**Netflix Data: Cleaning, Analysis and Visualization**

|  |  |
| --- | --- |
| Project Title | Netflix Data: Cleaning, Analysis and Visualization |
| Tools | Jupyter Notebook |
| Domain | Data Analyst & Data scientist |
| Project Difficulties level | intermediate |

**1.Dataset Description:**

The dataset contains information about Netflix shows and movies, with 8,790 records and 10 attributes. Below is a detailed description of each column:

1. show\_id – A unique identifier for each show or movie.
2. type – Indicates whether the entry is a "Movie" or "TV Show."
3. title – The name of the movie or TV show.
4. director – The director(s) of the show or movie.
5. country – The country where the content was produced.
6. date\_added – The date when the content was added to Netflix.
7. release\_year – The year the show or movie was originally released.
8. rating – The content rating (e.g., PG-13, TV-MA, TV-PG).
9. duration – Duration of the movie (in minutes) or number of seasons for TV shows.
10. listed\_in – The genres or categories associated with the content.

**2.Problem Statement:**

With the rise of digital streaming platforms, Netflix has become a global leader in the entertainment industry. However, understanding content trends, user preferences, and catalog composition is essential for making data-driven decisions.

This project aims to analyze the Netflix dataset to uncover insights regarding:

* The distribution of movies and TV shows by country, genre, and year.
* Trends in content addition over the years.
* The impact of content ratings on the availability of different types of shows.
* Possible factors influencing content popularity and duration.
* Predictive modeling to classify and recommend content based on historical data.

By leveraging data preprocessing, exploratory data analysis (EDA), and machine learning techniques, this analysis provides valuable insights into Netflix's content strategy and helps in making informed business decisions.

**3. Data Preprocessing:**

* Missing Value Analysis: Checked for missing values in the dataset.
* Duplicate Removal: Identified and removed any duplicate rows.
* Data Type Conversion: Converted the date\_added column to datetime format.
* Feature Engineering:
  + Extracted year\_added, month\_added, and day\_added from date\_added to analyze trends.
  + Applied Label Encoding for categorical variables when required.

**4. Exploratory Data Analysis (EDA):**

* Content Distribution:
  + Analyzed the count of Movies vs. TV Shows on Netflix.
  + Identified the top countries producing Netflix content.
  + Examined the most common genres using the listed\_in column.
* Trend Analysis:
  + Visualized the number of content additions over the years.
  + Analyzed the distribution of content based on release\_year.
* Content Ratings:
  + Examined the most frequent content ratings (e.g., TV-MA, PG-13, TV-PG).
* Duration Analysis:
  + Compared the duration of movies in minutes.
  + Examined the number of seasons for TV shows.

**5. Machine Learning Model:**

* Objective: Classify Netflix content into categories using a Random Forest Classifier.
* Data Preparation:
  + Selected relevant features for classification.
  + Standardized numerical features using StandardScaler.
  + Encoded categorical variables with LabelEncoder.
* **Model Training & Evaluation:**
* Utilized **Random Forest Classifier** with **GridSearchCV** for hyperparameter tuning.
* Tuned parameters: n\_estimators, max\_depth, min\_samples\_split, and min\_samples\_leaf.
* Applied **5-fold cross-validation** to enhance model performance and prevent overfitting.
* Selected the **best estimator** and trained the model on the dataset.
* Evaluated the model using **accuracy score (70.3%)** and a **classification report**, highlighting precision, recall, and F1-score.

**6.Conclusion:**

This project analyzed the Netflix dataset to understand trends in content distribution, ratings, and availability by country. Here are the key findings:

* Content Distribution: Netflix has more movies than TV shows, with about 70% of the content being movies and 30% being TV shows.
* Ratings and Trends: Most content is rated "TV-MA" and "TV-14," indicating a focus on mature audiences. The number of new releases has increased over the years, with certain months seeing more additions.
* Country-Wise Analysis: The U.S. has the largest content library, followed by India and the U.K. Netflix is popular in English-speaking countries but is also growing globally.
* Machine Learning Model: A Random Forest Classifier was trained to predict whether a piece of content is a movie or a TV show, using features like release year, month added, and rating. The model performed well after tuning.

