{

"metadata": {

"kernelspec": {

"name": "python",

"display\_name": "Python (Pyodide)",

"language": "python"

},

"language\_info": {

"codemirror\_mode": {

"name": "python",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.8"

}

},

"nbformat\_minor": 5,

"nbformat": 4,

"cells": [

{

"id": "5b312935",

"cell\_type": "markdown",

"source": "# IMBD Top Movies ",

"metadata": {}

},

{

"id": "84b7eeaa",

"cell\_type": "markdown",

"source": "# 1. Data Collection\n\*\*\*\n\n## Task: Data Collection:\n\n### Done By: Norah Mohammed Alwohaibi\n\n### Date: 12-March-2024 \n\nIn this section we will collect the row data, from IMDB website (https://www.imdb.com/list/ls063676189/) by web scrapping using this code below:\n### First, import libraries:",

"metadata": {}

},

{

"id": "13a0cb2a",

"cell\_type": "code",

"source": "import requests\nfrom bs4 import BeautifulSoup#for html\nimport pandas as pd\nimport numpy as np\nimport csv\nimport pandas as pd\nfrom sklearn.model\_selection import train\_test\_split\nfrom sklearn.preprocessing import OneHotEncoder, LabelEncoder\nfrom sklearn.compose import ColumnTransformer\nfrom sklearn.pipeline import Pipeline",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 45

},

{

"id": "7f9a8b08",

"cell\_type": "markdown",

"source": "## Load the dataset and import libraries:",

"metadata": {}

},

{

"id": "1eed96ec",

"cell\_type": "code",

"source": "import pandas as pd\nfrom collections import Counter\nimport matplotlib.pyplot as plt\n%pip install seaborn\nimport seaborn as sns\nimport numpy as np\n\n\n# Load the dataset\nfile\_path = 'movies.csv'\ndata = pd.read\_csv(file\_path)\n\n# Display the first few rows of the dataset\ndata.head()\n\n",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 1,

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

"data": {

"text/plain": " Title Release year \\\n0 The Shawshank Redemption 1994 \n1 The Dark Knight 2008 \n2 Inception 2010 \n3 Fight Club 1999 \n4 Pulp Fiction 1994 \n\n Plot summary \\\n0 Over the course of several years, two convicts... \n1 When the menace known as the Joker wreaks havo... \n2 A thief who steals corporate secrets through t... \n3 An insomniac office worker and a devil-may-car... \n4 The lives of two mob hitmen, a boxer, a gangst... \n\n Genre Rating Runtime IMDb rating Metascore \\\n0 Drama 9.3 142 min 9.3 82.0 \n1 Action, Crime, Drama 9.0 152 min 9.0 84.0 \n2 Action, Adventure, Sci-Fi 8.8 148 min 8.8 74.0 \n3 Drama 8.8 139 min 8.8 67.0 \n4 Crime, Drama 8.9 154 min 8.9 95.0 \n\n Director Stars \\\n0 Frank Darabont Tim Robbins, Morgan Freeman, Bob Gunton, Willi... \n1 Christopher Nolan Christian Bale, Heath Ledger, Aaron Eckhart, M... \n2 Christopher Nolan Leonardo DiCaprio, Joseph Gordon-Levitt, Ellio... \n3 David Fincher Brad Pitt, Edward Norton, Meat Loaf, Zach Grenier \n4 Quentin Tarantino John Travolta, Uma Thurman, Samuel L. Jackson,... \n\n Votes Gross \n0 2,869,913 $28.34M \n1 2,851,842 $534.86M \n2 2,532,959 $292.58M \n3 2,305,364 $37.03M \n4 2,204,248 $107.93M ",

"text/html": "<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>Title</th>\n <th>Release year</th>\n <th>Plot summary</th>\n <th>Genre</th>\n <th>Rating</th>\n <th>Runtime</th>\n <th>IMDb rating</th>\n <th>Metascore</th>\n <th>Director</th>\n <th>Stars</th>\n <th>Votes</th>\n <th>Gross</th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <th>0</th>\n <td>The Shawshank Redemption</td>\n <td>1994</td>\n <td>Over the course of several years, two convicts...</td>\n <td>Drama</td>\n <td>9.3</td>\n <td>142 min</td>\n <td>9.3</td>\n <td>82.0</td>\n <td>Frank Darabont</td>\n <td>Tim Robbins, Morgan Freeman, Bob Gunton, Willi...</td>\n <td>2,869,913</td>\n <td>$28.34M</td>\n </tr>\n <tr>\n <th>1</th>\n <td>The Dark Knight</td>\n <td>2008</td>\n <td>When the menace known as the Joker wreaks havo...</td>\n <td>Action, Crime, Drama</td>\n <td>9.0</td>\n <td>152 min</td>\n <td>9.0</td>\n <td>84.0</td>\n <td>Christopher Nolan</td>\n <td>Christian Bale, Heath Ledger, Aaron Eckhart, M...</td>\n <td>2,851,842</td>\n <td>$534.86M</td>\n </tr>\n <tr>\n <th>2</th>\n <td>Inception</td>\n <td>2010</td>\n <td>A thief who steals corporate secrets through t...</td>\n <td>Action, Adventure, Sci-Fi</td>\n <td>8.8</td>\n <td>148 min</td>\n <td>8.8</td>\n <td>74.0</td>\n <td>Christopher Nolan</td>\n <td>Leonardo DiCaprio, Joseph Gordon-Levitt, Ellio...</td>\n <td>2,532,959</td>\n <td>$292.58M</td>\n </tr>\n <tr>\n <th>3</th>\n <td>Fight Club</td>\n <td>1999</td>\n <td>An insomniac office worker and a devil-may-car...</td>\n <td>Drama</td>\n <td>8.8</td>\n <td>139 min</td>\n <td>8.8</td>\n <td>67.0</td>\n <td>David Fincher</td>\n <td>Brad Pitt, Edward Norton, Meat Loaf, Zach Grenier</td>\n <td>2,305,364</td>\n <td>$37.03M</td>\n </tr>\n <tr>\n <th>4</th>\n <td>Pulp Fiction</td>\n <td>1994</td>\n <td>The lives of two mob hitmen, a boxer, a gangst...</td>\n <td>Crime, Drama</td>\n <td>8.9</td>\n <td>154 min</td>\n <td>8.9</td>\n <td>95.0</td>\n <td>Quentin Tarantino</td>\n <td>John Travolta, Uma Thurman, Samuel L. Jackson,...</td>\n <td>2,204,248</td>\n <td>$107.93M</td>\n </tr>\n </tbody>\n</table>\n</div>"

},

"metadata": {}

}

],

"execution\_count": 1

},

{

"id": "a620c38b",

"cell\_type": "markdown",

"source": "\n\n# 2. Data Processing and Cleaning: \n\*\*\*\n\n## Task: Data Cleaning:\n### Done By: Deem Alshaye\n### Date: 16-March-2024 \n\n### Handling missing values:",

"metadata": {}

},

{

"id": "6035a254",

"cell\_type": "code",

"source": "# Step 1: Check for Missing Values\ndata.isnull().sum()\n\n",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 2,

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

"data": {

"text/plain": "Title 0\nRelease year 0\nPlot summary 0\nGenre 0\nRating 0\nRuntime 0\nIMDb rating 0\nMetascore 27\nDirector 0\nStars 0\nVotes 0\nGross 39\ndtype: int64"

},

"metadata": {}

}

],

"execution\_count": 2

},

{

"id": "ab0f83f2",

"cell\_type": "markdown",

"source": "What was done: Checked for the presence of missing values in each column of the dataset.\nWhy it was necessary: Identifying missing values is crucial to assess the quality of the data and decide on appropriate handling techniques.\n\ntools used:pandas",

"metadata": {}

},

{

"id": "829d3c9b",

"cell\_type": "markdown",

"source": "### Data Type Conversion",

"metadata": {}

},

{

"id": "ab5b61ef",

"cell\_type": "code",

"source": "# Step 2: Data Type Conversion\n# Convert 'Release year' to integer\ndata['Release year'] = pd.to\_numeric(data['Release year'], errors='coerce')\n\n# Convert 'IMDb rating' to float\ndata['IMDb rating'] = pd.to\_numeric(data['IMDb rating'], errors='coerce')\n\n# Convert 'votes' to integer\ndata['Votes'] = data['Votes'].str.replace(',', '').astype(int)\n\n",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 3

},

{

"id": "1dc72226",

"cell\_type": "markdown",

"source": "What was done: Converted the 'Release year' column to integer and 'Votes' column to integer and the 'IMDb rating' column to float.\nWhy it was necessary: Ensuring that data types are consistent with the data they represent is important for analysis and computation. For example, the release year should be an integer, and ratings should be floating-point numbers.\n\ntools used:pandas",

"metadata": {}

},

{

"id": "0caecabb",

"cell\_type": "markdown",

"source": "### Remove Unnecessary Characters:",

"metadata": {}

},

{

"id": "c7b3dc63",

"cell\_type": "code",

"source": "# Step 3: Remove Unnecessary Characters\n# Remove 'min' from 'Runtime' and convert to integer\ndata['Runtime'] = data['Runtime'].str.replace(' min', '').astype(float)\n\n# Remove '$' and 'M' from 'Gross', convert to float, and multiply by 1 million\ndata['Gross'] = data['Gross'].str.replace('[\\$\\,M]', '', regex=True).astype(float) \* 1e6\n",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 4

},

{

"id": "88b21cd3",

"cell\_type": "markdown",

"source": "What was done: Removed non-numeric characters from the 'Runtime' and 'Gross' columns and converted them to appropriate numeric types.\nWhy it was necessary: Numeric columns containing non-numeric characters cannot be used for mathematical operations. Cleaning these columns allows for accurate calculations and analysis.\ntools used:pandas",

"metadata": {}

},

{

"id": "5fcd8d8a",

"cell\_type": "markdown",

"source": "### Splitting Columns ",

"metadata": {}

},

{

"id": "0f821d03",

"cell\_type": "code",

"source": "# Step 4: Splitting Columns \ndata['Stars'] = data['Stars'].str.split(', ')\n# Split the 'Genre' column into a list of genres\ndata['Genre'] = data['Genre'].str.split(', ')\n",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 5

},

{

"id": "32e7b9ed",

"cell\_type": "markdown",

"source": "What was done: Split the 'Stars' column into a list of individual names.\nWhy it was necessary: The 'Stars' column contained multiple names in a single string, which is not ideal for analysis. Splitting the names into a list allows for easier manipulation and analysis of individual names.\n\ntools used:pandas",

"metadata": {}

},

{

"id": "2fa09155",

"cell\_type": "markdown",

"source": "### Handling missing values",

"metadata": {}

},

{

"id": "79dc780e",

"cell\_type": "code",

"source": "# Handling missing values\n\n# For 'Metascore', we'll fill missing values with the median\nmetascore\_median = data['Metascore'].median()\ndata['Metascore'].fillna(metascore\_median, inplace=True)\n\n# For 'Gross', we'll fill missing values with the median\ngross\_median = data['Gross'].median()\ndata['Gross'].fillna(gross\_median, inplace=True)\n\n# Fill missing values in 'Release year' with the median\nrelease\_year\_median = data['Release year'].median()\ndata['Release year'].fillna(release\_year\_median, inplace=True)\n\n# Check if there are any missing values left\ndata.isnull().sum()\n",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 6,

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

"data": {

"text/plain": "Title 0\nRelease year 0\nPlot summary 0\nGenre 0\nRating 0\nRuntime 0\nIMDb rating 0\nMetascore 0\nDirector 0\nStars 0\nVotes 0\nGross 0\ndtype: int64"

},

"metadata": {}

}

],

"execution\_count": 6

},

{

"id": "57db4d84",

"cell\_type": "markdown",

"source": "What was done: Filled missing values in the 'Metascore', 'Gross', and 'Release year' columns with their respective median values. Why it was necessary: Missing values can affect the results of analyses and computations. Filling them with a central tendency measure is a common practice to maintain the overall distribution of the data.\n\ntools used:pandas",

