**Movie Analytics dashboards**



**Project Report**

**Submitted by : Submitted to :**

**Bhavishya(23csu071) Mrs. Bhavna Sharma**

**Ayushmaan(23csu067)**

**Aryan Sengar(23csu055)**

**Ayush(23csu065)**

**Introduction**

The code presented is a Python-based graphical application for visualizing and analysing data from an IMDb Top 1000 movies dataset. It uses several libraries to perform data processing, visualization, and GUI creation. The application enables users to explore genre distributions, release year trends, top directors, IMDb ratings distributions, and filter movies by genre.

**CODE-**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg

import tkinter as tk

from tkinter import ttk, END, BOTH, Label

# Load Dataset

try:

    # Replace this path with the correct path to your CSV file

    file\_path = r'C:\Users\Bhavishya\Desktop\Python\imdb\_top\_1000.csv'

    imdb\_data = pd.read\_csv(file\_path)

    # Clean and process data

    imdb\_data['Released\_Year'] = pd.to\_numeric(imdb\_data['Released\_Year'], errors='coerce')

    imdb\_data = imdb\_data.dropna(subset=['Released\_Year']).reset\_index(drop=True)

except Exception as e:

    print(f"Error loading dataset: {e}")

    imdb\_data = pd.DataFrame()

# Functions for Visualizations

def show\_genre\_distribution():

    print("Displaying genre distribution...")

    genres = imdb\_data['Genre'].str.split(', ').dropna()

    all\_genres = [genre for sublist in genres for genre in sublist]

    genre\_counts = pd.Series(all\_genres).value\_counts()

    if genre\_counts.empty:

        display\_message("No genre data available.")

        return

    fig, ax = plt.subplots(figsize=(6, 4))

    sns.barplot(x=genre\_counts.values[:10], y=genre\_counts.index[:10], palette='viridis', ax=ax)

    ax.set\_title('Top 10 Movie Genres')

    ax.set\_xlabel('Number of Movies')

    ax.set\_ylabel('Genre')

    display\_plot(fig)

def show\_release\_year\_trend():

    print("Displaying release year trend...")

    movies\_by\_year = imdb\_data['Released\_Year'].value\_counts().sort\_index()

    if movies\_by\_year.empty:

        display\_message("No release year data available.")

        return

    fig, ax = plt.subplots(figsize=(6, 4))

    ax.plot(movies\_by\_year.index, movies\_by\_year.values, marker='o', linestyle='-', color='blue')

    ax.set\_title('Movies Released Over Time')

    ax.set\_xlabel('Year')

    ax.set\_ylabel('Number of Movies')

    ax.grid(True)

    display\_plot(fig)

def show\_top\_directors():

    print("Displaying top directors...")

    top\_directors = imdb\_data['Director'].value\_counts().head(10)

    if top\_directors.empty:

        display\_message("No director data available.")

        return

    fig, ax = plt.subplots(figsize=(6, 4))

    sns.barplot(x=top\_directors.values, y=top\_directors.index, palette='mako', ax=ax)

    ax.set\_title('Top 10 Directors')

    ax.set\_xlabel('Number of Movies')

    ax.set\_ylabel('Director')

    display\_plot(fig)

def show\_ratings\_distribution():

    print("Displaying ratings distribution...")

    if imdb\_data['IMDB\_Rating'].isnull().all():

        display\_message("No ratings data available.")

        return

    fig, ax = plt.subplots(figsize=(6, 4))

    sns.histplot(imdb\_data['IMDB\_Rating'], bins=10, kde=True, color='orange', ax=ax)

    ax.set\_title('Distribution of IMDb Ratings')

    ax.set\_xlabel('IMDb Rating')

    ax.set\_ylabel('Frequency')

    display\_plot(fig)

def filter\_movies\_by\_genre():

    print("Filtering movies by genre...")

    selected\_genre = genre\_var.get()

    filtered\_movies = imdb\_data[imdb\_data['Genre'].str.contains(selected\_genre, na=False, case=False)]

    for widget in table\_frame.winfo\_children():

        widget.destroy()

    if not filtered\_movies.empty:

        table = ttk.Treeview(table\_frame, columns=("Title", "Rating", "Year"), show='headings', height=10)

        table.heading("Title", text="Series Title")

        table.heading("Rating", text="IMDB Rating")

        table.heading("Year", text="Released Year")

        # Adding rows to the table

        for \_, row in filtered\_movies[['Series\_Title', "IMDB\_Rating", 'Released\_Year']].iterrows():

            table.insert("", END, values=(row['Series\_Title'], row['IMDB\_Rating'], int(row['Released\_Year'])))

        table.pack(fill=BOTH, expand=True)

    else:

        Label(table\_frame, text="No Movies Found in Selected Genre!", fg="red", font=("Arial", 12)).pack()

def display\_plot(fig):

    # Clear previous plots

    for widget in plot\_frame.winfo\_children():

        widget.destroy()

    canvas = FigureCanvasTkAgg(fig, master=plot\_frame)

    canvas.draw()

    canvas.get\_tk\_widget().pack(fill=BOTH, expand=True)

def display\_message(message):

    for widget in plot\_frame.winfo\_children():

        widget.destroy()

    Label(plot\_frame, text=message, fg="red", font=("Arial", 14)).pack()

# Main Window

root = tk.Tk()

root.title("Movie Analytics Dashboard")

root.geometry("1200x800")

root.configure(bg="#d6eaf3")  # Light blue background

# Title

title\_label = tk.Label(root, text="🎬 Movie Analytics Dashboard", font=("Arial", 20, "bold"), bg="#d6eaf3", fg="black")

title\_label.pack(pady=20)

# Button Frame

button\_frame = tk.Frame(root, bg="#d6eaf3")

button\_frame.pack(pady=10)

ttk.Button(button\_frame, text="Genre Distribution", command=show\_genre\_distribution).grid(row=0, column=0, padx=15, pady=10)

ttk.Button(button\_frame, text="Release Year Trend", command=show\_release\_year\_trend).grid(row=0, column=1, padx=15, pady=10)

ttk.Button(button\_frame, text="Top Directors", command=show\_top\_directors).grid(row=0, column=2, padx=15, pady=10)

ttk.Button(button\_frame, text="Ratings Distribution", command=show\_ratings\_distribution).grid(row=0, column=3, padx=15, pady=10)

# Filter Movies by Genre

filter\_frame = tk.Frame(root, bg="#d6eaf3")

filter\_frame.pack(pady=20)

genre\_label = tk.Label(filter\_frame, text="Select Genre:", font=("Arial", 14), bg="#d6eaf3")

genre\_label.pack(side=tk.LEFT, padx=10)

genre\_var = tk.StringVar()

genres = imdb\_data['Genre'].str.split(', ').dropna()

unique\_genres = sorted(set([genre for sublist in genres for genre in sublist]))

genre\_dropdown = ttk.Combobox(filter\_frame, textvariable=genre\_var, values=unique\_genres, state="readonly", width=20, font=("Arial", 12))

genre\_dropdown.pack(side=tk.LEFT, padx=10)

ttk.Button(filter\_frame, text="Filter Movies", command=filter\_movies\_by\_genre).pack(side=tk.LEFT, padx=10)

# Plot and Table Frames

plot\_frame = tk.Frame(root, bg="#d6eaf3", width=1000, height=500)

plot\_frame.pack(pady=20, fill=BOTH, expand=True)