**Code and Results:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split, cross\_val\_score

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

# Load dataset

file\_path = 'netflix.csv'

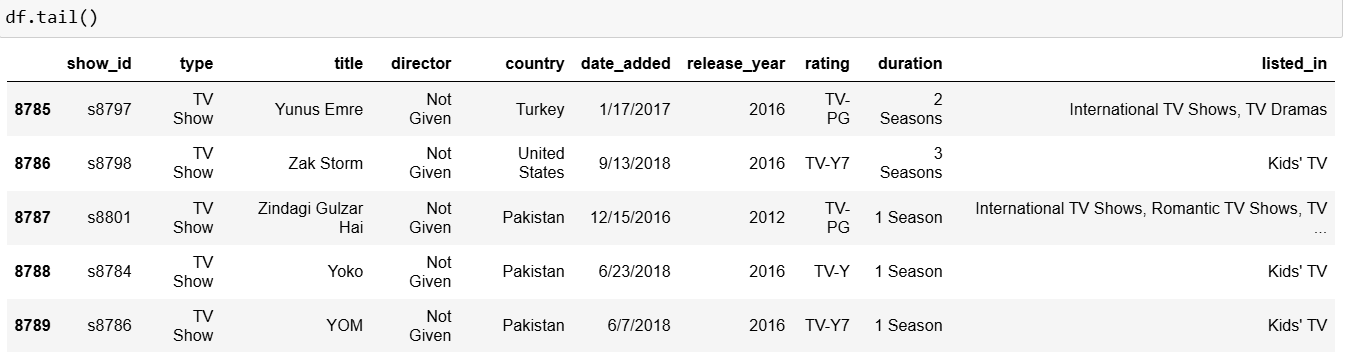
df = pd.read\_csv(file\_path)

# Display basic info about dataset

df.head()# top 5 rows

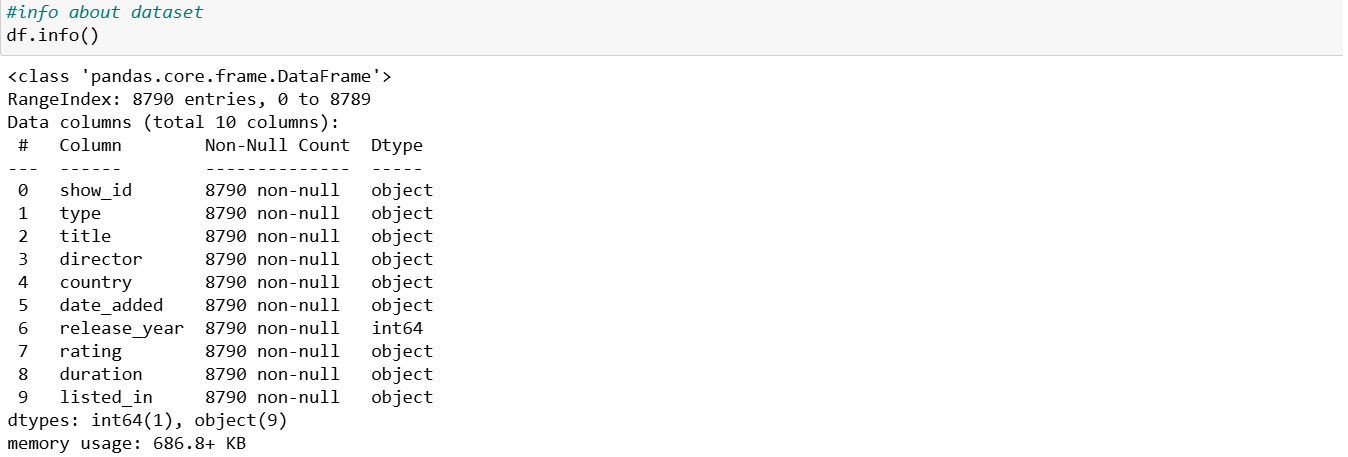


df.tail()