"metadata": {}

},

{

"id": "f7b398ef",

"cell\_type": "markdown",

"source": "### Extracting",

"metadata": {}

},

{

"id": "bf097f07",

"cell\_type": "code",

"source": "# Extract four-digit year from each value in the \"Release year\" column\ndata['Release year'] = data['Release year'].astype(str).str.extract(r'(\\d{4})')\n\n# Convert the extracted years to integers\ndata['Release year'] = data['Release year'].astype(int)\n\n\n# Verify the changes by printing unique values in ascending order\nprint(sorted(data['Release year'].unique()))",

"metadata": {

"scrolled": true,

"trusted": true

},

"outputs": [

{

"name": "stdout",

"text": "[1921, 1922, 1925, 1927, 1931, 1933, 1934, 1936, 1937, 1939, 1940, 1941, 1942, 1944, 1946, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016]\n",

"output\_type": "stream"

}

],

"execution\_count": 7

},

{

"id": "7ad10008",

"cell\_type": "markdown",

"source": "What was done: The code snippet extracts the four-digit year from each value in the \"Release year\" column of the DataFrame `data` and assigns it back to the same column. It uses the `astype(str)` method to convert the values to strings, and then applies the `str.extract(r'(\\d{4})')` method with a regular expression pattern `(\\d{4})` to extract the four-digit year and then convert it into integer.\n\nWhy it was necessary: This operation is necessary to ensure that the \"Release year\" column contains only the year information in a consistent format (four digits). This can be useful for various data analysis and visualization tasks where having the year in a standardized format is important.\n\ntools used:pandas",

"metadata": {}

},

{

"id": "33363647-5603-47d7-945e-35f51a1a8e1e",

"cell\_type": "markdown",

"source": "### cleaned data:",

"metadata": {}

},

{

"id": "3e7fb7e7",

"cell\_type": "code",

"source": "# Display the cleaned dataset \ndata.head()",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 8,

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

"data": {

"text/plain": " Title Release year \\\n0 The Shawshank Redemption 1994 \n1 The Dark Knight 2008 \n2 Inception 2010 \n3 Fight Club 1999 \n4 Pulp Fiction 1994 \n\n Plot summary \\\n0 Over the course of several years, two convicts... \n1 When the menace known as the Joker wreaks havo... \n2 A thief who steals corporate secrets through t... \n3 An insomniac office worker and a devil-may-car... \n4 The lives of two mob hitmen, a boxer, a gangst... \n\n Genre Rating Runtime IMDb rating Metascore \\\n0 [Drama] 9.3 142.0 9.3 82.0 \n1 [Action, Crime, Drama] 9.0 152.0 9.0 84.0 \n2 [Action, Adventure, Sci-Fi] 8.8 148.0 8.8 74.0 \n3 [Drama] 8.8 139.0 8.8 67.0 \n4 [Crime, Drama] 8.9 154.0 8.9 95.0 \n\n Director Stars \\\n0 Frank Darabont [Tim Robbins, Morgan Freeman, Bob Gunton, Will... \n1 Christopher Nolan [Christian Bale, Heath Ledger, Aaron Eckhart, ... \n2 Christopher Nolan [Leonardo DiCaprio, Joseph Gordon-Levitt, Elli... \n3 David Fincher [Brad Pitt, Edward Norton, Meat Loaf, Zach Gre... \n4 Quentin Tarantino [John Travolta, Uma Thurman, Samuel L. Jackson... \n\n Votes Gross \n0 2869913 28340000.0 \n1 2851842 534860000.0 \n2 2532959 292580000.0 \n3 2305364 37030000.0 \n4 2204248 107930000.0 ",

"text/html": "<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>Title</th>\n <th>Release year</th>\n <th>Plot summary</th>\n <th>Genre</th>\n <th>Rating</th>\n <th>Runtime</th>\n <th>IMDb rating</th>\n <th>Metascore</th>\n <th>Director</th>\n <th>Stars</th>\n <th>Votes</th>\n <th>Gross</th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <th>0</th>\n <td>The Shawshank Redemption</td>\n <td>1994</td>\n <td>Over the course of several years, two convicts...</td>\n <td>[Drama]</td>\n <td>9.3</td>\n <td>142.0</td>\n <td>9.3</td>\n <td>82.0</td>\n <td>Frank Darabont</td>\n <td>[Tim Robbins, Morgan Freeman, Bob Gunton, Will...</td>\n <td>2869913</td>\n <td>28340000.0</td>\n </tr>\n <tr>\n <th>1</th>\n <td>The Dark Knight</td>\n <td>2008</td>\n <td>When the menace known as the Joker wreaks havo...</td>\n <td>[Action, Crime, Drama]</td>\n <td>9.0</td>\n <td>152.0</td>\n <td>9.0</td>\n <td>84.0</td>\n <td>Christopher Nolan</td>\n <td>[Christian Bale, Heath Ledger, Aaron Eckhart, ...</td>\n <td>2851842</td>\n <td>534860000.0</td>\n </tr>\n <tr>\n <th>2</th>\n <td>Inception</td>\n <td>2010</td>\n <td>A thief who steals corporate secrets through t...</td>\n <td>[Action, Adventure, Sci-Fi]</td>\n <td>8.8</td>\n <td>148.0</td>\n <td>8.8</td>\n <td>74.0</td>\n <td>Christopher Nolan</td>\n <td>[Leonardo DiCaprio, Joseph Gordon-Levitt, Elli...</td>\n <td>2532959</td>\n <td>292580000.0</td>\n </tr>\n <tr>\n <th>3</th>\n <td>Fight Club</td>\n <td>1999</td>\n <td>An insomniac office worker and a devil-may-car...</td>\n <td>[Drama]</td>\n <td>8.8</td>\n <td>139.0</td>\n <td>8.8</td>\n <td>67.0</td>\n <td>David Fincher</td>\n <td>[Brad Pitt, Edward Norton, Meat Loaf, Zach Gre...</td>\n <td>2305364</td>\n <td>37030000.0</td>\n </tr>\n <tr>\n <th>4</th>\n <td>Pulp Fiction</td>\n <td>1994</td>\n <td>The lives of two mob hitmen, a boxer, a gangst...</td>\n <td>[Crime, Drama]</td>\n <td>8.9</td>\n <td>154.0</td>\n <td>8.9</td>\n <td>95.0</td>\n <td>Quentin Tarantino</td>\n <td>[John Travolta, Uma Thurman, Samuel L. Jackson...</td>\n <td>2204248</td>\n <td>107930000.0</td>\n </tr>\n </tbody>\n</table>\n</div>"

},

"metadata": {}

}

],

"execution\_count": 8

},

{

"id": "ece7e4da-0408-4ddd-a1d8-48b593a1cf73",

"cell\_type": "markdown",

"source": "# 3. Preprocessing\n\*\*\*\n## Task: Preprocessing: \n### Done By: Arwa Mesloub\n### Date: 20-March-2024 \n### Normilazation:",

"metadata": {}

},

{

"id": "c2f08af6-b8a0-457d-921a-bdf00a47fb74",

"cell\_type": "code",

"source": "from sklearn.preprocessing import MinMaxScaler\n\n# Create a MinMaxScaler object with the custom range (1 to 10)\ncustom\_scaler = MinMaxScaler(feature\_range=(1, 10))\n\n# Normalize the specified columns in the original 'data' DataFrame\ncolumns\_to\_normalize = ['Metascore', 'Votes', 'Gross']\ndata[columns\_to\_normalize] = custom\_scaler.fit\_transform(data[columns\_to\_normalize])\ndata.head()\nprint(data.dtypes)\n",

"metadata": {

"trusted": true

},

"outputs": [

{

"name": "stdout",

"text": "Title object\nRelease year int32\nPlot summary object\nGenre object\nRating float64\nRuntime float64\nIMDb rating float64\nMetascore float64\nDirector object\nStars object\nVotes float64\nGross float64\ndtype: object\n",

"output\_type": "stream"

}

],

"execution\_count": 9

},

{

"id": "8dcf3fd9-ee91-40e3-b9b8-b0a096bde647",

"cell\_type": "markdown",

"source": "What was done: The 'MetaScore', 'Votes', and 'Gross' columns were normalized to a scale of 1-10.\n\nWhy it was necessary: the normalization was done to ensure that the data is presented consistently for comparison and visualization. Since IMDB ratings typically fall within the 1, 10 range aligning the 'MetaScore' 'Votes' and 'Gross columns, on the scale makes it easier to analyze and understand. By bringing all values to a scale it simplifies the process of gauging the significance or size of these variables relative, to each other.\ntools used:pandas",

"metadata": {}

},

{

"id": "18f69e17-8c86-4586-a1a4-6b1d3e4458a2",

"cell\_type": "markdown",

"source": "### Discretization:",

"metadata": {}

},

{

"id": "5a06cde1-2d01-4917-af24-717a0bf8b2ba",

"cell\_type": "code",

"source": "\n\n# Drop rows with NaN values in 'IMDb rating' column\ndata = data.dropna(subset=['IMDb rating'])\n\n# Define the bins and labels for discretization\nbins = [0, 6.9, 8.4, 10]\nlabels = ['Low', 'Medium', 'High']\n\n# Discretize the 'IMDb rating' column\ndata['Rating Category'] = pd.cut(data['IMDb rating'], bins=bins, labels=labels, include\_lowest=True)\n\n# Display the updated dataset with the new 'Rating Category' column\ndata.head()\n",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 10,

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

"data": {

"text/plain": " Title Release year \\\n0 The Shawshank Redemption 1994 \n1 The Dark Knight 2008 \n2 Inception 2010 \n3 Fight Club 1999 \n4 Pulp Fiction 1994 \n\n Plot summary \\\n0 Over the course of several years, two convicts... \n1 When the menace known as the Joker wreaks havo... \n2 A thief who steals corporate secrets through t... \n3 An insomniac office worker and a devil-may-car... \n4 The lives of two mob hitmen, a boxer, a gangst... \n\n Genre Rating Runtime IMDb rating Metascore \\\n0 [Drama] 9.3 142.0 9.3 8.219780 \n1 [Action, Crime, Drama] 9.0 152.0 9.0 8.417582 \n2 [Action, Adventure, Sci-Fi] 8.8 148.0 8.8 7.428571 \n3 [Drama] 8.8 139.0 8.8 6.736264 \n4 [Crime, Drama] 8.9 154.0 8.9 9.505495 \n\n Director Stars \\\n0 Frank Darabont [Tim Robbins, Morgan Freeman, Bob Gunton, Will... \n1 Christopher Nolan [Christian Bale, Heath Ledger, Aaron Eckhart, ... \n2 Christopher Nolan [Leonardo DiCaprio, Joseph Gordon-Levitt, Elli... \n3 David Fincher [Brad Pitt, Edward Norton, Meat Loaf, Zach Gre... \n4 Quentin Tarantino [John Travolta, Uma Thurman, Samuel L. Jackson... \n\n Votes Gross Rating Category \n0 10.000000 1.272308 High \n1 9.941789 6.139261 High \n2 8.914598 3.811287 High \n3 8.181464 1.355807 High \n4 7.855748 2.037057 High ",

