data['title'].tolist()
print(list_of_all_titles)

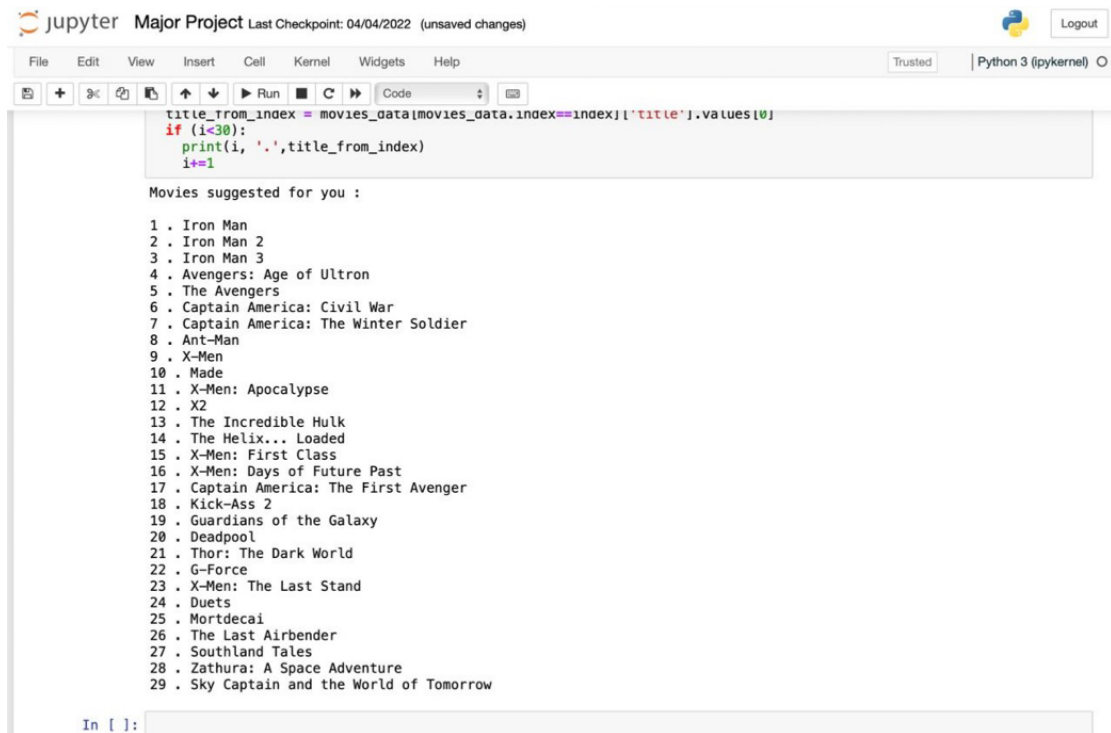
In [*]: # finding the close match for the movie name given by the user
find_close_match = difflib.get_close_matches(movie_name, list_of_all_titles)
print(find_close_match)

In [*]: close_match = find_close_match[0]
print(close_match)

In [*]: # finding the index of the movie with title
index_of_the_movie = movies_data[movies_data.title == close_match]['index'].values[0]
print(index of the movie)

```

## 5.1 Test Case:-



The image shows a Jupyter Notebook interface. At the top, the header includes the Jupyter logo, the text "Major Project", the last checkpoint "04/04/2022", and "(unsaved changes)". On the right, there is a "Logout" button. Below the header is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". A toolbar contains icons for file operations, a "Run" button, and a "Code" dropdown. The main area contains a code cell with the following Python code:

```
title_from_index = movies_data[movies_data.index==index]['title'].values[0]
if (i<30):
    print(i, '.',title_from_index)
    i+=1
```

Below the code cell, the output is displayed. It starts with the text "Movies suggested for you :", followed by a numbered list of 29 movie titles:

- 1 . Iron Man
- 2 . Iron Man 2
- 3 . Iron Man 3
- 4 . Avengers: Age of Ultron
- 5 . The Avengers
- 6 . Captain America: Civil War
- 7 . Captain America: The Winter Soldier
- 8 . Ant-Man
- 9 . X-Men
- 10 . Made
- 11 . X-Men: Apocalypse
- 12 . X2
- 13 . The Incredible Hulk
- 14 . The Helix... Loaded
- 15 . X-Men: First Class
- 16 . X-Men: Days of Future Past
- 17 . Captain America: The First Avenger
- 18 . Kick-Ass 2
- 19 . Guardians of the Galaxy
- 20 . Deadpool
- 21 . Thor: The Dark World
- 22 . G-Force
- 23 . X-Men: The Last Stand
- 24 . Duets
- 25 . Mortdecai
- 26 . The Last Airbender
- 27 . Southland Tales
- 28 . Zathura: A Space Adventure
- 29 . Sky Captain and the World of Tomorrow

At the bottom of the notebook, there is an input prompt "In [ ]:" followed by a text box.

## **CHAPTER-6**

### **CONCLUSION AND FUTURE ENHANCEMENTS**

We have illustrated the modelling of a movie recommendation system by making the use of content-based filtering in the movie recommendation system.

Recommendation systems have become the most essential fount of a relevant and reliable source of information in the world of internet. Simple ones consider one or a few parameters while the more complex ones make use of more parameters to filter the results and make it more user friendly. With the inclusion of advanced deep learning and other filtering techniques like collaborative filtering and hybrid filtering a strong movie recommendation system can be built. This can be a major step towards the further development of this model as it will not only become more efficient to use but also increase the business value even further.

#### **Future Work**

- Moving forward, we hope to incorporate richer data and features to allow for a stronger understanding of the input text as well as improved performance. Our dataset was quite limited and didn't include any other user information or metadata for posts, and that additional data would be hugely important for recommending accurate movies.
- Furthermore, we will add a new algorithm that will work alongside cosine similarity which will give more accurate results.

## **REFERENCES**

- 1.Movie Recommendation System using Cosine Similarity and  
...<https://www.ijeat.org> > uploads > papers
- 2.<https://ieeexplore.ieee.org/document/9544794>

## Appendix

### Data Cleaning and Lemmatizing

```
# importing dependencies here
import numpy as np
import pandas as pd
import os
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
import re
import nltk
from nltk.corpus import stopwords
nltk.download("stopwords")
from nltk.stem import WordNetLemmatizer
import time

# reading the dataset
df = pd.read_csv(os.path.join("../", "data", "mbti_1.csv"))
personality_data, df_holdout = train_test_split(
    df, random_state=42, test_size=0.01, stratify=df["type"]
)
personality_data.to_csv(os.path.join("../", "data",
    "personality_data.csv"), index=False)
df_holdout.to_csv(os.path.join("../", "data", "df_holdout.csv"),
    index=False)

#reading the dataset that will be used for training and testing model
personality_data = pd.read_csv(os.path.join("../", "data",
    "personality_data.csv"))

# checking for missing values
personality_data.isnull().sum()

# to handle the class imbalance better, converting the 16 classes
into 4 more balanced classes
personality_data["is_Extrovert"] = personality_data["type"].apply(
    lambda x: 1 if x[0] == "E" else 0
)
personality_data["is_Sensing"] = personality_data["type"].apply(
    lambda x: 1 if x[1] == "S" else 0
)
personality_data["is_Thinking"] = personality_data["type"].apply(
    lambda x: 1 if x[2] == "T" else 0
)
personality_data["is_Judging"] = personality_data["type"].apply(
    lambda x: 1 if x[3] == "J" else 0
)

#rearranging the data frame columns
personality_data = personality_data[
    ["type", "is_Extrovert", "is_Sensing", "is_Thinking",
    "is_Judging", "posts"]
]

