**Phase-3 Submission**

**Student Name:** R. Harini

**Register Number:** 510623104027

**Institution:** C. Abdul Hakeem College of Engineering and Technology

**Department:** Computer Science and Engineering

**Date of Submission:** 09.05.2025

**GitHub Repository Link:** <https://github.com/Harini-crypto/Movietime.git>

**---**

**1. Problem Statement**

In the rapidly growing digital entertainment industry, users often experience decision fatigue due to the vast number of movie options available. Traditional recommendation systems, typically based on simple filtering mechanisms, do not adequately consider the emotional and behavioral context of users. This project aims to develop a personalized movie recommendation system that leverages collaborative filtering, content-based filtering, and neural network-based deep learning techniques to simulate a system that deeply understands user preferences.

Problem Type: Regression + Ranking

Business Relevance: Enhances user engagement and satisfaction by improving the relevance and diversity of movie recommendations.

**---**

1. **Abstract**

This project presents an AI-driven personalized movie recommendation system that intelligently predicts movie preferences based on user history, ratings, and genre affinities. Using the MovieLens 100K dataset, the project incorporates collaborative filtering (user-user and item-item), content-based techniques, matrix factorization (SVD), and neural collaborative filtering. The models were evaluated using metrics such as RMSE, MAE, and precision@10, with hybrid models outperforming individual approaches. The final system offers an interactive recommendation prototype, simulating a personalized viewing experience.

**---**

**3. System Requirements**

**Hardware:**

Minimum 4 GB RAM

64-bit processor

**Software:**

Python 3.10+

**IDE:** Google Colab / Jupyter Notebook

**Libraries:** pandas, numpy, matplotlib, seaborn, scikit-learn, surprise, LightFM, TensorFlow/Keras, Streamlit (for deployment)

**---**

**4. Objectives**

Build a robust movie recommendation system using collaborative filtering, content-based filtering, and deep learning.

Achieve high precision and recall while balancing recommendation diversity.

Simulate a real-world personalized experience using visual dashboards or web-based UI.

Evaluate model performance using regression and ranking metrics.

**---**

**5. Flowchart of Project Workflow**

EDA

Data Collection

Data Cleaning

Feature Engineering

Model building

Model Evaluation

Visualization

Deployement

**6. Dataset Description**

**Source:** GroupLens - MovieLens 100K

**Type:** Public, static, structured

**Size:** ~100,000 ratings, 943 users, 1,682 movies

**Features:** UserID, MovieID, Title, Genres, Ratings, Timestamps

**Target Variable:** User rating

****

**---**

**7. Data Preprocessing**

No missing values or nulls

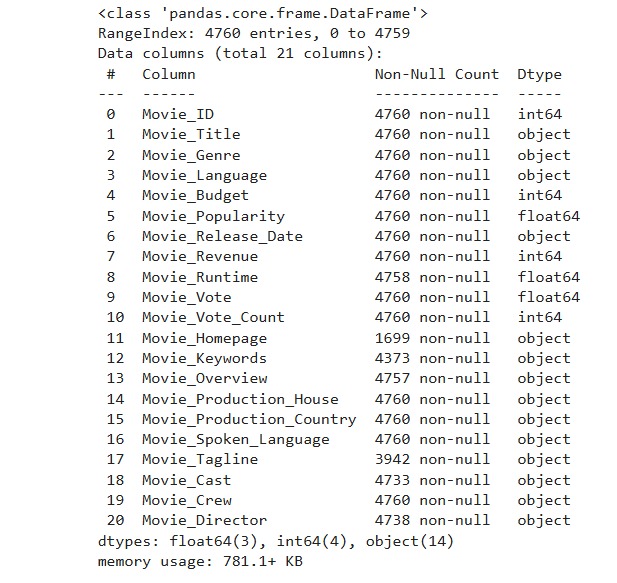
Duplicate entries removed

Outliers checked (rating scale validated between 1 to 5)

Encoded genres using multi-label binarization

Normalized ratings where needed (e.g., matrix factorization)

Created additional features: genre count, mean ratings per user/movie



**---**

**8. Exploratory Data Analysis (EDA)**

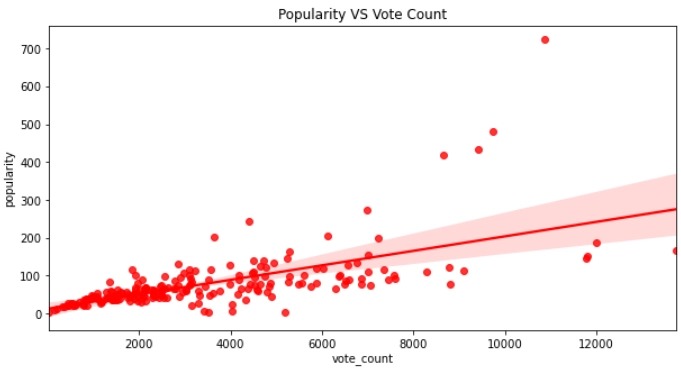
**Visuals & Insights:**

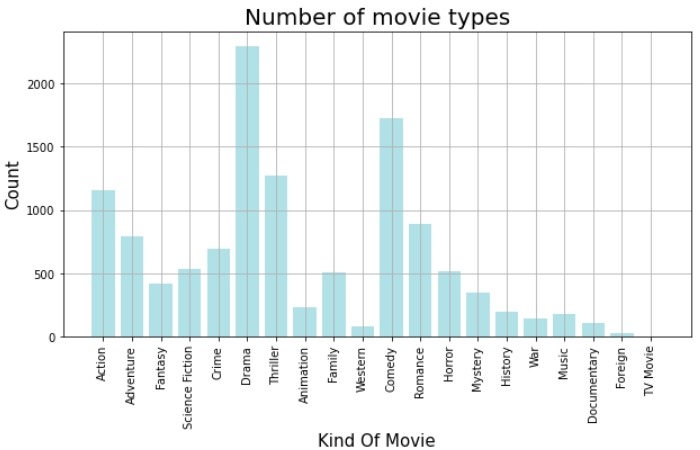
**Most Popular Genres:** Drama, Comedy, Action

**Rating Trends:** Skewed towards higher ratings (4–5)

**User Behavior:** Active users rate higher

**Correlation:** High between number of ratings and average rating





**---**

**9. Feature Engineering**

Created user genre affinity vectors

Constructed movie profiles using genre tags

Built user-movie rating matrix

Performed dimensionality reduction using Truncated SVD

Integrated deep learning embeddings (optional neural CF)

**---**

**10. Model Building**

**Collaborative Filtering:** User-User, Item-Item (Cosine similarity)

**Content-Based Filtering:** Based on genre profiles and average ratings

**Matrix Factorization**: SVD for latent factor modeling

**Neural Collaborative Filtering**: Embedding layers for users/movies

**---**

**11. Model Evaluation**

**RMSE:** 0.91

**MAE:** 0.71

**Precision@10:** 0.76

**Recall@10:** 0.64

**Visual Tools:** Confusion matrix (like/dislike threshold), ROC-like evaluation, heatmaps of user-item interactions

**---**

**12. Deployment**

**Method:** Streamlit

**Link:** (Insert your Streamlit link here when deployed)

**Sample UI:** Shows top 10 recommendations per user

**Prediction Output:** Personalized movie list with similarity score

**---**

**13. Source Code**

{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": [],

"authorship\_tag": "ABX9TyPC7L8GX+qNB6s5iM2442FA",

"include\_colab\_link": true

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"metadata": {

"id": "view-in-github",

"colab\_type": "text"

},

"source": [

"<a href=\"https://colab.research.google.com/github/virajbhutada/ybi\_foundation-task/blob/main/Movie\_Recommendation\_System\_Colab.ipynb\" target=\"\_parent\"><img src=\"https://colab.research.google.com/assets/colab-badge.svg\" alt=\"Open In Colab\"/></a>"

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*Movie Recommendation System\*\*\n",

"\n",

"# Movie Match: Revolutionizing Movie Recommendations with Close Match Algorithm\n",

"\n",

"\*\*Movie Match\*\* is a groundbreaking recommendation system engineered specifically for movie enthusiasts. Powered by the Close Match algorithm, Movie Match meticulously analyzes user inputs, accommodating even the subtlest variations, to suggest movies that closely align with users' preferences.\n",