"text/html": "<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>Title</th>\n <th>Release year</th>\n <th>Plot summary</th>\n <th>Genre</th>\n <th>Rating</th>\n <th>Runtime</th>\n <th>IMDb rating</th>\n <th>Metascore</th>\n <th>Director</th>\n <th>Stars</th>\n <th>Votes</th>\n <th>Gross</th>\n <th>Rating Category</th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <th>0</th>\n <td>The Shawshank Redemption</td>\n <td>1994</td>\n <td>Over the course of several years, two convicts...</td>\n <td>[Drama]</td>\n <td>9.3</td>\n <td>142.0</td>\n <td>9.3</td>\n <td>8.219780</td>\n <td>Frank Darabont</td>\n <td>[Tim Robbins, Morgan Freeman, Bob Gunton, Will...</td>\n <td>10.000000</td>\n <td>1.272308</td>\n <td>High</td>\n </tr>\n <tr>\n <th>1</th>\n <td>The Dark Knight</td>\n <td>2008</td>\n <td>When the menace known as the Joker wreaks havo...</td>\n <td>[Action, Crime, Drama]</td>\n <td>9.0</td>\n <td>152.0</td>\n <td>9.0</td>\n <td>8.417582</td>\n <td>Christopher Nolan</td>\n <td>[Christian Bale, Heath Ledger, Aaron Eckhart, ...</td>\n <td>9.941789</td>\n <td>6.139261</td>\n <td>High</td>\n </tr>\n <tr>\n <th>2</th>\n <td>Inception</td>\n <td>2010</td>\n <td>A thief who steals corporate secrets through t...</td>\n <td>[Action, Adventure, Sci-Fi]</td>\n <td>8.8</td>\n <td>148.0</td>\n <td>8.8</td>\n <td>7.428571</td>\n <td>Christopher Nolan</td>\n <td>[Leonardo DiCaprio, Joseph Gordon-Levitt, Elli...</td>\n <td>8.914598</td>\n <td>3.811287</td>\n <td>High</td>\n </tr>\n <tr>\n <th>3</th>\n <td>Fight Club</td>\n <td>1999</td>\n <td>An insomniac office worker and a devil-may-car...</td>\n <td>[Drama]</td>\n <td>8.8</td>\n <td>139.0</td>\n <td>8.8</td>\n <td>6.736264</td>\n <td>David Fincher</td>\n <td>[Brad Pitt, Edward Norton, Meat Loaf, Zach Gre...</td>\n <td>8.181464</td>\n <td>1.355807</td>\n <td>High</td>\n </tr>\n <tr>\n <th>4</th>\n <td>Pulp Fiction</td>\n <td>1994</td>\n <td>The lives of two mob hitmen, a boxer, a gangst...</td>\n <td>[Crime, Drama]</td>\n <td>8.9</td>\n <td>154.0</td>\n <td>8.9</td>\n <td>9.505495</td>\n <td>Quentin Tarantino</td>\n <td>[John Travolta, Uma Thurman, Samuel L. Jackson...</td>\n <td>7.855748</td>\n <td>2.037057</td>\n <td>High</td>\n </tr>\n </tbody>\n</table>\n</div>"

},

"metadata": {}

}

],

"execution\_count": 10

},

{

"id": "5f61fa55-bb91-49d9-9282-16dbda943490",

"cell\_type": "markdown",

"source": "Was done: we discretized 'IMDb rating' into the categories (low, medium, high ) 'Low' for ratings below 7.0, 'Medium' for ratings between 7.0 and 8.4, and 'High' for ratings 8.5 and above. Why it was necessary: provides a simplified representation of the ratings, allowing for easier analysis and comparison. we can identify patterns within each category\ntools used:pandas",

"metadata": {}

},

{

"id": "4a456dbe-a3dd-4ff1-84c7-971e3ef07c38",

"cell\_type": "markdown",

"source": "### Feature Selection:",

"metadata": {}

},

{

"id": "d1baba67",

"cell\_type": "code",

"source": "# Drop the 'Plot summary' and 'IMDb rating' columns\ndata.drop(columns=['Plot summary', 'IMDb rating'], inplace=True)\n\n\ndata.head()",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 11,

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

"data": {

"text/plain": " Title Release year Genre \\\n0 The Shawshank Redemption 1994 [Drama] \n1 The Dark Knight 2008 [Action, Crime, Drama] \n2 Inception 2010 [Action, Adventure, Sci-Fi] \n3 Fight Club 1999 [Drama] \n4 Pulp Fiction 1994 [Crime, Drama] \n\n Rating Runtime Metascore Director \\\n0 9.3 142.0 8.219780 Frank Darabont \n1 9.0 152.0 8.417582 Christopher Nolan \n2 8.8 148.0 7.428571 Christopher Nolan \n3 8.8 139.0 6.736264 David Fincher \n4 8.9 154.0 9.505495 Quentin Tarantino \n\n Stars Votes Gross \\\n0 [Tim Robbins, Morgan Freeman, Bob Gunton, Will... 10.000000 1.272308 \n1 [Christian Bale, Heath Ledger, Aaron Eckhart, ... 9.941789 6.139261 \n2 [Leonardo DiCaprio, Joseph Gordon-Levitt, Elli... 8.914598 3.811287 \n3 [Brad Pitt, Edward Norton, Meat Loaf, Zach Gre... 8.181464 1.355807 \n4 [John Travolta, Uma Thurman, Samuel L. Jackson... 7.855748 2.037057 \n\n Rating Category \n0 High \n1 High \n2 High \n3 High \n4 High ",

"text/html": "<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>Title</th>\n <th>Release year</th>\n <th>Genre</th>\n <th>Rating</th>\n <th>Runtime</th>\n <th>Metascore</th>\n <th>Director</th>\n <th>Stars</th>\n <th>Votes</th>\n <th>Gross</th>\n <th>Rating Category</th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <th>0</th>\n <td>The Shawshank Redemption</td>\n <td>1994</td>\n <td>[Drama]</td>\n <td>9.3</td>\n <td>142.0</td>\n <td>8.219780</td>\n <td>Frank Darabont</td>\n <td>[Tim Robbins, Morgan Freeman, Bob Gunton, Will...</td>\n <td>10.000000</td>\n <td>1.272308</td>\n <td>High</td>\n </tr>\n <tr>\n <th>1</th>\n <td>The Dark Knight</td>\n <td>2008</td>\n <td>[Action, Crime, Drama]</td>\n <td>9.0</td>\n <td>152.0</td>\n <td>8.417582</td>\n <td>Christopher Nolan</td>\n <td>[Christian Bale, Heath Ledger, Aaron Eckhart, ...</td>\n <td>9.941789</td>\n <td>6.139261</td>\n <td>High</td>\n </tr>\n <tr>\n <th>2</th>\n <td>Inception</td>\n <td>2010</td>\n <td>[Action, Adventure, Sci-Fi]</td>\n <td>8.8</td>\n <td>148.0</td>\n <td>7.428571</td>\n <td>Christopher Nolan</td>\n <td>[Leonardo DiCaprio, Joseph Gordon-Levitt, Elli...</td>\n <td>8.914598</td>\n <td>3.811287</td>\n <td>High</td>\n </tr>\n <tr>\n <th>3</th>\n <td>Fight Club</td>\n <td>1999</td>\n <td>[Drama]</td>\n <td>8.8</td>\n <td>139.0</td>\n <td>6.736264</td>\n <td>David Fincher</td>\n <td>[Brad Pitt, Edward Norton, Meat Loaf, Zach Gre...</td>\n <td>8.181464</td>\n <td>1.355807</td>\n <td>High</td>\n </tr>\n <tr>\n <th>4</th>\n <td>Pulp Fiction</td>\n <td>1994</td>\n <td>[Crime, Drama]</td>\n <td>8.9</td>\n <td>154.0</td>\n <td>9.505495</td>\n <td>Quentin Tarantino</td>\n <td>[John Travolta, Uma Thurman, Samuel L. Jackson...</td>\n <td>7.855748</td>\n <td>2.037057</td>\n <td>High</td>\n </tr>\n </tbody>\n</table>\n</div>"

},

"metadata": {}

}

],

"execution\_count": 11

},

{

"id": "f1c89331-7a9a-4278-a513-b35ca8f28609",

"cell\_type": "markdown",

"source": "What was done: The 'Plot Summary'and 'IMDb rating' columns was dropped from the dataset.\n\nWhy it was necessary: The 'Plot Summary' column was deemed not helpful for the analysis or modeling objectives. Removing it simplifies the dataset, reducing complexity and focusing on more relevant features,and The 'IMDb rating' column was dropped because it was a duplicate of the 'Rating' column.\ntools used:pandas",

"metadata": {}

},

{

"id": "10973fa7",

"cell\_type": "markdown",

"source": "\n# 4. Statistical Summaries & EDA\n\*\*\*",

"metadata": {}

},

{

"id": "7b616592",

"cell\_type": "markdown",

"source": "## Task: Statistical Summaries\n### Done by : Leen Alharbi \n### Date: 20-3-2024",

"metadata": {}

},

{

"id": "53a63663",

"cell\_type": "code",

"source": "#Data visualization and analysis\n#Statistical description of numerical columns\nmovie\_DF= pd.DataFrame(data)\nmovie\_DF.describe()",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 12,

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

"data": {

"text/plain": " Release year Rating Runtime Metascore Votes \\\ncount 1997.000000 1997.000000 1997.000000 1997.000000 1997.000000 \nmean 2000.522784 7.006610 114.147722 6.239282 1.706129 \nstd 14.139624 0.847551 21.259413 1.713948 0.934589 \nmin 1921.000000 1.900000 64.000000 1.000000 1.000000 \n25% 1997.000000 6.400000 99.000000 4.956044 1.170325 \n50% 2004.000000 7.100000 111.000000 6.241758 1.385879 \n75% 2009.000000 7.600000 126.000000 7.527473 1.823659 \nmax 2016.000000 9.300000 238.000000 10.000000 10.000000 \n\n Gross \ncount 1997.000000 \nmean 1.784810 \nstd 0.819826 \nmin 1.000000 \n25% 1.253859 \n50% 1.544088 \n75% 2.054737 \nmax 10.000000 ",

"text/html": "<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>Release year</th>\n <th>Rating</th>\n <th>Runtime</th>\n <th>Metascore</th>\n <th>Votes</th>\n <th>Gross</th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <th>count</th>\n <td>1997.000000</td>\n <td>1997.000000</td>\n <td>1997.000000</td>\n <td>1997.000000</td>\n <td>1997.000000</td>\n <td>1997.000000</td>\n </tr>\n <tr>\n <th>mean</th>\n <td>2000.522784</td>\n <td>7.006610</td>\n <td>114.147722</td>\n <td>6.239282</td>\n <td>1.706129</td>\n <td>1.784810</td>\n </tr>\n <tr>\n <th>std</th>\n <td>14.139624</td>\n <td>0.847551</td>\n <td>21.259413</td>\n <td>1.713948</td>\n <td>0.934589</td>\n <td>0.819826</td>\n </tr>\n <tr>\n <th>min</th>\n <td>1921.000000</td>\n <td>1.900000</td>\n <td>64.000000</td>\n <td>1.000000</td>\n <td>1.000000</td>\n <td>1.000000</td>\n </tr>\n <tr>\n <th>25%</th>\n <td>1997.000000</td>\n <td>6.400000</td>\n <td>99.000000</td>\n <td>4.956044</td>\n <td>1.170325</td>\n <td>1.253859</td>\n </tr>\n <tr>\n <th>50%</th>\n <td>2004.000000</td>\n <td>7.100000</td>\n <td>111.000000</td>\n <td>6.241758</td>\n <td>1.385879</td>\n <td>1.544088</td>\n </tr>\n <tr>\n <th>75%</th>\n <td>2009.000000</td>\n <td>7.600000</td>\n <td>126.000000</td>\n <td>7.527473</td>\n <td>1.823659</td>\n <td>2.054737</td>\n </tr>\n <tr>\n <th>max</th>\n <td>2016.000000</td>\n <td>9.300000</td>\n <td>238.000000</td>\n <td>10.000000</td>\n <td>10.000000</td>\n <td>10.000000</td>\n </tr>\n </tbody>\n</table>\n</div>"

},

"metadata": {}

}

],

"execution\_count": 12

},

{

"id": "db01466e",

"cell\_type": "markdown",

"source": "Beginning with the 'Release year', it's evident that the dataset encompasses movies spanning from as early as 1921 to as recent as 2016, with a mean release year around 2000.52 and a standard deviation of approximately 14.14, indicating a spread around the average. The 'Rating' attribute showcases an average rating of 7.01, with a standard deviation of 0.85, suggesting a relatively narrow spread of ratings compared to the mean. 'Runtime' statistics reveal an average duration of around 114.15 minutes, with a standard deviation of about 21.26, indicating variability in movie lengths. 'Metascore', likely representing scores from Metacritic, shows a mean of 6.24, with a standard deviation of 1.71. 'Votes' and 'Gross' statistics reflect attributes related to audience engagement and financial performance, respectively, with mean values of 1.71 and 1.78, and standard deviations of 0.93 and 0.82, respectively. These statistics collectively provide insights into the distribution and variability of movie-related attributes, offering valuable information for further analysis and understanding of the dataset.",

