


1. Upload the Dataset

```
from google.colab import files
```

2. Load the Dataset

```
import pandas as pd
```

```
df = pd.read_csv('netflix_titles.csv')
df.head()
```



	show_id	type	title	director	cast	country	date_added	release_year
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel	NaN	September 24, 2021	2021

Next steps: [Generate code with df](#) [View recommended plots](#) [New interactive sheet](#)

3. Data Exploration

```
df.shape # Check number of rows and columns
df.columns # Column names
df.info() # Data types and non-null values
df.describe(include='all') # Summary of statistics
```

```

↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
#   Column                Non-Null Count  Dtype
---  -
0   show_id                8807 non-null   object
1   type                   8807 non-null   object
2   title                  8807 non-null   object
3   director               6173 non-null   object
4   cast                   7982 non-null   object
5   country                7976 non-null   object
6   date_added             8797 non-null   object
7   release_year           8807 non-null   int64
8   rating                 8803 non-null   object
9   duration               8804 non-null   object
10  listed_in              8807 non-null   object
11  description             8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB

```

	show_id	type	title	director	cast	country	date_added	release_y
count	8807	8807	8807	6173	7982	7976	8797	8807.0000
unique	8807	2	8807	4528	7692	748	1767	N
top	s8807	Movie	Zubaan	Rajiv Chilaka	David Attenborough	United States	January 1, 2020	N
freq	1	6131	1	19	19	2818	109	N
mean	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2014.1800
std	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8.8190
min	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1925.0000
25%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2013.0000
50%	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2017.0000

4. Check Missing Values and Duplicates

```

df.isnull().sum()
df.duplicated().sum()

```

```

↳ np.int64(0)

