**1.Types of recommendation system:**

1. Content Based: (content similarity)

-on the basis of content similarity it recommends you. Eg if u are listening romantic song then u will recommend you romantic song you only.

1. Collaborating filtering: (user similarity)

-on the basis of user interest it recommend you, for example there are two user A and B what happen is that both watched same movie and rated the movie similarly so when A watch new movie then as the movie taste of A and B is same then B will be recommended the same new movie.

1. Hybrid:

-content + collaborating

**2. Which type of recommendations we are going to apply to our project?**

Content based

**3.Project flow**

Data -> preprocessing ->model -> website-> deploy

**4.features in movie**

1. **budget**: The budget allocated for making the movie.
2. **genres**: A list of genres associated with the movie.
3. **homepage**: URL of the movie's official website.
4. **id**: Unique identifier for the movie.
5. **keywords**: Keywords or tags associated with the movie.
6. **original\_language**: The original language of the movie.
7. **original\_title**: The original title of the movie.
8. **overview**: A brief overview or summary of the movie's plot.
9. **popularity**: Popularity score of the movie.
10. **production\_companies**: A list of production companies involved in making the movie.
11. **production\_countries**: A list of countries where the movie was produced.
12. **release\_date**: The release date of the movie.
13. **revenue**: The revenue generated by the movie.
14. **runtime**: The duration of the movie in minutes.
15. **spoken\_languages**: A list of languages spoken in the movie.
16. **status**: The status of the movie (e.g., released, in production).
17. **tagline**: A tagline or slogan associated with the movie.
18. **title**: The title of the movie.
19. **vote\_average**: The average rating given to the movie.
20. **vote\_count**: The number of votes/ratings the movie has received.

5.features in credits

1.movie\_id

2. title

3.cast

4.crew

6.What are the different datasets used in this project?

I have choosen 2 types of datasets

* Movies (‘tmdb \_5000\_movies’)
* Credits(‘tmdb\_5000\_credits)

7. movies = movies.merge(credits,on='title')

There are different features present in both the datasets so it is better we must merge them ,hence we are adding credits with movies on the basis of title as title is the common feature in both.

8. movies['original\_language'].value\_counts()

It will count the number of movies present in the different languages

9. movies.info()

Output:

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, movie\_id, cast, crew

Out of all these we will select only few features as per our requirement movie\_id, title, overview, generes, keywordss, cast, crew.

We havenot selectedoriginal\_language because in the count we get that the original language English is 4510 which is around 95% of the total dataset and rest 5% which is not possible , rest remaining feature doesn’t matter for us.

10. movies = movies[['movie\_id','title','overview','genres','keywords','cast','crew']]

We set our new data frame

11. main moto of this project

We have to find tag so that it will create a paragraph for a movie to get recommendation.we basically add the name of the generes like “action” to overview then name of the keywords e.g culture to overview , then add top 3 main cats of the movie to the overview then atlast add producer name from the crew, and this will create a best paragraph that will attracts the viewers. The combination of all these form tag which is known as corpous.

12.movies.iloc[0]

The `iloc` function in pandas is used for integer-location based indexing. It allows you to access elements in a DataFrame or Series using their integer positions. Here's a brief overview of how it works:

- `iloc[row\_index]`: Access a single row at the specified integer position `row\_index`.

- `iloc[start\_index:end\_index]`: Access a range of rows from `start\_index` to `end\_index-1`.

- `iloc[row\_index, column\_index]`: Access a single element at the specified integer position `row\_index` and `column\_index`.

- `iloc[:, column\_index]`: Access all rows of a specific column at the specified integer position `column\_index`.

- `iloc[row\_index\_list, column\_index\_list]`: Access a subset of rows and columns specified by lists of integer positions.

For example, `df.iloc[0]` would return the first row of the DataFrame `df`, `df.iloc[:, 0]` would return the first column of the DataFrame, and `df.iloc[0, 0]` would return the value in the first row and first column of the DataFrame.

13. import ast

def convert(obj):

L=[]

for i in ast.literal\_eval(obj):

L.append(i['name'])

return L

This Python function `convert` takes a string `obj` as input, which represents a list of dictionaries. It then converts each dictionary in the list to extract the value associated with the key `'name'` and appends it to a new list `L`. Finally, it returns the list `L` containing the extracted names.