"metadata": {}

},

{

"id": "9fa1a308",

"cell\_type": "code",

"source": "movie\_DF.head()",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 13,

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

"data": {

"text/plain": " Title Release year Genre \\\n0 The Shawshank Redemption 1994 [Drama] \n1 The Dark Knight 2008 [Action, Crime, Drama] \n2 Inception 2010 [Action, Adventure, Sci-Fi] \n3 Fight Club 1999 [Drama] \n4 Pulp Fiction 1994 [Crime, Drama] \n\n Rating Runtime Metascore Director \\\n0 9.3 142.0 8.219780 Frank Darabont \n1 9.0 152.0 8.417582 Christopher Nolan \n2 8.8 148.0 7.428571 Christopher Nolan \n3 8.8 139.0 6.736264 David Fincher \n4 8.9 154.0 9.505495 Quentin Tarantino \n\n Stars Votes Gross \\\n0 [Tim Robbins, Morgan Freeman, Bob Gunton, Will... 10.000000 1.272308 \n1 [Christian Bale, Heath Ledger, Aaron Eckhart, ... 9.941789 6.139261 \n2 [Leonardo DiCaprio, Joseph Gordon-Levitt, Elli... 8.914598 3.811287 \n3 [Brad Pitt, Edward Norton, Meat Loaf, Zach Gre... 8.181464 1.355807 \n4 [John Travolta, Uma Thurman, Samuel L. Jackson... 7.855748 2.037057 \n\n Rating Category \n0 High \n1 High \n2 High \n3 High \n4 High ",

"text/html": "<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>Title</th>\n <th>Release year</th>\n <th>Genre</th>\n <th>Rating</th>\n <th>Runtime</th>\n <th>Metascore</th>\n <th>Director</th>\n <th>Stars</th>\n <th>Votes</th>\n <th>Gross</th>\n <th>Rating Category</th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <th>0</th>\n <td>The Shawshank Redemption</td>\n <td>1994</td>\n <td>[Drama]</td>\n <td>9.3</td>\n <td>142.0</td>\n <td>8.219780</td>\n <td>Frank Darabont</td>\n <td>[Tim Robbins, Morgan Freeman, Bob Gunton, Will...</td>\n <td>10.000000</td>\n <td>1.272308</td>\n <td>High</td>\n </tr>\n <tr>\n <th>1</th>\n <td>The Dark Knight</td>\n <td>2008</td>\n <td>[Action, Crime, Drama]</td>\n <td>9.0</td>\n <td>152.0</td>\n <td>8.417582</td>\n <td>Christopher Nolan</td>\n <td>[Christian Bale, Heath Ledger, Aaron Eckhart, ...</td>\n <td>9.941789</td>\n <td>6.139261</td>\n <td>High</td>\n </tr>\n <tr>\n <th>2</th>\n <td>Inception</td>\n <td>2010</td>\n <td>[Action, Adventure, Sci-Fi]</td>\n <td>8.8</td>\n <td>148.0</td>\n <td>7.428571</td>\n <td>Christopher Nolan</td>\n <td>[Leonardo DiCaprio, Joseph Gordon-Levitt, Elli...</td>\n <td>8.914598</td>\n <td>3.811287</td>\n <td>High</td>\n </tr>\n <tr>\n <th>3</th>\n <td>Fight Club</td>\n <td>1999</td>\n <td>[Drama]</td>\n <td>8.8</td>\n <td>139.0</td>\n <td>6.736264</td>\n <td>David Fincher</td>\n <td>[Brad Pitt, Edward Norton, Meat Loaf, Zach Gre...</td>\n <td>8.181464</td>\n <td>1.355807</td>\n <td>High</td>\n </tr>\n <tr>\n <th>4</th>\n <td>Pulp Fiction</td>\n <td>1994</td>\n <td>[Crime, Drama]</td>\n <td>8.9</td>\n <td>154.0</td>\n <td>9.505495</td>\n <td>Quentin Tarantino</td>\n <td>[John Travolta, Uma Thurman, Samuel L. Jackson...</td>\n <td>7.855748</td>\n <td>2.037057</td>\n <td>High</td>\n </tr>\n </tbody>\n</table>\n</div>"

},

"metadata": {}

}

],

"execution\_count": 13

},

{

"id": "8be8ffca",

"cell\_type": "code",

"source": "#include other categorical columns\nmovie\_DF.describe(include=['O']) ",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 14,

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

"data": {

"text/plain": " Title Genre Director \\\ncount 1997 1997 1997 \nunique 1976 246 880 \ntop Cinderella [Action, Adventure, Sci-Fi] Steven Spielberg \nfreq 2 90 23 \n\n Stars \ncount 1997 \nunique 1980 \ntop [Talia Shire, Burt Young, Carl Weathers] \nfreq 3 ",

"text/html": "<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>Title</th>\n <th>Genre</th>\n <th>Director</th>\n <th>Stars</th>\n </tr>\n </thead>\n <tbody>\n <tr>\n <th>count</th>\n <td>1997</td>\n <td>1997</td>\n <td>1997</td>\n <td>1997</td>\n </tr>\n <tr>\n <th>unique</th>\n <td>1976</td>\n <td>246</td>\n <td>880</td>\n <td>1980</td>\n </tr>\n <tr>\n <th>top</th>\n <td>Cinderella</td>\n <td>[Action, Adventure, Sci-Fi]</td>\n <td>Steven Spielberg</td>\n <td>[Talia Shire, Burt Young, Carl Weathers]</td>\n </tr>\n <tr>\n <th>freq</th>\n <td>2</td>\n <td>90</td>\n <td>23</td>\n <td>3</td>\n </tr>\n </tbody>\n</table>\n</div>"

},

"metadata": {}

}

],

"execution\_count": 14

},

{

"id": "6eed6a34",

"cell\_type": "code",

"source": "movie\_DF.info()",

"metadata": {

"trusted": true

},

"outputs": [

{

"name": "stdout",

"text": "<class 'pandas.core.frame.DataFrame'>\nRangeIndex: 1997 entries, 0 to 1996\nData columns (total 11 columns):\n # Column Non-Null Count Dtype \n--- ------ -------------- ----- \n 0 Title 1997 non-null object \n 1 Release year 1997 non-null int32 \n 2 Genre 1997 non-null object \n 3 Rating 1997 non-null float64 \n 4 Runtime 1997 non-null float64 \n 5 Metascore 1997 non-null float64 \n 6 Director 1997 non-null object \n 7 Stars 1997 non-null object \n 8 Votes 1997 non-null float64 \n 9 Gross 1997 non-null float64 \n 10 Rating Category 1997 non-null category\ndtypes: category(1), float64(5), int32(1), object(4)\nmemory usage: 119.1+ KB\n",

"output\_type": "stream"

}

],

"execution\_count": 15

},

{

"id": "27354b30",

"cell\_type": "markdown",

"source": "The 'Release year' column is of integer type, while 'Rating', 'Runtime', 'Metascore', 'Votes', and 'Gross' are represented as floating-point numbers. Categorical data is present in the 'Rating Category' column. Textual data, such as movie titles, genres, director names, and actor names, is stored as objects. The DataFrame provides a comprehensive overview of movie-related attributes, facilitating data analysis and exploration. The memory usage of the DataFrame is approximately 158.2 KB.",

"metadata": {}

},

{

"id": "e02c76ef",

"cell\_type": "code",

"source": "movie\_DF.columns",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 16,

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

"data": {

"text/plain": "Index(['Title', 'Release year', 'Genre', 'Rating', 'Runtime', 'Metascore',\n 'Director', 'Stars', 'Votes', 'Gross', 'Rating Category'],\n dtype='object')"

},

"metadata": {}

}

],

"execution\_count": 16

},

{

"id": "cd0f61e1",

"cell\_type": "markdown",

"source": "## Task:Statistical Summaries visualization\n### Done by : Leen Alharbi\n### Date: 20-3-2024\n\n### Plots of Distribution & Scatter Plots using the libraries matplotlib.pyplot and seaborn",

"metadata": {}

},

{

"id": "af143ee6",

"cell\_type": "code",

"source": "# Plot of Distribution of release year\nrelease\_years = data['Release year']\n\nplt.figure(figsize=(14, 7))\nsns.histplot(release\_years, bins=30, kde=True)\nplt.title('Distribution of Release Years')\nplt.xlabel('Release Year')\nplt.ylabel('Frequency')\nplt.xticks(rotation=45) \nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1400x700 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 17

},

{

"id": "9945f6e8",

"cell\_type": "markdown",

"source": "Increasing Trend Over Time: There is a clear increasing trend in the number of movies released over the years. The number of movies starts to grow more significantly in the latter half of the 20th century, with a noticeable surge from the 1990s onwards.\n\nPeaks and Declines: There are distinct peaks in certain years, where the number of movies released is especially high, followed by declines. This could indicate years where the movie industry was particularly active or years that were significant for cinema in terms of output.\n\nRecent Dominance: The most recent years in the dataset, which seem to be the late 2000s and early 2010s, dominate the dataset. This could suggest either a bias towards more recent movies in the dataset collection process or an actual increase in movie production in recent years.\n",

"metadata": {}

},

{

"id": "7ac48b33",

"cell\_type": "code",

"source": "# Plot distribution of 'Runtime'\nsns.set\_style(\"whitegrid\")\n\n\nplt.figure(figsize=(50, 45))\n\nplt.subplot(3, 2, 2)\n\nsns.histplot(data=data, x='Runtime', bins=30, kde=True, color='skyblue')\n\nplt.title('Distribution of Runtime', fontsize=40)\nplt.xlabel('Runtime', fontsize=20) \nplt.ylabel('Frequency', fontsize=25)\n\nplt.xticks(rotation=90, fontsize=20)\n\nplt.tight\_layout(pad=1.0)\n\n\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 5000x4500 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 18

},

{

"id": "25a5646d",

"cell\_type": "markdown",

"source": "Normal-like Distribution: The histogram shows what appears to be a roughly normal distribution of movie runtimes. There's a clear central peak with frequencies tapering off on either side. This indicates that most movies have a duration close to the average runtime, with fewer films at the extremes of very short or very long.\n\nMost Common Runtimes: The peak of the distribution suggests that the most common movie runtimes are around the center of the dataset's range. Given standard film lengths, this is likely in the vicinity of 90 to 120 minutes, which is typical for feature-length films.\n\nLong-Tail on the Right: There's a long tail to the right of the distribution, indicating that while most movies are of standard length, there's a number of films with significantly longer runtimes. These could be epic films, extended versions, or movies that are outliers in terms of content and structure.\n\nFew Very Short or Very Long Films: There are relatively few very short (under 80 minutes) or very long (over 150 minutes) films, which reflects traditional movie industry standards and audience expectations",

"metadata": {}

},

{

"id": "e9077045",

"cell\_type": "code",

"source": "# Plot distribution of 'Metascore'\n\nsns.histplot(data=data, x='Metascore', bins=30, kde=True, color='purple')\n\nplt.title('Distribution of Metascore', fontsize=20)\nplt.xlabel('Metascore', fontsize=20) \n\n",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 19,

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

"data": {

"text/plain": "Text(0.5, 0, 'Metascore')"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 640x480 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 19

},

{

"id": "a139f0a8",

"cell\_type": "markdown",

"source": "Central Tendency: The distribution has a pronounced peak slightly above 5 on the Metascore scale. This peak indicates the most common range of scores that movies in this dataset received.\n\nSkewness: The distribution seems to have a slight right skew, with a tail extending more toward the higher scores. This suggests that there are a few movies with very high Metascores compared to the rest.\n\nNormality: Despite the skew, the overall shape of the distribution has a resemblance to a normal distribution, suggesting that Metascore values are fairly evenly spread around the mean.",

"metadata": {}

},

{

"id": "f30737e9",

"cell\_type": "code",

"source": "# Plot distribution of 'Gross'\nsns.histplot(data=data, x='Gross', bins=30, kde=True, color='pink')\nplt.title('Distribution of Gross Revenue')",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 20,

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

"data": {

"text/plain": "Text(0.5, 1.0, 'Distribution of Gross Revenue')"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 640x480 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 20