```

5. Visualize a Few Features

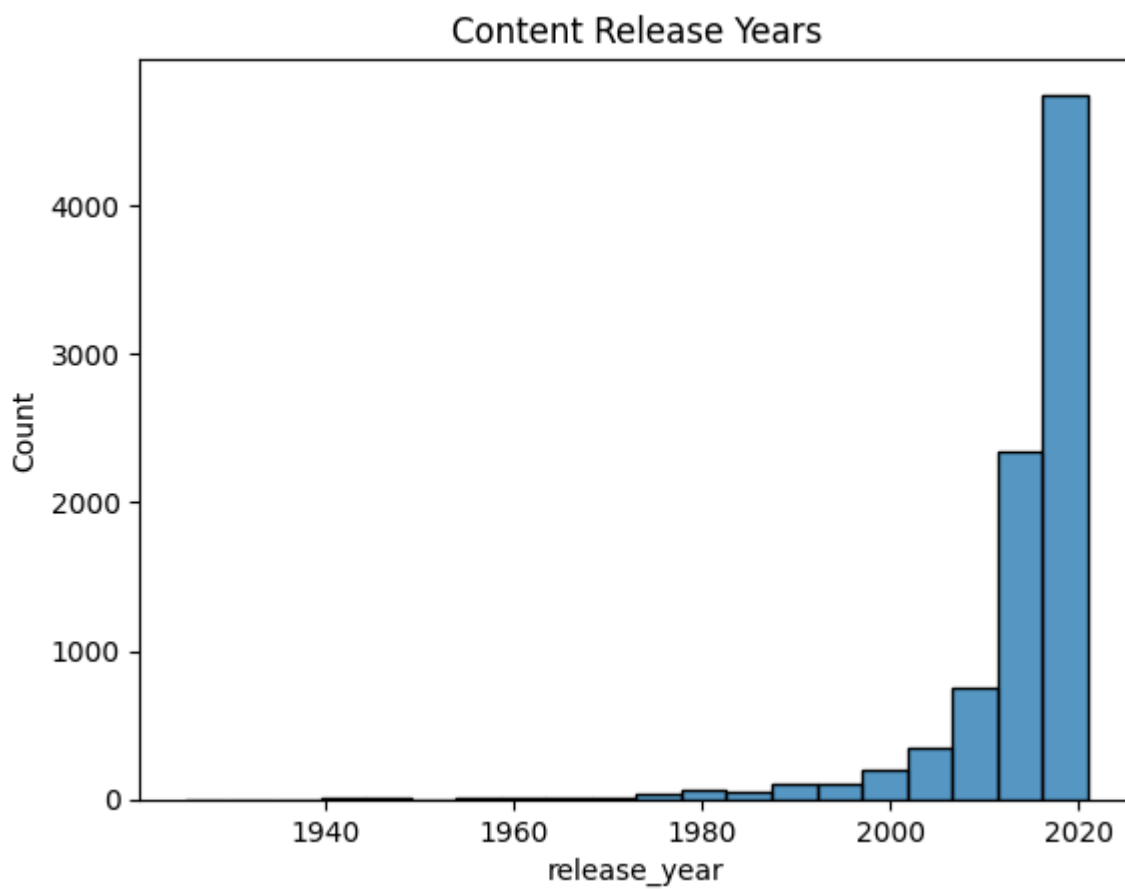
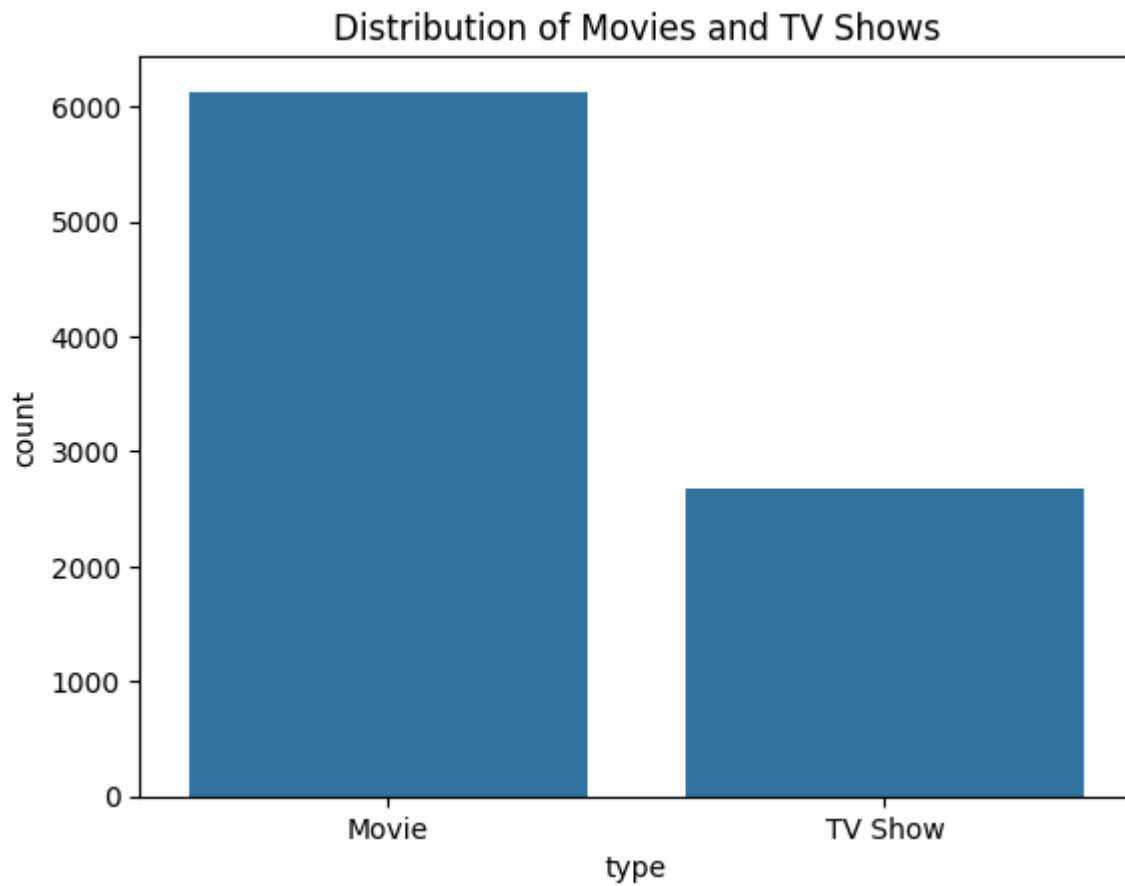
```

import seaborn as sns
import matplotlib.pyplot as plt

