Netflix Insights and Metrics

Week 1&2: Cleaning and Normalisation

1. Dataset Overview

The dataset was loaded from **`netflix_titles.csv`** into a Pandas DataFrame. The initial shape of the dataset is (8807 rows, 12 columns). It includes 8,807 titles with 12 attributes:

- **title**
- **type** (Movie / TV Show)
- **director**
- **cast**
- **country**
- **date_added**
- **release_year**
- **rating**
- **duration**
- **listed_in (genres)**
- **description**

---

2. Data Cleaning Steps (Pandas)

2.1 Duplicate Removal

- Checked duplicates using df_read.duplicated().sum().
- Found **0 duplicates** and dropped them using:

df_read = df_read.drop_duplicates()

- The **shape of the dataset remained (8807, 12)**.

2.2 Missing Value Handling

- Identified missing values using *df_read.isnull().sum()*.
- Filled missing values in key fields with default indicators using the *.fillna()* function:

- Filled missing values in key fields:
    - `director` → *Unknown*
    - `cast` → *Not Available*

- `country` → *Unknown*
- `date_added` → *Not Available*
- `rating` → *Not Rated*
- `duration` → *Unknown*
- Dropped rows missing critical fields: `title`, `type`.

2.3 Standardization

- Trimmed whitespaces and normalized formatting:
  - `duration` cleaned (stripped, converted to Title Case).

    df_read['duration'] = df_read['duration'].str.strip().str.title()

  - `cast` standardized by stripping whitespace.

    df_read['cast'] = df_read['cast'].str.strip()

2.4 DataFrame Consolidation

- Saved the cleaned dataset as `df_clean`.
- The **final shape of the cleaned dataset (df_clean) is (8807 rows, 12 columns)**.
- Exported to CSV as **`netflix_cleaned.csv`** for downstream use with:

df_clean.to_csv("netflix_cleaned.csv", index=False)

---

3. Key Insights

All key insights were generated using the .value_counts() method in Pandas.

Content Distribution (Movies vs. TV Shows)

- Counted how many entries were **Movies** vs. **TV Shows**, showing Netflix's balance of formats using

type_distribution = df_clean['type'].value_counts()

- The analysis found: **6,131 Movies** (69.6%) and **2,676 TV Shows** (30.4%).

Top Directors

- Extracted the Top 10 directors using:

top_directors = df_clean['director'].value_counts().head(10)

- Top directors included **Rajiv Chilaka** (**19 titles**), **Raúl Campos, Jan Suter** (**18 titles**), and **Marcus Raboy** (**16 titles**).

Geographical Spread

- Extracted the Top 10 countries with most content using:

top_countries = df_clean['country'].value_counts().head(10)

- The top countries are the **United States** (**2,818 titles**), **India** (**972 titles**), and the **United Kingdom** (**419 titles**).

Ratings

- Listed the **Top 10 most common ratings**, highlighting Netflix's most frequent audience classifications using:

top_ratings = df_clean['rating'].value_counts().head(10)

- The most frequent audience classifications are **TV-MA** (**3,207 titles**), **TV-14** (**2,160 titles**), and **TV-PG** (**863 titles**).

---

4. Potential Applications

- **Content Strategy** → Use director and country-level insights to plan future acquisitions.
- **Genre & Rating Focus** → Explore dominant categories for personalized recommendations.
- **Regional Growth** → Understand high-content countries to strengthen global strategy.
- **Recommendation Systems** → Combine attributes like type, country, and rating to build content filters.

---

5. Data Normalization and Feature Engineering

To prepare the dataset for machine learning, categorical data was encoded into numerical formats, and key features were normalized. This process was performed on a copy of the cleaned DataFrame (df_clean).

5.1 Label Encoding for 'rating'

- Converted the categorical rating column into numerical labels using LabelEncoder.
- This assigns a unique integer to each rating category (e.g., TV-MA, TV-14), creating the new rating_label column.

label_encoder = LabelEncoder()

df_normalised['rating_label']=label_encoder.fit_transform(df_normalised['rating'].astype(str))

5.2 One-Hot Encoding for 'type'

- Applied OneHotEncoder to the type column to create binary features for 'Movie' and 'TV Show'.
- This avoids implying an incorrect ordinal relationship between the two categories.
- The resulting columns were merged back into the main DataFrame.

onehot_encoder = OneHotEncoder(sparse_output=False, drop=None)

type_encoded = onehot_encoder.fit_transform(df_normalised[['type']])

df_type_onehot = pd.DataFrame(type_encoded,

columns=onehot_encoder.get_feature_names_out(['type'])) df_normalised = pd.concat([df_normalised, df_type_onehot], axis=1)

5.3 Ordinal Encoding for 'country'

- Transformed the country column into ranked numerical values using OrdinalEncoder.

- The encoding order was based on the frequency of content from each country (value_counts()).
- A new country_ordinal column was added to the DataFrame.

country_order = df_normalised['country'].value_counts().index.tolist()

ordinal_encoder = OrdinalEncoder(categories=[country_order])

df_normalised['country_ordinal']=ordinal_encoder.fit_transform(df_normalised[['country']])

5.4 Normalization (Min-Max Scaling)

- Applied MinMaxScaler to scale numerical features to a standard range between 0 and 1.
- This step is important for machine learning algorithms sensitive to the scale of input features.
- The following newly created columns were normalized:
    - **rating_label**
    - **country_ordinal**

scaler = MinMaxScaler() cols_to_normalize = ['rating_label', 'country_ordinal']

df_normalised[cols_to_normalize] = scaler.fit_transform(df_normalised[cols_to_normalize])