# converting posts into lower case
personality_data["clean_posts"] =
personality_data["posts"].str.lower()
# replacing ||| with space
personality_data["clean_posts"] =
personality_data["clean_posts"].str.replace(
    re.compile(r"\\|\\|\\|"), " "
)
personality_data["clean_posts"] =
personality_data["clean_posts"].str.replace(
```

```

re.compile(r"https?:\/\/(www)?(?:[A-Za-z_0-9-]+)([\S]*)"), ""
)
# dropping emails
personality_data["clean_posts"] =
personality_data["clean_posts"].str.replace(
    re.compile(r"\S+@\S+"), ""
)
# dropping punctuations
personality_data["clean_posts"] =
personality_data["clean_posts"].str.replace(
    re.compile(r"^[a-z\s]"), " "
)
# dropping MBTIs mentioned in the posts. There are quite a few
mentions of these types in these posts.
mbti = personality_data["type"].unique()
for type_word in mbti:
    personality_data["clean_posts"] =
personality_data["clean_posts"].str.replace(
    type_word.lower(), ""
)
#Lemmatizing
lemmatizer = WordNetLemmatizer()
personality_data["clean_posts"] =
personality_data["clean_posts"].apply(
    lambda x: " ".join(
        [
            lemmatizer.lemmatize(word)
            for word in x.split(" ")
            if word not in stopwords.words("english")
        ]
    )
)

```

### Sentiment Analysis and POS Tagging

```

analyzer = SentimentIntensityAnalyzer()
nlp_sentiment_score = []
for post in personality_data["clean_posts"]:
    score = analyzer.polarity_scores(post)
    nlp_sentiment_score.append(score)
# segregating the individual sentiment scores - compound, positive,
negative and neutral
personality_data["compound_sentiment"] = [
    score["compound"] for score in nlp_sentiment_score
]
personality_data["pos_sentiment"] = [score["pos"] for score in
nlp_sentiment_score]
personality_data["neg_sentiment"] = [score["neg"] for score in
nlp_sentiment_score]
personality_data["neu_sentiment"] = [score["neu"] for score in
nlp_sentiment_score]
#Sentiment scores have negative values that Naive Bayes can't handle.
So scaling it.
min_max_scaler = MinMaxScaler()
personality_data["compound_sentiment"] = min_max_scaler.fit_transform(
    np.array(personality_data["compound_sentiment"]).reshape(-1, 1)
)
personality_data["pos_sentiment"] = min_max_scaler.fit_transform(
    np.array(personality_data["pos_sentiment"]).reshape(-1, 1)
)
personality_data["neg_sentiment"] = min_max_scaler.fit_transform(
    np.array(personality_data["neg_sentiment"]).reshape(-1, 1)
)

```



```

)
personality_data["neu_sentiment"] = min_max_scaler.fit_transform(
    np.array(personality_data["neu_sentiment"]).reshape(-1, 1)
)
# creating a tag_posts column that will have each post as a separate
list in a row. tag_posts will be a list of 50 lists.
# replacing urls with domain name
personality_data["tag_posts"] = personality_data["posts"].str.replace(
    re.compile(r"https?:\/\/(www)?(?:[A-Za-z_0-9-]+)([\S])"),
    lambda match: match.group(2),
)
# replacing ||| with space
personality_data["tag_posts"] = [
    post for post in personality_data["tag_posts"].str.split("\\|\\|\\|")
]
personality_data["tagged_words"] =
personality_data["tag_posts"].apply(
    lambda x: [nltk.pos_tag(word_tokenize(line)) for line in x]
)