```

```
# Type of content
sns.countplot(x='type', data=df)
plt.title('Distribution of Movies and TV Shows')
plt.show()

# Release year distribution
sns.histplot(df['release_year'], bins=20)
plt.title('Content Release Years')
plt.show()
```



6. Identify Target and Features

```
target = 'type'  
features = ['release_year', 'rating', 'duration', 'listed_in']
```

7. Convert Categorical Columns to Numerical

```
df['duration'] = df['duration'].str.extract('(\d+)').astype(float)
df['rating'] = df['rating'].astype(str)
df['listed_in'] = df['listed_in'].astype(str)
```

8. One-Hot Encoding

```
df_encoded = pd.get_dummies(df[features], drop_first=True)
```

9. Feature Scaling

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded)
```

10. Train-Test Split

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

y = LabelEncoder().fit_transform(df[target])
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_st
```

11. Model Building

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
model.fit(X_train, y_train)
```



▼ RandomForestClassifier ⓘ ?
RandomForestClassifier()

12. Evaluation

```
from sklearn.metrics import classification_report, accuracy_score
```

```
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))
```



	precision	recall	f1-score	support
0	1.00	1.00	1.00	1214
1	1.00	1.00	1.00	548
accuracy			1.00	1762
macro avg	1.00	1.00	1.00	1762
weighted avg	1.00	1.00	1.00	1762

Accuracy: 1.0

13. Make Predictions from New Input

```
new_data = df_encoded.iloc[0:1]
new_scaled = scaler.transform(new_data)
model.predict(new_scaled)
```



array([0])

14. Convert to DataFrame and Encode

```
def preprocess_input(data):
    data = pd.get_dummies(data, drop_first=True)
    data = scaler.transform(data)
    return data
```

15. Predict the Final Grade

```
def predict_type(input_data):
    processed = preprocess_input(input_data)
    prediction = model.predict(processed)
    return "TV Show" if prediction[0] == 1 else "Movie"
```

16. Deployment - Build an Interactive App (Gradio)

```
!pip install gradio
import gradio as gr
```

```

Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-p
Collecting ruff>=0.9.3 (from gradio)
  Downloading ruff-0.11.9-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.m
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (f
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: hf-xet<2.0.0,>=1.1.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/d
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.1
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-package
Downloading gradio-5.29.1-py3-none-any.whl (54.1 MB)
_____ 54.1/54.1 MB 15.3 MB/s eta 0:00:00
Downloading gradio_client-1.10.1-py3-none-any.whl (323 kB)
_____ 323.1/323.1 kB 15.7 MB/s eta 0:00:00
Downloading aiofiles-24.1.0-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.12-py3-none-any.whl (95 kB)
_____ 95.2/95.2 kB 2.8 MB/s eta 0:00:00
Downloading groovy-0.1.2-py3-none-any.whl (14 kB)
Downloading python_multipart-0.0.20-py3-none-any.whl (24 kB)
Downloading ruff-0.11.9-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11
_____ 11.5/11.5 MB 91.4 MB/s eta 0:00:00
Downloading safehttpx-0.1.6-py3-none-any.whl (8.7 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.46.2-py3-none-any.whl (72 kB)
_____ 72.0/72.0 kB 3.8 MB/s eta 0:00:00
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)

```

17. Create the Prediction Function

```
def predict_netflix_type(release_year, rating, duration, listed_in):  
    input_df = pd.DataFrame([[release_year, rating, duration, listed_in]],  
                             columns=['release_year', 'rating', 'duration', 'listed_in'])  
    input_df = pd.get_dummies(input_df)  
    # Align with training features  
    input_df = input_df.reindex(columns=df_encoded.columns, fill_value=0)  
    scaled = scaler.transform(input_df)  
    prediction = model.predict(scaled)  
    return "TV Show" if prediction[0] == 1 else "Movie"
```

18. Create the Gradio Interface

```
interface = gr.Interface(  
    fn=predict_netflix_type,  
    inputs=[  
        gr.Number(label="Release Year"),  
        gr.Textbox(label="Rating (e.g., TV-MA, PG-13)"),  
        gr.Number(label="Duration (minutes or seasons)"),  
        gr.Textbox(label="Genre (e.g., Drama, Comedy)"),  
    ],  
    outputs="text",  
    title="Netflix Content Type Predictor"  
)  
  
interface.launch()
```