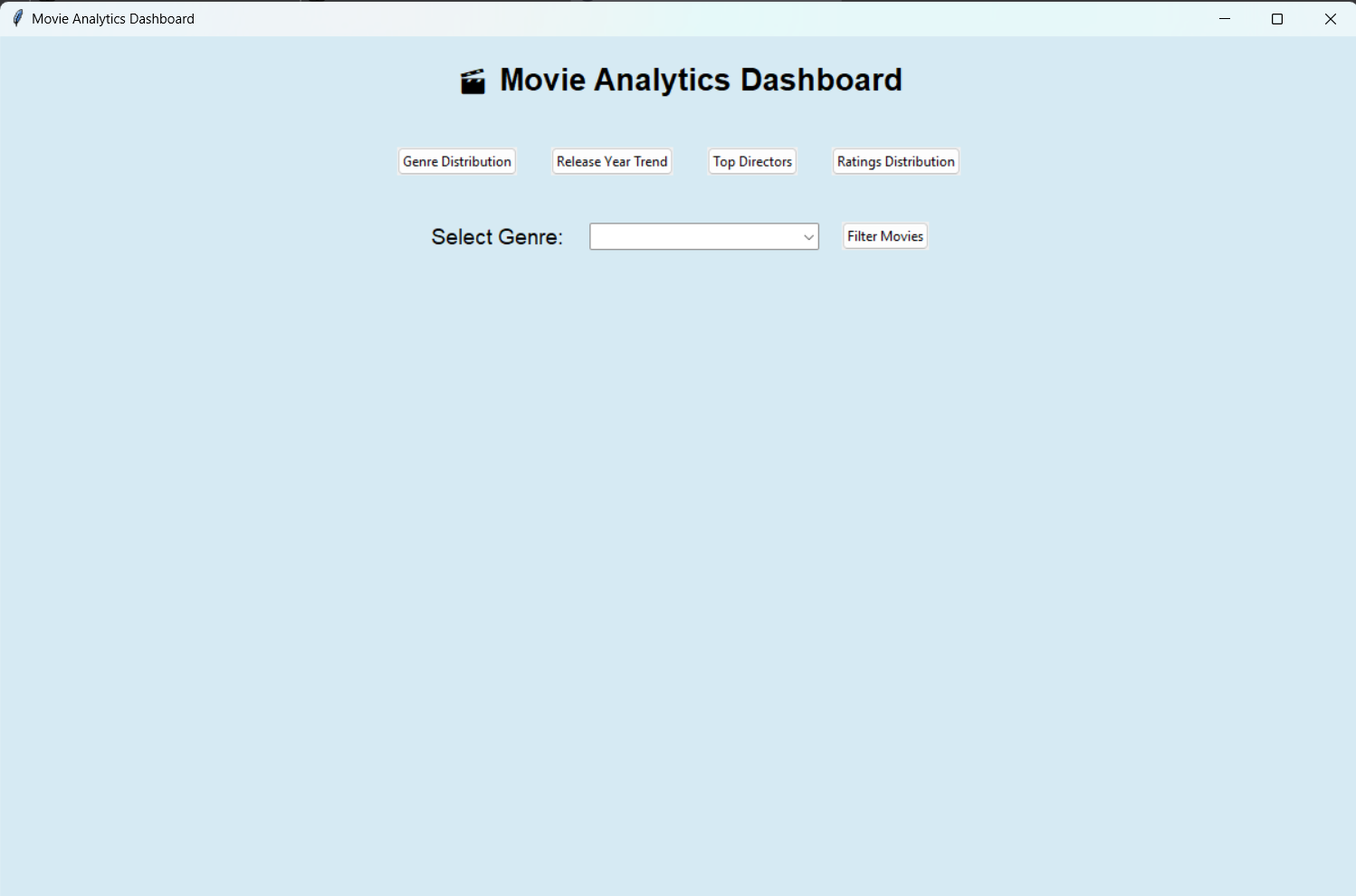
table\_frame = tk.Frame(root, bg="#d6eaf3", width=1000, height=300)