},

{

"id": "62dfc9aa",

"cell\_type": "markdown",

"source": "The histogram for the distribution of gross revenue indicate:\n\nRight-Skewed Distribution: The graph shows that the gross revenue distribution is heavily skewed to the right, meaning there are a large number of movies that have relatively low earnings and a small number that have very high earnings.\n\nMost Movies Earn Less: The peak at the lower end of the gross scale suggests that most movies earn on the lower side of the revenue spectrum.\n\nBlockbusters Are Rare: The long tail to the right indicates that movies with extremely high gross revenues (blockbusters) are relatively rare compared to the total number of movies.",

"metadata": {}

},

{

"id": "684b81b1",

"cell\_type": "markdown",

"source": "## Bar plot to discover trending genres among movies in the dataset",

"metadata": {}

},

{

"id": "74d258d7",

"cell\_type": "code",

"source": "data\_exploded = data.explode('Genre')\n\n\ngenre\_counts = data\_exploded['Genre'].value\_counts()\n\ntop\_10\_genres = genre\_counts.head(10)\n\ntop\_10\_genres.plot(kind='bar', title='Top 10 Genres Among Movies')",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 21,

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

"data": {

"text/plain": "<AxesSubplot:title={'center':'Top 10 Genres Among Movies'}>"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 640x480 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 21

},

{

"id": "84083d4e",

"cell\_type": "markdown",

"source": "Drama Dominates: The genre with the highest count is Drama, indicating that it's the most prevalent genre among the movies in the dataset.\n\nAction and Comedy Are Popular: Following Drama, Action and Comedy genres also have significant counts, suggesting these are popular genres.\n\nVariety in Top Genres: The presence of Adventure, Crime, Thriller, Romance, Sci-Fi, Mystery, and Fantasy shows there is a good mix of genres represented in the top ten.\n\nFantasy Less Common: Fantasy appears to be the least common among the top genres, but still makes it to the top ten list.\n\nImplications for Production: The popularity of genres like Drama and Action could influence production decisions in the film industry, as these genres might be associated with a higher likelihood of commercial success.",

"metadata": {}

},

{

"id": "f872c0a5",

"cell\_type": "markdown",

"source": "## Bar plot to discover genres with the highest rating among movies in the dataset",

"metadata": {}

},

{

"id": "d623af41",

"cell\_type": "code",

"source": "exploded\_df = data.explode('Genre')\n\ngenre\_ratings = exploded\_df.groupby('Genre')['Rating'].mean().sort\_values(ascending=False)\n\n\ntop\_n = 10 \ngenre\_ratings.head(top\_n).plot(kind='bar', figsize=(12, 8), color='skyblue')\n\n\nplt.title('Top Genres by Average Rating')\nplt.xlabel('Genre')\nplt.ylabel('Average Rating')\nplt.xticks(rotation=45)\n\n\nplt.tight\_layout()\nplt.show()\n",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1200x800 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 22

},

{

"id": "9784f3ca",

"cell\_type": "markdown",

"source": "Consistent High Ratings: All the genres shown have high average ratings, which likely means they are well-received by audiences or critics.\n\nClose Range: The average ratings are in a close range, suggesting that no single genre drastically outperforms the others in terms of average rating.\n\nPopularity vs. Rating: This chart does not necessarily reflect the popularity of the genres, but rather how highly they are rated on average.\n\nRepresentation Issue: The genres might contain varying numbers of movies, which isn't depicted in this type of visualization. A genre with a high average rating but few movies could be overrepresented compared to a genre with many movies and a slightly lower average rating.",

"metadata": {}

},

{

"id": "8cd747d6",

"cell\_type": "markdown",

"source": "## Bar plot to discover genres with the highest Average metascore among movies in the dataset",

"metadata": {}

},

{

"id": "e40d1e23",

"cell\_type": "code",

"source": "\ndata\_exploded = data.explode('Genre')\n\n\navg\_metascore = data\_exploded.groupby('Genre')['Metascore'].mean()\n\nsorted\_avg\_metascore = avg\_metascore.sort\_values(ascending=False)\n\n\nN = 5\ntop\_N\_genres = sorted\_avg\_metascore.head(N)\n\ntop\_N\_genres.plot(kind='bar', title='Top 5 Genres by Average Metascore', color='green')\nplt.ylabel('Average Metascore')\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 640x480 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 23

},

{

"id": "dc3f573d",

"cell\_type": "markdown",

"source": "Highest Rated Genre: \"Film-Noir\" appears as the highest-rated genre, suggesting that, on average, films in this genre are critically acclaimed or well-received by critics.\n\nClose Ratings: The genres \"War,\" \"Western,\" \"History,\" and \"Animation\" have very similar average Metascores. The small differences between the bars indicate that these genres tend to receive comparable critical acclaim.\n\nRange of Metascores: All the genres have high average Metascores, with the lowest among the top genres being close to 7 and the highest above 8. This suggests that movies in these genres are generally rated favorably.\n\nDiversity of Genres: The genres presented cover a diverse range of film types, from the classic \"Film-Noir\" to modern \"Animation.\" This diversity indicates that critically acclaimed films are not confined to a particular style or era.\n\nGenre Representation: The chart suggests that \"Film-Noir,\" despite being a less common genre today, still holds a place as the top-rated genre in terms of Metascore. It's noteworthy that classic genres like \"War\" and \"Western\" continue to be well-regarded, while \"History\" and \"Animation\" also receive high critical praise.",

"metadata": {}

},

{

"id": "8c913b58",

"cell\_type": "markdown",

"source": "## Bar plot to discover genres with the highest total gross among movies in the dataset",

"metadata": {}

},

{

"id": "4b07a717",

"cell\_type": "code",

"source": "data\_exploded = data.explode('Genre')\n\ngenre\_gross = data\_exploded.groupby('Genre')['Gross'].sum()\n\ntop\_n\_genres\_by\_gross = genre\_gross.sort\_values(ascending=False).head(10)\n\ntop\_n\_genres\_by\_gross.plot(kind='bar', title='Top 10 Genres by Total Gross')\nplt.xlabel('Genre')\nplt.ylabel('Total Gross')\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 640x480 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 24

},

{

"id": "1043965e",

"cell\_type": "markdown",

"source": "Genre Popularity: Drama, Action, and Adventure genres lead in total gross revenue, indicating their widespread popularity and potential box office appeal.\n\nRevenue Distribution: There's a notable drop-off in total gross revenue after the top three genres, with Comedy, Crime, and Thriller following behind. This suggests that while there are standout genres that consistently perform well at the box office, there is a diverse range of genres that also contribute significantly to the industry's revenue.\n\nComparatively Lower Grossing Genres: Sci-Fi, Romance, Fantasy, and Mystery have lower total gross revenues in comparison to the leading genres. Despite this, their presence in the top 10 indicates they have a solid market presence.\n\nMarket Trends: The chart may reflect market trends and audience preferences, with Drama and Action films likely drawing larger audiences. The high performance of the Adventure genre could also be attributed to the popularity of blockbuster franchises and family-friendly films.\n\nDiversity in High-Grossing Genres: The top-grossing genres are diverse, covering a range of storytelling styles from the character-driven Drama to the spectacle-driven Action and Adventure genres.",

"metadata": {}

},

{

"id": "227fec38",

"cell\_type": "markdown",

"source": "## Task: Visualize the trend of movie ratings, Runtime and gross earnings over time.\n### Done by: Shaden Albader\n### Date: 20-3-2024\n\n Summary:\n - Pandas is used to handle the dataset.\n - Matplotlib is used for plotting line graphs.\n - Datetime is used to convert the 'Release year' column to datetime format.",

"metadata": {}

},

{

"id": "9501c0d0",

"cell\_type": "markdown",

"source": "### Visualize the trend of movie ratings over time",

"metadata": {}

},

{

"id": "39f6d1b1",

"cell\_type": "code",

"source": "data['Release year'] = pd.to\_datetime(data['Release year'], format='%Y')\ndata.sort\_values(by='Release year', inplace=True)\n\n# Ratings over time\nplt.plot(data['Release year'], data['Rating'])\nplt.xlabel('Release Year')\nplt.ylabel('Rating')\nplt.title('Rating Trend over Time')\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 640x480 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 25

},

{

"id": "a3dff91d",

"cell\_type": "markdown",

"source": "Consistently High Ratings: The line plot shows that, overall, movies have received consistently high ratings across all the years considered in the dataset. This indicates that the majority of movies tend to be well-received by viewers.\n\nClustering of Lines in the Later Years: The cluster of lines in the later years (1990-2020) suggests a higher concentration of movies released during this period. This could be due to factors such as increased movie production, expanded access to distribution channels, or the growth of the film industry.\n\nVariation in Ratings: Despite the overall high ratings, the presence of some low ratings in the later years indicates that there are movies in that period that did not receive favorable ratings. This variation might be attributed to factors such as the diversity of movie genres, different audience preferences, or the quality and reception of individual films.\n\nImpact of Changing Trends: The presence of low ratings in the later years could be influenced by changing cinematic trends, evolving audience tastes, or the emergence of new storytelling techniques. These factors can contribute to a wider range of ratings, including both high and low scores.",

"metadata": {}

},

{

"id": "51f5ed4d",

"cell\_type": "markdown",

"source": "### Visualize the trend of runtime of movies over time.",

"metadata": {}

},

{

"id": "0885a2f5",

"cell\_type": "code",

"source": "# Runtime over time\nplt.plot(data['Release year'], data['Runtime'])\nplt.xlabel('Release Year')\nplt.ylabel('Runtime')\nplt.title('Runtime Trend over Time')\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 640x480 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 26

},

{

"id": "602574fa",

"cell\_type": "markdown",

"source": "Consistent Runtime Range: The plot indicates that the majority of movies, especially those released after 1960, have runtimes ranging from approximately 90 to 175 minutes. This suggests that filmmakers and studios have tended to adhere to a certain duration range when producing movies during this period.\n\nVariations in Runtime: Despite the consistent range, the plot also reveals that there are movies with runtimes both higher and lower than the typical range. This indicates that filmmakers have made creative choices to deviate from the average runtime, potentially to better suit the storytelling needs or artistic vision of their films.\n\nEarly Years: In the earlier years depicted on the left side of the plot, movies generally had shorter runtimes that did not exceed 150 minutes. This could be due to factors such as technological limitations, storytelling conventions, or audience preferences during that era. It is interesting to note that longer movies were less common during this period.\n\nIncreasing Runtimes: As we move towards the more recent years on the right side of the plot, there seems to be a slight increase in the average runtime of movies. This could be attributed to various factors such as advancements in filmmaking techniques, the willingness of audiences to engage with longer narratives, and the evolving tastes and expectations of viewers.\n\nImpact of Genre: It's important to consider the influence of different movie genres on the observed runtime trends. Certain genres, such as action or comedy, tend to have shorter runtimes, while others like drama or epic films often require longer durations to fully develop their narratives. Therefore, the plot may reflect the average runtime trend across multiple genres, but individual genres might have their own unique patterns.\n\nOutliers: The plot contain outliers, represented by movies with significantly higher or lower runtimes than the average. These outliers could be attributed to factors such as experimental films, documentaries, or special cases where filmmakers intentionally deviate from the standard runtime conventions to achieve specific artistic or storytelling goals.",

"metadata": {}

},

{

"id": "1b167e5e",

"cell\_type": "markdown",

"source": "### Visualize the trend of gross earnings of movies over time.\n",

"metadata": {}

},

{

"id": "ef0658b6",

"cell\_type": "code",

"source": "# Gross earnings over time\nplt.plot(data['Release year'], data['Gross'])\nplt.xlabel('Release Year')\nplt.ylabel('Gross Earnings')\nplt.title('Gross Earnings Trend over Time')\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 640x480 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 27

