Netflix Insights and Metrics

Week 1&2:

Netflix Insights

1. **Content Distribution**
   - The dataset comprises **~8,800 titles**, a combination of both **Movies (~70%)** and **TV Shows (~30%)**.
   - Movies dominate Netflix's catalog, but TV Shows have been increasing in recent years, signaling Netflix's shift towards episodic content.
2. **Temporal Trends**
   - Titles span multiple decades, with older classics alongside recent Netflix Originals.
   - A sharp rise in content is observed post-2015, aligning with Netflix's global expansion strategy.
3. **Genre Representation**
   - A wide variety of genres exist.
   - **Top genres:** Dramas, Documentaries, Comedies.
   - **Emerging genres:** International TV, Stand-up Comedy, and Romantic TV Shows — reflecting user demand.
4. **Geographical Spread**
   - Content originates from over 100 countries, showcasing Netflix's global production and licensing reach.
   - Major contributors**:** United States, India, United Kingdom, Japan, South Korea.
5. **Rating Distribution**
   - Titles are spread across maturity ratings (TV-MA, R, PG-13, TV-14, etc.).
   - A strong presence of mature-rated content (TV-MA, R) indicates a focus on adult audiences, but family-friendly segments (TV-Y, PG) are also well represented.
6. **Missing Data Observations**
   - Director and cast columns had significant missing values, likely due to incomplete metadata.
   - Rating and Duration had gaps, which were filled systematically for consistency.

Netflix Metrics/Scope

1. **Trend Analysis**
   - Evaluate the evolution of Movies vs. TV Shows, genres, and ratings over years.
   - Guide Netflix in shaping its content acquisition and production strategies.
2. **Genre Popularity & Recommendations**
   - Identify top genres globally and regionally.
   - Enable personalized recommendations based on user preferences.

3. **Geographical Expansion Strategy**
   - Assess country-level contributions to Netflix's catalog.
   - Support regional expansion and localized content production.
4. **Content Duration Insights**
   - Distinguish average movie length vs. average TV Show seasons.
   - Inform viewer engagement and content planning.
5. **Data Quality Improvement**
   - Enhance metadata completeness for directors and casts.
   - Support enriched recommendation systems and talent-based content analysis.

Dataset Loading

The Netflix dataset is sourced from Kaggle and loaded into the workspace for preprocessing and analysis.

- **Dataset Source:** Kaggle — Netflix Movies and TV Shows Dataset.
- **Dataset Size:** ~8,800 titles across multiple years and genres.
- **Key Columns:** type, title, director, cast, country, release_year, rating, duration, listed_in, date_added.

Loading the dataset using pandas:

```
import pandas as pd

df_read = pd.read_csv("/Volumes/workspace/default/netflix/netflix_titles.csv")

display(df_read.head())
```

output:

show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
s1	Movie	Example Movie Title	John Doe	Actor A, Actor B	United States	2020-01-01	2019	PG-13	90 min	Dramas, Comedies	A short description of the movie.
s2	TV Show	Example Show Title	Jane Smith	Actor C, Actor D	India	2019-06-10	2018	TV-MA	2 Seasons	TV Dramas, International TV Shows	A short description of the show.

The dataset provides a rich set of features that allow for multi-dimensional analysis of Netflix's content strategy, such as genre diversity, rating distributions, and country-wise availability.

Data Cleaning Steps Using Pandas

Step 1 — Null Handling/Handling missing Values

Purpose:
Missing values in a dataset can distort analysis and predictions. We handle missing data to make the dataset complete and reliable.

Actions Taken:

- Dropped rows missing date_added values (important for temporal analysis).
- Filled missing director and cast values with "Not Available" to keep information consistent.
- Filled missing country with "Unknown" to represent missing geographical data.
- Filled missing duration with "0" to keep numeric processing consistent.
- Filled missing rating values with the **most common rating** (mode) to maintain category balance.

Code:

```
df_read['director'] = df_read['director'].fillna("Not Available")

df_read['cast'] = df_read['cast'].fillna("Not Available")

df_read['country'] = df_read['country'].fillna("Unknown")

df_read['duration'] = df_read['duration'].fillna("0")

mode_rating = df_read['rating'].mode()[0]

df_read['rating'] = df_read['rating'].fillna(mode_rating)
```

Example Output — Missing Values Check:

Missing values per column:

```
show_id      0

type         0

title        0

director     0

cast         0
```

Step 2 — Remove Duplicates

Purpose:
Duplicate rows can bias analysis. Removing them ensures the dataset represents unique entries only.

ActionTaken:
Removed exact duplicate rows from the dataset.

Code:

```
df_cleaned = df_read.drop_duplicates()

print(df_cleaned.shape)
```

Example Output:

```
(7787, 12)
```

(Original dataset size: 8000 rows → After cleaning: 7787 rows)

Step 3 — Whitespace Cleaning

Purpose:
Extra spaces in categorical columns can cause incorrect grouping and encoding. Cleaning spaces ensures consistency.

ActionsTaken:
Trimmed spaces from type and rating columns.

Code:

```
df_cleaned['type'] = df_cleaned['type'].str.strip()

df_cleaned['rating'] = df_cleaned['rating'].str.strip()
```

Example-Output:
Before cleaning: " Movie ", " PG-13 " → After cleaning: "Movie", "PG-13".