#info about dataset

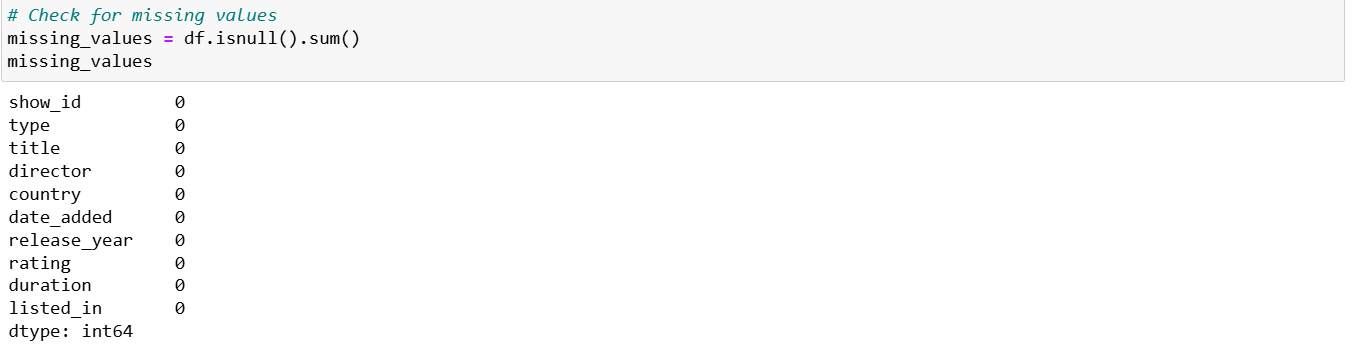
df.info()



# Check for missing values

missing\_values = df.isnull().sum()

missing\_values



# Check for duplicate rows

duplicates = df.duplicated().sum()

print(f"Number of duplicate rows: {duplicates}")

Number of duplicate rows: 0

# Convert 'date\_added' to datetime

df['date\_added'] = pd.to\_datetime(df['date\_added'])

df



# Extract year, month, and day from 'date\_added'

df['year\_added'] = df['date\_added'].dt.year

df['month\_added'] = df['date\_added'].dt.month

df['day\_added'] = df['date\_added'].dt.day

# Visualization - Distribution of Content Type

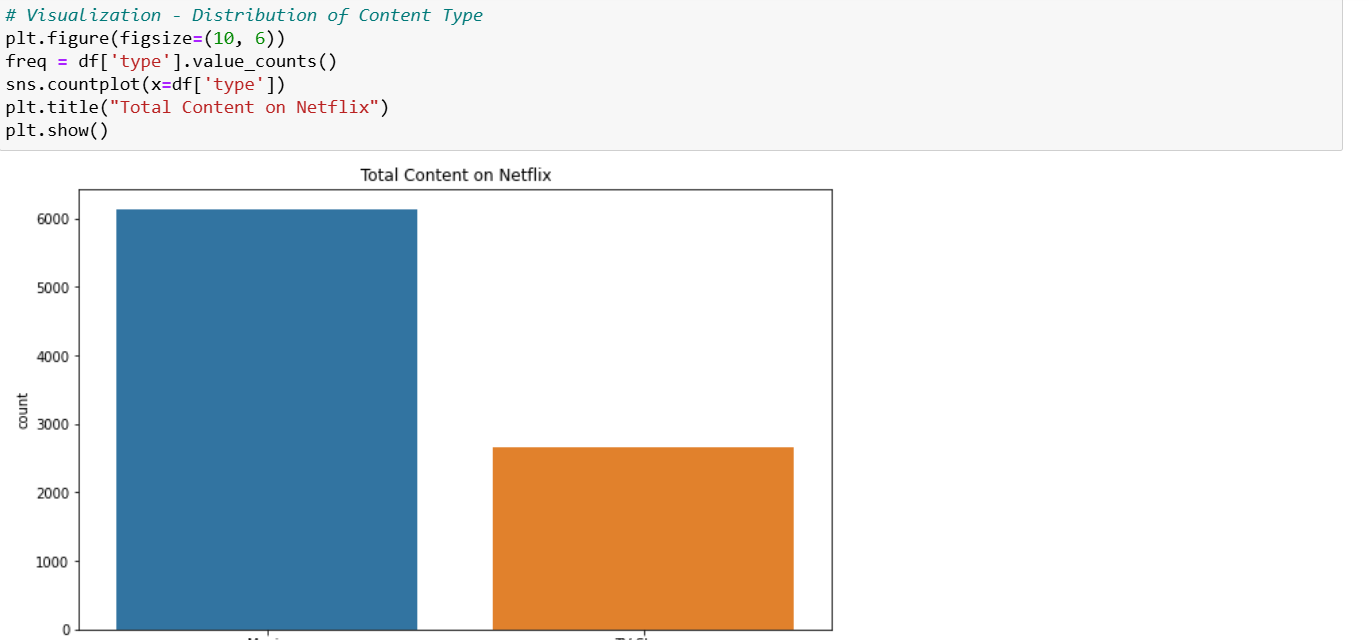
plt.figure(figsize=(10, 6))

freq = df['type'].value\_counts()

sns.countplot(x=df['type'])

plt.title("Total Content on Netflix")

plt.show()

