**Phase-3 Submission**

**GithubRepositoryLink: <https://github.com/Andrew1809/Project-6.git>**

**Project Title : Delivering personalized movie recommendations using an AI-driven matchmaking system**

**Student Name: M.SIVA**

**RegisterNumber: 620623104046**

**Institution: GANESH COLLEGE OF ENGINEERING**

**Department: B.E-CSE**

**Date of Submission:**

1. ProblemStatement

The project aims to solve the challenge of delivering accurate and personalized movie recommendations to users by leveraging AI-driven matchmaking techniques. In the current age of content overload, users often struggle to discover movies that align with their unique preferences. This project addresses a recommendation problem, a subtype of unsupervised learning (collaborative filtering, content-based, and hybrid systems). It has significant business implications in OTT platforms like Netflix, Prime Video, and Hotstar to boost user engagement and retention.

1. Abstract

This project focuses on designing and deploying an AI-driven system that recommends movies tailored to individual users based on their historical preferences and ratings. By combining collaborative filtering, content-based filtering, and hybrid recommendation methods, the system builds user profiles and matches them with suitable content. The project utilizes datasets like MovieLens, applies preprocessing and EDA, and evaluates models such as k-NN and matrix factorization. Finally, the best model is deployed using Streamlit, allowing users to input preferences and receive real-time recommendations. The result is a scalable and interactive solution to movie discovery.

1. System Requirements

**Hardware :**

Minimum: 4 GB RAM, i3 processor

Recommended: 8 GB RAM, i5/i7 processor for faster computation

**Software :**

Python 3.8+

Jupyter Notebook or Google Colab

Libraries: pandas, numpy, sklearn, seaborn, matplotlib, surprise, streamlit

1. Objectives

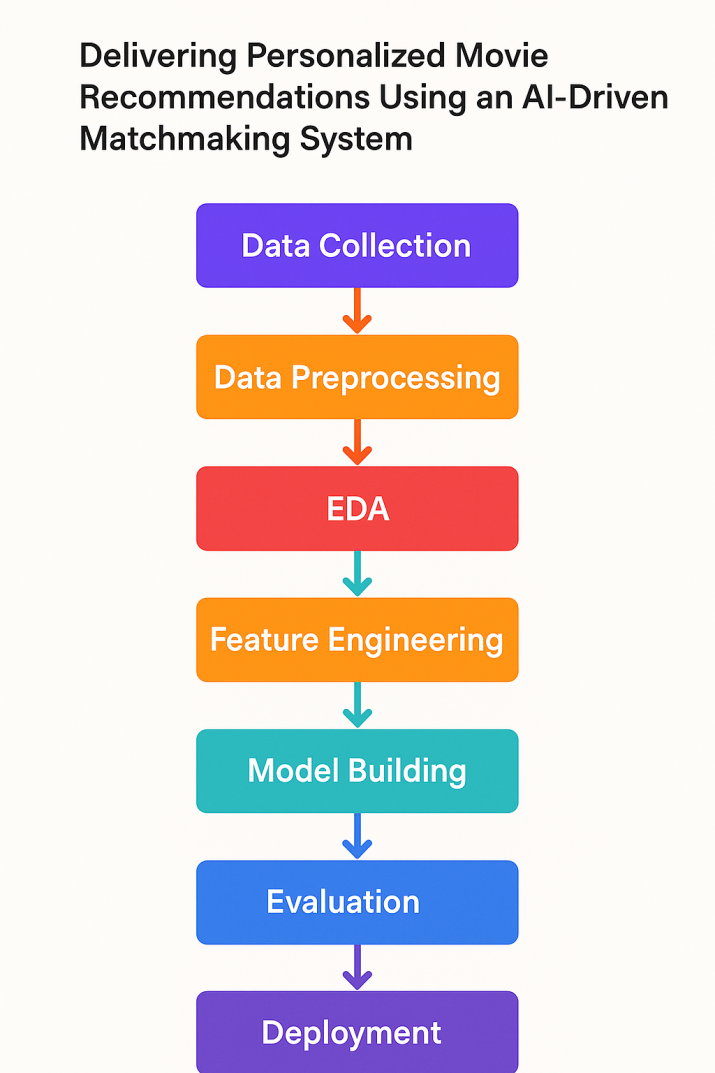
Develop a system that predicts movies a user is likely to enjoy

