Understanding User Preferences & Enhancing Content Recommendations on Netflix

Group Members :

Kavisha Elvitigala IT2020069
Fathima Zahara IT2020063
R.I.Hewapathirana IT2020079
Pramoda Udakanajali IT2020065
A.L.D.P.Maheshika IT2020006

Outline

- Identified problem
- Data Understanding
- Data preprocessing
- Data Mining
- Evaluation and Interpretation
- Deployment

The problem identified

- Netflix faces the challenge of recommending personalized content to its users, aiming to enhance its user engagement and retaining subscribers.
- Hence, despite the availability of the user ratings and the user history, there is a need for algorithms to accurately predict user preference and give recommendations to individuals.
- Further, in order to achieve this it is important to analyze the data related to the Netflix viewers and the contents to get a better understanding of user behavior patterns.

Data Understanding

Data collection – Keggal dataset (title.csv)

Data exploration

- A thorough data exploration using the **pandas** and **numpy** libraries to gain a comprehensive understanding of the dataset.

- The steps involved are as followed
  - Checking dimensions
  - Inspecting data in the dataset
  - Checking existing columns and variable types
  - Checking for null and unique values
  - Statistical details for numerical columns
  - Identifying categorical and numerical variables

Data Cleaning

- **Remove Duplicates**
  - Identify and remove duplicate rows to ensure data integrity.
- **Handle Missing Values**
  - Identify missing values and decide on a strategy: remove, fill with mean/median/mode, or use interpolation.
- **Correct Data Types**
  - Ensure each column has the correct data type (e.g., integers, floats, dates).
- **Standardize Formats**
  - Ensure consistency in data formats (e.g., date formats, text capitalization).
- **Fix Inconsistencies**
  - Correct inconsistencies in data entries (e.g., spelling errors, inconsistent naming conventions).

Data Cleaning

- **Remove Outliers**
  - Identify and handle outliers that can skew analysis results.
- **Filter Unnecessary Data**
  - Remove irrelevant or redundant columns that do not add value to the analysis.
- **Normalize Data**
  - Scale numerical data to a standard range (e.g., 0-1) to ensure comparability.
- **Validate Data**
  - Check for logical consistency and validity (e.g., dates in chronological order, valid ranges for scores).

Data Transformation

- **Feature Engineering**
  - Create new features from existing ones to provide additional insights or improve model performance.
- **Scaling**
  - Scale numerical features to a similar range (e.g., using Min-Max scaling or Standardization) to prevent certain features from dominating the model.
- **Encoding Categorical Variables**
  - Convert categorical variables into numerical representations using techniques like one-hot encoding or label encoding for model compatibility.

Descriptive Analysis

- Descriptive analysis is a crucial step in understanding and summarizing the main features of a dataset. It involves the use of statistical techniques to describe and summarize data in a meaningful way. This helps in uncovering patterns, trends, and insights, which are essential for further data analysis or decision-making processes. Here's an overview of what descriptive analysis entails:

1. Descriptive Statistics for Numerical Variables

Descriptive statistics for numerical variables provide a summary of the data distribution and include measures such as:

- **Count**: The number of observations in the dataset.
- **Mean**: The average value of the data.
- **Standard Deviation (std)**: A measure of the dispersion or spread of the data.
- **Minimum (min)**: The smallest value in the dataset.
- **Maximum (max)**: The largest value in the dataset.
- **Percentiles (25%, 50%, 75%)**: These values divide the data into quarters, providing insights into the data distribution. The 50th percentile is the median.

For example, in the Netflix dataset:

1.Summary Statistics

```
[ ] numerical_variables = ['release_year', 'runtime', 'imdb_score', 'imdb_votes', 'tmdb_popularity', 'tmdb_score']
    numerical_statistics = df[numerical_variables].describe()

    print("Numerical Variables Statistics:")
    print(numerical_statistics)
```

Output

```
[ ]
Numerical Variables Statistics:
       release_year      runtime   imdb_score    imdb_votes  tmdb_popularity  \
count   5429.000000  5429.000000  5429.000000  5.429000e+03      5429.000000
mean    2016.324922    78.296740     6.505648  2.307513e+04        23.363137
std        6.982593    38.791231     1.149268  9.202175e+04        83.049843
min     1954.000000     0.000000     1.500000  5.000000e+00         0.600000
25%     2016.000000    45.000000     5.800000  5.320000e+02         2.950000
50%     2018.000000    85.000000     6.600000  2.325000e+03         7.289000
75%     2020.000000   105.000000     7.300000  1.051500e+04        17.830000
max     2022.000000   240.000000     9.000000  2.294231e+06      2274.044000

       tmdb_score
count  5429.000000
mean      6.820413
std       1.120670
min       1.000000
25%       6.200000
50%       6.822991
75%       7.500000
max      10.000000
```

```
[ ] categorical_statistics = {}
    for var in categorical_variables:
        counts = df[var].value_counts()
        percentages = (counts / counts.sum()) * 100
        categorical_statistics[var] = pd.DataFrame({'Counts': counts, 'Percentage': percentages})


    print("\nCategorical Variables Statistics:")
    for var, stats in categorical_statistics.items():
        print(f"\n{var.capitalize()} Statistics:")
        print(stats)
```

Output

```
[ ]
Categorical Variables Statistics:

Type Statistics:
       Counts  Percentage
type
MOVIE    3475   64.008105
SHOW     1954   35.991895

Genres Statistics:
                                                  Counts  Percentage
genres
['comedy']                                           442    8.141463
['drama']                                            280    5.157488
['documentation']                                    270    4.973292
['comedy', 'drama']                                  124    2.284030
['drama', 'romance']                                 122    2.247191
...                                                  ...         ...
['thriller', 'crime', 'drama', 'western']              1    0.018420
['drama', 'scifi', 'fantasy', 'horror']                1    0.018420
['horror', 'fantasy', 'thriller']                      1    0.018420
['drama', 'action', 'war', 'history']                  1    0.018420
['documentation', 'music', 'reality']                  1    0.018420

[1700 rows x 2 columns]

Production_countries Statistics:
                                  Counts  Percentage
production_countries
['US']                              1827   33.926893
```