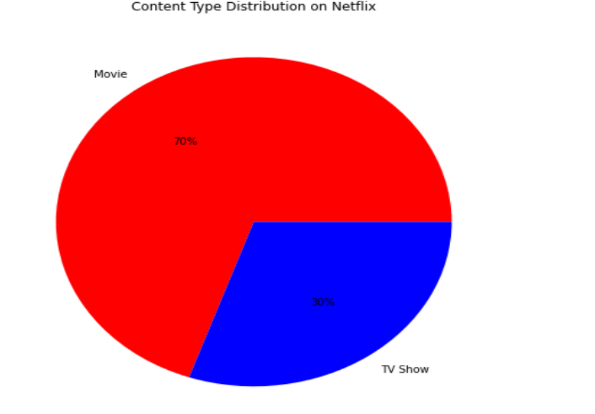


plt.figure(figsize=(8, 8))

plt.pie(freq, labels=freq.index, autopct='%.0f%%', colors=['red', 'blue'])

plt.title("Content Type Distribution on Netflix")

plt.show()



# Visualization - Rating Distribution

ratings = df['rating'].value\_counts().reset\_index()

plt.figure(figsize=(12, 6))

plt.bar(ratings['rating'], ratings['count'], color='skyblue')

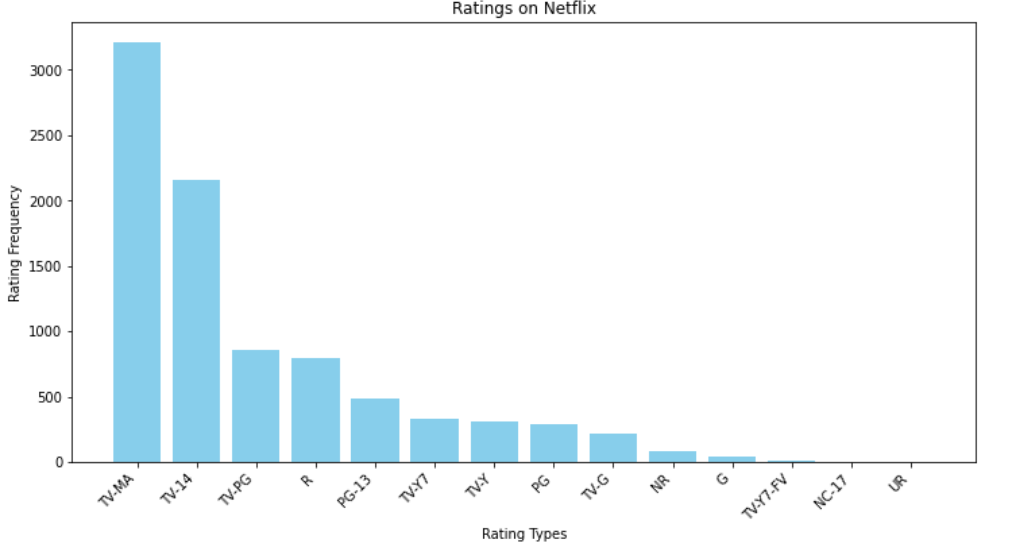
plt.xticks(rotation=45, ha='right')

plt.xlabel("Rating Types")

plt.ylabel("Rating Frequency")

plt.title("Ratings on Netflix")

plt.show()

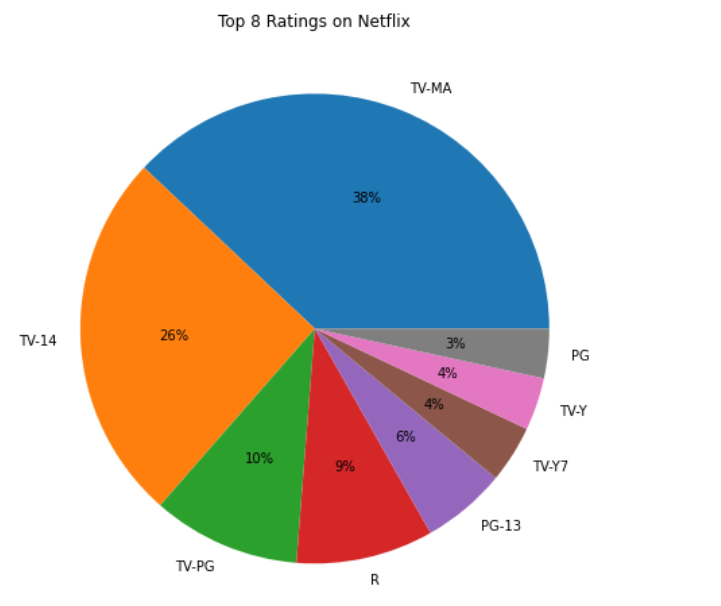


plt.figure(figsize=(8, 8))

plt.pie(ratings['count'][:8], labels=ratings['rating'][:8], autopct='%.0f%%')

plt.title("Top 8 Ratings on Netflix")

plt.show()