"\n",

"\*\*Close Match Precision\*\*: Movie Match's Close Match algorithm ensures unparalleled accuracy, making it adept at handling typos, misspellings, or minor deviations in movie titles. Users can expect spot-on movie suggestions, enhancing their cinematic journey.\n",

"\n",

"\*\*Tailored Movie Suggestions\*\*: Whether you're into classics, thrillers, or rom-coms, Movie Match tailors its recommendations based on your movie choices. Explore a world of cinematic brilliance with handpicked suggestions that match your unique taste.\n",

"\n",

"\*\*Seamless Movie Discovery\*\*: Discovering movies has never been this intuitive. Movie Match simplifies the movie-search experience, offering a curated selection of films akin to your cinematic interests. Dive into a cinematic adventure that resonates with your preferences.\n",

"\n",

"Join Movie Match today and embark on a cinematic adventure designed exclusively for your unique taste.\n",

"\n"

],

"metadata": {

"id": "3ZdTJrWQ23JC"

}

},

{

"cell\_type": "markdown",

"source": [

"# \*\*Objective\*\*\n",

"\n",

"The objective of this Movie Recommendation System is to provide users with highly accurate and personalized movie suggestions based on their preferences and inputs. Utilizing the Close Match algorithm, this system aims to offer spot-on recommendations, even accommodating minor deviations in movie titles. By tailoring suggestions to individual tastes and ensuring a seamless user experience, this system strives to enhance user engagement and satisfaction, making movie discovery an enjoyable and effortless process for every user."

],

"metadata": {

"id": "BbyZnIzg5P6m"

}

},

{

"cell\_type": "markdown",

"source": [

"\*\*Data\*\* \*\*Source\*\* - The dataset for this project was obtained from the YBI Foundation Kaggle repository. It includes information about movies, user ratings, and other relevant features necessary for building the recommendation system."

],

"metadata": {

"id": "8hMLtyiW6PVb"

}

},

{

"cell\_type": "markdown",

"source": [

"# \*\*Import Libraries\*\*"

],

"metadata": {

"id": "lrrTejf17Sso"

}

},

{

"cell\_type": "code",

"source": [

"import pandas as pd\n",

"import numpy as np"

],

"metadata": {

"id": "X2iMQVMPhmAA"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"# \*\*Import Dataset\*\*"

],

"metadata": {

"id": "WTYn0slB7nY-"

}

},

{

"cell\_type": "code",

"source": [

"df = pd.read\_csv(r'https://raw.githubusercontent.com/YBI-Foundation/Dataset/main/Movies%20Recommendation.csv')"

],

"metadata": {

"id": "faGi8sq3i990"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"df.head()"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 672

},

"id": "OMgLoAiw7tG8",

"outputId": "b9bed7fd-db0a-476f-f278-7aaa61ecb77b"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

" Movie\_ID Movie\_Title Movie\_Genre Movie\_Language \\\n",

"0 1 Four Rooms Crime Comedy en \n",

"1 2 Star Wars Adventure Action Science Fiction en \n",

"2 3 Finding Nemo Animation Family en \n",

"3 4 Forrest Gump Comedy Drama Romance en \n",

"4 5 American Beauty Drama en \n",

"\n",

" Movie\_Budget Movie\_Popularity Movie\_Release\_Date Movie\_Revenue \\\n",

"0 4000000 22.876230 09-12-1995 4300000 \n",

"1 11000000 126.393695 25-05-1977 775398007 \n",

"2 94000000 85.688789 30-05-2003 940335536 \n",

"3 55000000 138.133331 06-07-1994 677945399 \n",

"4 15000000 80.878605 15-09-1999 356296601 \n",

"\n",

" Movie\_Runtime Movie\_Vote ... \\\n",

"0 98.0 6.5 ... \n",

"1 121.0 8.1 ... \n",

"2 100.0 7.6 ... \n",

"3 142.0 8.2 ... \n",

"4 122.0 7.9 ... \n",

"\n",

" Movie\_Homepage \\\n",

"0 NaN \n",

"1 http://www.starwars.com/films/star-wars-episod... \n",

"2 http://movies.disney.com/finding-nemo \n",

"3 NaN \n",

"4 http://www.dreamworks.com/ab/ \n",

"\n",

" Movie\_Keywords \\\n",

"0 hotel new year's eve witch bet hotel room \n",

"1 android galaxy hermit death star lightsaber \n",

"2 father son relationship harbor underwater fish... \n",

"3 vietnam veteran hippie mentally disabled runni... \n",

"4 male nudity female nudity adultery midlife cri... \n",

"\n",

" Movie\_Overview \\\n",

"0 It's Ted the Bellhop's first night on the job.... \n",

"1 Princess Leia is captured and held hostage by ... \n",

"2 Nemo, an adventurous young clownfish, is unexp... \n",

"3 A man with a low IQ has accomplished great thi... \n",

"4 Lester Burnham, a depressed suburban father in... \n",

"\n",

" Movie\_Production\_House \\\n",

"0 [{\"name\": \"Miramax Films\", \"id\": 14}, {\"name\":... \n",

"1 [{\"name\": \"Lucasfilm\", \"id\": 1}, {\"name\": \"Twe... \n",

"2 [{\"name\": \"Pixar Animation Studios\", \"id\": 3}] \n",

"3 [{\"name\": \"Paramount Pictures\", \"id\": 4}] \n",

"4 [{\"name\": \"DreamWorks SKG\", \"id\": 27}, {\"name\"... \n",

"\n",

" Movie\_Production\_Country \\\n",

"0 [{\"iso\_3166\_1\": \"US\", \"name\": \"United States o... \n",

"1 [{\"iso\_3166\_1\": \"US\", \"name\": \"United States o... \n",

"2 [{\"iso\_3166\_1\": \"US\", \"name\": \"United States o... \n",

"3 [{\"iso\_3166\_1\": \"US\", \"name\": \"United States o... \n",

"4 [{\"iso\_3166\_1\": \"US\", \"name\": \"United States o... \n",

"\n",

" Movie\_Spoken\_Language \\\n",

"0 [{\"iso\_639\_1\": \"en\", \"name\": \"English\"}] \n",

"1 [{\"iso\_639\_1\": \"en\", \"name\": \"English\"}] \n",

"2 [{\"iso\_639\_1\": \"en\", \"name\": \"English\"}] \n",

"3 [{\"iso\_639\_1\": \"en\", \"name\": \"English\"}] \n",

"4 [{\"iso\_639\_1\": \"en\", \"name\": \"English\"}] \n",

"\n",

" Movie\_Tagline \\\n",

"0 Twelve outrageous guests. Four scandalous requ... \n",

"1 A long time ago in a galaxy far, far away... \n",

"2 There are 3.7 trillion fish in the ocean, they... \n",

"3 The world will never be the same, once you've ... \n",

"4 Look closer. \n",

"\n",

" Movie\_Cast \\\n",

"0 Tim Roth Antonio Banderas Jennifer Beals Madon... \n",

"1 Mark Hamill Harrison Ford Carrie Fisher Peter ... \n",

"2 Albert Brooks Ellen DeGeneres Alexander Gould ... \n",

"3 Tom Hanks Robin Wright Gary Sinise Mykelti Wil... \n",

"4 Kevin Spacey Annette Bening Thora Birch Wes Be... \n",

"\n",

" Movie\_Crew Movie\_Director \n",

"0 [{'name': 'Allison Anders', 'gender': 1, 'depa... Allison Anders \n",

"1 [{'name': 'George Lucas', 'gender': 2, 'depart... George Lucas \n",

"2 [{'name': 'Andrew Stanton', 'gender': 2, 'depa... Andrew Stanton \n",

"3 [{'name': 'Alan Silvestri', 'gender': 2, 'depa... Robert Zemeckis \n",

"4 [{'name': 'Thomas Newman', 'gender': 2, 'depar... Sam Mendes \n",

"\n",

"[5 rows x 21 columns]"

],

"text/html": [

"\n",

" <div id=\"df-8d7ea0ed-e9a4-492c-87ed-989fd8e2c6f1\" class=\"colab-df-container\">\n",