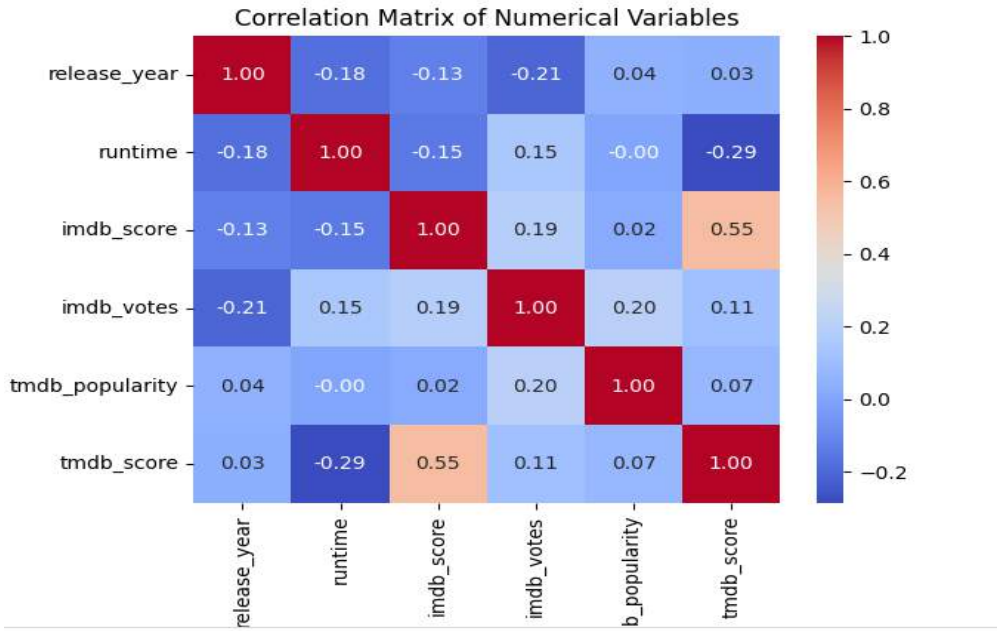
2.Relationship Analysis

1.Correlation analysis between numerical variables

```
[ ]  numerical_variables = ['release_year', 'runtime', 'imdb_score', 'imdb_votes', 'tmdb_popularity', 'tmdb_score']
     correlation_matrix = df[numerical_variables].corr()
```

```
[ ]  sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
     plt.title("Correlation Matrix of Numerical Variables")
     plt.show()
```

Output



2.Cross tabulation between categorical variables

```
[ ]  categorical_variables = ['type', 'genres', 'production_countries']

     for var in categorical_variables:
         cross_tab = pd.crosstab(df[var], df['type'])

         print(f"\nCross-tabulation for {var.capitalize()} and Type:\n")
         print(cross_tab)
```

Output

```
Cross-tabulation for Type and Type:

type    MOVIE  SHOW
type
MOVIE    3475     0
SHOW        0  1954

Cross-tabulation for Genres and Type:

type                                                  MOVIE  SHOW
genres
['action', 'animation', 'comedy', 'drama', 'fam...        0     1
['action', 'animation', 'comedy', 'drama', 'fam...        0     2
['action', 'animation', 'comedy', 'family', 'fa...        1     1
['action', 'animation', 'comedy', 'family', 'mu...        0     1
['action', 'animation', 'comedy', 'family']               0     3
...                                                     ...   ...
['western', 'history', 'drama']                           1     0
['western', 'horror', 'action']                           1     0
['western', 'thriller', 'horror']                         1     0
['western']                                               0     1
[]                                                        1     5

[1700 rows x 2 columns]

Cross-tabulation for Production_countries and Type:
```

Data Visualization

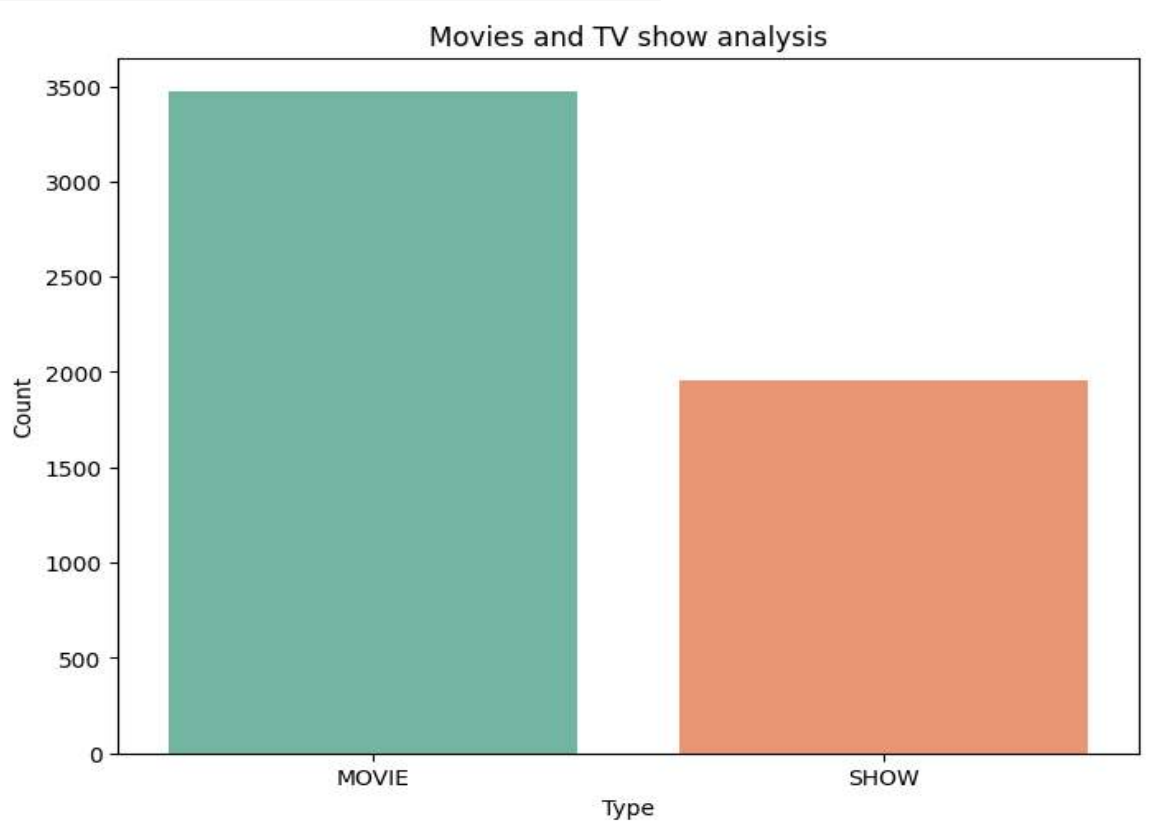
Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. In the context of the Netflix dataset, data visualization helps in summarizing and communicating the insights from the data effectively.