Here's how the function works:

1. It uses `ast.literal\_eval()` to safely evaluate the input string `obj` as a Python expression, which should be a list of dictionaries.

2. It iterates over each dictionary in the list obtained from the evaluated expression.

3. For each dictionary, it extracts the value associated with the key `'name'` and appends it to the list `L`.

4. After processing all dictionaries, it returns the list `L` containing the extracted names.

14. cv.get\_feature\_names\_out()

Which means it will return feature name generated by the vectorizer.

15.in result we will get some name like action and actions which result same meaning so need to change these kind of words so here we will apply steming. Means for action, actions,actioned it will take one word action.

**APP:**

**16.** import streamlit as st  
 Streamlit is a Python library that enables you to create interactive web applications directly from your Python scripts. It's designed to be simple and intuitive, allowing you to build data-driven apps quickly and easily. With Streamlit, you can create web interfaces for your machine learning models, data visualizations, and more, all with just a few lines of code.

Streamlit provides a variety of components and widgets for creating interactive elements such as sliders, dropdowns, buttons, and text inputs. You can also incorporate popular data science libraries like Pandas, Matplotlib, and Plotly to analyze and visualize data within your app.

One of the key features of Streamlit is its automatic updating. Whenever you make changes to your code and save the script, Streamlit will automatically detect those changes and update the web app in real-time, without requiring you to restart the server or refresh the browser.

Overall, Streamlit simplifies the process of creating web applications with Python, making it accessible to data scientists, machine learning engineers, and developers alike.

17.st.title(‘movie recommendation’)

The `st.title()` function in Streamlit is used to create a title for your web application. It's a simple way to add a prominent heading that captures the main purpose or theme of your app.

In your case, you've used `st.title('movie recommendation')` to set the title of your web application to "movie recommendation". This indicates to users what the focus of your app is — suggesting movies based on certain criteria, user preferences, or other factors.

Titles are typically displayed at the top of the web page or app interface, making them one of the first things users see when they interact with your application. By setting a clear and descriptive title like "movie recommendation", you provide users with an immediate understanding of what they can expect from your app.

18.import pickle

Importing `pickle` in Python allows you to serialize and deserialize Python objects, essentially converting them into byte streams for storage or transmission. This module is particularly useful for saving machine learning models, complex data structures, or any other Python objects to disk, and later reloading them into memory.

Once you import `pickle`, you can use its functions like `pickle.dump()` to serialize objects to a file and `pickle.load()` to deserialize objects from a file back into memory. This can be handy for saving trained machine learning models so you can reuse them later without needing to retrain them every time.

19. pickle.dump(new\_df,open('movie\_dict.pkl','wb'))

**This line of code serializes the DataFrame `new\_df` using the `pickle.dump()` function and saves it to a file named `'movies.pkl'`. Let me break it down for you:**

**- `pickle.dump()`: This function from the `pickle` module is used to serialize objects and write them to a file. In this case, you're using it to serialize the DataFrame `new\_df`.**

**- `new\_df`: This is the DataFrame object that you want to serialize and save.**

**- `open('movies.pkl', 'wb')`: This part of the code opens a file named `'movies.pkl'` in binary write mode (`'wb'`). The `'wb'` mode indicates that the file will be opened for writing in binary mode, which is necessary when working with pickle files.**

**- `pickle.dump(new\_df, open('movies.pkl', 'wb'))`: This expression serializes the DataFrame `new\_df` and writes it to the file `'movies.pkl'` using pickle serialization.**

**After executing this line of code, the DataFrame `new\_df` will be saved to a file named `'movies.pkl'` in binary format, ready to be loaded and deser ialized later using `pickle.load()`.**

**20. Try to copy it from c-drive ->users->Sanjiv->ml-project->movie-recommendation->movie\_dict // movies.pkl doesn’t work so we make new pkl called movie\_dict where we are using dictonary (new\_df.to\_dict())**

**And paste it to pycharm movie-recommendation.**

1. pickle.dump(new\_df.to\_dict(),open('movie\_dict.pkl','wb'))