***Source Code***

1.Upload the Dataset

from google.colab import files

uploaded = files.upload() # Upload movies.csv

2. Load the Dataset

import pandas as pd

df = pd.read\_csv("movies.csv")

df.head()

3. Data Exploration

print("Shape:", df.shape)

print(df.info())

print(df.describe(include="all").T)

print(df.sample(5))

4. Check for Missing Values and Duplicates

print("Missing values:\n", df.isnull().sum())

print("Duplicates:", df.duplicated().sum())

df = df.drop\_duplicates()

5. Visualize a Few Features

import matplotlib.pyplot as plt

import seaborn as sns

# Example: distribution of genres

if "genres" in df.columns:

top\_genres = df['genres'].value\_counts().head(10)

sns.barplot(x=top\_genres.values, y=top\_genres.index)

plt.title("Top Genres")

plt.show()

6.Identify Target and Features

features = ["movieId", "title", "genres"]

print("Features:", features)

7.Convert Categorical Columns to Numerical

from sklearn.preprocessing import LabelEncoder

if "genres" in df.columns:

le = LabelEncoder()

df["genres\_encoded"] = le.fit\_transform(df["genres"].astype(str))

8. One-Hot Encoding

from sklearn.preprocessing import OneHotEncoder

if "genres" in df.columns:

ohe = OneHotEncoder()

genres\_matrix = ohe.fit\_transform(df[["genres"]])

print("One-hot encoded genres shape:", genres\_matrix.shape)

9.Feature Scaling

from sklearn.preprocessing import StandardScaler

import numpy as np

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(genres\_matrix.toarray())

10. Train-Test Split

from sklearn.model\_selection import train\_test\_split

X = scaled\_features

y = df["genres\_encoded"] # pseudo target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

11. Model Building

from sklearn.neighbors import NearestNeighbors

model = NearestNeighbors(metric="cosine", algorithm="brute")

model.fit(X\_train)

12. Evaluation

distances, indices = model.kneighbors(X\_test[:5], n\_neighbors=5)

print("Example Recommendations (indices):", indices)

13. Make Predictions from New Input

def recommend(movie\_title):

movie\_idx = df[df["title"].str.contains(movie\_title, case=False, na=False)].index[0]

movie\_vector = X[movie\_idx].reshape(1, -1)

distances, indices = model.kneighbors(movie\_vector, n\_neighbors=6)

recs = df.iloc[indices[0][1:]]["title"].values

return recs

print("Recommendations:", recommend("Toy Story"))

14. Convert to DataFrame and Encode

input\_dict = {"title": "Inception", "genres": "Action"}

input\_df = pd.DataFrame([input\_dict])

input\_df["genres\_encoded"] = le.transform(input\_df["genres"])

15. Predict the Final Grade

def predict\_score(movie\_title):

movie\_idx = df[df["title"].str.contains(movie\_title, case=False, na=False)].index[0]

movie\_vector = X[movie\_idx].reshape(1, -1)

distances, indices = model.kneighbors(movie\_vector, n\_neighbors=2)

return 1 - distances[0][1] # similarity score

16. Create a Prediction Function

def gradio\_recommender(movie\_title):

try:

recs = recommend(movie\_title)

return "\n".join(recs)

except:

return "Movie not found. Try another."

17. Create the Gradio Interface

import gradio as gr

iface = gr.Interface(

fn=gradio\_recommender,

inputs=gr.Textbox(label="Enter a Movie Title"),

outputs="text",

title="AI Movie Recommendation System",

description="Type a movie title and get similar recommendations!"

)

iface.launch(share=True)