# creating list of unique POS tags
tag_set = set()

for i, data in personality_data["tagged_words"].iteritems():
    for tup in data[0]:
        tag_set.add(tup[1])
tag_list = list(tag_set)
def pos_cat(x, tag):
    return [len([y for y in line if y[1] == tag]) for line in x]

for col in tag_list:
    personality_data["POS_" + col + "_mean"] =
personality_data["tagged_words"].apply(
    lambda x: np.mean(pos_cat(x, col))
)
    personality_data["POS_" + col + "_std"] =
personality_data["tagged_words"].apply(
    lambda x: np.std(pos_cat(x, col))
)
# grouping pos tags based on stanford list
tags_dict = {
    "ADJ": ["JJ", "JJR", "JJS"],
    "ADP": ["EX", "TO"],
    "ADV": ["RB", "RBR", "RBS", "WRB"],
    "CONJ": ["CC", "IN"],
    "DET": ["DT", "PDT", "WDT"],
    "NOUN": ["NN", "NNS", "NNP", "NNPS"],
    "NUM": ["CD"],
    "PRT": ["RP"],
    "PRON": ["PRP", "PRP$", "WP", "WP$"],
    "VERB": ["MD", "VB", "VBD", "VBG", "VBN", "VBP", "VBZ"],
    ".": ["#", "$", "'", "(", ")", ",", ".", ":"],
    "X": ["FW", "LS", "UH"],
}
def stanford_tag(x, tag):
    tags_list = [len([y for y in line if y[1] in tags_dict[col]]) for
line in x]
    return tags_list

for col in tags_dict.keys():

```

```

    personality_data[col + "_avg"] =
personality_data["tagged_words"].apply(
    lambda x: np.median(stanford_tag(x, col))
)

```

## ML models

```

#Setting Predictors and target variable
# setting X to clean_posts, compound sentiment score, pos tags and
various other counts
X =
personality_data[["clean_posts", "compound_sentiment", "ADJ_avg", "ADP_av
g", "ADV_avg", "CONJ_avg", "DET_avg", "NOUN_avg", "NUM_avg", "PRT_avg", "PRON
_avg", "VERB_avg", "qm", "em", "colons", "emojis", "word_count", "unique_word
s", "upper", "link_count", "ellipses", "img_count", ]]
# setting y to four target classes -> is_Extrovert, is_Sensing,
is_Thinking, is_Judging
y = personality_data.iloc[:, 1:5]

```

### #Setting up preprocessor for vectorization and selecting best counts and scores

```

# preprocessing steps for selecting best k columns/features from
counts & scores and for vectorizing words
counts_n_scores =
["compound_sentiment", "ADJ_avg", "ADP_avg", "ADV_avg", "CONJ_avg", "DET_av
g", "NOUN_avg", "NUM_avg", "PRT_avg", "PRON_avg", "VERB_avg", "qm", "em", "col
ons", "emojis", "word_count", "unique_words", "upper", "link_count", "ellips
es", "img_count"]
# for selecting k best features from features other than words
best_k_features = make_pipeline(MinMaxScaler(), SelectKBest(f_classif,
k=10))
# setting up preprocessing for TF-IDF vectorizer
preprocessor_tf = ColumnTransformer(
    transformers=[
        (
            "tfidf",
            TfidfVectorizer(min_df=25, max_df=0.85,
stop_words=additional_stopwords),
            "clean_posts",
        ),
        ("selectbest", best_k_features, counts_n_scores),
    ],
    remainder="passthrough",
)
# setting up preprocessing for COUNT vectorizer
preprocessor_ct = ColumnTransformer(
    transformers=[
        (
            "ct_vect",
            CountVectorizer(min_df=25, max_df=0.85,
stop_words=additional_stopwords),
            "clean_posts",
        ),
        ("selectbest", best_k_features, counts_n_scores),
    ],
    remainder="passthrough",
)

```

```

# setting up the personality dictionary for printing scores for each
class
mbti_type = {
    "is_Extrovert": "Extrovert vs Introvert",

```

```

    "is_Sensing": "Sensing vs Intuition",
    "is_Thinking": "Thinking vs Feeling",
    "is_Judging": "Judging vs Perceiving",
}

# function to build the model for predicting each of the 4 target
classes
def build_model(model, X, target, vectorizer_name):

    for col in target.columns:

        print(f"\n{mbti_type[col]}")
        target = y[col]