Importance of Data Visualization

1. **Simplifies Complex Data**: Transforms large and complex datasets into visual formats that are easier to understand.
2. **Identifies Trends and Patterns**: This makes it easier to see patterns, trends, and correlations that might not be obvious from raw data.
3. **Facilitates Comparison**: Allows for quick comparisons between different datasets or different aspects of the same dataset.
4. **Aids Decision-Making**: Supports better decision-making by presenting data clearly and concisely.
5. **Engages Audience**: More engaging and easier to interpret than tables of numbers, making it effective for presentations and reports.

Analysis of movies and tv-shows

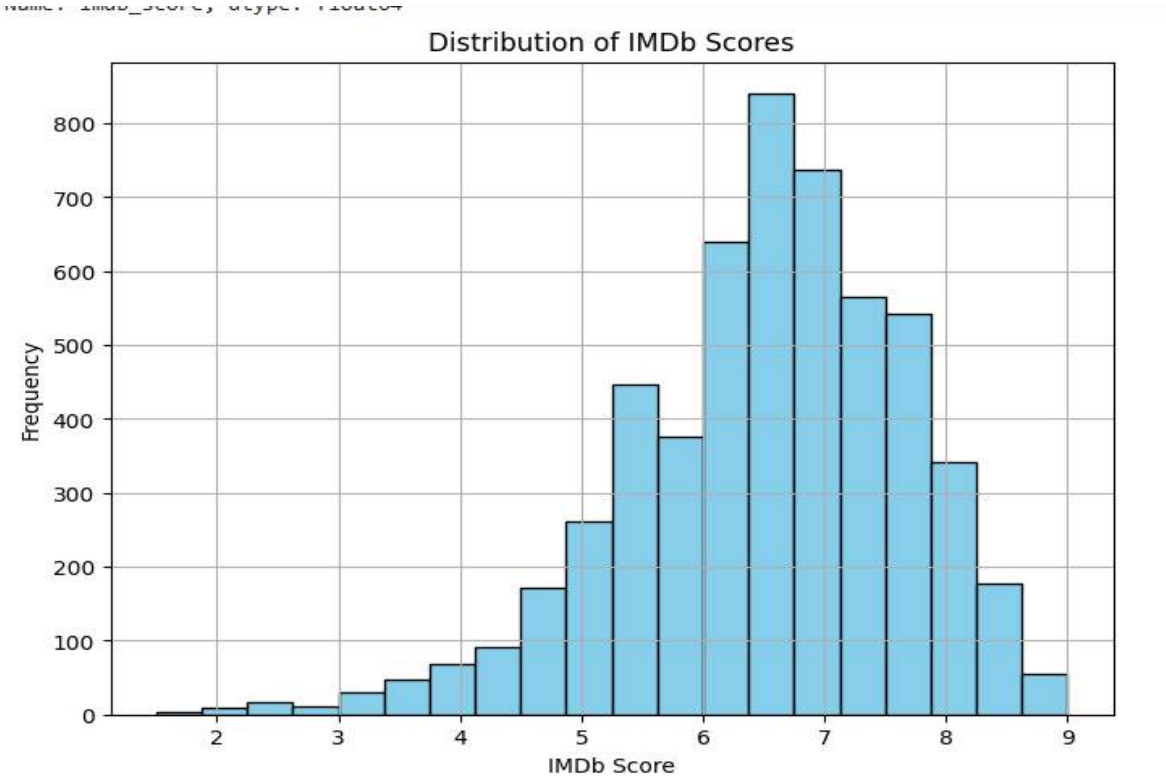
```
plt.figure(figsize=(8, 6))
ax = sns.countplot(x="type", hue="type", data=df, palette="Set2")
plt.title('Movies and TV show analysis')
plt.xlabel('Type')
plt.ylabel('Count')
plt.show()
```



1.IMDb Score Analysis

```
[ ]  imdb_score_stats = df['imdb_score'].describe()
     print("IMDb Score Summary Statistics:")
     print(imdb_score_stats)

     plt.figure(figsize=(8, 6))
     plt.hist(df['imdb_score'], bins=20, color='skyblue', edgecolor='black')
     plt.xlabel('IMDb Score')
     plt.ylabel('Frequency')
     plt.title('Distribution of IMDb Scores')
     plt.grid(True)
     plt.show()
```

