```
{
 "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_fo
undation-
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",
"      <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",
"      <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)\t0.1392030610963816 4\n",
" (0, 21158)\t0.1420505305318727 2\n",
" (0, 24701)\t0.1135742394262492 7\n",
" (0, 14943)\t0.091376722056839\n",
" (0, 18098)\t0.0620043066698574 2\n",
" (0, 26738)\t0.175053052455033\n",
" (0, 9790)\t0.0871255209565566 5\n",
" (0, 26675)\t0.1116831168780693\n",
" (0, 13401)\t0.1374887652926309 6\n",
" (0, 24105)\t0.1072639549318099 6\n",
" (0, 18192)\t0.0727876194215237 2\n",
" (0, 6172)\t0.1197021245107388 5\n",
" (0, 9626)\t0.1175791043581882 6\n",
" (0, 11960)\t0.2013402989996113 4\n",
" (0, 12801)\t0.153033881819968 2\n",
" (0, 2292)\t0.1954632929283795\n",
" (0, 15172)\t0.153769176399498 2\n",
" (0, 18196)\t0.0857902986998748 5\n",
" :\t:\n",
" (4757, 1839)\t0.1932762908310767 2\n",
" (4757, 5410)\t0.1973475915040059 6\n",
" (4757, 11350)\t0.2158229488651412 2\n",
" (4757, 22017)\t0.164640024791853 1\n",
" (4757, 17789)\t0.1888134193725854 4\n",
" (4757, 9484)\t0.141116477972563 8\n",
" (4757, 14176)\t0.233083199004581 6\n",
" (4757, 11762)\t0.173213889364726 45\n",
" (4757, 14052)\t0.177631235341000 7\n",
" (4757, 24232)\t0.109477844352038 87\n",
" (4757, 24746)\t0.097449407898142 22\n",
" (4757, 13079)\t0.124003747141451 13\n",
" (4757, 17721)\t0.148908535366771 2\n",
" (4758, 8651)\t1.0\n",
" (4759, 18229)\t0.335273421837652 24\n",
" (4759, 22434)\t0.335273421837652 24\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",
       "       [0.01438634, 1.        , 0.01438634, ..., 0.        ,
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"
    }
   }
```