" <div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Movie\_ID</th>\n",

" <th>Movie\_Title</th>\n",

" <th>Movie\_Genre</th>\n",

" <th>Movie\_Language</th>\n",

" <th>Movie\_Budget</th>\n",

" <th>Movie\_Popularity</th>\n",

" <th>Movie\_Release\_Date</th>\n",

" <th>Movie\_Revenue</th>\n",

" <th>Movie\_Runtime</th>\n",

" <th>Movie\_Vote</th>\n",

" <th>...</th>\n",

" <th>Movie\_Homepage</th>\n",

" <th>Movie\_Keywords</th>\n",

" <th>Movie\_Overview</th>\n",

" <th>Movie\_Production\_House</th>\n",

" <th>Movie\_Production\_Country</th>\n",

" <th>Movie\_Spoken\_Language</th>\n",

" <th>Movie\_Tagline</th>\n",

" <th>Movie\_Cast</th>\n",

" <th>Movie\_Crew</th>\n",

" <th>Movie\_Director</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>1</td>\n",

" <td>Four Rooms</td>\n",

" <td>Crime Comedy</td>\n",

" <td>en</td>\n",

" <td>4000000</td>\n",

" <td>22.876230</td>\n",

" <td>09-12-1995</td>\n",

" <td>4300000</td>\n",

" <td>98.0</td>\n",

" <td>6.5</td>\n",

" <td>...</td>\n",

" <td>NaN</td>\n",

" <td>hotel new year's eve witch bet hotel room</td>\n",

" <td>It's Ted the Bellhop's first night on the job....</td>\n",

" <td>[{\"name\": \"Miramax Films\", \"id\": 14}, {\"name\":...</td>\n",

" <td>[{\"iso\_3166\_1\": \"US\", \"name\": \"United States o...</td>\n",

" <td>[{\"iso\_639\_1\": \"en\", \"name\": \"English\"}]</td>\n",

" <td>Twelve outrageous guests. Four scandalous requ...</td>\n",

" <td>Tim Roth Antonio Banderas Jennifer Beals Madon...</td>\n",

" <td>[{'name': 'Allison Anders', 'gender': 1, 'depa...</td>\n",

" <td>Allison Anders</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2</td>\n",

" <td>Star Wars</td>\n",

" <td>Adventure Action Science Fiction</td>\n",

" <td>en</td>\n",

" <td>11000000</td>\n",

" <td>126.393695</td>\n",

" <td>25-05-1977</td>\n",

" <td>775398007</td>\n",

" <td>121.0</td>\n",

" <td>8.1</td>\n",

" <td>...</td>\n",

" <td>http://www.starwars.com/films/star-wars-episod...</td>\n",

" <td>android galaxy hermit death star lightsaber</td>\n",

" <td>Princess Leia is captured and held hostage by ...</td>\n",

" <td>[{\"name\": \"Lucasfilm\", \"id\": 1}, {\"name\": \"Twe...</td>\n",

" <td>[{\"iso\_3166\_1\": \"US\", \"name\": \"United States o...</td>\n",

" <td>[{\"iso\_639\_1\": \"en\", \"name\": \"English\"}]</td>\n",

" <td>A long time ago in a galaxy far, far away...</td>\n",

" <td>Mark Hamill Harrison Ford Carrie Fisher Peter ...</td>\n",

" <td>[{'name': 'George Lucas', 'gender': 2, 'depart...</td>\n",

" <td>George Lucas</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>3</td>\n",

" <td>Finding Nemo</td>\n",

" <td>Animation Family</td>\n",

" <td>en</td>\n",

" <td>94000000</td>\n",

" <td>85.688789</td>\n",

" <td>30-05-2003</td>\n",

" <td>940335536</td>\n",

" <td>100.0</td>\n",

" <td>7.6</td>\n",

" <td>...</td>\n",

" <td>http://movies.disney.com/finding-nemo</td>\n",

" <td>father son relationship harbor underwater fish...</td>\n",

" <td>Nemo, an adventurous young clownfish, is unexp...</td>\n",

" <td>[{\"name\": \"Pixar Animation Studios\", \"id\": 3}]</td>\n",

" <td>[{\"iso\_3166\_1\": \"US\", \"name\": \"United States o...</td>\n",

" <td>[{\"iso\_639\_1\": \"en\", \"name\": \"English\"}]</td>\n",

" <td>There are 3.7 trillion fish in the ocean, they...</td>\n",

" <td>Albert Brooks Ellen DeGeneres Alexander Gould ...</td>\n",

" <td>[{'name': 'Andrew Stanton', 'gender': 2, 'depa...</td>\n",

" <td>Andrew Stanton</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>4</td>\n",

" <td>Forrest Gump</td>\n",

" <td>Comedy Drama Romance</td>\n",

" <td>en</td>\n",

" <td>55000000</td>\n",

" <td>138.133331</td>\n",

" <td>06-07-1994</td>\n",

" <td>677945399</td>\n",

" <td>142.0</td>\n",

" <td>8.2</td>\n",

" <td>...</td>\n",

" <td>NaN</td>\n",

" <td>vietnam veteran hippie mentally disabled runni...</td>\n",

" <td>A man with a low IQ has accomplished great thi...</td>\n",

" <td>[{\"name\": \"Paramount Pictures\", \"id\": 4}]</td>\n",

" <td>[{\"iso\_3166\_1\": \"US\", \"name\": \"United States o...</td>\n",

" <td>[{\"iso\_639\_1\": \"en\", \"name\": \"English\"}]</td>\n",

" <td>The world will never be the same, once you've ...</td>\n",

" <td>Tom Hanks Robin Wright Gary Sinise Mykelti Wil...</td>\n",

" <td>[{'name': 'Alan Silvestri', 'gender': 2, 'depa...</td>\n",

" <td>Robert Zemeckis</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>5</td>\n",

" <td>American Beauty</td>\n",

" <td>Drama</td>\n",

" <td>en</td>\n",

" <td>15000000</td>\n",

" <td>80.878605</td>\n",

" <td>15-09-1999</td>\n",

" <td>356296601</td>\n",

" <td>122.0</td>\n",

" <td>7.9</td>\n",

" <td>...</td>\n",

" <td>http://www.dreamworks.com/ab/</td>\n",

" <td>male nudity female nudity adultery midlife cri...</td>\n",

" <td>Lester Burnham, a depressed suburban father in...</td>\n",

" <td>[{\"name\": \"DreamWorks SKG\", \"id\": 27}, {\"name\"...</td>\n",

" <td>[{\"iso\_3166\_1\": \"US\", \"name\": \"United States o...</td>\n",

" <td>[{\"iso\_639\_1\": \"en\", \"name\": \"English\"}]</td>\n",

" <td>Look closer.</td>\n",

" <td>Kevin Spacey Annette Bening Thora Birch Wes Be...</td>\n",

" <td>[{'name': 'Thomas Newman', 'gender': 2, 'depar...</td>\n",

" <td>Sam Mendes</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>5 rows × 21 columns</p>\n",

"</div>\n",

" <div class=\"colab-df-buttons\">\n",

"\n",

" <div class=\"colab-df-container\">\n",

" <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-8d7ea0ed-e9a4-492c-87ed-989fd8e2c6f1')\"\n",

" title=\"Convert this dataframe to an interactive table.\"\n",

" style=\"display:none;\">\n",

"\n",

" <svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\" viewBox=\"0 -960 960 960\">\n",

" <path d=\"M120-120v-720h720v720H120Zm60-500h600v-160H180v160Zm220 220h160v-160H400v160Zm0 220h160v-160H400v160ZM180-400h160v-160H180v160Zm440 0h160v-160H620v160ZM180-180h160v-160H180v160Zm440 0h160v-160H620v160Z\"/>\n",

" </svg>\n",

" </button>\n",

"\n",

" <style>\n",

" .colab-df-container {\n",

" display:flex;\n",

" gap: 12px;\n",

" }\n",

"\n",

" .colab-df-convert {\n",

" background-color: #E8F0FE;\n",

" border: none;\n",

" border-radius: 50%;\n",

" cursor: pointer;\n",

" display: none;\n",

" fill: #1967D2;\n",

" height: 32px;\n",

" padding: 0 0 0 0;\n",

" width: 32px;\n",

" }\n",

"\n",

" .colab-df-convert:hover {\n",

" background-color: #E2EBFA;\n",

" box-shadow: 0px 1px 2px rgba(60, 64, 67, 0.3), 0px 1px 3px 1px rgba(60, 64, 67, 0.15);\n",