        X_train, X_test, y_train, y_test = train_test_split(
            X, target, test_size=0.2, random_state=42, stratify=target
        )

        # model training
        model.fit(X_train, y_train)

        # y_hat
        y_pred = model.predict(X_test)

        # y_probability
        y_proba = model.predict_proba(X_test)[: , 1]

        # precision recall score
        average_precision = average_precision_score(y_test, y_proba)

        # model evaluation
        print(
            f"Geometric Mean Score: {geometric_mean_score(y_test,
y_pred, average='weighted'):.2f}"
        )
        print(f"ROC-AUC Score: {roc_auc_score(y_test, y_proba):.2f}")
        print(f"Average Precision-Recall Score:
{average_precision:.2f}")
        print(classification_report_imbalanced(y_test, y_pred))

TF-IDF Logistic Regression
tfidf_logistic_regression = imb_make_pipeline(
    preprocessor_tf, RandomUnderSampler(), LogisticRegressionCV()
)
build_model(tfidf_logistic_regression, X, y, "tfidf")

Count Vectorized Logistic Regression
ct_logistic_regression = imb_make_pipeline(
    preprocessor_ct, RandomUnderSampler(), LogisticRegressionCV()
)
build_model(ct_logistic_regression, X, y, "ct_vect")

TF-IDF Logistic Lasso
tfidf_logistic_regression_lasso = imb_make_pipeline(
    preprocessor_tf, RandomUnderSampler(),
    LogisticRegressionCV(penalty='l1', solver='saga' )
)

build_model(tfidf_logistic_regression_lasso, X, y, "tfidf")

Count Vectorized Logistic Lasso
ct_logistic_regression_lasso = imb_make_pipeline(
    preprocessor_ct, RandomUnderSampler(),
    LogisticRegressionCV(penalty='l1', solver='saga' )
)

```

```

)
build_model(ct_logistic_regression_lasso, X, y, "ct_vect")
TF-IDF Logistic Ridge
tfidf_logistic_regression_ridge = imb_make_pipeline(
    preprocessor_tf, RandomUnderSampler(),
    LogisticRegressionCV(penalty='l2', solver='saga' )
)
build_model(tfidf_logistic_regression_ridge, X, y, "tfidf")
Count Vectorized Logistic Ridge
ct_logistic_regression_ridge = imb_make_pipeline(
    preprocessor_ct, RandomUnderSampler(),
    LogisticRegressionCV(penalty='l2', solver='saga' )
)
build_model(ct_logistic_regression_ridge, X, y, "ct_vect")
TF-IDF Support Vector Classifier
tfidf_svc = imb_make_pipeline(
    preprocessor_tf, RandomUnderSampler(), DenseTransformer(),
    SVC(kernel='linear', probability=True)
)
build_model(tfidf_svc, X, y, "tfidf")
Count Vectorized Support Vector Classifier
ct_svc = imb_make_pipeline(
    preprocessor_ct, DenseTransformer(), RandomUnderSampler(),
    SVC(kernel='linear', probability=True)
)
build_model(ct_svc, X, y, "ct_vect")
TF-IDF Naive Bayes
tfidf_nb = imb_make_pipeline(
    preprocessor_tf, DenseTransformer(), RandomUnderSampler(),
    MultinomialNB(),
)
build_model(tfidf_nb, X, y, "tfidf")
Count Vectorized Naive Bayes
ct_nb = imb_make_pipeline(
    preprocessor_ct, DenseTransformer(), RandomUnderSampler(),
    MultinomialNB(),
)
build_model(ct_nb, X, y, "ct_vect")
TF-IDF Random Forest
tfidf_rf = imb_make_pipeline(
    preprocessor_tf, DenseTransformer(),
    RandomUnderSampler(),
    RandomForestClassifier(n_estimators=100, max_depth=10),
)
build_model(tfidf_rf, X, y, "tfidf")
Count Vectorized Random Forest
ct_rf = imb_make_pipeline(

```