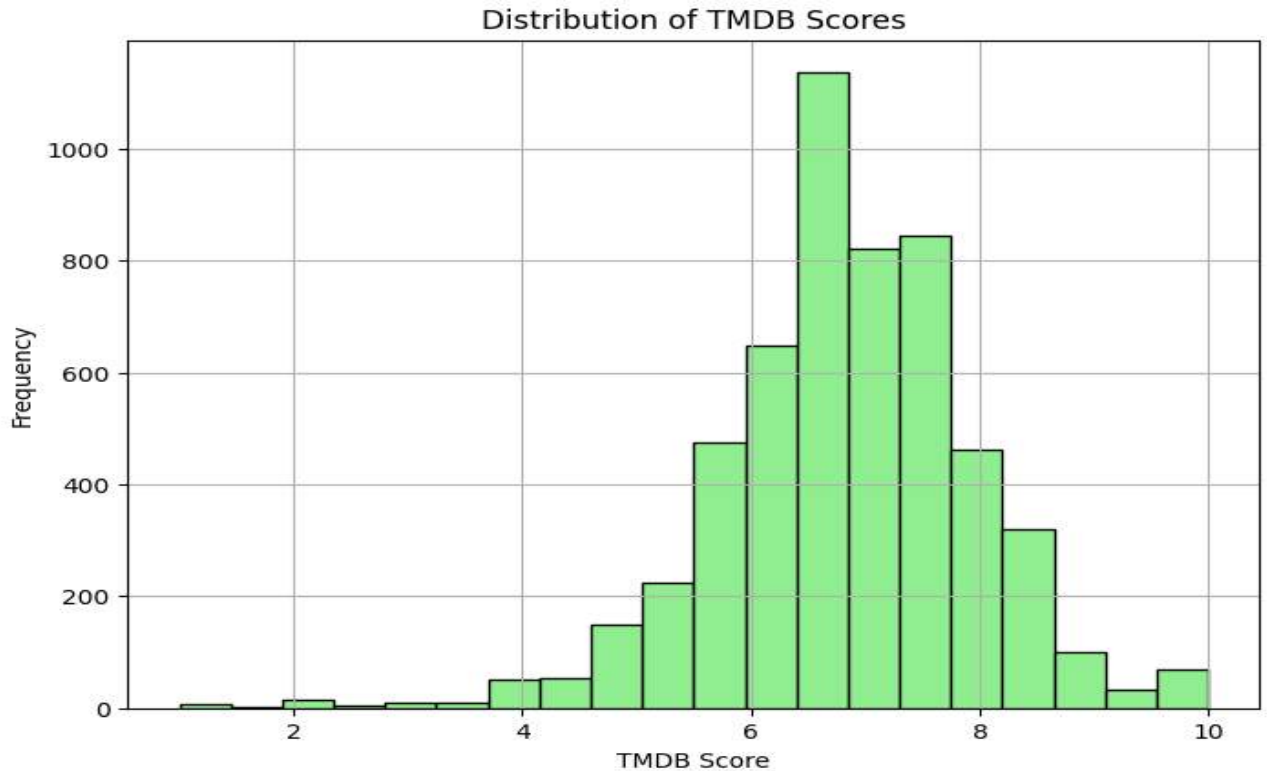
```
IMDb Score Summary Statistics:
count    5429.000000
mean        6.505648
std         1.149268
min         1.500000
25%         5.800000
50%         6.600000
75%         7.300000
max         9.000000
Name: imdb_score, dtype: float64
```



2.TMDB Score Analysis

```
[ ]  tmdb_score_stats = df['tmdb_score'].describe()
     print("\nTMDB Score Summary Statistics:")
     print(tmdb_score_stats)

     plt.figure(figsize=(8, 6))
     plt.hist(df['tmdb_score'], bins=20, color='lightgreen', edgecolor='black')
     plt.xlabel('TMDB Score')
     plt.ylabel('Frequency')
     plt.title('Distribution of TMDB Scores')
     plt.grid(True)
     plt.show()
```

```
TMDB Score Summary Statistics:
count    5429.000000
mean        6.820413
std         1.120670
min         1.000000
25%         6.200000
50%         6.822991
75%         7.500000
max        10.000000
Name: tmdb_score, dtype: float64
```