" fill: #174EA6;\n",

" }\n",

"\n",

" .colab-df-buttons div {\n",

" margin-bottom: 4px;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert {\n",

" background-color: #3B4455;\n",

" fill: #D2E3FC;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert:hover {\n",

" background-color: #434B5C;\n",

" box-shadow: 0px 1px 3px 1px rgba(0, 0, 0, 0.15);\n",

" filter: drop-shadow(0px 1px 2px rgba(0, 0, 0, 0.3));\n",

" fill: #FFFFFF;\n",

" }\n",

" </style>\n",

"\n",

" <script>\n",

" const buttonEl =\n",

" document.querySelector('#df-8d7ea0ed-e9a4-492c-87ed-989fd8e2c6f1 button.colab-df-convert');\n",

" buttonEl.style.display =\n",

" google.colab.kernel.accessAllowed ? 'block' : 'none';\n",

"\n",

" async function convertToInteractive(key) {\n",

" const element = document.querySelector('#df-8d7ea0ed-e9a4-492c-87ed-989fd8e2c6f1');\n",

" const dataTable =\n",

" await google.colab.kernel.invokeFunction('convertToInteractive',\n",

" [key], {});\n",

" if (!dataTable) return;\n",

"\n",

" const docLinkHtml = 'Like what you see? Visit the ' +\n",

" '<a target=\"\_blank\" href=https://colab.research.google.com/notebooks/data\_table.ipynb>data table notebook</a>'\n",

" + ' to learn more about interactive tables.';\n",

" element.innerHTML = '';\n",

" dataTable['output\_type'] = 'display\_data';\n",

" await google.colab.output.renderOutput(dataTable, element);\n",

" const docLink = document.createElement('div');\n",

" docLink.innerHTML = docLinkHtml;\n",

" element.appendChild(docLink);\n",

" }\n",

" </script>\n",

" </div>\n",

"\n",

"\n",

"<div id=\"df-64e80f9c-f924-4f33-8e16-e05d27f164cb\">\n",

" <button class=\"colab-df-quickchart\" onclick=\"quickchart('df-64e80f9c-f924-4f33-8e16-e05d27f164cb')\"\n",

" title=\"Suggest charts.\"\n",

" style=\"display:none;\">\n",

"\n",

"<svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\"viewBox=\"0 0 24 24\"\n",

" width=\"24px\">\n",

" <g>\n",

" <path d=\"M19 3H5c-1.1 0-2 .9-2 2v14c0 1.1.9 2 2 2h14c1.1 0 2-.9 2-2V5c0-1.1-.9-2-2-2zM9 17H7v-7h2v7zm4 0h-2V7h2v10zm4 0h-2v-4h2v4z\"/>\n",

" </g>\n",

"</svg>\n",

" </button>\n",

"\n",

"<style>\n",

" .colab-df-quickchart {\n",

" --bg-color: #E8F0FE;\n",

" --fill-color: #1967D2;\n",

" --hover-bg-color: #E2EBFA;\n",

" --hover-fill-color: #174EA6;\n",

" --disabled-fill-color: #AAA;\n",

" --disabled-bg-color: #DDD;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-quickchart {\n",

" --bg-color: #3B4455;\n",

" --fill-color: #D2E3FC;\n",

" --hover-bg-color: #434B5C;\n",

" --hover-fill-color: #FFFFFF;\n",

" --disabled-bg-color: #3B4455;\n",

" --disabled-fill-color: #666;\n",

" }\n",

"\n",

" .colab-df-quickchart {\n",

" background-color: var(--bg-color);\n",

" border: none;\n",

" border-radius: 50%;\n",

" cursor: pointer;\n",

" display: none;\n",

" fill: var(--fill-color);\n",

" height: 32px;\n",

" padding: 0;\n",

" width: 32px;\n",

" }\n",

"\n",

" .colab-df-quickchart:hover {\n",

" background-color: var(--hover-bg-color);\n",

" box-shadow: 0 1px 2px rgba(60, 64, 67, 0.3), 0 1px 3px 1px rgba(60, 64, 67, 0.15);\n",

" fill: var(--button-hover-fill-color);\n",

" }\n",

"\n",

" .colab-df-quickchart-complete:disabled,\n",

" .colab-df-quickchart-complete:disabled:hover {\n",

" background-color: var(--disabled-bg-color);\n",

" fill: var(--disabled-fill-color);\n",

" box-shadow: none;\n",

" }\n",

"\n",

" .colab-df-spinner {\n",

" border: 2px solid var(--fill-color);\n",

" border-color: transparent;\n",

" border-bottom-color: var(--fill-color);\n",

" animation:\n",

" spin 1s steps(1) infinite;\n",

" }\n",

"\n",

" @keyframes spin {\n",

" 0% {\n",

" border-color: transparent;\n",

" border-bottom-color: var(--fill-color);\n",

" border-left-color: var(--fill-color);\n",

" }\n",

" 20% {\n",

" border-color: transparent;\n",

" border-left-color: var(--fill-color);\n",

" border-top-color: var(--fill-color);\n",

" }\n",

" 30% {\n",

" border-color: transparent;\n",

" border-left-color: var(--fill-color);\n",

" border-top-color: var(--fill-color);\n",

" border-right-color: var(--fill-color);\n",

" }\n",

" 40% {\n",

" border-color: transparent;\n",

" border-right-color: var(--fill-color);\n",

" border-top-color: var(--fill-color);\n",

" }\n",

" 60% {\n",

" border-color: transparent;\n",

" border-right-color: var(--fill-color);\n",

" }\n",

" 80% {\n",

" border-color: transparent;\n",

" border-right-color: var(--fill-color);\n",

" border-bottom-color: var(--fill-color);\n",

" }\n",

" 90% {\n",

" border-color: transparent;\n",

" border-bottom-color: var(--fill-color);\n",

" }\n",

" }\n",

"</style>\n",

"\n",

" <script>\n",

" async function quickchart(key) {\n",

" const quickchartButtonEl =\n",

" document.querySelector('#' + key + ' button');\n",

" quickchartButtonEl.disabled = true; // To prevent multiple clicks.\n",

" quickchartButtonEl.classList.add('colab-df-spinner');\n",

" try {\n",

" const charts = await google.colab.kernel.invokeFunction(\n",

" 'suggestCharts', [key], {});\n",

" } catch (error) {\n",

" console.error('Error during call to suggestCharts:', error);\n",

" }\n",

" quickchartButtonEl.classList.remove('colab-df-spinner');\n",

" quickchartButtonEl.classList.add('colab-df-quickchart-complete');\n",

" }\n",

" (() => {\n",

" let quickchartButtonEl =\n",

" document.querySelector('#df-64e80f9c-f924-4f33-8e16-e05d27f164cb button');\n",

" quickchartButtonEl.style.display =\n",

" google.colab.kernel.accessAllowed ? 'block' : 'none';\n",

" })();\n",

" </script>\n",

"</div>\n",

" </div>\n",

" </div>\n"

]

},

"metadata": {},

"execution\_count": 3

}

]

},

{

"cell\_type": "code",

"source": [

"df.info()"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "6kaMuKi870HW",

"outputId": "b9ed9cfa-afd6-416b-e013-401e6ca87912"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"<class 'pandas.core.frame.DataFrame'>\n",