},

{

"id": "f5208cdb",

"cell\_type": "markdown",

"source": "Increasing Trend: The plot clearly indicates that the gross earnings of movies have been increasing over the years. This suggests that, on average, movies have been generating higher revenues in later years compared to earlier years.\n\nHigher Gross Earnings: In the later years represented on the right side of the plot, there is a notable increase in the gross earnings of movies. This indicates that movies released during these years have been more commercially successful, generating significantly higher revenues.\n\nClustered Range: The plot shows that the majority of movies have gross earnings that fall within the range of 1 to 5 billion dollars. This suggests that this range is common for movies in terms of their financial performance. It is important to note that the gross earnings are scaled in billions, as mentioned in the assumption.\n\nHigh Gross Earnings Outliers: In the later years, there are movies that have exceptionally high gross earnings, reaching up to 10 billion dollars or more. These movies can be considered as outliers or blockbuster hits that have achieved significant commercial success. Such movies often have a combination of factors like a highly anticipated franchise, a large fan base, critical acclaim, effective marketing, and global appeal.\n\nInfluence of Successful Franchises: The increased gross earnings in later years could be attributed to the success of popular movie franchises. Sequels, spin-offs, and cinematic universes have become more prevalent in recent years, attracting a dedicated fan base and generating substantial revenues.\n\nGlobal Box Office Growth: The rise in gross earnings over time can also be attributed to the growth of the global box office. The expansion of international markets, particularly in countries like China, has contributed to the overall increase in movie revenues.",

"metadata": {}

},

{

"id": "5953c009",

"cell\_type": "markdown",

"source": "## Task: Identify the top directors by average rating, total gross earnings, average rating, and gross earnings.\n### Done by: Shaden Albader\n### Date: 20-3-2024\n",

"metadata": {}

},

{

"id": "853c515c",

"cell\_type": "markdown",

"source": "### Identify the top directors by average rating.\n",

"metadata": {}

},

{

"id": "600f7014",

"cell\_type": "code",

"source": "director\_ratings = data.groupby('Director')['Rating'].mean()\ndirector\_earnings = data.groupby('Director')['Gross'].sum()\n\ndirector\_ratings = director\_ratings.sort\_values(ascending=False)\ndirector\_earnings = director\_earnings.sort\_values(ascending=False)",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 28

},

{

"id": "45219fe3",

"cell\_type": "code",

"source": "plt.figure(figsize=(12, 6))\ndirector\_ratings[:10].plot(kind='bar', color='green')\nplt.xlabel('Director')\nplt.ylabel('Average Rating')\nplt.title('Top 10 Directors by Average Rating')\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1200x600 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 29

},

{

"id": "af2a794e",

"cell\_type": "markdown",

"source": "High Average Ratings: The plot indicates that the top 10 directors have achieved high average ratings for their movies. This suggests that their work has been consistently well-received and highly regarded by viewers or critics.\n\nClose Range of Average Ratings: The plot shows that the average ratings of these directors are in a close range. This implies that the movies directed by these filmmakers have been perceived to have similar levels of quality and excellence. The narrow range of average ratings suggests a certain level of consistency in the quality of their work.\n\nSubjectivity of Ratings: It's important to note that movie ratings can be subjective, and individual preferences may vary. While the average ratings indicate a high level of appreciation for these directors' work, it's possible that different viewers or critics may have slightly different opinions on the quality and impact of their movies.\n\nReputation and Influence: The fact that these directors have achieved high average ratings highlights their reputation and influence in the film industry. Their work has likely left a significant impact on audiences, critics, and the broader filmmaking community.",

"metadata": {}

},

{

"id": "5e93fe5d",

"cell\_type": "markdown",

"source": "### Identify the top directors by total gross earnings.\n",

"metadata": {}

},

{

"id": "76d7a4e5",

"cell\_type": "code",

"source": "plt.figure(figsize=(12, 6))\ndirector\_earnings[:10].plot(kind='bar')\nplt.xlabel('Director')\nplt.ylabel('Total Gross Earnings')\nplt.title('Top 10 Directors by Total Gross Earnings')\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1200x600 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 30

},

{

"id": "a3b5ba0d",

"cell\_type": "markdown",

"source": "The plot reveals that Steven Spielberg has the highest total earnings among all directors. This indicates that his movies, collectively, have generated the highest amount of box office revenue compared to other directors in the plot. Spielberg's success highlights his status as one of the most commercially successful and influential directors in the industry.\n\nDouble the Earnings: The analysis reveals that Spielberg's total earnings are nearly double the earnings of the other directors on the plot. This significant difference emphasizes his exceptional financial success and underscores his impact on the box office. Spielberg's ability to consistently deliver commercially successful movies has set him apart from his peers.\n\nProximity of Other Directors: The rest of the directors, excluding Spielberg, appear to have similar levels of total earnings, as indicated by their relatively close bar heights. While Spielberg stands out with his significantly higher earnings, the other directors' earnings are relatively comparable to each other.",

"metadata": {}

},

{

"id": "cfdcf733",

"cell\_type": "markdown",

"source": "### Identify the top actors by average rating.\n",

"metadata": {}

},

{

"id": "4814628c",

"cell\_type": "code",

"source": "data\_exploded = data.explode('Stars')\n\n# Calculate the mean rating for each individual actor\nstar\_ratings = data\_exploded.groupby('Stars')['Rating'].mean()\n\n# Calculate the sum of gross earnings for each individual actor (if needed)\n# star\_earnings = data\_exploded.groupby('Stars')['Gross'].sum()\n\n# Sort the ratings in descending order to get the top-rated actors\nstar\_ratings = star\_ratings.sort\_values(ascending=False)\n\n# Plot the top 10 actors by average rating\nplt.figure(figsize=(12, 6))\nstar\_ratings[:10].plot(kind='bar')\nplt.xlabel('Actor')\nplt.ylabel('Average Rating')\nplt.title('Top 10 Actors by Average Rating')\nplt.xticks(rotation=45, ha='right') # Rotate the x-axis labels for better readability\nplt.tight\_layout() # Adjust the layout to make room for the x-axis labels\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1200x600 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 31

},

{

"id": "8d32b02c",

"cell\_type": "markdown",

"source": "The top 10 actors have consistently received high average ratings for their performances, indicating that their acting skills are highly regarded by viewers and critics.\n\nThe average ratings for these actors are closely grouped together, suggesting that the movies they appear in are perceived to have similar levels of acting quality.\n\nThe ratings of these actors are very similar, with only slight variations among them. This indicates that they consistently deliver performances of a consistently high standard.\n\nThese actors demonstrate a consistent level of strong acting skills and the ability to effectively embody their roles.",

"metadata": {}

},

{

"id": "bb70c5c6",

"cell\_type": "markdown",

"source": "### Identify the top actors by total gross earnings.\n",

"metadata": {}

},

{

"id": "0afdf227",

"cell\_type": "code",

"source": "# If the 'Stars' column has actors grouped together, we'll split them into individual rows\ndata\_exploded = data.explode('Stars')\n\n# Calculate the sum of gross earnings for each individual actor\nstar\_earnings = data\_exploded.groupby('Stars')['Gross'].sum()\n\n# Sort the gross earnings in descending order to get the top earners\nstar\_earnings = star\_earnings.sort\_values(ascending=False)\n\n# Plot the top 10 stars by total gross earnings\nplt.figure(figsize=(12, 6))\nstar\_earnings[:10].plot(kind='bar', color='green')\nplt.xlabel('Actor')\nplt.ylabel('Total Gross Earnings')\nplt.title('Top 10 Actors by Total Gross Earnings')\nplt.xticks(rotation=45, ha='right') # Rotate the x-axis labels for better readability\nplt.tight\_layout() # Adjust the layout to make room for the x-axis labels\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1200x600 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 32

},

{

"id": "020488c8",

"cell\_type": "markdown",

"source": "The graph shows that Tom Hanks stands out as the highest-earning actor in terms of total gross earnings. His movies have generated significant box office revenue, indicating his exceptional commercial success and popularity among audiences.\n\nFollowing Tom Hanks, Johnny Depp and Robert De Niro have also achieved high total gross earnings. Their movies have performed well at the box office, contributing to their financial success and establishing them as influential and bankable actors.\n\nProximity of Earnings: The information provided indicates that the remaining actors on the plot are relatively close to each other in terms of total gross earnings. While the specific values may vary, their earnings fall within a similar range, suggesting comparable levels of commercial success among these actors.",

"metadata": {}

},

{

"id": "7549d890",

"cell\_type": "markdown",

"source": "## EDA",

"metadata": {}

},

{

"id": "77c686af",

"cell\_type": "markdown",

"source": "## Task: visualize the correlation coefficients between pairs of variables in a dataset\n### Done by: Norah Mohammed Alwohaibi\n### Date: 21-3-2024\n\n### Lets explore the relationship between the attributes.",

"metadata": {}

},

{

"id": "0e18a676",

"cell\_type": "code",

"source": "\ncorrs = data[['Release year', 'Rating', 'Runtime', 'Metascore', 'Votes', 'Gross']].corr()\n\nplt.figure(figsize=(10, 8))\nsns.heatmap(corrs, annot=True, cmap='coolwarm', vmax=1, vmin=-1)\nplt.title('Correlation Heatmap of Movie Attributes')\nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"name": "stderr",

"text": "<ipython-input-33-87c724cf1236>:1: FutureWarning: The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric\_only to silence this warning.\n corrs = data[['Release year', 'Rating', 'Runtime', 'Metascore', 'Votes', 'Gross']].corr()\n",

"output\_type": "stream"

},

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1000x800 with 2 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 33

},

{

"id": "01e21b16",

"cell\_type": "markdown",

"source": "From the figuer above, we can interprut that a correlation coefficient of 0.74 between \"Rating\" and \"Metascore\" indicates a strong positive correlation between the IMDb rating of movies and their Metascore ratings. This suggests that movies with higher IMDb ratings tend to also have higher Metascore ratings, indicating agreement between both rating systems.\n\nRegarding the correlation coefficient of 0.49 between \"Votes\" and \"Rating,\" it suggests a moderate positive correlation between the number of votes a movie receives and its IMDb rating. This implies that movies with higher IMDb ratings tend to attract more votes, indicating a level of popularity or viewer engagement.\n\nA correlation coefficient of 0.33 between \"runtime\" and \"votes\" indicates a moderate positive correlation between the runtime of movies and the number of votes they receive. This suggests that movies with longer runtimes tend to receive more votes, Similarly, a correlation coefficient of 0.31 between \"rating\" and \"gross\" indicates a moderate positive correlation between the rating of movies and their gross earnings. This suggests that movies with higher ratings tend to have higher gross earnings, but again, it's not a very strong relationship.",

"metadata": {}

},

{

"id": "01cb2ed3",

"cell\_type": "markdown",

"source": "## Task: Reflect on the relationship using scatter plot between one pair of varibles\n### Done by : Leen Alharbi\n### Date: 20-3-2024",

"metadata": {}

},

{

"id": "ae1cdc1c",

"cell\_type": "markdown",

"source": "### First, Scatter plot to reflect on the relationship between Gross and Rating",

"metadata": {}

},

{

"id": "88495999",

"cell\_type": "code",

"source": "plt.figure(figsize=(10,8))\nsns.regplot(x='Rating',y='Gross',data=data,color='yellow')",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 34,

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

"data": {

"text/plain": "<AxesSubplot:xlabel='Rating', ylabel='Gross'>"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1000x800 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 34

},

{

"id": "ff340e6f",

"cell\_type": "markdown",

"source": "Density of Data Points: The majority of the data points are clustered at the lower end of the Gross revenue axis, suggesting that most movies do not earn extremely high gross revenues, which is consistent with the industry where a few movies earn the majority of the revenue.\n\nScatter Distribution: There is a wide scatter in Gross revenue across all IMDb ratings. Movies with mid-range IMDb ratings (around 5 to 8) show a wide variation in Gross revenue, indicating that a moderate IMDb rating does not necessarily predict a movie's financial success.\n\nHigh Grossing Outliers: There are a few outliers with very high Gross revenues, which appear to be spread across a range of IMDb ratings from around 6 to 9. These are likely blockbuster hits that, regardless of their IMDb rating, managed to achieve significant box office success.\n\nLack of Strong Correlation: There does not appear to be a strong direct correlation between IMDb ratings and Gross revenue. While there are movies with high ratings that have high Gross revenues, there are also many with high ratings that do not gross as much, and vice versa.\n\nRating Threshold for High Gross: There seems to be a threshold IMDb rating (around 6) below which very few movies achieve high gross revenues. Most of the higher grossing movies have a rating above this threshold.\n\nNo Clear Trend: There is no clear upward or downward trend in the plot that would suggest higher ratings correspond to higher gross revenues or vice versa.\n\nIn summary, the plot suggests that while there may be a general positive trend (movies with higher ratings also tend to have higher gross revenues), there are many exceptions, and the relationship is not strong enough to make definitive predictions about a movie's financial success based solely on its IMDb rating. Other factors not depicted in the scatter plot.",