table\_frame.pack(pady=20, fill=BOTH, expand=True)

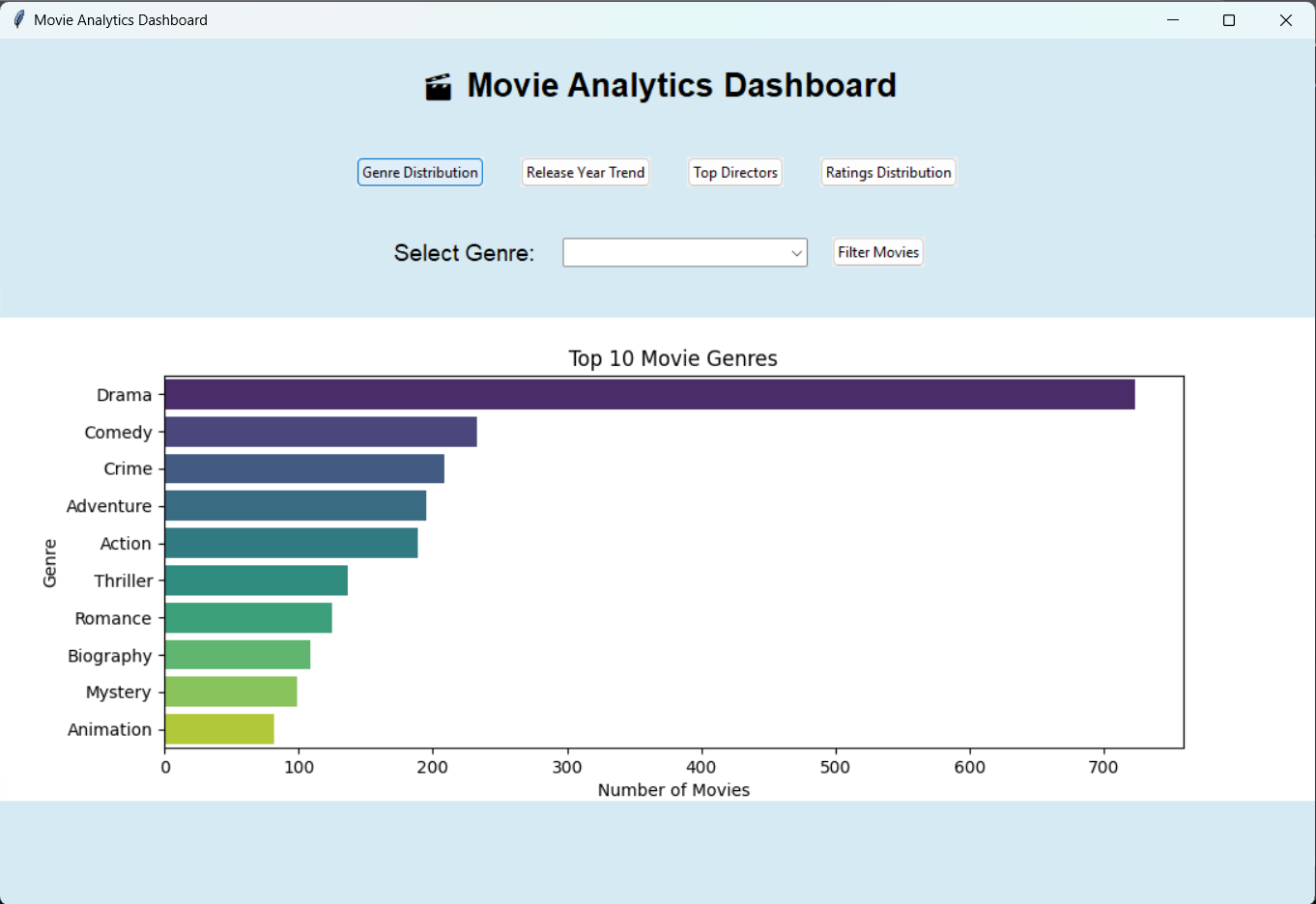
root.mainloop()