"RangeIndex: 4760 entries, 0 to 4759\n",

"Data columns (total 21 columns):\n",

" # Column Non-Null Count Dtype \n",

"--- ------ -------------- ----- \n",

" 0 Movie\_ID 4760 non-null int64 \n",

" 1 Movie\_Title 4760 non-null object \n",

" 2 Movie\_Genre 4760 non-null object \n",

" 3 Movie\_Language 4760 non-null object \n",

" 4 Movie\_Budget 4760 non-null int64 \n",

" 5 Movie\_Popularity 4760 non-null float64\n",

" 6 Movie\_Release\_Date 4760 non-null object \n",

" 7 Movie\_Revenue 4760 non-null int64 \n",

" 8 Movie\_Runtime 4758 non-null float64\n",

" 9 Movie\_Vote 4760 non-null float64\n",

" 10 Movie\_Vote\_Count 4760 non-null int64 \n",

" 11 Movie\_Homepage 1699 non-null object \n",

" 12 Movie\_Keywords 4373 non-null object \n",

" 13 Movie\_Overview 4757 non-null object \n",

" 14 Movie\_Production\_House 4760 non-null object \n",

" 15 Movie\_Production\_Country 4760 non-null object \n",

" 16 Movie\_Spoken\_Language 4760 non-null object \n",

" 17 Movie\_Tagline 3942 non-null object \n",

" 18 Movie\_Cast 4733 non-null object \n",

" 19 Movie\_Crew 4760 non-null object \n",

" 20 Movie\_Director 4738 non-null object \n",

"dtypes: float64(3), int64(4), object(14)\n",

"memory usage: 781.1+ KB\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"df.shape"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "CSrojM8U74QK",

"outputId": "34516396-1158-401c-fc29-bba2ba47bb34"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"(4760, 21)"

]

},

"metadata": {},

"execution\_count": 5

}

]

},

{

"cell\_type": "code",

"source": [

"df.columns"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "eeDm-5Vr761m",

"outputId": "64319bae-e401-48ff-cd64-d05d25f89de2"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"Index(['Movie\_ID', 'Movie\_Title', 'Movie\_Genre', 'Movie\_Language',\n",

" 'Movie\_Budget', 'Movie\_Popularity', 'Movie\_Release\_Date',\n",

" 'Movie\_Revenue', 'Movie\_Runtime', 'Movie\_Vote', 'Movie\_Vote\_Count',\n",

" 'Movie\_Homepage', 'Movie\_Keywords', 'Movie\_Overview',\n",

" 'Movie\_Production\_House', 'Movie\_Production\_Country',\n",

" 'Movie\_Spoken\_Language', 'Movie\_Tagline', 'Movie\_Cast', 'Movie\_Crew',\n",

" 'Movie\_Director'],\n",

" dtype='object')"

]

},

"metadata": {},

"execution\_count": 6

}

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*Get Feature Selection\*\*\n",

"\n"

],

"metadata": {

"id": "\_yVpd4kAkS-6"

}

},

{

"cell\_type": "code",

"source": [

"df\_features = df[[ 'Movie\_Genre', 'Movie\_Keywords', 'Movie\_Tagline', 'Movie\_Cast', 'Movie\_Director']].fillna('')"

],

"metadata": {

"id": "Wcb1Zj72keAQ"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"df\_features.shape"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "vfYbeers8FuI",

"outputId": "49ea06c5-7dcf-452a-9274-fb2346722d9e"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"(4760, 5)"

]

},

"metadata": {},

"execution\_count": 8

}

]

},

{

"cell\_type": "code",

"source": [

"df\_features"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 423

},

"id": "lb9jHzZw8LC3",

"outputId": "33133255-69db-49e4-c736-b7680a12b7cf"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

" Movie\_Genre \\\n",

"0 Crime Comedy \n",

"1 Adventure Action Science Fiction \n",

"2 Animation Family \n",

"3 Comedy Drama Romance \n",

"4 Drama \n",

"... ... \n",

"4755 Horror \n",

"4756 Comedy Family Drama \n",

"4757 Thriller Drama \n",

"4758 Family \n",

"4759 Documentary \n",

"\n",

" Movie\_Keywords \\\n",

"0 hotel new year's eve witch bet hotel room \n",

"1 android galaxy hermit death star lightsaber \n",

"2 father son relationship harbor underwater fish... \n",

"3 vietnam veteran hippie mentally disabled runni... \n",

"4 male nudity female nudity adultery midlife cri... \n",

"... ... \n",

"4755 \n",

"4756 \n",

"4757 christian film sex trafficking \n",

"4758 \n",

"4759 music actors legendary perfomer classic hollyw... \n",

"\n",

" Movie\_Tagline \\\n",

"0 Twelve outrageous guests. Four scandalous requ... \n",

"1 A long time ago in a galaxy far, far away... \n",

"2 There are 3.7 trillion fish in the ocean, they... \n",

"3 The world will never be the same, once you've ... \n",

"4 Look closer. \n",

"... ... \n",

"4755 The hot spot where Satan's waitin'. \n",

"4756 It’s better to stand out than to fit in. \n",

"4757 She never knew it could happen to her... \n",

"4758 \n",

"4759 \n",

"\n",

" Movie\_Cast Movie\_Director \n",

"0 Tim Roth Antonio Banderas Jennifer Beals Madon... Allison Anders \n",

"1 Mark Hamill Harrison Ford Carrie Fisher Peter ... George Lucas \n",

"2 Albert Brooks Ellen DeGeneres Alexander Gould ... Andrew Stanton \n",

"3 Tom Hanks Robin Wright Gary Sinise Mykelti Wil... Robert Zemeckis \n",

"4 Kevin Spacey Annette Bening Thora Birch Wes Be... Sam Mendes \n",

"... ... ... \n",

"4755 Lisa Hart Carroll Michael Des Barres Paul Drak... Pece Dingo \n",

"4756 Roni Akurati Brighton Sharbino Jason Lee Anjul... Frank Lotito \n",

"4757 Nicole Smolen Kim Baldwin Ariana Stephens Brys... Jaco Booyens \n",

"4758 \n",

"4759 Tony Oppedisano Simon Napier-Bell \n",

"\n",

"[4760 rows x 5 columns]"

],

"text/html": [

"\n",

" <div id=\"df-cfe11a2f-a6f0-4f8c-b791-07d9958c9b70\" class=\"colab-df-container\">\n",

" <div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>Movie\_Genre</th>\n",

" <th>Movie\_Keywords</th>\n",

" <th>Movie\_Tagline</th>\n",

" <th>Movie\_Cast</th>\n",

" <th>Movie\_Director</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>Crime Comedy</td>\n",

" <td>hotel new year's eve witch bet hotel room</td>\n",

" <td>Twelve outrageous guests. Four scandalous requ...</td>\n",

" <td>Tim Roth Antonio Banderas Jennifer Beals Madon...</td>\n",

" <td>Allison Anders</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>Adventure Action Science Fiction</td>\n",

" <td>android galaxy hermit death star lightsaber</td>\n",

" <td>A long time ago in a galaxy far, far away...</td>\n",

" <td>Mark Hamill Harrison Ford Carrie Fisher Peter ...</td>\n",

" <td>George Lucas</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>Animation Family</td>\n",

" <td>father son relationship harbor underwater fish...</td>\n",

" <td>There are 3.7 trillion fish in the ocean, they...</td>\n",

" <td>Albert Brooks Ellen DeGeneres Alexander Gould ...</td>\n",

" <td>Andrew Stanton</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>Comedy Drama Romance</td>\n",

" <td>vietnam veteran hippie mentally disabled runni...</td>\n",

" <td>The world will never be the same, once you've ...</td>\n",

" <td>Tom Hanks Robin Wright Gary Sinise Mykelti Wil...</td>\n",

" <td>Robert Zemeckis</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>Drama</td>\n",

" <td>male nudity female nudity adultery midlife cri...</td>\n",

" <td>Look closer.</td>\n",

" <td>Kevin Spacey Annette Bening Thora Birch Wes Be...</td>\n",

" <td>Sam Mendes</td>\n",

" </tr>\n",

" <tr>\n",

" <th>...</th>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" <td>...</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4755</th>\n",

" <td>Horror</td>\n",

" <td></td>\n",

" <td>The hot spot where Satan's waitin'.</td>\n",

" <td>Lisa Hart Carroll Michael Des Barres Paul Drak...</td>\n",

" <td>Pece Dingo</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4756</th>\n",

" <td>Comedy Family Drama</td>\n",

" <td></td>\n",

" <td>It’s better to stand out than to fit in.</td>\n",

" <td>Roni Akurati Brighton Sharbino Jason Lee Anjul...</td>\n",

" <td>Frank Lotito</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4757</th>\n",

" <td>Thriller Drama</td>\n",

" <td>christian film sex trafficking</td>\n",

" <td>She never knew it could happen to her...</td>\n",

" <td>Nicole Smolen Kim Baldwin Ariana Stephens Brys...</td>\n",

" <td>Jaco Booyens</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4758</th>\n",