```

preprocesser_ct, RandomUnderSampler(),
RandomForestClassifier(n_estimators=100, max_depth=10),
)
build_model(ct_rf, X, y, "ct_vect")

```

### **Final Model - Logistic Regression with TF-IDF Vectorization**

```

for col in y.columns:

    print(f"\n{mbti_type[col]}")

    target = y[col]

    tfidf_logistic_regression = imb_make_pipeline(
        preprocesser_tf,
        RandomUnderSampler(),
        LogisticRegressionCV()
    )

    # training the data on entire dataset
    tfidf_logistic_regression.fit(X, target)
    # feature importance
    coef = tfidf_logistic_regression[-1].coef_[0]
    word =
    tfidf_logistic_regression[0].named_transformers_["tfidf"].get_feature_
_names()
    word_list = list(zip(word, coef))
    result = pd.DataFrame(word_list, columns=["word",
"coef"]).set_index("word")
    result =
    result.reindex(result.coef.abs().sort_values(ascending=False).index)[
0:21]
    print(result)

    # plotting feature importance
    result["coef"] = result["coef"].apply(lambda x: abs(x))
    result.sort_values("coef", inplace=True)
    result.plot(kind="barh", color="#61BED6", title=mbti_type[col])

    # saving the model
    dump(tfidf_logistic_regression, f"clf_{col}.joblib")

```

### **Final Model Testing**

```

def predict(s):
    return len(s.split(" "))
def predict_e(s):
    X = prep_data(s)
    # loading the 4 models

```

```

    EorI_model = load(os.path.join("..", "models",
"clf_is_Extrovert.joblib"))
    SorN_model = load(os.path.join("..", "models",
"clf_is_Sensing.joblib"))
    TorF_model = load(os.path.join("..", "models",
"clf_is_Thinking.joblib"))
    JorP_model = load(os.path.join("..", "models",
"clf_is_Judging.joblib"))
    # predicting
    EorI_pred = EorI_model.predict(X)
    print("preds", EorI_pred)
    SorN_pred = SorN_model.predict(X)
    print("preds", SorN_pred)
    TorF_pred = TorF_model.predict(X)
    print("preds", TorF_pred)
    JorP_pred = JorP_model.predict(X)
    print("preds", JorP_pred)
    # combining the predictions from the 4 models
    result = combine_classes(EorI_pred, SorN_pred, TorF_pred,
JorP_pred)

    return result[0]

