**Netflix Data Analysis**

**Objective:** Analyze Netflix data to find trends in content distribution, viewership, and genre popularity.

**📌 Step 1: Define the Problem Statement**

Decide on key insights you want to extract from the Netflix dataset. Some possible questions:  
✅ What are the most popular genres over time?  
✅ How do IMDb ratings compare to Netflix ratings?  
✅ Which countries have the highest Netflix content?  
✅ What is the trend of movie/TV show releases over the years?

For this guide, we will focus on **Netflix Content Analysis (Genres, Countries, Trends).**

**📌 Step 2: Collect a Dataset**

You need a dataset with details about movies and TV shows available on Netflix. Good sources:

* [**Netflix Movies & TV Shows (Kaggle)**](https://www.kaggle.com/datasets/shivamb/netflix-shows)

💡 **Dataset Features (Example)**:

| **Column Name** | **Description** |
| --- | --- |
| Show\_ID | Unique ID for each show/movie |
| Type | Movie or TV Show |
| Title | Name of the content |
| Director | Director of the show/movie |
| Cast | Main actors |
| Country | Country of production |
| Date Added | Date when added to Netflix |
| Release Year | Year of release |
| Rating | Age rating (e.g., PG-13, R) |
| Duration | Length of movie/TV show |
| Genre | Main genre |
| Description | Short description |

**📌 Step 3: Data Cleaning & Preprocessing**

**🔹 Load Data in Python**

python

CopyEdit

import pandas as pd

# Load the dataset

df = pd.read\_csv('netflix\_titles.csv')

# Check basic information

df.info()

df.head()

**🔹 Handle Missing Values**

python

CopyEdit

# Check for missing values

print(df.isnull().sum())

# Fill missing values (example: replacing NaN with 'Unknown')

df.fillna('Unknown', inplace=True)

**🔹 Convert Date Columns**

python

CopyEdit

# Convert 'date\_added' to datetime format

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

# Extract Year and Month

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

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

**🔹 Standardize Text Data**

python

CopyEdit

# Convert text columns to lowercase

df['title'] = df['title'].str.lower()

df['genre'] = df['genre'].str.lower()

df['country'] = df['country'].str.lower()

**📌 Step 4: Exploratory Data Analysis (EDA)**

**🔹 Summary Statistics**

python

CopyEdit

print(df.describe())

**🔹 Count of Movies vs. TV Shows**

python

CopyEdit

import seaborn as sns

import matplotlib.pyplot as plt

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

plt.title('Distribution of Movies vs. TV Shows')

plt.show()

**🔹 Most Popular Genres**

python

CopyEdit

# Count top genres

top\_genres = df['genre'].value\_counts().head(10)

# Plot

plt.figure(figsize=(10,5))

sns.barplot(x=top\_genres.values, y=top\_genres.index, palette='coolwarm')

plt.title('Top 10 Genres on Netflix')

plt.xlabel('Count')

plt.ylabel('Genre')

plt.show()

**🔹 Country-wise Content Distribution**

python

CopyEdit

# Count content by country

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

# Plot

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

sns.barplot(x=top\_countries.index, y=top\_countries.values, palette='magma')

plt.xticks(rotation=45)

plt.title('Top 10 Countries Producing Netflix Content')

plt.xlabel('Country')

plt.ylabel('Count')

plt.show()

**🔹 Content Trend Over the Years**

python

CopyEdit

# Count content by release year

df['release\_year'].value\_counts().sort\_index().plot(kind='line', figsize=(10,5))

plt.title('Content Release Trend Over Years')

plt.xlabel('Year')

plt.ylabel('Number of Movies/Shows')

plt.show()

**📌 Step 5: Sentiment Analysis on Descriptions (Optional)**

**🔹 Install NLP Libraries**

bash

CopyEdit

pip install textblob

**🔹 Perform Sentiment Analysis**

python

CopyEdit

from textblob import TextBlob

# Function to get sentiment polarity

def get\_sentiment(text):

return TextBlob(text).sentiment.polarity

# Apply function

df['sentiment'] = df['description'].apply(get\_sentiment)

# Plot sentiment distribution

sns.histplot(df['sentiment'], bins=30, kde=True)

plt.title('Sentiment Distribution of Netflix Descriptions')

plt.show()

**📌 Step 6: Data Visualization & Dashboard**

If you want to present insights in **Power BI or Tableau**, export the cleaned data.

python

CopyEdit

df.to\_csv('cleaned\_netflix\_data.csv', index=False)

You can create:

* **Bar Charts**: Top genres and country-wise content
* **Heatmaps**: Seasonal trends in content addition
* **Pie Charts**: TV Shows vs. Movies distribution

**📌 Step 7: Build a Recommendation System (Optional)**

If you want to suggest movies based on genres or ratings, you can use **TF-IDF (Text Similarity).**

**🔹 Install NLP Libraries**

bash

CopyEdit

pip install sklearn

**🔹 Build a Simple Content-Based Recommender**

python

CopyEdit

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

# TF-IDF Vectorizer

tfidf = TfidfVectorizer(stop\_words='english')

tfidf\_matrix = tfidf.fit\_transform(df['description'])

# Compute similarity

cosine\_sim = cosine\_similarity(tfidf\_matrix, tfidf\_matrix)

# Function to recommend movies

def recommend\_movies(title, df, cosine\_sim):

idx = df[df['title'] == title.lower()].index[0]

sim\_scores = list(enumerate(cosine\_sim[idx]))

sim\_scores = sorted(sim\_scores, key=lambda x: x[1], reverse=True)

sim\_scores = sim\_scores[1:6]

movie\_indices = [i[0] for i in sim\_scores]

return df['title'].iloc[movie\_indices]

# Example Usage

print(recommend\_movies('Stranger Things', df, cosine\_sim))

**📌 Step 8: Deploy Model as a Web App (Optional)**

**🔹 Flask API for Recommendations**

python

CopyEdit

from flask import Flask, request, jsonify

app = Flask(\_\_name\_\_)

@app.route('/recommend', methods=['POST'])

def recommend():

data = request.get\_json()

recommendations = recommend\_movies(data['title'], df, cosine\_sim)

return jsonify({'Recommendations': list(recommendations)})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**📌 Step 9: Summary of Requirements**

**📂 Tools Required:**

✅ Python (pandas, sklearn, seaborn, matplotlib)  
✅ Power BI / Tableau (for dashboards)  
✅ Flask / Streamlit (if deployment is needed)