# Visualization - Monthly and Yearly Releases

monthly\_releases = df['month\_added'].value\_counts().sort\_index()

yearly\_releases = df['year\_added'].value\_counts().sort\_index()

plt.figure(figsize=(12, 6))

plt.plot(monthly\_releases.index, monthly\_releases.values, marker='o', label='Monthly Releases')

plt.xlabel("Month")

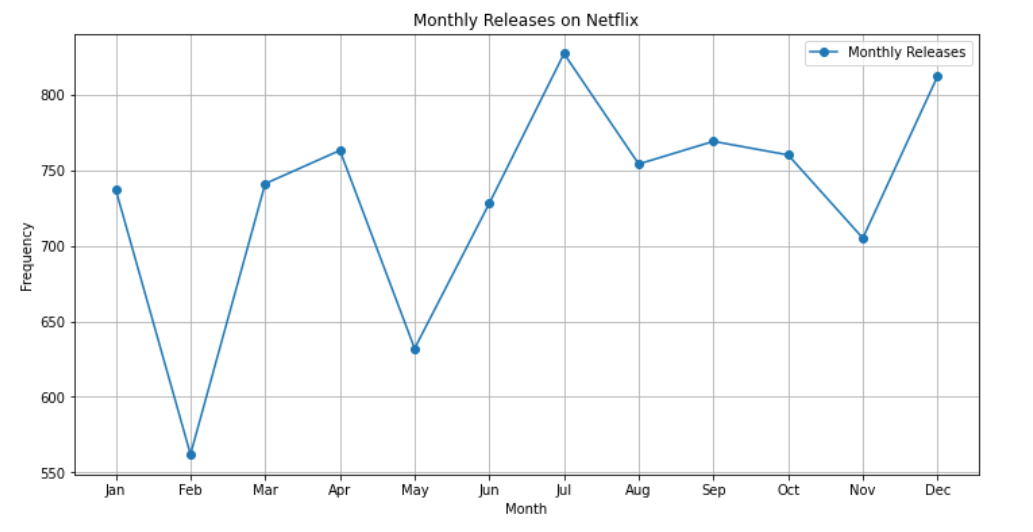
plt.ylabel("Frequency")

plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])

plt.title("Monthly Releases on Netflix")

plt.legend()

plt.grid()



plt.figure(figsize=(12, 6))

plt.plot(yearly\_releases.index, yearly\_releases.values, marker='o', label='Yearly Releases', color='green')

plt.xlabel("Year")

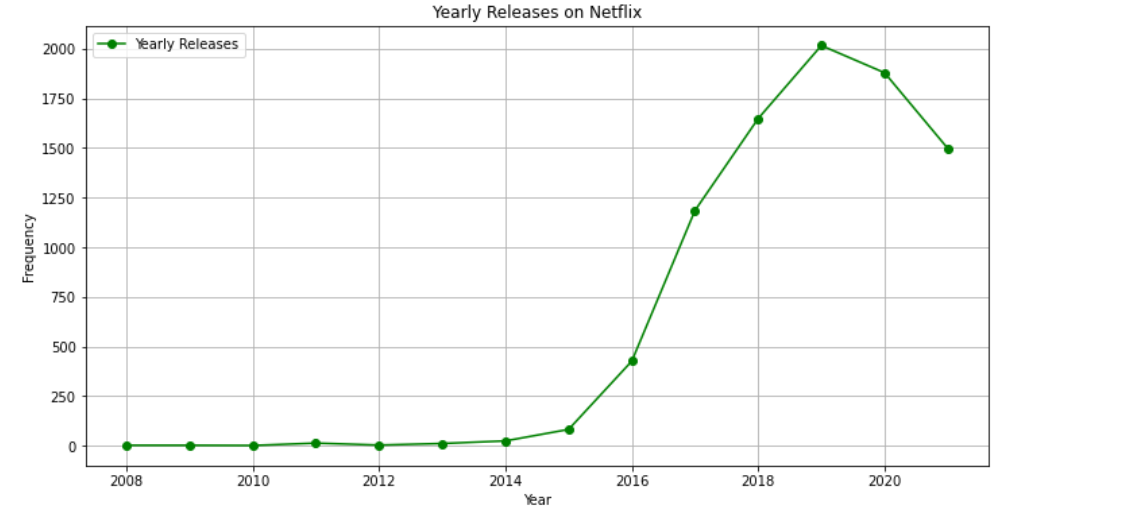
plt.ylabel("Frequency")

plt.title("Yearly Releases on Netflix")

plt.legend()

plt.grid()

plt.show()