" <td>Family</td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" <td></td>\n",

" </tr>\n",

" <tr>\n",

" <th>4759</th>\n",

" <td>Documentary</td>\n",

" <td>music actors legendary perfomer classic hollyw...</td>\n",

" <td></td>\n",

" <td>Tony Oppedisano</td>\n",

" <td>Simon Napier-Bell</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"<p>4760 rows × 5 columns</p>\n",

"</div>\n",

" <div class=\"colab-df-buttons\">\n",

"\n",

" <div class=\"colab-df-container\">\n",

" <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-cfe11a2f-a6f0-4f8c-b791-07d9958c9b70')\"\n",

" title=\"Convert this dataframe to an interactive table.\"\n",

" style=\"display:none;\">\n",

"\n",

" <svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\" viewBox=\"0 -960 960 960\">\n",

" <path d=\"M120-120v-720h720v720H120Zm60-500h600v-160H180v160Zm220 220h160v-160H400v160Zm0 220h160v-160H400v160ZM180-400h160v-160H180v160Zm440 0h160v-160H620v160ZM180-180h160v-160H180v160Zm440 0h160v-160H620v160Z\"/>\n",

" </svg>\n",

" </button>\n",

"\n",

" <style>\n",

" .colab-df-container {\n",

" display:flex;\n",

" gap: 12px;\n",

" }\n",

"\n",

" .colab-df-convert {\n",

" background-color: #E8F0FE;\n",

" border: none;\n",

" border-radius: 50%;\n",

" cursor: pointer;\n",

" display: none;\n",

" fill: #1967D2;\n",

" height: 32px;\n",

" padding: 0 0 0 0;\n",

" width: 32px;\n",

" }\n",

"\n",

" .colab-df-convert:hover {\n",

" background-color: #E2EBFA;\n",

" box-shadow: 0px 1px 2px rgba(60, 64, 67, 0.3), 0px 1px 3px 1px rgba(60, 64, 67, 0.15);\n",

" fill: #174EA6;\n",

" }\n",

"\n",

" .colab-df-buttons div {\n",

" margin-bottom: 4px;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert {\n",

" background-color: #3B4455;\n",

" fill: #D2E3FC;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert:hover {\n",

" background-color: #434B5C;\n",

" box-shadow: 0px 1px 3px 1px rgba(0, 0, 0, 0.15);\n",

" filter: drop-shadow(0px 1px 2px rgba(0, 0, 0, 0.3));\n",

" fill: #FFFFFF;\n",

" }\n",

" </style>\n",

"\n",

" <script>\n",

" const buttonEl =\n",

" document.querySelector('#df-cfe11a2f-a6f0-4f8c-b791-07d9958c9b70 button.colab-df-convert');\n",

" buttonEl.style.display =\n",

" google.colab.kernel.accessAllowed ? 'block' : 'none';\n",

"\n",

" async function convertToInteractive(key) {\n",

" const element = document.querySelector('#df-cfe11a2f-a6f0-4f8c-b791-07d9958c9b70');\n",

" const dataTable =\n",

" await google.colab.kernel.invokeFunction('convertToInteractive',\n",

" [key], {});\n",

" if (!dataTable) return;\n",

"\n",

" const docLinkHtml = 'Like what you see? Visit the ' +\n",

" '<a target=\"\_blank\" href=https://colab.research.google.com/notebooks/data\_table.ipynb>data table notebook</a>'\n",

" + ' to learn more about interactive tables.';\n",

" element.innerHTML = '';\n",

" dataTable['output\_type'] = 'display\_data';\n",

" await google.colab.output.renderOutput(dataTable, element);\n",

" const docLink = document.createElement('div');\n",

" docLink.innerHTML = docLinkHtml;\n",

" element.appendChild(docLink);\n",

" }\n",

" </script>\n",

" </div>\n",

"\n",

"\n",

"<div id=\"df-cf53ec91-7e37-4144-a35c-3a9c5b11144e\">\n",

" <button class=\"colab-df-quickchart\" onclick=\"quickchart('df-cf53ec91-7e37-4144-a35c-3a9c5b11144e')\"\n",

" title=\"Suggest charts.\"\n",

" style=\"display:none;\">\n",

"\n",

"<svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\"viewBox=\"0 0 24 24\"\n",

" width=\"24px\">\n",

" <g>\n",

" <path d=\"M19 3H5c-1.1 0-2 .9-2 2v14c0 1.1.9 2 2 2h14c1.1 0 2-.9 2-2V5c0-1.1-.9-2-2-2zM9 17H7v-7h2v7zm4 0h-2V7h2v10zm4 0h-2v-4h2v4z\"/>\n",

" </g>\n",

"</svg>\n",

" </button>\n",

"\n",

"<style>\n",

" .colab-df-quickchart {\n",

" --bg-color: #E8F0FE;\n",

" --fill-color: #1967D2;\n",

" --hover-bg-color: #E2EBFA;\n",

" --hover-fill-color: #174EA6;\n",

" --disabled-fill-color: #AAA;\n",

" --disabled-bg-color: #DDD;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-quickchart {\n",

" --bg-color: #3B4455;\n",

" --fill-color: #D2E3FC;\n",

" --hover-bg-color: #434B5C;\n",

" --hover-fill-color: #FFFFFF;\n",

" --disabled-bg-color: #3B4455;\n",

" --disabled-fill-color: #666;\n",

" }\n",

"\n",

" .colab-df-quickchart {\n",

" background-color: var(--bg-color);\n",

" border: none;\n",

" border-radius: 50%;\n",

" cursor: pointer;\n",

" display: none;\n",

" fill: var(--fill-color);\n",

" height: 32px;\n",

" padding: 0;\n",

" width: 32px;\n",

" }\n",

"\n",

" .colab-df-quickchart:hover {\n",

" background-color: var(--hover-bg-color);\n",

" box-shadow: 0 1px 2px rgba(60, 64, 67, 0.3), 0 1px 3px 1px rgba(60, 64, 67, 0.15);\n",

" fill: var(--button-hover-fill-color);\n",

" }\n",

"\n",

" .colab-df-quickchart-complete:disabled,\n",

" .colab-df-quickchart-complete:disabled:hover {\n",

" background-color: var(--disabled-bg-color);\n",

" fill: var(--disabled-fill-color);\n",

" box-shadow: none;\n",

" }\n",

"\n",

" .colab-df-spinner {\n",

" border: 2px solid var(--fill-color);\n",

" border-color: transparent;\n",

" border-bottom-color: var(--fill-color);\n",

" animation:\n",

" spin 1s steps(1) infinite;\n",

" }\n",

"\n",

" @keyframes spin {\n",

" 0% {\n",

" border-color: transparent;\n",

" border-bottom-color: var(--fill-color);\n",

" border-left-color: var(--fill-color);\n",

" }\n",

" 20% {\n",

" border-color: transparent;\n",

" border-left-color: var(--fill-color);\n",

" border-top-color: var(--fill-color);\n",

" }\n",

" 30% {\n",

" border-color: transparent;\n",

" border-left-color: var(--fill-color);\n",

" border-top-color: var(--fill-color);\n",

" border-right-color: var(--fill-color);\n",

" }\n",

" 40% {\n",

" border-color: transparent;\n",

" border-right-color: var(--fill-color);\n",

" border-top-color: var(--fill-color);\n",

" }\n",

" 60% {\n",

" border-color: transparent;\n",

" border-right-color: var(--fill-color);\n",

" }\n",

" 80% {\n",

" border-color: transparent;\n",

" border-right-color: var(--fill-color);\n",

" border-bottom-color: var(--fill-color);\n",

" }\n",

" 90% {\n",

" border-color: transparent;\n",

" border-bottom-color: var(--fill-color);\n",

" }\n",

" }\n",

"</style>\n",

"\n",

" <script>\n",

" async function quickchart(key) {\n",

" const quickchartButtonEl =\n",

" document.querySelector('#' + key + ' button');\n",

" quickchartButtonEl.disabled = true; // To prevent multiple clicks.\n",

" quickchartButtonEl.classList.add('colab-df-spinner');\n",

" try {\n",

" const charts = await google.colab.kernel.invokeFunction(\n",

" 'suggestCharts', [key], {});\n",

" } catch (error) {\n",

" console.error('Error during call to suggestCharts:', error);\n",

" }\n",

" quickchartButtonEl.classList.remove('colab-df-spinner');\n",

" quickchartButtonEl.classList.add('colab-df-quickchart-complete');\n",

" }\n",

" (() => {\n",

" let quickchartButtonEl =\n",

" document.querySelector('#df-cf53ec91-7e37-4144-a35c-3a9c5b11144e button');\n",

" quickchartButtonEl.style.display =\n",

" google.colab.kernel.accessAllowed ? 'block' : 'none';\n",

" })();\n",

" </script>\n",

"</div>\n",

" </div>\n",

" </div>\n"