"metadata": {}

},

{

"id": "c673995e",

"cell\_type": "markdown",

"source": "### Next Scatter plot between Gross and Metascore",

"metadata": {}

},

{

"id": "0cce8799",

"cell\_type": "code",

"source": "plt.figure(figsize=(10,8))\nsns.regplot(x='Metascore',y='Gross',data=data,color='blue')",

"metadata": {

"trusted": true

},

"outputs": [

{

"execution\_count": 35,

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

"data": {

"text/plain": "<AxesSubplot:xlabel='Metascore', ylabel='Gross'>"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1000x800 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 35

},

{

"id": "e0b45cd4",

"cell\_type": "markdown",

"source": "Weak to No Apparent Correlation: The visualization does not show a clear linear relationship between Gross revenue and Metascore. The data points are widely spread out, indicating that a higher Metascore doesn't consistently align with higher Gross revenue, and vice versa.\n\nHigh Grossing Films Across Metascores: There are films with a wide range of Metascores that have achieved high Gross revenue. This indicates that factors other than critical acclaim, as measured by Metascore, may have a substantial impact on financial success.\n\nCluster of Lower Grossing Films: There is a concentration of data points at the lower end of the Gross revenue scale across various Metascores. This means most movies do not achieve enormous box office success regardless of their critical reception.\n\nOutliers: Some films show exceptionally high Gross revenue, which could be attributed to blockbuster status, but these do not necessarily have the highest Metascores. Conversely, some films with high Metascores do not show exceptionally high Gross revenue.\n\nIn summary, the relationship between Gross revenue and Metascore in this data appears to be non-linear and weak. Critical acclaim, as assessed by Metascore, is not a guaranteed predictor of financial performance at the box office. The success in terms of revenue is likely influenced by a multitude of factors, including but not limited to, marketing, distribution, audience appeal, star power, and competition at the time of release.",

"metadata": {}

},

{

"id": "ffb987b4",

"cell\_type": "markdown",

"source": "### Next Scatter Plot to show the relationship between Runtime and Gross",

"metadata": {}

},

{

"id": "7a5fd266",

"cell\_type": "code",

"source": "plt.figure(figsize=(10,8))\nsns.regplot(x='Runtime',y='Gross',data=data,color='red')",

"metadata": {

"scrolled": true,

"trusted": true

},

"outputs": [

{

"execution\_count": 36,

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

"data": {

"text/plain": "<AxesSubplot:xlabel='Runtime', ylabel='Gross'>"

},

"metadata": {}

},

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1000x800 with 1 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 36

},

{

"id": "f0c6daef",

"cell\_type": "markdown",

"source": "Weak Positive Correlation: The upward slope of the trend line suggests there is a weak positive correlation between runtime and gross. In general, as movies get longer, they might gross slightly more, but this relationship is not strong.\n\nConcentration of Data Points: There is a significant concentration of data points at the lower end of the Gross axis, indicating that most movies have a lower gross revenue. This is typical for the film industry, where a majority of movies do not become blockbuster hits.\n\nSpread of Gross Revenues: For any given runtime, there is a wide spread of gross revenues. This implies that runtime alone is not a strong predictor of a movie's financial success.\n\nOutliers: Some movies with long runtimes appear to have higher gross revenues, indicating that certain longer films can be quite successful. However, there are also long movies that do not gross as much.\n\nHigh Grossing Short Films: There are also outliers with shorter runtimes that have high gross revenues. These could be highly successful short films or films with significant cultural impact.\n\nPopular Movie Durations: The dense cluster of data points around the 90-120 minute mark corresponds with the typical runtime for feature films and indicates that most movies adhere to this industry standard.\n\nRuntime Limitation for Success: The trend line and data points suggest that there's no strong threshold effect where movies have to be a certain length to achieve higher gross revenue.\n\nThe relationship depicted by the trend line is a generalization, and the substantial scatter around it suggests that there are many other factors at play that influence a movie's gross revenue, such as genre, marketing, the presence of star actors, competition at the time of release, and overall market conditions.",

"metadata": {}

},

{

"id": "39668c33",

"cell\_type": "markdown",

"source": "## Task: Ploting Annual Movies trends\n### Done by : Leen Alharbi\n### Date: 20-3-2024",

"metadata": {}

},

{

"id": "983f1a48",

"cell\_type": "code",

"source": "#Annual Movies trends\n\n\nmovie\_DF['Release year'] = pd.to\_numeric(movie\_DF['Release year'], errors='coerce')\n\n\nannual\_stats = movie\_DF.groupby('Release year').agg({\n 'Rating': 'mean',\n 'Gross': 'sum'\n})\n\nannual\_stats = annual\_stats.reset\_index()\n\n\nfig, ax1 = plt.subplots(figsize=(14, 7))\n\ncolor = 'tab:blue'\nax1.set\_xlabel('Release year')\nax1.set\_ylabel('Average Rating', color=color)\nax1.plot(annual\_stats['Release year'], annual\_stats['Rating'], color=color)\nax1.tick\_params(axis='y', labelcolor=color)\n\nax2 = ax1.twinx() \n\n\ncolor = 'tab:red'\nax2.set\_ylabel('Total Gross', color=color) \nax2.plot(annual\_stats['Release year'], annual\_stats['Gross'], color=color)\nax2.tick\_params(axis='y', labelcolor=color)\n\n\nplt.title('Annual Movie Trends: Average Rating and Total Gross')\nfig.tight\_layout() \nplt.show()",

"metadata": {

"trusted": true

},

"outputs": [

{

"output\_type": "display\_data",

"data": {

"text/plain": "<Figure size 1400x700 with 2 Axes>",

"image/png": ""

},

"metadata": {}

}

],

"execution\_count": 37

},

{

"id": "c9171b4a",

"cell\_type": "markdown",

"source": "\n1- Average Rating Trend: The blue line shows fluctuations in average ratings over time. There are periods, especially mid-century, where average ratings spike, which could correspond to years with highly acclaimed movies. Starting around the 1980s, there's a noticeable downward trend in average ratings.\n\n2- Total Gross Trend: The red line indicates total gross revenue, which generally increases over time. There's a significant spike in gross revenue around the 2010s, followed by a sharp decline. The spike may indicate a year or period with exceptionally high-grossing films, while the decline could be due to incomplete data for recent years, a change in movie consumption patterns, or external factors affecting the film industry.\n\n3- Correlation between Rating and Gross: There doesn't seem to be a strong correlation between the average ratings and total gross revenue. High-grossing years don't necessarily coincide with years that had higher average ratings.\n\n\n\n4- The decline in average ratings over the past few decades could indicate either a change in movie production quality, a shift in critical perception, or possibly a larger volume of movies with lower ratings being produced.\n5-The sharp increase and subsequent decrease in total gross suggest that external factors (such as the rise of digital streaming platforms, changes in consumer behavior, economic factors, or data availability) might significantly impact box office performance.\n6- Data Outliers: The extreme spike in total gross could represent an outlier year with abnormally high revenue, possibly due to blockbuster releases, changes in ticket prices, or the way revenue was reported or collected.",

"metadata": {}

},

{

"id": "b6bf0502-3697-4035-9327-739b39740c0c",

"cell\_type": "markdown",

"source": "## Task: Balnace data:\n\nTo help the model provide accurate and high-quality results, the initial stage in this task is to encode textual values into numerical ones before balancing the data. Using a technique known as one-hot encoding That handled through the MultiLabelBinarize() , we encode the 'Genre' from a list to multiple binary columns.\r\nFollowing that, we assign numerical numbers to \"stars\" and \"director.\"\r\n\r\n'Title' was removed as the final step of encoding since it has no bearing on the contents.\r\n\r\nNext, we separated the class label  \"rating category,\" into training and testing sets, making sure to apply logistic regression models and SMOTE for balance. Lastly, we computed accuracy/recall to verify the balance.",

"metadata": {}

},

{

"id": "f8e8c86b-6659-42a9-8d9b-be3d0988d9ce",

"cell\_type": "code",

"source": "print(data.info())\nprint(data.head())\nprint(data.dtypes)",

"metadata": {

"trusted": true

},

"outputs": [

{

"name": "stdout",

"text": "<class 'pandas.core.frame.DataFrame'>\nInt64Index: 1997 entries, 1926 to 227\nData columns (total 31 columns):\n # Column Non-Null Count Dtype \n--- ------ -------------- ----- \n 0 Title 1997 non-null object \n 1 Release year 1997 non-null datetime64[ns]\n 2 Rating 1997 non-null float64 \n 3 Runtime 1997 non-null float64 \n 4 Metascore 1997 non-null float64 \n 5 Director 1997 non-null object \n 6 Stars 1997 non-null object \n 7 Votes 1997 non-null float64 \n 8 Gross 1997 non-null float64 \n 9 Rating Category 1997 non-null category \n 10 Action 1997 non-null int32 \n 11 Adventure 1997 non-null int32 \n 12 Animation 1997 non-null int32 \n 13 Biography 1997 non-null int32 \n 14 Comedy 1997 non-null int32 \n 15 Crime 1997 non-null int32 \n 16 Drama 1997 non-null int32 \n 17 Family 1997 non-null int32 \n 18 Fantasy 1997 non-null int32 \n 19 Film-Noir 1997 non-null int32 \n 20 History 1997 non-null int32 \n 21 Horror 1997 non-null int32 \n 22 Music 1997 non-null int32 \n 23 Musical 1997 non-null int32 \n 24 Mystery 1997 non-null int32 \n 25 Romance 1997 non-null int32 \n 26 Sci-Fi 1997 non-null int32 \n 27 Sport 1997 non-null int32 \n 28 Thriller 1997 non-null int32 \n 29 War 1997 non-null int32 \n 30 Western 1997 non-null int32 \ndtypes: category(1), datetime64[ns](1), float64(5), int32(21), object(3)\nmemory usage: 298.5+ KB\nNone\n Title Release year Rating Runtime Metascore Director \\\n1926 The Kid 1921-01-01 8.2 68.0 6.241758 Charles Chaplin \n1894 Nosferatu 1922-01-01 7.9 94.0 6.241758 F.W. Murnau \n1909 The Gold Rush 1925-01-01 8.1 95.0 6.241758 Charles Chaplin \n1167 Metropolis 1927-01-01 8.3 153.0 9.802198 Fritz Lang \n1204 City Lights 1931-01-01 8.5 87.0 9.901099 Charles Chaplin \n\n Stars Votes Gross \\\n1926 [Edna Purviance, Jackie Coogan, Carl Miller] 1.187510 1.052367 \n1894 [Max Schreck, Alexander Granach, Gustav von Wa... 1.093125 1.544088 \n1909 [Mack Swain, Tom Murray, Henry Bergman] 1.135890 1.052367 \n1167 [Brigitte Helm, Alfred Abel, Gustav FrÃ¶hlich,... 1.350098 1.011915 \n1204 [Virginia Cherrill, Florence Lee, Harry Myers] 1.384030 1.000192 \n\n Rating Category ... Horror Music Musical Mystery Romance Sci-Fi \\\n1926 Medium ... 0 1 0 0 0 0 \n1894 Medium ... 0 0 0 0 1 0 \n1909 Medium ... 0 0 0 0 0 0 \n1167 Medium ... 0 0 0 0 0 0 \n1204 High ... 0 0 0 0 0 0 \n\n Sport Thriller War Western \n1926 0 0 0 0 \n1894 0 0 0 0 \n1909 0 0 0 0 \n1167 0 0 0 0 \n1204 0 1 0 0 \n\n[5 rows x 31 columns]\nTitle object\nRelease year datetime64[ns]\nRating float64\nRuntime float64\nMetascore float64\nDirector object\nStars object\nVotes float64\nGross float64\nRating Category category\nAction int32\nAdventure int32\nAnimation int32\nBiography int32\nComedy int32\nCrime int32\nDrama int32\nFamily int32\nFantasy int32\nFilm-Noir int32\nHistory int32\nHorror int32\nMusic int32\nMusical int32\nMystery int32\nRomance int32\nSci-Fi int32\nSport int32\nThriller int32\nWar int32\nWestern int32\ndtype: object\n",