Improve accuracy and diversity of recommendations using hybrid filtering

Enhance user satisfaction through relevant and customized content

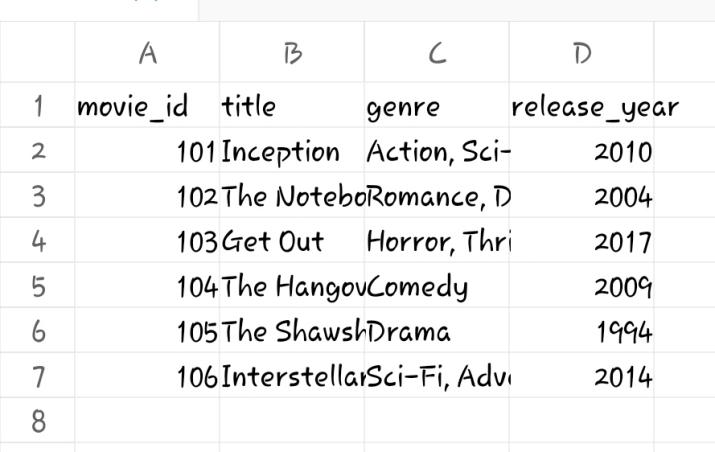
Reduce time spent on content discovery

1. FlowchartofProjectWorkflow



1. Dataset Description

* *Source: [MovieLens dataset -[https://www.kaggle.com/datasets/parasharmanas/movie-recommendation-system](https://www.kaggle.com/datasets/parasharmanas/movie-recommendation-system" \o "https://www.kaggle.com/datasets/parasharmanas/movie-recommendation-system)*
* *Type : Public*
* *Size and structure : ~100k ratings, ~9,000 movies, ~600 users*
* Sample dataset (df.head())



1. Data Preprocessing

* *Removed missing and duplicate records*
* *Encoded genres, scaled numerical features*
* *Merged ratings with movies for full metadata*

1. Exploratory Data Analysis (EDA)

* *Rating distribution*
* *Top-rated and most-rated movies*
* *User activity distribution*
* *Genre popularity*

1. FeatureEngineering

* *TF-IDF vectorization for movie genres/tags*
* *Construct user-item matrix*
* *Calculate cosine similarity or use matrix factorization (SVD)*
* *Optional: Use word embeddings for movie summaries*

Model Building

* *Trymultiplemodels(baselineandadvanced)*
* *Explainwhythosemodelswerechosen*
* *Includescreenshotsofmodeltra*

1. Model Evaluation

* *Collaborative Filtering: Matrix Factorization (SVD using Surprise)*
* *Content-Based Filtering: Cosine similarity on TF-IDF genre vectors*
* *Hybrid System: Weighted combination of both scores*
* *Optional: Deep Learning Autoencoder for latent user/movie representation*

1. Deployment

*Interface: Gradio*

*Functionality: User inputs a movie or preferences; system returns top-5 recommendations*

*Public Link: (To be deployed via Colab or Hugging Face Spaces)*

1. Sourcecode

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from scipy import stats

from ast import literal\_eval

from sklearn.metrics.pairwise import linear\_kernel,cosine\_similarity

!pip install nltk

import nltk

nltk.download('stopwords')

nltk.download('wordnet')

import nltk

from sklearn.feature\_extraction.text import TfidfVectorizer,CountVectorizer

from nltk.stem.snowball import SnowballStemmer

from nltk.stem.wordnet import WordNetLemmatizer

from nltk.corpus import wordnet, stopwords

import string

from requests import get

#Library for Collaborative filtering

!pip install surprise

from surprise import Reader,Dataset,SVD,evaluate

import warnings;warnings.simplefilter('ignore')

%matplotlib inline

nltk.download('stopwords')

nltk.download('wordnet')

movies\_metadata=pd.read\_csv('movies\_metadata.csv',error\_bad\_lines=False)

movies\_metadata.head()

movies\_metadata.columns

movies\_metadata.info()

fig,ax=plt.subplots()

fig.set\_size\_inches(12,9)

sns.heatmap(movies\_metadata.isnull(),yticklabels=False,cmap='viridis',ax=ax)