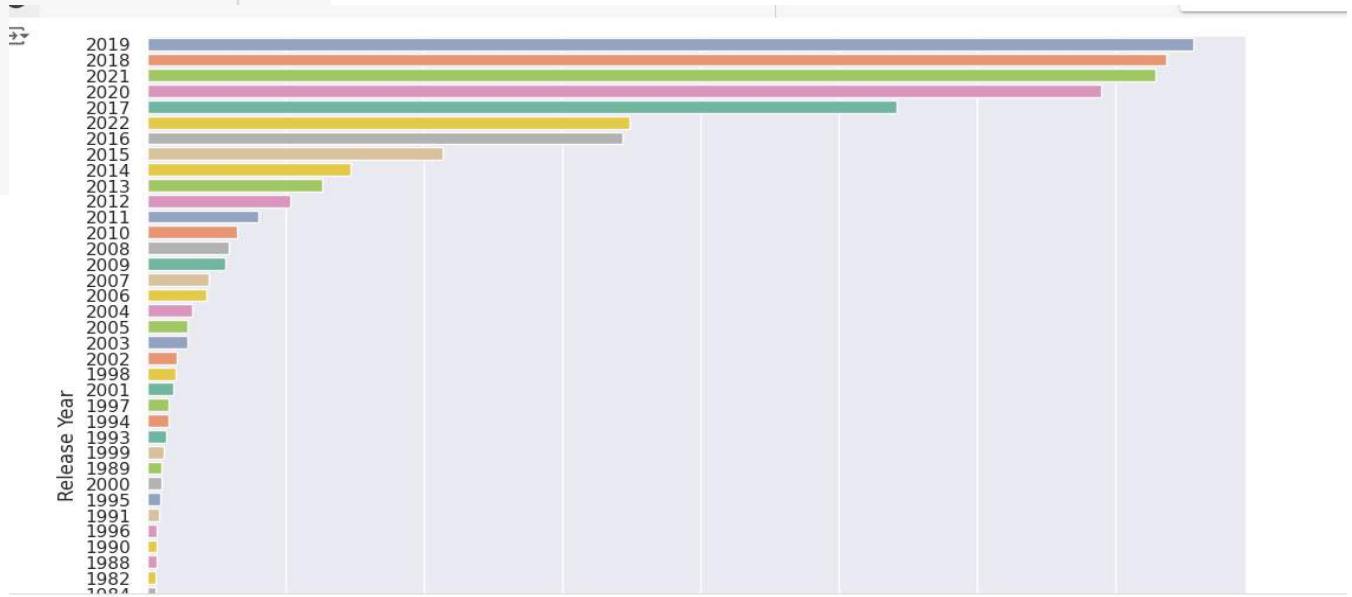


Yearwise analysis

```
[ ] num_years = df['release_year'].nunique()

    print("Number of unique years in the dataset:", num_years)
```

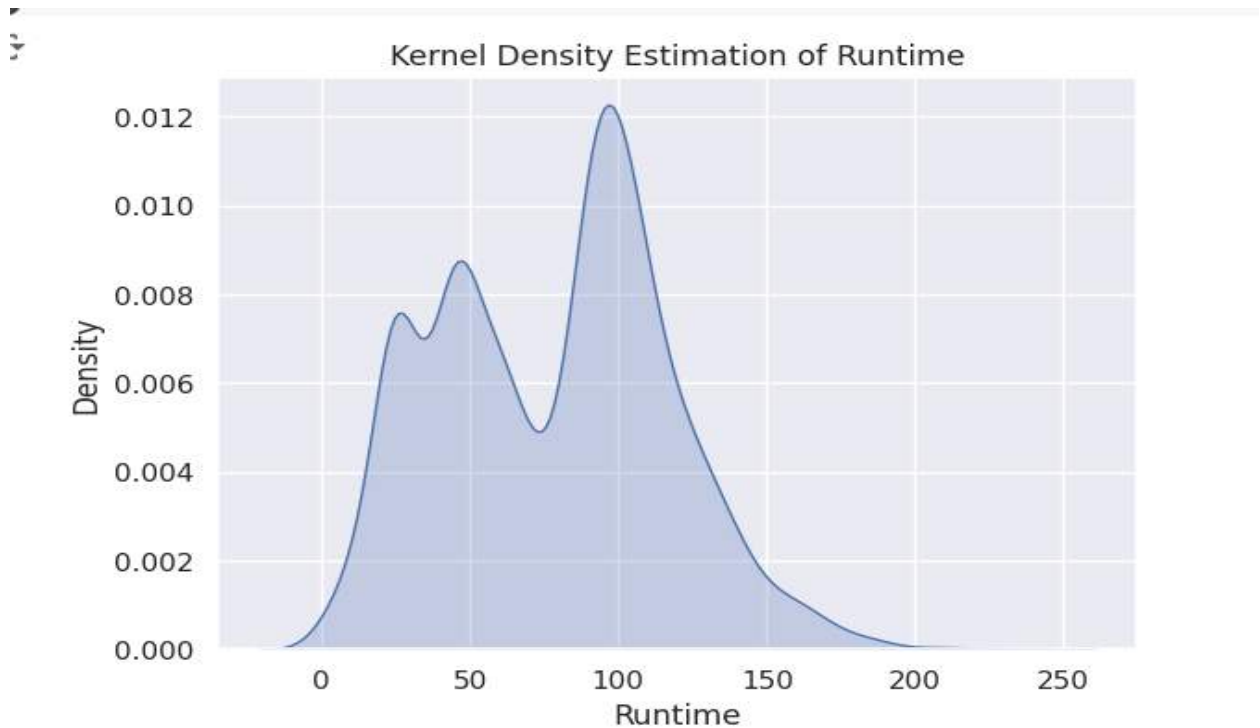
```
Number of unique years in the dataset: 62
```

```
[ ] plt.figure(figsize=(12, 10))
    sns.set(style="darkgrid")
    ax = sns.countplot(y="release_year", data=df, hue="release_year", palette="Set2",
                       order=df['release_year'].value_counts().index[0:52], legend=False)
    ax.set_ylabel('Release Year')
    ax.set_xlabel('Count')
    plt.show()
```



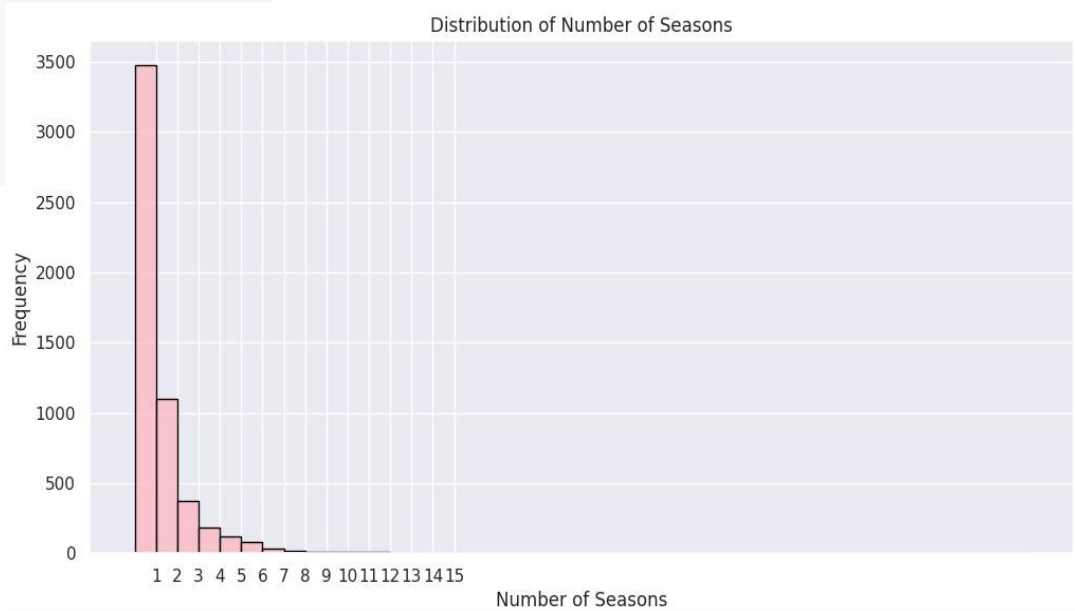
Analysis of duration of movies

```
sns.set(style="darkgrid")
sns.kdeplot(data=df['runtime'], fill=True)
plt.xlabel('Runtime')
plt.ylabel('Density')
plt.title('Kernel Density Estimation of Runtime')
plt.show()
```