]

},

"metadata": {},

"execution\_count": 9

}

]

},

{

"cell\_type": "code",

"source": [

"x = df\_features['Movie\_Genre'] + '' + df\_features['Movie\_Keywords'] + '' + df\_features['Movie\_Tagline'] +''+ df\_features['Movie\_Cast']+ ''+ df\_features['Movie\_Director']"

],

"metadata": {

"id": "\_ytxKvp08OFN"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"x"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "98Dglwm\_nY7S",

"outputId": "05af5df1-e31e-4ccd-8182-b48d04c59f84"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"<4760x27466 sparse matrix of type '<class 'numpy.float64'>'\n",

"\twith 111276 stored elements in Compressed Sparse Row format>"

]

},

"metadata": {},

"execution\_count": 22

}

]

},

{

"cell\_type": "code",

"source": [

"x.shape"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "q-mCZkWo8TxF",

"outputId": "499bae12-b64d-4699-8614-0d76909beeb7"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"(4760,)"

]

},

"metadata": {},

"execution\_count": 11

}

]

},

{

"cell\_type": "markdown",

"source": [

"# Get Feature Text Conversion to Tokens"

],

"metadata": {

"id": "6ktGaplEnqQ6"

}

},

{

"cell\_type": "code",

"source": [

"from sklearn.feature\_extraction.text import TfidfVectorizer\n"

],

"metadata": {

"id": "n3RfUY3an2Hf"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"tfidf = TfidfVectorizer()\n"

],

"metadata": {

"id": "DhXgmLLVofc9"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"x = tfidf.fit\_transform(x)\n"

],

"metadata": {

"id": "pRHHv0PPolfF"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"x.shape"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "CPSJBOc3o1FJ",

"outputId": "56a7ddfa-48aa-4e39-db70-8bf22b4de41f"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"(4760, 27466)"

]

},

"metadata": {},

"execution\_count": 15

}

]

},

{

"cell\_type": "code",

"source": [

"print(x)"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "w\_a-F7a4o5-Z",

"outputId": "15efdbb5-8d37-4274-91a2-fa5b6cec26f8"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

" (0, 1028)\t0.16196019146631543\n",

" (0, 24785)\t0.1954632929283795\n",

" (0, 15844)\t0.14205053053187272\n",

" (0, 15553)\t0.17099186675469502\n",

" (0, 2132)\t0.18002354204307464\n",

" (0, 13312)\t0.09914387783149516\n",

" (0, 1887)\t0.14106037409792174\n",

" (0, 1216)\t0.13920306109638164\n",

" (0, 21158)\t0.14205053053187272\n",

" (0, 24701)\t0.11357423942624927\n",

" (0, 14943)\t0.091376722056839\n",

" (0, 18098)\t0.06200430666985742\n",

" (0, 26738)\t0.175053052455033\n",

" (0, 9790)\t0.08712552095655665\n",

" (0, 26675)\t0.1116831168780693\n",

" (0, 13401)\t0.13748876529263096\n",

" (0, 24105)\t0.10726395493180996\n",

" (0, 18192)\t0.07278761942152372\n",

" (0, 6172)\t0.11970212451073885\n",

" (0, 9626)\t0.11757910435818826\n",

" (0, 11960)\t0.20134029899961134\n",

" (0, 12801)\t0.1530338818199682\n",

" (0, 2292)\t0.1954632929283795\n",

" (0, 15172)\t0.1537691763994982\n",

" (0, 18196)\t0.08579029869987485\n",

" :\t:\n",

" (4757, 1839)\t0.19327629083107672\n",

" (4757, 5410)\t0.19734759150400596\n",

" (4757, 11350)\t0.21582294886514122\n",

" (4757, 22017)\t0.1646400247918531\n",

" (4757, 17789)\t0.18881341937258544\n",

" (4757, 9484)\t0.1411164779725638\n",

" (4757, 14176)\t0.2330831990045816\n",

" (4757, 11762)\t0.17321388936472645\n",

" (4757, 14052)\t0.1776312353410007\n",

" (4757, 24232)\t0.10947784435203887\n",

" (4757, 24746)\t0.09744940789814222\n",

" (4757, 13079)\t0.12400374714145113\n",

" (4757, 17721)\t0.1489085353667712\n",

" (4758, 8651)\t1.0\n",

" (4759, 18229)\t0.33527342183765224\n",

" (4759, 22434)\t0.33527342183765224\n",

" (4759, 18841)\t0.33527342183765224\n",

" (4759, 6950)\t0.33527342183765224\n",

" (4759, 345)\t0.31978160936741457\n",

" (4759, 14742)\t0.31978160936741457\n",

" (4759, 12139)\t0.2778063685558062\n",

" (4759, 4446)\t0.282306565154911\n",

" (4759, 17552)\t0.3087899934962816\n",

" (4759, 9955)\t0.21805075638656476\n",

" (4759, 2285)\t0.21465229435984196\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*Get Similarity Score using Cosine Similarity\*\*\n",

"\n",

"cosine\_similarity computes the L2-normalized dot product of vectors. Euclidean (L2) normalization projects the vectors onto the unit sphere, and their dot product is then the cosine of the angle between the points denoted by the vectors.\n"

],

"metadata": {

"id": "F1s2B2S9pO2H"

}

},

{

"cell\_type": "code",

"source": [

" from sklearn.metrics.pairwise import cosine\_similarity\n"

],

"metadata": {

"id": "1zFMR2LZpg\_G"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"Similarity\_Score = cosine\_similarity(x)\n"

],

"metadata": {

"id": "6Dw7zlukppaW"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"Similarity\_Score"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "tkTqhzqQqAZ6",

"outputId": "c369172a-cb4f-4840-d3db-4a5e16789a55"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"array([[1. , 0.01438634, 0.03807033, ..., 0. , 0. ,\n",

" 0. ],\n",

" [0.01438634, 1. , 0.00844858, ..., 0. , 0. ,\n",

" 0. ],\n",

" [0.03807033, 0.00844858, 1. , ..., 0. , 0. ,\n",

" 0. ],\n",

" ...,\n",

" [0. , 0. , 0. , ..., 1. , 0. ,\n",

" 0. ],\n",

" [0. , 0. , 0. , ..., 0. , 1. ,\n",

" 0. ],\n",

" [0. , 0. , 0. , ..., 0. , 0. ,\n",

" 1. ]])"

]

},

"metadata": {},

"execution\_count": 19

}

]

},

{

"cell\_type": "code",

"source": [

"Similarity\_Score.shape"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "\_Cj3mKCWqJqO",

"outputId": "2546b158-ed95-4836-f419-f4e39cd60bd2"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"(4760, 4760)"

]

},

"metadata": {},

"execution\_count": 20

}

]

},

{

"cell\_type": "markdown",

"source": [

"\*\*Get Movie Name as Input from User and Validate for Closest Spelling\*\*"

],

"metadata": {

"id": "07ygpSC0qN01"

}

},

{

"cell\_type": "code",

"source": [

"Favourite\_Movie\_Name = input(' Enter your favourite movie name :')\n"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "UjzxHG6QrA2e",

"outputId": "adfc14f0-ee5e-4f9c-f976-85f9d25b80f2"

},

"execution\_count": null,

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

" Enter your favourite movie name :Star Wars\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"All\_Movies\_Title\_List = df['Movie\_Title'].tolist()"

],

"metadata": {

"id": "Qtq2jnvfshlJ"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"import difflib"