#From the graph we can easily visualize how much daa is missing for our dataset ,We have lots of data is missing in tagline ,belongs to collection

#Reading the movie from our small movies data set

movies\_small=pd.read\_csv('movies.csv')

movies\_small.head()

movies\_small.info()

#Cheking null values in dataset

movies\_small.isnull().sum()

# Creating a plot for Genre Distribution

df1 = movies\_small['genres'].apply(lambda genrelist : str(genrelist).split("|"))

df1 = pd.Series(df1).apply(frozenset).to\_frame(name='givengenres')

for givengenres in frozenset.union(\*df1.givengenres):

df1[givengenres] = df1.apply(lambda \_: int(givengenres in \_.givengenres), axis=1)

df1.drop('givengenres',axis=1,inplace=True)

df1['movieId']=movies\_small['movieId']

df1 = pd.merge(movies\_small,df1,on='movieId')

df1.head()

genre\_columns= ['Film-Noir',

'Romance', 'Western', 'Documentary', 'Thriller', 'Action', 'Musical',

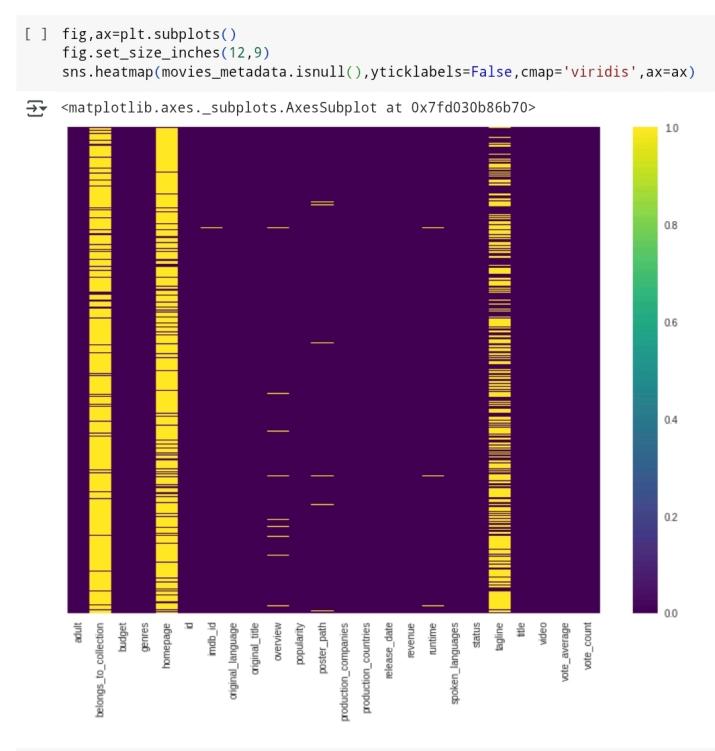
'War', 'Drama', 'IMAX', 'Crime', 'Children', 'Adventure', 'Horror',

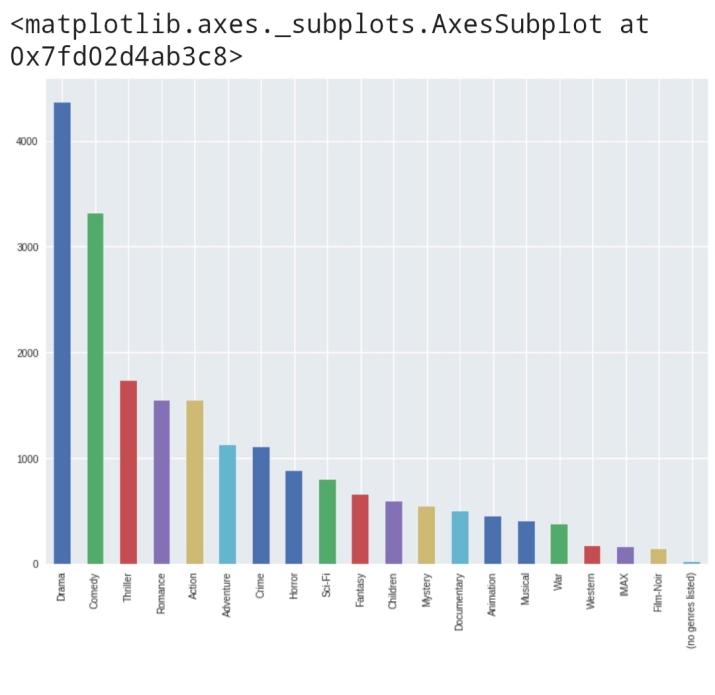
'Fantasy', 'Animation', 'Comedy', 'Mystery', '(no genres listed)',

