**#Upload the Dataset**

from google.colab import files

uploaded = files.upload()# Upload the Dataset

**#load the dataset**

import pandas as pd

# Try reading the CSV file with a different encoding

try:

df = pd.read\_csv('netflix\_titles.csv', encoding='utf-8')

except UnicodeDecodeError:

print("UTF-8 decoding failed, trying latin-1...")

try:

df = pd.read\_csv('netflix\_titles.csv', encoding='latin-1')

except UnicodeDecodeError:

print("Latin-1 decoding failed, trying cp1252...")

try:

df = pd.read\_csv('netflix\_titles.csv', encoding='cp1252')

except UnicodeDecodeError:

print("Could not decode the file with utf-8, latin-1, or cp1252. Please check the file encoding.")

df = None # Set df to None if decoding fails

if df is not None:

print("File read successfully!")

display(df.head())

**#data exploration**

df.info()

df.describe(include='all')

df.columns

**# Check for Missing Values and Duplicates**

# Missing values

df.isnull().sum()

# Duplicates

df.duplicated().sum()

**# Visualize a Few Features**

import seaborn as sns

import matplotlib.pyplot as plt

# Example: Show count of content types

sns.countplot(data=df, x='type')

plt.title('Count of Content Types')

plt.show()

# Example: Show distribution of release years

sns.histplot(df['release\_year'], kde=True)

plt.title('Distribution of Release Year')

plt.show()

**#Convert Categorical Columns to Numerical (Label Encoding)**

from sklearn.preprocessing import LabelEncoder

df\_encoded = df.copy()

label\_encoders = {}

for column in df\_encoded.select\_dtypes(include='object').columns:

le = LabelEncoder()

df\_encoded[column] = df\_encoded[column].astype(str)

df\_encoded[column] = le.fit\_transform(df\_encoded[column])

**#One-Hot Encoding**

df\_ohe = pd.get\_dummies(df, drop\_first=True)

df\_ohe.head()

**#Feature Scaling**

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(df\_ohe.select\_dtypes(include='number'))

df\_scaled = pd.DataFrame(scaled\_features, columns=df\_ohe.select\_dtypes(include='number').columns)

**# Train-Test Split**

from sklearn.model\_selection import train\_test\_split

X = df\_scaled

y = df['type'] # Replace with appropriate label

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

**@Model Building**

from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

**#Evaluation**

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

y\_pred = model.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

**# Make Predictions from New Input**

sample\_input = X\_test.iloc[0].values.reshape(1, -1)

prediction = model.predict(sample\_input)

prediction

**# Convert to DataFrame and Encode**

# Convert user input to DataFrame and encode it

new\_data = pd.DataFrame([sample\_input[0]], columns=X.columns)

**Final Grade (the example context)# Predict**

final\_prediction = model.predict(new\_data)

print("Predicted Class:", final\_prediction[0])

**# Deployment - Building an Interactive App**

!pip install gradio

**# Create a Prediction Function**

def predict\_type(feature\_list):

df\_input = pd.DataFrame([feature\_list], columns=X.columns)

prediction = model.predict(df\_input)

return prediction[0]**#Create the Gradio Interface**

import gradio as gr

interface = gr.Interface(fn=predict\_type,

inputs=[gr.Text(label=col) for col in X.columns], # Changed gr.inputs.Text to gr.Text

outputs="text")

interface.launch()