],

"metadata": {

"id": "r6dk2ESPsqLB"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"Movie\_Recommendation = difflib.get\_close\_matches (Favourite\_Movie\_Name, All\_Movies\_Title\_List)\n",

"print(Movie\_Recommendation)\n"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "9NGk22JGsvan",

"outputId": "0aa5bb91-1d9f-45d2-99f9-c37b48156926"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"['Star Wars', 'Star Trek', 'State Fair']\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"Close\_Match = Movie\_Recommendation[0]\n",

"print (Close\_Match)"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "RNLvKO979-nq",

"outputId": "32974c64-4c05-4dc8-c156-81218ceaf74f"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Star Wars\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"\n",

"Index\_of\_Close\_Match\_Movie = df [df.Movie\_Title == Close\_Match]['Movie\_ID'].values[0]\n",

"print(Index\_of\_Close\_Match\_Movie)\n"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "tzQUudQmtYiQ",

"outputId": "e729dadc-62c3-4d9b-a310-9a7c54209884"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"2\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"# getting a list of similar movies\n",

"\n",

"Recommendation\_Score = list(enumerate(Similarity\_Score[Index\_of\_Close\_Match\_Movie]))\n",

"print (Recommendation\_Score)\n",

"\n",

"\n"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "rz-GrRlxuOEW",

"outputId": "558bf5b2-5c7c-4811-ff3e-4a20d46764a1"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

]

}

]

},

{

"cell\_type": "code",

"source": [

" len(Recommendation\_Score)\n",

"\n"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "qQPdvpMevJnh",

"outputId": "510411d8-0cea-48f7-80c5-7a045de70bc7"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"4760"

]

},

"metadata": {},

"execution\_count": 30

}

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*Get All Movies Sorted Based on Recommendation Score for your Favourite Movie\*\*\n"

],

"metadata": {

"id": "gxu7GYlmvUeP"

}

},

{

"cell\_type": "code",

"source": [

"#sorting the movies based on their similarity score\n",

"\n",

"Sorted\_Similar\_Movies = sorted(Recommendation\_Score, key = lambda x:x[1], reverse=True)\n",

"print (Sorted\_Similar\_Movies)\n"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "JDksS\_ukvMbC",

"outputId": "c113a429-7fc6-4569-e399-f2c313cfc556"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text":

}

]

},

{

"cell\_type": "code",

"source": [

"# print the name of similar movies based on the index\n",

"\n",

"print('Top 30 Movies Suggested for You :\\ n ')\n",

"\n",

"i=1\n",

"\n",

"for movie in Sorted\_Similar\_Movies:\n",

" index = movie[0]\n",

" title\_from\_index = df [df.index==index]['Movie\_Title'].values[0]\n",

" if (i<31):\n",

" print(i, '.',title\_from\_index)\n",

" i+=1"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "FnM9k7sWvi-R",

"outputId": "e3f7edd4-c232-46f2-e1ab-b837e802819f"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Top 30 Movies Suggested for You :\\ n \n",

"1 . Finding Nemo\n",

"2 . Big Fish\n",

"3 . John Carter\n",

"4 . Spider-Man\n",

"5 . Shark Tale\n",

"6 . Flight of the Intruder\n",

"7 . El Mariachi\n",

"8 . Shooting Fish\n",

"9 . The Shaggy Dog\n",

"10 . The Muse\n",

"11 . Freaky Friday\n",

"12 . American Dreamz\n",

"13 . The Outsiders\n",

"14 . Mr. Peabody & Sherman\n",

"15 . The Simpsons Movie\n",

"16 . Tora! Tora! Tora!\n",

"17 . Happy Feet\n",

"18 . xXx: State of the Union\n",

"19 . Indie Game: The Movie\n",

"20 . The English Patient\n",

"21 . Because of Winn-Dixie\n",

"22 . Atlantis: The Lost Empire\n",

"23 . Ponyo\n",

"24 . Evan Almighty\n",

"25 . White Chicks\n",

"26 . The Mask\n",

"27 . The Rookie\n",

"28 . Troy\n",

"29 . Jonah: A VeggieTales Movie\n",

"30 . Mallrats\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*Top 10 Movies Recommended Based on Your Favorite Movie\*\*\n"

],

"metadata": {

"id": "f3WmuvMQyHUg"

}

},

{

"cell\_type": "code",

"source": [

"import difflib\n",

"\n",

"Movie\_Name = input('Enter your favorite movie name: ')\n",

"\n",

"list\_of\_all\_titles = df['Movie\_Title'].tolist()\n",

"\n",

"# Find close matches to the input movie name\n",

"close\_matches = difflib.get\_close\_matches(Movie\_Name, list\_of\_all\_titles)\n",

"\n",

"if close\_matches:\n",

" closest\_match = close\_matches[0] # Get the closest match\n",

" Index\_of\_Movie = df[df.Movie\_Title == closest\_match]['Movie\_ID'].values[0]\n",

"\n",

" Recommendation\_Score = list(enumerate(Similarity\_Score[Index\_of\_Movie]))\n",

"\n",

" sorted\_similar\_movies = sorted(Recommendation\_Score, key=lambda x: x[1], reverse=True)\n",

"\n",

" print('Top 10 Movies suggested for you: \\n')\n",

"\n",

" i = 1\n",

"\n",

" for movie in sorted\_similar\_movies:\n",

" index = movie[0]\n",

" if index < len(df):\n",

" title\_from\_index = df[df.Movie\_ID == index]['Movie\_Title'].values[0]\n",

" print(i, '.', title\_from\_index)\n",

" i += 1\n",

" else:\n",

" print(\"Invalid index:\", index)\n",

"\n",

" if i > 10:\n",

" break\n",

"else:\n",

" print('No close matches found for the entered movie name.')\n"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "dzYW8eWH0tKP",

"outputId": "5bd630d0-812c-47c1-c0a2-45a52360b78f"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Enter your favorite movie name: Forest Gump\n",

"Top 10 Movies suggested for you: \n",

"\n",

"1 . Forrest Gump\n",

"2 . Heaven is for Real\n",

"3 . Rampage\n",

"4 . Miss Potter\n",

"5 . Juno\n",

"6 . From Paris with Love\n",

"7 . Hannibal Rising\n",

"8 . Herbie Fully Loaded\n",

"9 . Just Go with It\n",

"10 . Ghosts of Mars\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"\n",

"\n",

"\n",

"Movie Match, the advanced Movie Recommendation System, is powered by the \*\*Close Match algorithm\*\*, ensuring precise and personalized movie suggestions. By employing fuzzy matching techniques, it adeptly handles minor input variations, such as typos or incomplete titles. This innovative approach guarantees spot-on recommendations, tailored exclusively for each user's cinematic taste.\n",

"\n",

"\n",

"\n",

"In conclusion, Movie Match offers a seamless and immersive movie-watching experience. I'm delighted to assist you in discovering movies perfectly aligned with your preferences. Thank you for choosing Movie Match. For any inquiries or assistance, please don't hesitate to contact me. Happy movie watching!\n",

"\n",

"\n",

"Thank you for exploring our Movie Recommendation System! I hope you enjoy your personalized movie suggestions. If you have any questions or feedback, feel free to reach out. Happy movie watching!"

],

"metadata": {

"id": "i\_GCDc4MBITG"

}

}

]

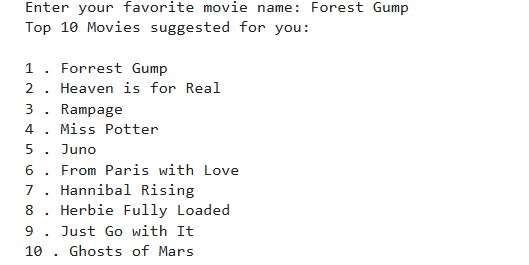
}

**14. Future Scope**

Integrate real-time feedback to adapt recommendations dynamically

Include user sentiment analysis from reviews for deeper personalization

Deploy fully functional web app with live user login and database



**---**

**15. Team Members and Roles**

**R. Harini –** Team Lead, Model Building, Deep Learning

**H. Ayisha Siddiqha –** Data Collection, Cleaning, Preprocessing

**P. Hemapriya –** EDA, Visualizations and insights documentation

**K. Magizharasi –** Model Evaluation, Final Report Writing and presentation