Distribution of number of seasons

```
# Set the style of the plot
sns.set(style="darkgrid")

min_season = int(df['seasons'].min())
max_season = int(df['seasons'].max())

# Determine the maximum value to show on the x-axis
max_ticks = min(15, max_season)

# Plot a histogram of the number of seasons with binwidth=1
plt.figure(figsize=(12, 6))
sns.histplot(data=df, x='seasons', bins=range(min_season, max_season + 1), kde=False, color='lightpink', edgecolor='black')
plt.xlabel('Number of Seasons')
plt.ylabel('Frequency')
plt.title('Distribution of Number of Seasons')

# Set x-axis ticks to show integers from 1 to the maximum number of seasons (up to 15)
plt.xticks(range(1, max_ticks + 1))

plt.show()
```



Predictive analysis

1. **Recommendation using Term Frequency and Inverse Document Frequency**

- **Goal**: To suggest movies to users based on their past behavior and preferences.
- **Technique Used**: TF-IDF (Term Frequency-Inverse Document Frequency)
- **How It Works**:
  - **Step 1**: Calculate the importance of each word in a movie description across all movies.
  - **Step 2**: Use cosine similarity to find movies with similar descriptions.
- **Output**: List of recommended movies tailored to the user's past choices.

TF-IDF in Recommendation System

•**Term Frequency (TF)**: Measures how frequently a term appears in a document.

•**Inverse Document Frequency (IDF)**: Measures how important a term is by considering the number of documents it appears in.

•**Combining TF and IDF**: Yields a score that reflects the importance of a term within a specific document in the context of the entire dataset.

•**Application**: Used to represent movie descriptions in a numerical format, facilitating similarity calculations.

Output

Example 1 :

```
Recommendations for 'Titanic':
8                      The Blue Lagoon
13                       Cairo Station
16                   Alexandria… Why?
20                 Alibaba Aur 40 Chor
21                     The Blazing Sun
22                          Dark Waters
26                   Beirut, Oh Beirut
27    The Return of the Prodigal Son
29                           Manoranjan
30                                Ujala
Name: title, dtype: object
```

Example 2 :

```
Recommendations for 'Peaky Blinders':
3                        The Dirty Dozen
4         Monty Python's Flying Circus
5                          Life of Brian
9                  The Guns of Navarone
11    Richard Pryor: Live in Concert
13                       Cairo Station
16                   Alexandria… Why?
17                               The Land
27    The Return of the Prodigal Son
28                            Khoon Khoon
Name: title, dtype: object
```

Predictive analysis

Predictive model using Random Forest Classifier

- **Objective**: To predict whether a movie will be successful or not.
- **Model Used**: Random Forest Classifier

- **How It Works**:
  - **Data Preparation**: Collect features to define what constitutes as a hit movie
  - **Training**: Training the model on historical data (Movies with IMDb score over 7.0, TMDb score over 7.0 and TMDb popularity > 50 are considered as successful movies)
  - **Prediction**: Use the trained model to predict the success of new movies

- **Key Features Considered**: IMDb score, TMDb score, IMDb votes, TMDb popularity

Predictive analysis

Random Forest Classifier Accuracy

- The accuracy of our Random Forest Classifier model is **99.82%**.

```
Accuracy: 0.9981583793738489
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      1061
           1       1.00      0.92      0.96        25

    accuracy                           1.00      1086
   macro avg       1.00      0.96      0.98      1086
weighted avg       1.00      1.00      1.00      1086
```

Output

Example 1 :

```
new_movie_name = 'The Door'

new_movie_data = pd.DataFrame({
    'imdb_score': [9.5],
    'imdb_votes': [1000],
    'tmdb_score': [7.],
    'tmdb_popularity': [80.0]
})
```

```
Movie 'The Door' is not predicted to be a hit.
```

Example 2 :

```
new_movie_name = 'The Call'

new_movie_data = pd.DataFrame({
    'imdb_score': [8.2],
    'imdb_votes': [100000],
    'tmdb_score': [7.6],
    'tmdb_popularity': [142.5]
})
```

```
Movie 'The Call' is predicted to be a hit!
```

Deployment

Technologies Used

1. Python
- High-level, easy-to-read programming language.
- Developed the backend logic, handling data processing and integration.

2. Flask
- Lightweight web framework for Python.
- Managed routing, template rendering, and HTTP request handling.

3. Bootstrap
- Front-end framework for responsive web design.
- Created a clean, responsive layout with pre-designed components.