if __name__ == "__main__":
    t = time.time()
    # sample test string. Type ISTP.
    string = "I plugged the data into tableau to see how the different
features or how various mathematical formulas relate to the Weight.
Once I had a few that didn't have a wide distribution, I just started
trying different models, even ones we hadn't gone over yet. There are
a LOT of regression models. I do not like this try everything''
method, it's inefficient and illogical."
    print(predict_e(string))
    print(f"Preprocessing Time: {(time.time() - t):.2f} seconds")

```

## II. Rubrics for Project

Focus Areas	C No	Criterion [c]	Exemplary 4	Satisfactory 3	Developing 2	Unsatisfactory 1
<b>Problem Formulation (PO1, PO2, PO6, PO7, PSO1)</b>	I	<b>Identify/Define Problem</b> Ability to identify a suitable problem and define the project objectives.	Demonstrates a skillful ability to identify / articulate a problem and the objectives are well defined and prioritized.	Demonstrates ability to Identify / articulate a problem and All major objectives are identified.	Demonstrates some ability to identify / articulate a problem that is partially connected to the issues and most major objectives are identified but one or two minor ones are missing or priorities are not established.	Demonstrates minimal or no ability to identify / articulate a problem and many major objectives are not identified.
	II	<b>Collection of Background Information:</b> Ability to gather background Information (existing knowledge, research, and/or indications of the problem)	Collects sufficient relevant background information from appropriate sources, and is able to identify pertinent/critical information;	Collects sufficient relevant background information from appropriate sources;	Collects some relevant background information from appropriate Sources.	Minimal or no ability to collect relevant background information
	III	<b>Define scope of the problem</b> Ability to identify problem scope suitable to the degree considering the impact on society and environment	Demonstrates a skillful ability to define the scope of problem accurately mentioning the relevant fields of engineering precisely. Considers, explains and evaluates the impact of engineering interventions on society and environment.	Demonstrates ability to define problem scope mentioning the relevant fields of engineering broadly. Considers and explains the impact of engineering interventions on society and environment	Demonstrates some ability to define problem scope mentioning some of the relevant fields . Some consideration of the impact of engineering interventions on society and environment.	Demonstrates minimal or no ability to define problem scope and fails to mention relevant fields of engineering. Minimal or no consideration of the impact of engineering interventions on society and environment

<b>Project Design (PO3, PSO1)</b>	IV	<b>Understanding the Design Process and Problem Solving:</b> Ability to explain the design process including the importance of needs, specifications, concept generation and to develop an approach to solve a problem.	Demonstrates a comprehensive ability to understand and explain a design process. Considers multiple approaches to solving a problem, and can articulate reason for choosing solution	Demonstrates an ability to understand and explain a design process. Considers multiple approaches to solving a problem, which is justified and considers consequences.	Demonstrates some ability to understand and explain a design process. Consider a few approaches to solving a problem; don't always consider consequences.	Demonstrates minimal or no ability to understand and explain a design process. Considers a single approach to solving a problem. Does not consider consequences.
<b>Build (PO4, PO5, PSO1)</b>	V	<b>Implementing Design Strategy and Evaluating Final Design:</b> Ability to execute a solution taking into consideration design requirements using appropriate tool (software/hardware);	Demonstrates a skillful ability to execute a solution taking into consideration all design requirements using the most relevant tool.	Demonstrates an ability to execute a solution taking into consideration design requirements using relevant tools.	Demonstrates some ability to execute a solution but not using the most relevant tool.	Demonstrates minimal or no ability to execute a solution. Solution does not directly attend to the problem.
<b>Test &amp; Deploy (PO4, PO5, PSO1)</b>	VI	To evaluate/confirm the functioning of the final design. To deploy the project on the target environment	Demonstrates a skillful ability to evaluate/confirm the functioning of the final design skillfully, with deliberation for further Improvement	Demonstrates an ability to evaluate/confirm the functioning of the final design. The evaluation is complete and has sufficient	Ability to evaluate/confirm the functioning of the final design, but the evaluation lacks depth and/or is incomplete.	Demonstrates minimal or no ability to evaluate/confirm the functioning of the final design.



			after deployment.	depth.		
<b>Ethical responsibility (PO8)</b>	VII	<b>Proper Use of Others' Work:</b> Ability to recognize, understand and apply proper ethical use of intellectual property, copyrighted materials, and research.	Always recognizes and applies proper ethical use of intellectual property, copyrighted materials, and others' research.	Recognizes and applies proper ethical use of intellectual property, copyrighted materials, and others' research.	Some recognition and application of proper ethical use of intellectual property, copyrighted materials, and others' research.	Minimal or no recognition and/or application of proper ethical use of intellectual property, Copyrighted materials, or others' research.