# Visualization - Top 10 Countries with Most Content

top\_countries = df['country'].value\_counts().head(10)

plt.figure(figsize=(12, 6))

plt.bar(top\_countries.index, top\_countries.values, color='purple')

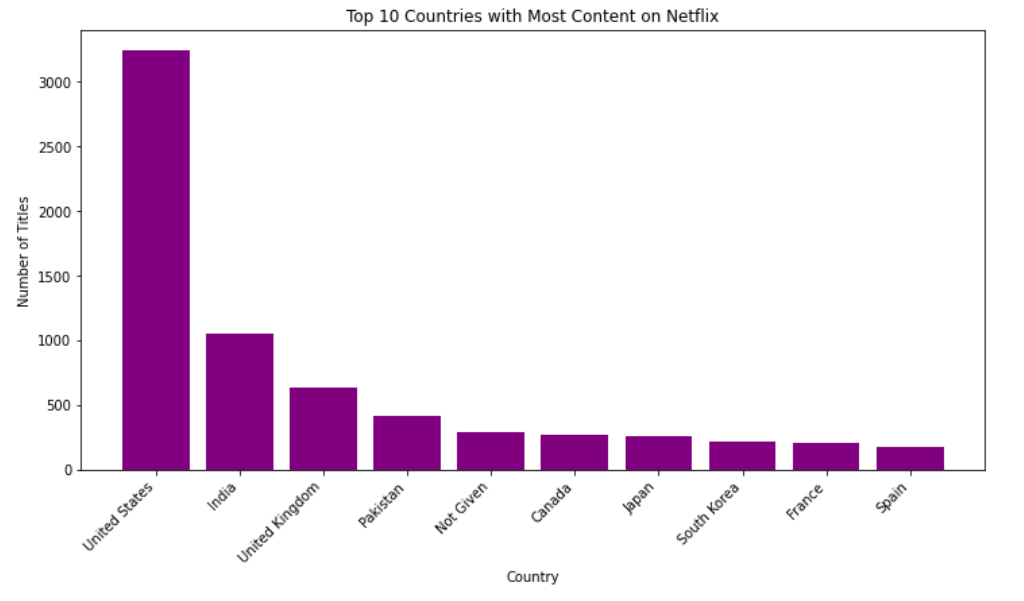
plt.xticks(rotation=45, ha='right')

plt.xlabel("Country")

plt.ylabel("Number of Titles")

plt.title("Top 10 Countries with Most Content on Netflix")

plt.show()



# Encode categorical variables

label\_enc = LabelEncoder()

df['type'] = label\_enc.fit\_transform(df['type'])

df['rating'] = label\_enc.fit\_transform(df['rating'])

df = df[['type', 'year\_added', 'month\_added', 'rating']].dropna()

# Split data into training and testing sets

X = df.drop(columns=['type'])

y = df['type']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Feature Scaling

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Hyperparameter tuning for RandomForest

param\_grid = {

'n\_estimators': [50, 100, 200],

'max\_depth': [None, 10, 20],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

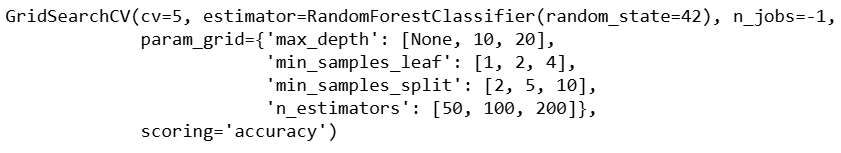
}

from sklearn.model\_selection import GridSearchCV

rf = RandomForestClassifier(random\_state=42)

grid\_search = GridSearchCV(rf, param\_grid, cv=5, scoring='accuracy', n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)



# Train best model

best\_rf = grid\_search.best\_estimator\_

y\_pred = best\_rf.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

print(f'Accuracy: {accuracy}\n')

print('Classification Report:\n', report)