Deployment

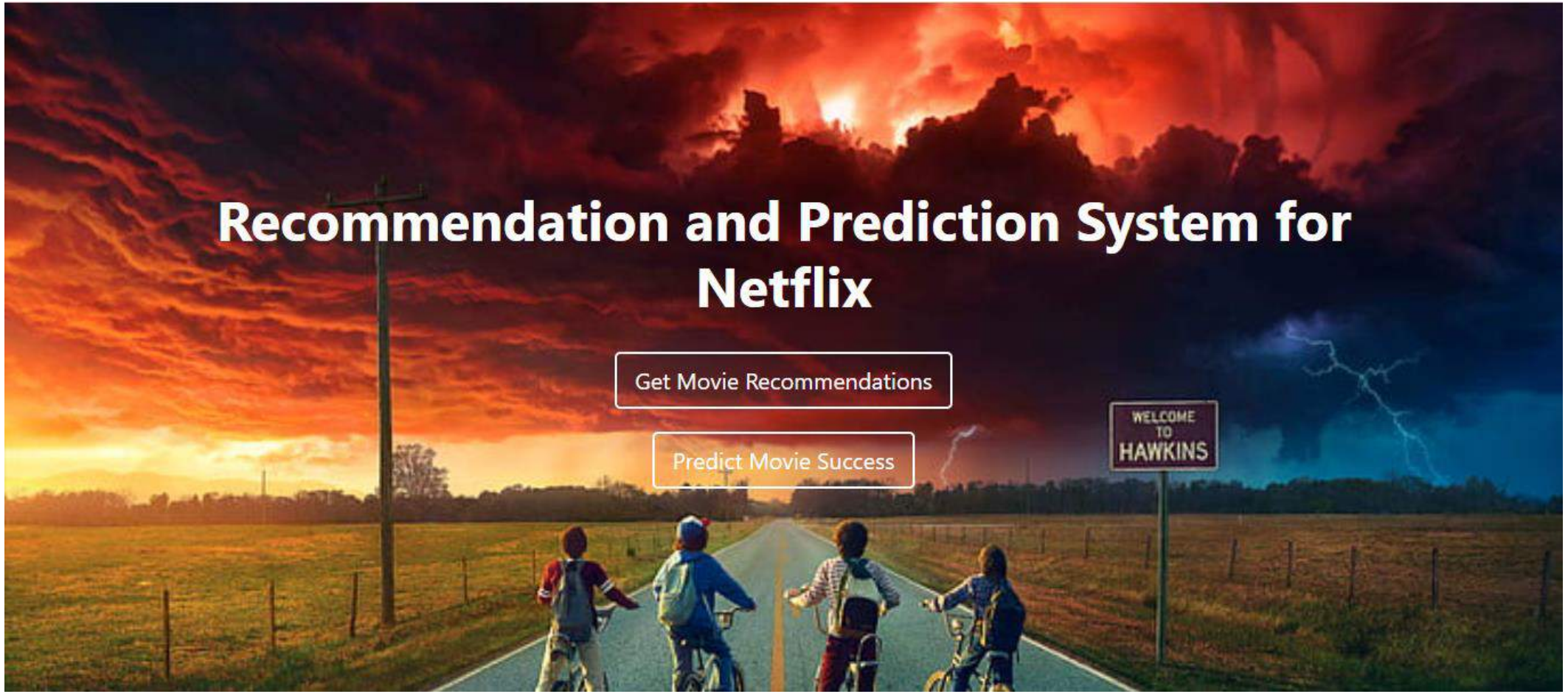
4. CSS

- Stylesheet language for visual presentation.
- Customized styles for background images, form transparency, and buttons