**Code snippets-**

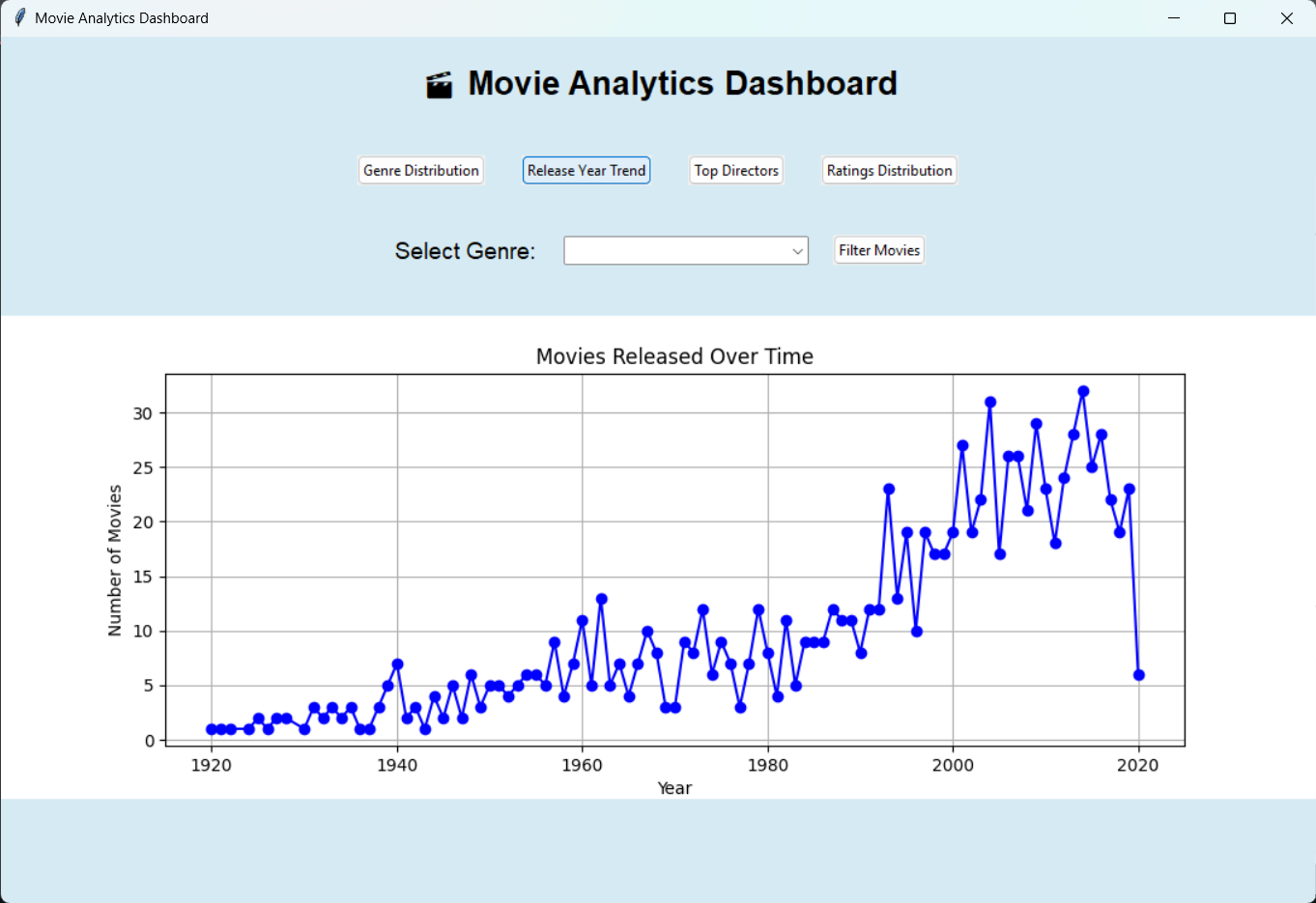
1)Starting Window



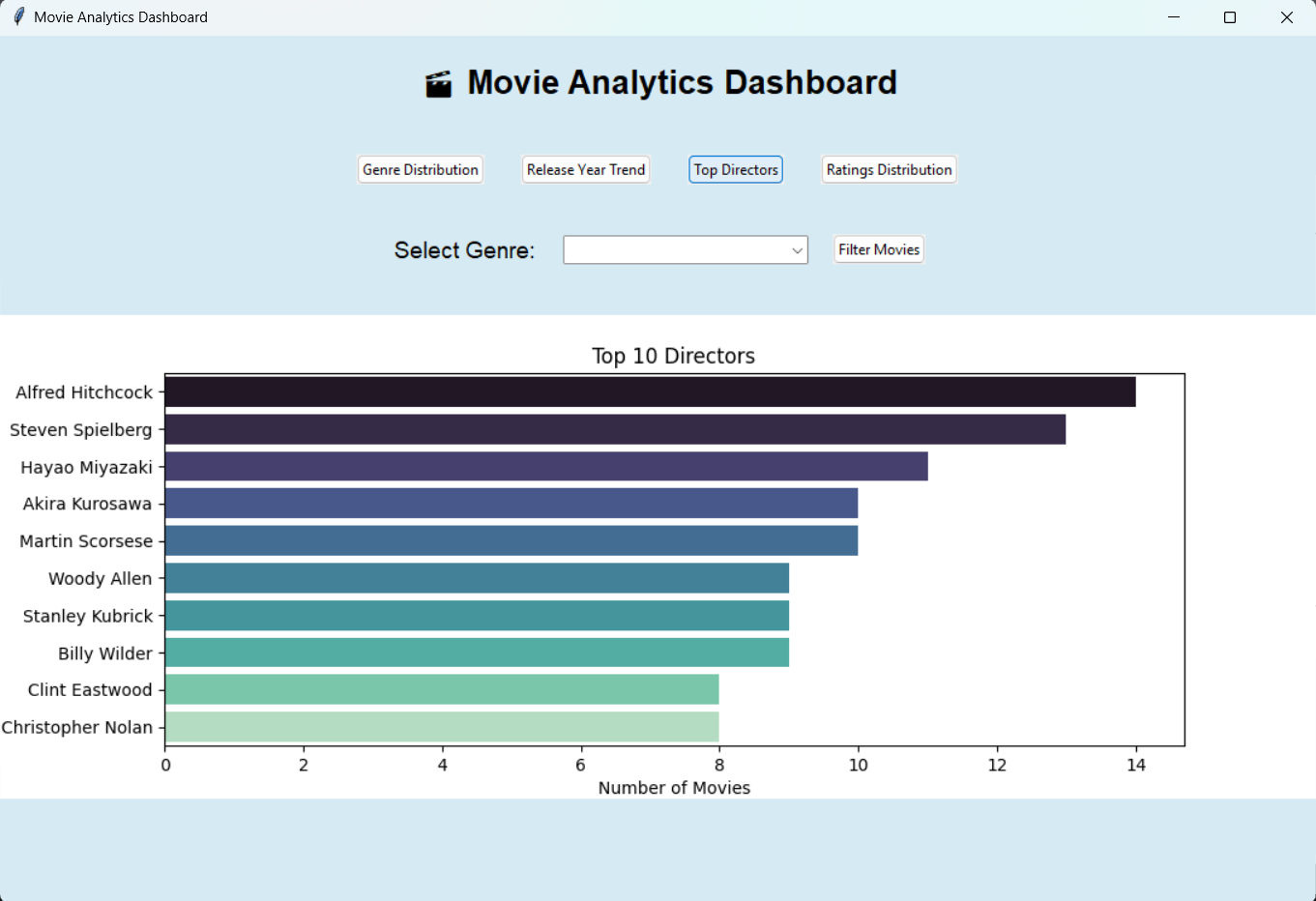
2)Genre Distribution table



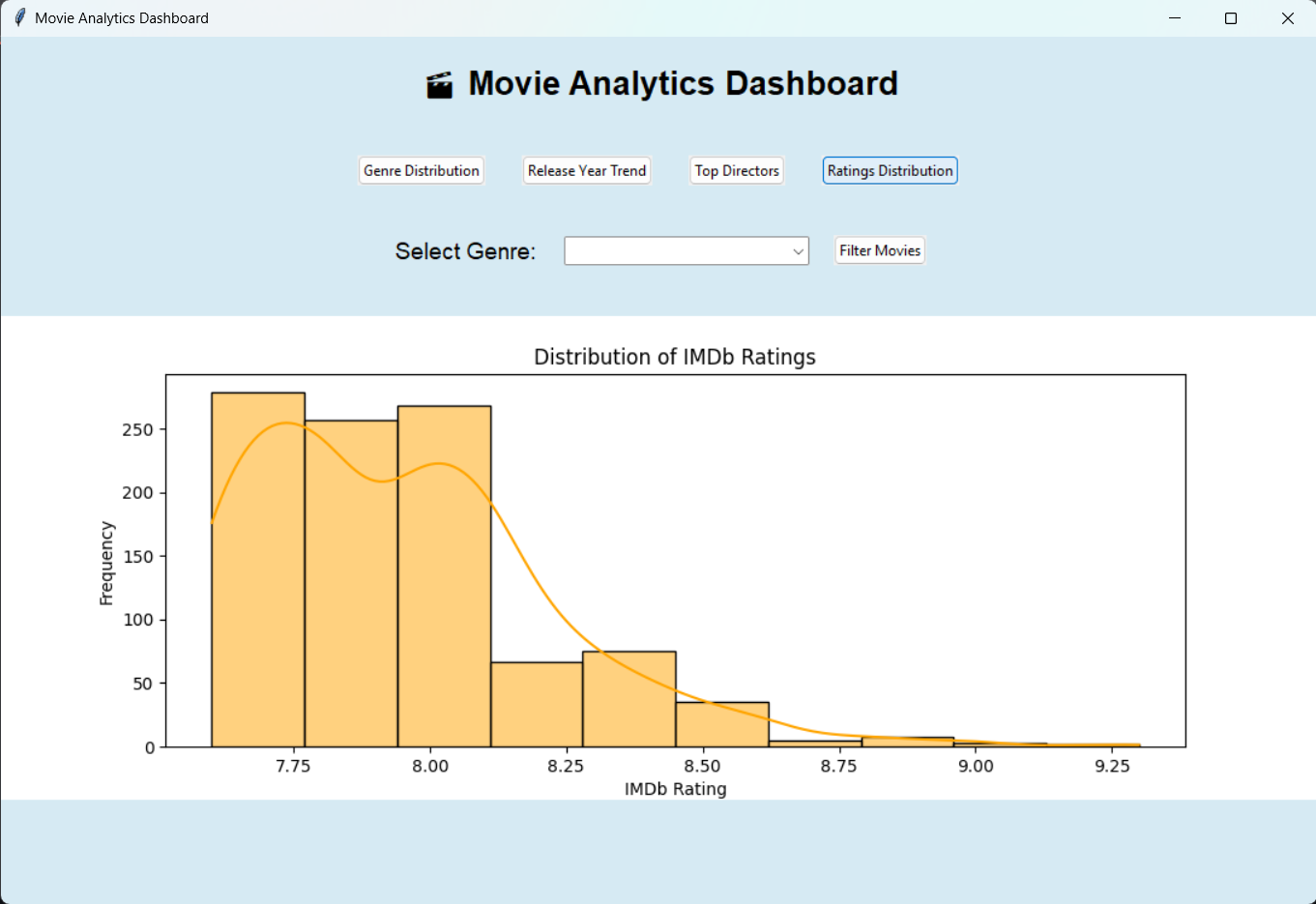
3)Release Year trend graph



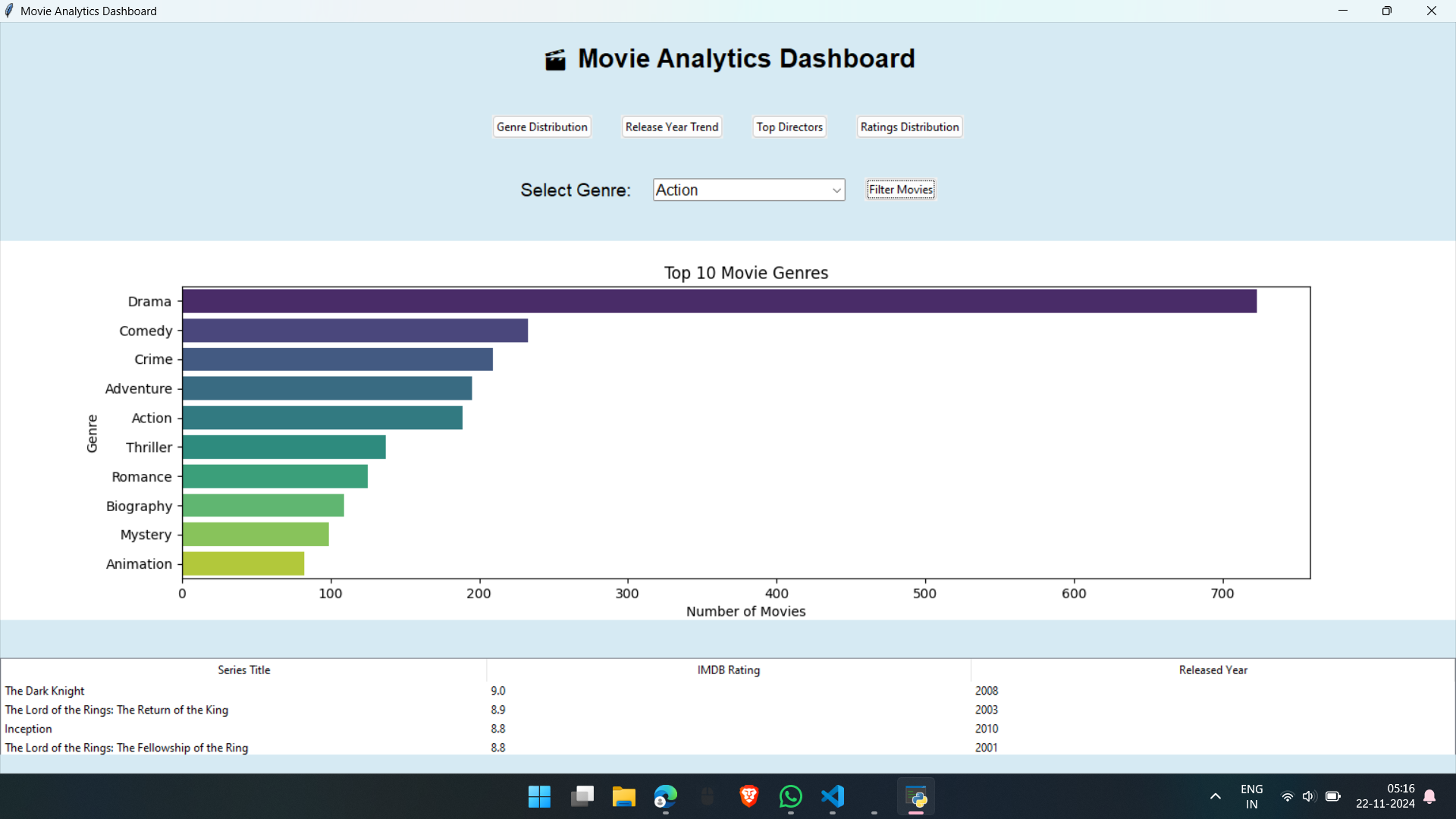
4)Top Directors chart



5)Rating distribution of all the records



6)Search by Genre



**Libraries Used**

**1. Pandas**

* **Purpose:** Data manipulation and analysis.
* **Usage:**
  + Loading the CSV file containing IMDb data.
  + Cleaning and transforming data (e.g., handling missing or invalid values).
  + Extracting and aggregating relevant data for visualizations and filtering.

**2. Matplotlib**

* **Purpose:** Data visualization.
* **Usage:**
  + Creating line plots and embedding them into the GUI.
  + Plotting trends such as movies released over time.

**3. Seaborn**

* **Purpose:** Advanced statistical data visualization built on Matplotlib.
* **Usage:**
  + Creating aesthetically pleasing bar plots and histograms.
  + Enhancing visualizations with color palettes and themes.