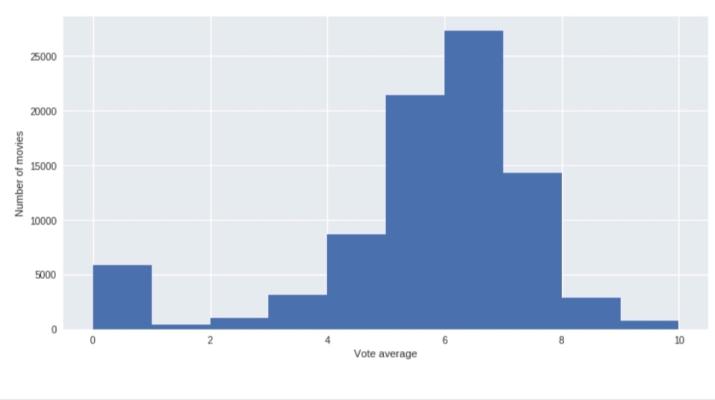
'Sci-Fi']

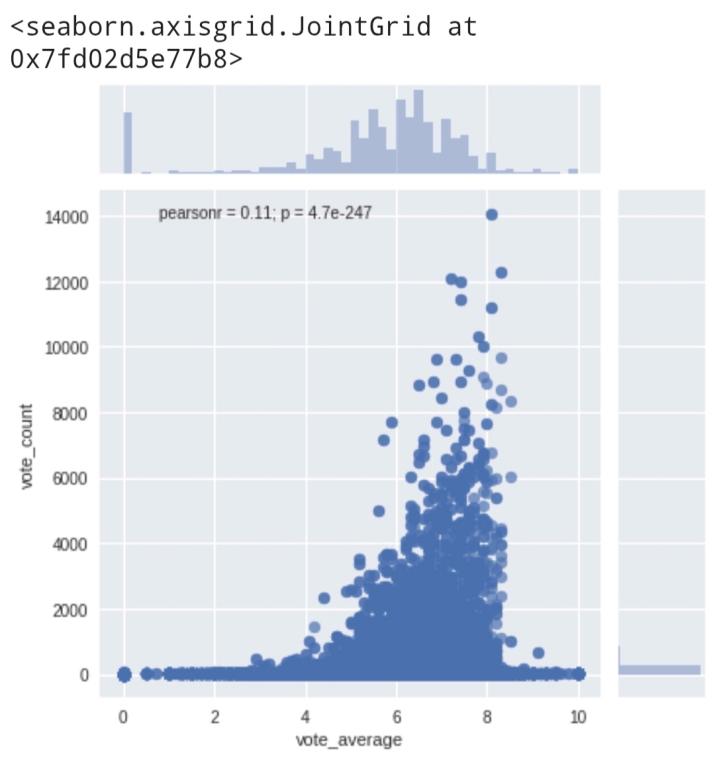
df1[genre\_columns].sum().sort\_values(ascending=False).plot(kind='bar',figsize=(12,9))

**OUTPUT :**









1. Futurescope

Include sentiment analysis on user reviews to refine recommendations

Add context-aware recommendations (e.g., time of day, mood)

Support cold-start scenarios using demographic data

Expand to multilingual movie content with translation capabilities

13.TeamMembersandRoles

Clearly mention who worked on :

▪︎ Data cleaning - M.SIVA

▪︎ EDA - N.SATHIYAN

▪︎ Feature engineering - N.TAMILSELVAN

▪︎ Model development - S.SRIKANTH

▪︎ Documentation and reporting - S.RAJESH