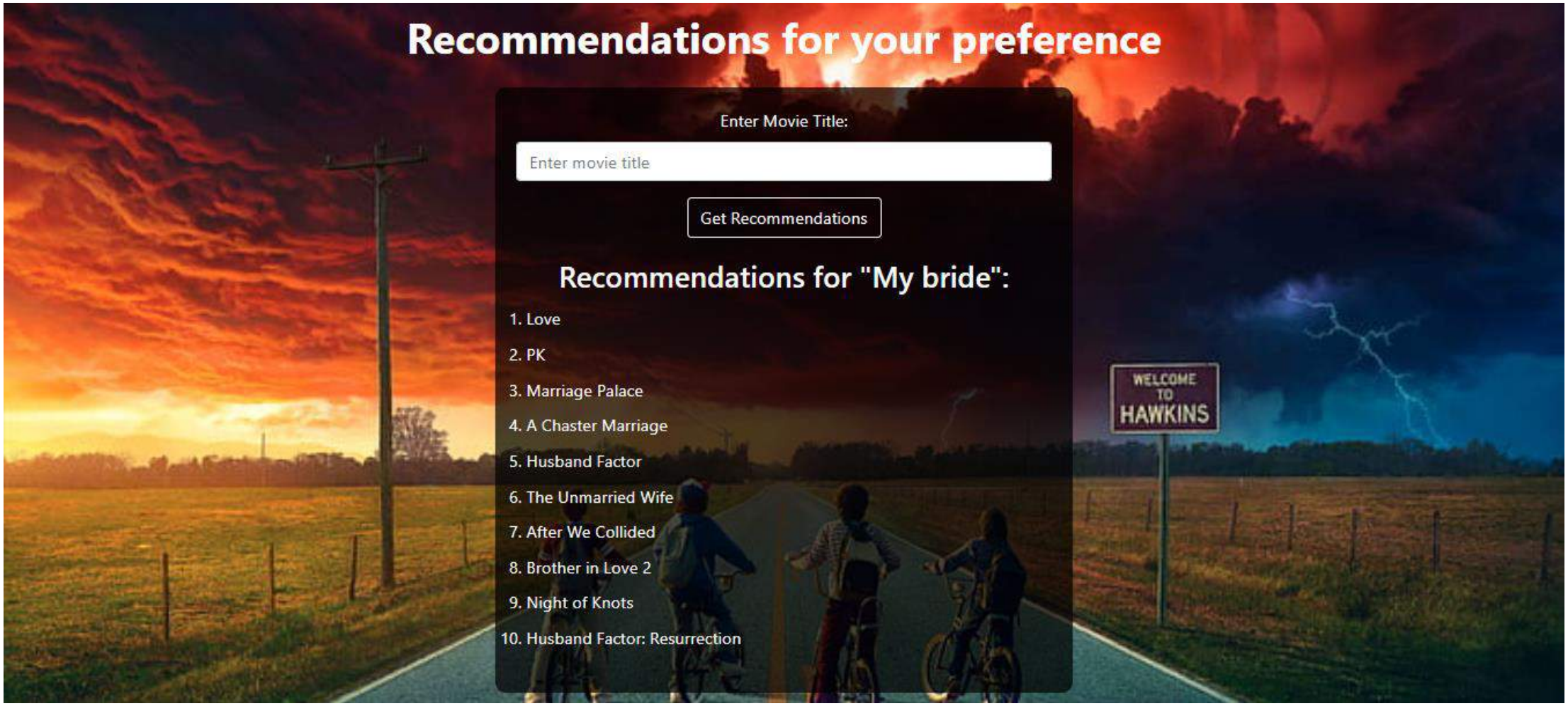
Integration Workflow

- **User Interaction**: User requests a page.
- **Flask Routing**: Routes request to Python function.
- **Data Processing**: Python handles data and computations.
- **Template Rendering**: Flask renders HTML with dynamic content.
- **Bootstrap & CSS**: Applies responsive layout and custom styles.
- **Response**: Styled HTML is sent back to the user’s browser.

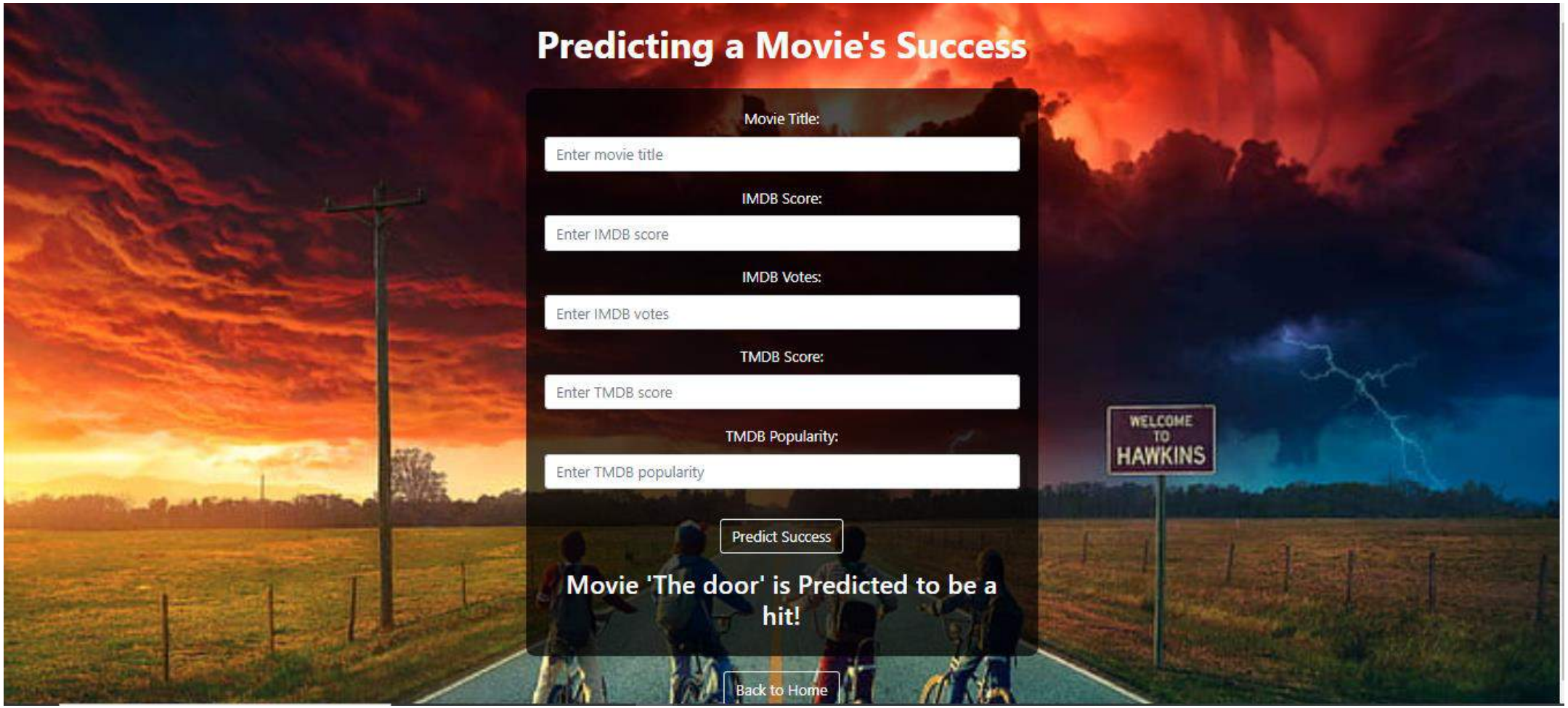
Home page


Recommendation and Prediction System for Netflix
Get Movie Recommendations
Predict Movie Success
WELCOME TO HAWKINS

Recommendations for user preference



Prediction for new movie's performance



Thank you


Safari File Edit View History Bookmarks Develop Window Help
NETFLIX
Browse
Kids
Netflix Originals
Stranger Things
98% Match 2017 TV-14 2 Seasons
A lost boy. A government lab steeped in terrible secrets. And a dark force that turns a town upside down.
MINDHUNTER
BLACK MIRROR
BEYOND STRANGER THINGS
OZARK
RIVERDALE
ORANGE IS THE BLACK
DYNASTY
SAVING CAPITALISM
CROWN