**4. Tkinter**

* **Purpose:** Building graphical user interfaces (GUIs).
* **Usage:**
  + Designing and structuring the application window.
  + Adding buttons, labels, dropdown menus, and frames for user interactions.
  + Displaying filtered movie data in a table format.

**5. Matplotlib Backends**

* **Purpose:** Integrating Matplotlib plots into the Tkinter GUI.
* **Usage:**
  + FigureCanvasTkAgg is used to embed Matplotlib plots directly within the Tkinter window.

**Functions Explained**

**1. show\_genre\_distribution()**

* **Purpose:** Display the distribution of movie genres.
* **Implementation:**
  + Extracts and counts individual genres from the dataset.
  + Plots the top 10 genres using a bar plot (Seaborn).
  + Displays the plot in the GUI.

**2. show\_release\_year\_trend()**

* **Purpose:** Visualize the trend of movie releases over time.
* **Implementation:**
  + Aggregates movies by their release years.
  + Plots the trend using a line chart (Matplotlib).
  + Embeds the chart in the GUI.

**3. show\_top\_directors()**

* **Purpose:** Display the top 10 directors by the number of movies.
* **Implementation:**
  + Counts the occurrences of directors in the dataset.
  + Plots a horizontal bar chart (Seaborn) to highlight the top 10.
  + Embeds the plot in the GUI.

**4. show\_ratings\_distribution()**

* **Purpose:** Display the distribution of IMDb ratings.
* **Implementation:**
  + Uses a histogram (Seaborn) with a KDE (Kernel Density Estimation) overlay.
  + Highlights the spread and frequency of IMDb ratings in the dataset.
  + Displays the plot in the GUI.

**5. filter\_movies\_by\_genre()**

* **Purpose:** Filter movies based on a selected genre and display them in a table.
* **Implementation:**
  + Extracts movies containing the selected genre.
  + Displays the movies in a Treeview table (Tkinter) with columns for title, rating, and year.
  + Handles cases where no movies match the selected genre by displaying an error message.

**6. display\_plot(fig)**

* **Purpose:** Render plots in the GUI.
* **Implementation:**
  + Clears previous plots in the plot frame.
  + Embeds the given figure (fig) using FigureCanvasTkAgg.

**7. display\_message(message)**

* **Purpose:** Display error or informational messages in the plot area.
* **Implementation:**
  + Clears the plot frame and displays a text message.

**Main Components**

**1. Dataset Loading and Cleaning**

* The dataset is loaded from a specified file path using Pandas.
* Non-numeric values in the Released\_Year column are coerced to NaN, and rows with missing values are dropped.
* Genres are split into individual components to allow for accurate aggregation.

**2. Graphical User Interface**

* Built using **Tkinter**, the GUI includes:
  + **Buttons** for invoking visualization functions.
  + **Dropdown menus** for filtering movies by genre.
  + **Frames** to organize plots and tables.

**3. Visualization**

* Seaborn and Matplotlib are used to create interactive and informative plots.
* Plots are seamlessly integrated into the GUI using FigureCanvasTkAgg.

**4. Interactive Features**

* Users can filter movies based on genres via a dropdown menu.
* Tabular views are dynamically generated to display filtered data.

**5. Error Handling**

* The application gracefully handles errors such as:
  + Missing data in the dataset.
  + Invalid genres or empty filtered results.
* Appropriate messages are displayed in the GUI.

**Future Improvements**

The current application provides a robust foundation for movie data analytics, but several enhancements can improve its functionality and user experience:

1. **Filtering by Additional Attributes**
   * Add options to filter movies by release year, IMDb rating, or director.
   * Use sliders or additional dropdown menus for more refined filtering.
2. **Combining Multiple Filters**
   * Allow users to apply multiple filters simultaneously (e.g., genre and release year).
   * Implement multi-select dropdowns or checkboxes for genres.
3. **Interactive Plots**
   * Enable users to interact with plots (e.g., clicking on a bar in the genre distribution chart to filter movies by that genre).
   * Use libraries like **Plotly** for advanced interactivity.
4. **Export Options**
   * Add functionality to export filtered data or visualizations as CSV files or images.
5. **Improved Aesthetics**
   * Enhance the visual appearance with themes using libraries like **ttkbootstrap**.
   * Add dynamic resizing and responsive layouts.
6. **Search Functionality**
   * Include a search bar for users to quickly find movies by title or keyword.

**Conclusion**

This code demonstrates a comprehensive approach to building a data-driven GUI application for movie analytics. Future improvements such as multi-attribute filtering, advanced interactivity, and aesthetic enhancements will further enhance its usability, making it a more versatile and engaging tool for users.