<b>Team Skills (PO9)</b>	VIII	<b>Individual Work Contributions and Time Management :</b> Ability to carry out individual Responsibilities and manage time (estimate, prioritize, establish deadlines/ milestones, follow timeline, plan for contingencies , adapt to change).	Designated jobs are accomplished by deadline; completed work is carefully and meticulously prepared and meets all requirements.	Designated jobs are accomplished by deadline; completed work meets requirements.	Designated jobs are accomplished by deadline; completed work meets most requirements.	Some Designated jobs are accomplished by deadline; completed work meets some requirements.
	IX	<b>Leadership Skills:</b> Ability to lead a team. (i) Mentors and accepts mentoring from others.	Exemplifies leadership skills.	Demonstrates leadership skills.	Demonstrates some leadership skills at times.	Demonstrates minimal or no leadership skills.

		(ii) Demonstrates capacity for initiative while respecting others' roles. (iii) Facilitates others' involvement. (iv) Evaluates team Effectiveness and plans for improvements				
	X	<b>Working with Others:</b> Ability to listen to, collaborate with, and champion the efforts of others.	Skillfully listens to, collaborates with, and champions the efforts of others.	Listens to, collaborates with, and champions the efforts of others.	Sometimes listens to, collaborates with, and champions others' efforts.	Rarely listens to, collaborates with, or champions others' efforts.
<b>Project Presentation (P10)</b>	XI	<b>Technical Writing Skills</b> Ability to communicate the main idea with clarity. Ability to use illustrations properly to support ideas (citations, position on page etc)	Main idea is clearly and precisely stated. Materials are seamlessly arranged in a logical sequence Illustrations are skillfully used to support ideas	Main idea is understandable. Material moves logically forward, Illustrations are properly used to support ideas	Main idea is somewhat understandable. Material has some logical order and is somewhat coherent or easy to follow. Illustrations are for the most part properly used to support ideas	Main idea is difficult to understand. Material has little logical order, and is often unclear, incoherent. Illustrations are used, but minimally support ideas. (not properly cited etc)
	XII	<b>Communication Skills for Oral Reports</b> Ability to present strong key ideas and	Presentation logically and skillfully structured. Key ideas are compelling, and	Presentation has clear structure and is easy to follow. Key ideas are	Presentation has some structure. Key ideas are generally identifiable,	Presentation rambles. Not organized; key ideas are difficult to identify, and are

		supporting details with clarity and concision. Maintain contact with audience, and ability to complete in the allotted time	articulated with exceptional clarity and concision. Introduction, supporting details and summary are clearly evident and memorable, and ascertain the credibility of the speaker. Presentation fits perfectly within time constraints.	clearly and concisely articulated, and are interesting. There is sufficient detail to ascertain speaker's authority, and presentation includes an introduction and summary. Presentation fits within time constraints, though the presenter might have to subtly rush or slow down.	although not very remarkable. Introduction, supporting details and/or summary may be too broad, too detailed or missing. Credibility of the speaker may be questionable at times. Presentation does not quite fit within time constraint; presenter has to rush or slow down at end	unremarkable. No clear introduction, supporting details and summary. Speaker has no credibility. Presentation is very short or unreasonably long.
<b>Project management (PO11, PSO2)</b>	XIII	Use of software project management principles and tools (versioning, time schedules etc )	Employ all the appropriate tools or engineering techniques. Clearly demonstrates mastery of several areas of the curriculum.	Employ the appropriate tools or engineering techniques	Employ some management tools	Does not make use of any tool
<b>Lifelong Learning (PO12, PSO2)</b>	XIV	<b>Extend Scope of Work:</b> Ability to extend the project through implementation in other study areas	Demonstrates a skillful ability to explore a subject/topic thoroughly, discusses the road map to extend the project in other areas.	Demonstrates an ability to explore a subject/topic, and shows possible areas in which project can be extended	Demonstrates some ability to explore a subject/topic, providing some knowledge of areas in which project can be extended	Demonstrates minimal or no ability to explore a subject/topic, and does not discuss future work clearly mentioning other areas

**Table 3 : Rubrics for Project**

**111. Rubrics Evaluation:**

	PO/PSO	PO1,PO2,PO6,PO7,PSO1			PO3,PSO1	PO4,PO5,PSO1	PO4,PO5,PSO1	PO8	PO9			PO10		PO11,PSO2	PO12,PSO2
Batch	Rubrics	R1			R2	R3	R4	R5	R6			R7		R8	R9
	Roll No	C I	C II	CI II	CIV	CV	CVI	CV II	CVI II	CI X	C X	C XI	CX II	CX III	CXI V

**Table 4 : Rubrics Evaluation**