Step 4 — Duration Extraction

Purpose:
The duration column contains both numeric and text values (e.g., “90 min”, “2 Seasons”). Splitting them allows quantitative analysis of durations.

Actions Taken:

- Extracted numeric part into duration_num.
- Extracted duration type (min, Season, Seasons) into duration_type.

Code:

```
df_read['duration_num'] = df_read['duration'].str.extract(r'(\d+)').astype(float)

df_cleaned['duration_type'] = df_cleaned['duration'].str.extract(r'(min|Season|Seasons)')
```

Example Output:

duration	**duration_num**	**duration_type**
90 min	90.0	min
2 Seasons	2.0	Seasons
1 Season	1.0	Season

Step 5 — Date Conversion

Purpose:
The date_added column must be in datetime format for time-series analysis.

ActionTaken:
Converted date_added to datetime, handling errors.

Code:

```
df_read['date_added'] = pd.to_datetime(df_read['date_added'], errors='coerce')
```

Example Output:

```
Earliest date added: 2008-01-01

Latest date added: 2023-07-15
```

Normalization

Normalization is a crucial preprocessing step that converts categorical text data into numerical representations. This process ensures data consistency, enables effective analysis, and prepares the dataset for machine learning and statistical modeling.

Step 1 — Label Encoding for Rating

Purpose:
Ratings are categorical (e.g., PG, TV-MA, R) and cannot be directly used in models. Label Encoding converts them into numerical labels while preserving their uniqueness.

Code:

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

df_normalized['rating_label'] = label_encoder.fit_transform(df_cleaned['rating'].astype(str))
```

Example Output:

rating	rating_label
PG	0
PG-13	1
TV-MA	2
R	3
TV-Y	4

Explanation:
Each unique rating value is assigned a numeric label starting from 0. This encoding allows numerical processing of ratings without losing their categorical nature.

Step 2 — One-Hot Encoding for Type

Purpose:
The type column (Movie or TV Show) is categorical with no ordinal relationship. One-hot encoding creates separate binary columns for each category.

Code:

```
from sklearn.preprocessing import OneHotEncoder
```

```
onehot_encoder = OneHotEncoder(sparse_output=False)

type_encoded_array = onehot_encoder.fit_transform(df_cleaned[['type']])

df_type_onehot = pd.DataFrame(type_encoded_array, columns=onehot_encoder.get_feature_names_out(['type']))

df_type_onehot.index = df_cleaned.index
```

Example Output Data:

type	type_Movie	type_TV Show
Movie	1.0	0.0
TV Show	0.0	1.0
Movie	1.0	0.0

Explanation:
One-hot encoding ensures no numerical ordering is assumed for categories. Each category becomes a separate column with binary values indicating presence (1) or absence (0).

Step 3 — One-Hot Encoding for Listed Genres

Purpose:
The listed_in column contains multiple genres for a title. Multi-hot encoding transforms each genre into a separate binary column.

Code:

```
listed_encoded_array = onehot_encoder.fit_transform(df_cleaned[['listed_in']])

df_listed_onehot = pd.DataFrame(listed_encoded_array, columns=onehot_encoder.get_feature_names_out(['listed_in']))

df_listed_onehot.index = df_cleaned.index
```

Example Output Data:

listed_in	listed_in_Comedies	listed_in_Dramas	listed_in_Crime TV Shows
Dramas, Comedies	1.0	1.0	0.0
Crime TV Shows	0.0	0.0	1.0
Documentaries	0.0	0.0	0.0

Explanation:
Each genre gets its own column. A value of 1 means the title belongs to that genre, allowing flexible genre-based analysis.

Step 4 — Ordinal Encoding for Country

Purpose:
Countries are categorical, but we can encode them based on their frequency in the dataset for analysis.

Code:

```
from sklearn.preprocessing import OrdinalEncoder

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

ordinal_encoder_country = OrdinalEncoder(categories=[country_order])

df_normalized['country_ordinal'] = ordinal_encoder_country.fit_transform(df_cleaned[['country']])
```

Example Output:

country	country_ordinal
United States	0.0
India	1.0
United Kingdom	2.0

Explanation:
The most frequent country gets the lowest ordinal value (0.0). This preserves frequency order without implying magnitude relationships.

Step 5 — Combine Normalized Columns

Purpose:
Combine all normalized columns into a final dataset for further analysis.

Code:

```
df_normalized = pd.concat([df_cleaned, df_type_onehot, df_listed_onehot], axis=1)

df_normalized.to_csv("/Volumes/workspace/default/netflix/netflix_normalized.csv", index=False)
```

Example Final Normalized Dataset:

show_id	type	rating	rating_label	type_Movie	type_TV Show	country	country_ordinal	listed_in_Comedies	listed_in_Dramas	duration_num	duration_type
s1	Movie	PG-13	1	1.0	0.0	United States	0.0	1.0	1.0	90.0	min
s2	TV Show	TV-MA	2	0.0	1.0	India	1.0	0.0	0.0	2.0	Seasons

Summary:
Normalization ensures the dataset is structured for analytics and modeling. Label encoding, one-hot encoding, and ordinal encoding make categorical data usable for algorithms and visualizations.