"output\_type": "stream"

}

],

"execution\_count": 40

},

{

"id": "85671100-a4b8-49e6-89b1-1dc783fc8ad1",

"cell\_type": "code",

"source": "import pandas as pd\nfrom sklearn.preprocessing import MultiLabelBinarizer\nmlb = MultiLabelBinarizer()\n# Assuming 'data' is DataFrame and 'Genre' is the column with lists\ngenres\_encoded = mlb.fit\_transform(data['Genre'])\ngenre\_labels = pd.DataFrame(genres\_encoded, columns=mlb.classes\_)\ndata = pd.concat([data.drop('Genre', axis=1), genre\_labels], axis=1)\n",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 39

},

{

"id": "19965f49-6b04-445a-b817-90871004411e",

"cell\_type": "code",

"source": "from sklearn.preprocessing import LabelEncoder\n\n# Calculate the frequency of each actor\nactor\_frequency = data['Stars'].explode().value\_counts().to\_dict()\n\n# Map frequency back to the DataFrame\ndata['Stars'] = data['Stars'].apply(lambda actors: sum([actor\_frequency[actor] for actor in actors]))\n",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 41

},

{

"id": "85277777-0621-4bb5-bf45-d206b72e2ad2",

"cell\_type": "code",

"source": "print(data.dtypes)",

"metadata": {

"trusted": true

},

"outputs": [

{

"name": "stdout",

"text": "Title object\nRelease year datetime64[ns]\nRating float64\nRuntime float64\nMetascore float64\nDirector object\nStars int64\nVotes float64\nGross float64\nRating Category category\nAction int32\nAdventure int32\nAnimation int32\nBiography int32\nComedy int32\nCrime int32\nDrama int32\nFamily int32\nFantasy int32\nFilm-Noir int32\nHistory int32\nHorror int32\nMusic int32\nMusical int32\nMystery int32\nRomance int32\nSci-Fi int32\nSport int32\nThriller int32\nWar int32\nWestern int32\ndtype: object\n",

"output\_type": "stream"

}

],

"execution\_count": 42

},

{

"id": "061f9d4c-ae32-47ec-9a30-66bce63417ad",

"cell\_type": "code",

"source": "# Properly drop the 'Title' column\ndata.drop('Title', axis=1, inplace=True)\ndata['Release year'] = data['Release year'].dt.year\n",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 43

},

{

"id": "9a2a0d92-79d7-4755-8431-a6fe72031798",

"cell\_type": "code",

"source": "# Create a label encoder object\nle = LabelEncoder()\n\n# Fit and transform the 'Director' column\ndata['Director'] = le.fit\_transform(data['Director'])\n# Now, 'Director' is converted to numeric labels\n",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 44

},

{

"id": "4ab8ae93-9b47-4908-a139-23b074ae5306",

"cell\_type": "code",

"source": "%pip install imbalanced-learn",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": 45

},

{

"id": "e264d669-05f2-48ed-b547-3c5813c336f5",

"cell\_type": "code",

"source": "import pandas as pd\nfrom sklearn.model\_selection import train\_test\_split\nfrom imblearn.over\_sampling import SMOTE\nfrom sklearn.linear\_model import LogisticRegression\nfrom sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score\nfrom sklearn.preprocessing import StandardScaler\n\n# Split data into features and target, maintaining the class distribution with stratified sampling\nX = data.drop('Rating Category', axis=1)\ny = data['Rating Category']\n\n# Use stratified sampling to ensure each class's proportion is maintained\nX\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=42)\n\n# Normalize the data to improve performance of logistic regression\nscaler = StandardScaler()\nX\_train\_scaled = scaler.fit\_transform(X\_train)\nX\_test\_scaled = scaler.transform(X\_test)\n\n# Use SMOTE to balance the training set\nsmote = SMOTE(random\_state=42)\nX\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train\_scaled, y\_train)\n\n# Train a Logistic Regression model on the rimport pandas as pd\nfrom sklearn.model\_selection import train\_test\_split\nfrom sklearn.preprocessing import StandardScaler\nfrom sklearn.linear\_model import LogisticRegression\nfrom sklearn.svm import LinearSVC\nfrom sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score\nfrom imblearn.over\_sampling import SMOTE\nfrom sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix\n\n\n# Assuming 'data' includes 'Rating Category' as the target and has multiple features including 'Metascore'\n\n# Prepare your features and target variable\nX = data.drop('Rating Category', axis=1) # All features\ny = data['Rating Category'] # Target variable\nX\_metascore = data[['Metascore']] # Only Metascore as feature\n\n# Use stratified sampling to ensure each class's proportion is maintained\nX\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=42)\nX\_train\_metascore, X\_test\_metascore = X\_train[['Metascore']], X\_test[['Metascore']]\n\n# Normalize the data to improve performance\nscaler = StandardScaler()\nX\_train\_scaled = scaler.fit\_transform(X\_train)\nX\_test\_scaled = scaler.transform(X\_test)\nX\_train\_metascore\_scaled = scaler.fit\_transform(X\_train\_metascore)\nX\_test\_metascore\_scaled = scaler.transform(X\_test\_metascore)\n\n# Use SMOTE to balance the training set\nsmote = SMOTE(random\_state=42)\nX\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train\_scaled, y\_train)\nX\_train\_metascore\_resampled, y\_train\_metascore\_resampled = smote.fit\_resample(X\_train\_metascore\_scaled, y\_train)\n\n# Train a Logistic Regression model on the resampled training data with all features\nmodel\_all\_features = LogisticRegression(random\_state=42, solver='lbfgs', multi\_class='multinomial', max\_iter=1000)\nmodel\_all\_features.fit(X\_train\_resampled, y\_train\_resampled)\n\n# Train another Logistic Regression model on the resampled training data with only Metascore\nmodel\_metascore = LogisticRegression(random\_state=42, solver='lbfgs', multi\_class='multinomial', max\_iter=1000)\nmodel\_metascore.fit(X\_train\_metascore\_resampled, y\_train\_metascore\_resampled)\n\n# Train a LinearSVC model on the resampled training data\nsvm\_model = LinearSVC(random\_state=42, max\_iter=1000)\nsvm\_model.fit(X\_train\_resampled, y\_train\_resampled)\ny\_pred\_svm = svm\_model.predict(X\_test\_scaled)\n\n# Perform predictions on the testing set with all features\ny\_pred\_all = model\_all\_features.predict(X\_test\_scaled)\n\n# Perform predictions on the testing set with only Metascore\ny\_pred\_metascore = model\_metascore.predict(X\_test\_metascore\_scaled)\n\n# Calculate and print the accuracy for both logistic regression models\naccuracy\_all = round(accuracy\_score(y\_test, y\_pred\_all) \* 100, 2)\naccuracy\_metascore = round(accuracy\_score(y\_test, y\_pred\_metascore) \* 100, 2)\n\n# Evaluate and print accuracy and other metrics for LinearSVC\naccuracy\_svm = accuracy\_score(y\_test, y\_pred\_svm)\nprecision\_svm = precision\_score(y\_test, y\_pred\_svm, average='macro')\nrecall\_svm = recall\_score(y\_test, y\_pred\_svm, average='macro')\nf1\_svm = f1\_score(y\_test, y\_pred\_svm, average='macro')\n\nprint(\"Accuracy with all features (Logistic Regression): \", accuracy\_all)\nprint(\"Accuracy with only Metascore (Logistic Regression): \", accuracy\_metascore)\nprint(\"LinearSVC Metrics:\")\nprint(f\"Accuracy: {accuracy\_svm:.2f}\")\nprint(f\"Precision: {precision\_svm:.2f}\")\nprint(f\"Recall: {recall\_svm:.2f}\")\nprint(f\"F1-score: {f1\_svm:.2f}\")\nmodel = LogisticRegression(random\_state=42, solver='lbfgs', multi\_class='multinomial', max\_iter=1000)\nmodel.fit(X\_train\_resampled, y\_train\_resampled)\n\n# Perform predictions on the testing set\ny\_pred = model.predict(X\_test\_scaled)\n\n# Calculate and print metrics\naccuracy = accuracy\_score(y\_test, y\_pred)\nprecision = precision\_score(y\_test, y\_pred, average='macro')\nrecall = recall\_score(y\_test, y\_pred, average='macro')\nf1 = f1\_score(y\_test, y\_pred, average='macro')\n\nprint(f\"Accuracy: {accuracy:.2f}\")\nprint(f\"Precision: {precision:.2f}\")\nprint(f\"Recall: {recall:.2f}\")\nprint(f\"F1-score: {f1:.2f}\")\n\n# Print confusion matrices for each model\ncm\_all\_features = confusion\_matrix(y\_test, y\_pred\_all)\ncm\_metascore = confusion\_matrix(y\_test, y\_pred\_metascore)\ncm\_svm = confusion\_matrix(y\_test, y\_pred\_svm)\n\nprint(\"Confusion Matrix for Logistic Regression (All Features):\")\nprint(cm\_all\_features)\nprint(\"\\nConfusion Matrix for Logistic Regression (Only Metascore):\")\nprint(cm\_metascore)\nprint(\"\\nConfusion Matrix for LinearSVC:\")\nprint(cm\_svm)\n",

"metadata": {

"trusted": true

},

"outputs": [

{

"name": "stderr",

"text": "/lib/python3.11/site-packages/sklearn/svm/\_classes.py:32: FutureWarning: The default value of `dual` will change from `True` to `'auto'` in 1.5. Set the value of `dual` explicitly to suppress the warning.\n warnings.warn(\n/lib/python3.11/site-packages/sklearn/svm/\_base.py:1250: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.\n warnings.warn(\n",

"output\_type": "stream"

},

{

"name": "stdout",

"text": "Accuracy with all features (Logistic Regression): 96.0\nAccuracy with only Metascore (Logistic Regression): 61.25\nLinearSVC Metrics:\nAccuracy: 0.95\nPrecision: 0.80\nRecall: 0.94\nF1-score: 0.84\nAccuracy: 0.96\nPrecision: 0.79\nRecall: 0.91\nF1-score: 0.83\nConfusion Matrix for Logistic Regression (All Features):\n[[ 8 0 2]\n [ 0 178 1]\n [ 12 1 198]]\n\nConfusion Matrix for Logistic Regression (Only Metascore):\n[[ 9 0 1]\n [ 3 147 29]\n [ 76 46 89]]\n\nConfusion Matrix for LinearSVC:\n[[ 9 0 1]\n [ 0 176 3]\n [ 12 2 197]]\n",

"output\_type": "stream"

}

],

"execution\_count": 51

},

{

"id": "2b726464-62fa-4dd0-b9a2-b67880fa5f17",

"cell\_type": "code",

"source": "print(data.dtypes)",

"metadata": {

"trusted": true

},

"outputs": [

{

"name": "stdout",

"text": "Release year int64\nRating float64\nRuntime float64\nMetascore float64\nDirector int32\nStars int64\nVotes float64\nGross float64\nRating Category category\nAction int32\nAdventure int32\nAnimation int32\nBiography int32\nComedy int32\nCrime int32\nDrama int32\nFamily int32\nFantasy int32\nFilm-Noir int32\nHistory int32\nHorror int32\nMusic int32\nMusical int32\nMystery int32\nRomance int32\nSci-Fi int32\nSport int32\nThriller int32\nWar int32\nWestern int32\ndtype: object\n",

"output\_type": "stream"

}

],

"execution\_count": 47

},

{

"id": "c81dad8f-36d6-41d9-aec4-d423eeb6e353",

"cell\_type": "code",

"source": "",

"metadata": {

"trusted": true

},

"outputs": [],

"execution\_count": null

}

]

}