Netflix Dataset Analysis

Introduction

This document provides a comprehensive overview of the preprocessing and normalization steps performed on the Netflix Movies and TV Shows dataset. The primary goal is to clean, structure, and encode the dataset to enable meaningful analysis, visualization, and predictive modeling. All preprocessing steps are performed using **Python Pandas** and **scikit-learn** in a Databricks

Dataset Information

File: netflix_analysis.csv

Column	Description
show_id	Unique identifier for each title
type	Movie or TV Show
title	Title of the content
director	Director(s) of the title
cast	Cast members
country	Country of production
date_added	Date added to Netflix
release_year	Year of release
rating	Content rating based on age group
duration	Duration in minutes (Movie) or seasons (TV Show)
listed_in	Genres (comma-separated)
description	Text description of the content

Netflix Dataset – Data Cleaning

1. Introduction

The raw Netflix dataset contains various inconsistencies such as missing values, duplicate entries, extra whitespace, and formatting issues. Cleaning the dataset ensures that the data is consistent, reliable, and ready for further processing (normalization, EDA, or machine learning).

The following sections explain the **data cleaning process** applied in Databricks using **Pandas**.

2. Data Cleaning

Step 1: Load Dataset

The dataset was imported into Pandas for inspection and cleaning.

```
import pandas as pd
import numpy as np

df = pd.read_csv("/Volumes/workspace/default/netflix_analysis/netflix_titles.csv")
```

- **Purpose:** Brings the raw data into memory for preprocessing.
- **Outcome:** DataFrame df containing Netflix titles.

Step 2: Initial Exploration

Exploration is necessary to understand the dataset's size, structure, and issues.

```
print(df.shape)       # Number of rows & columns
print(df.info())      # Data types & null counts
print(df.head())      # Preview first 5 rows
print(df.describe())  # Summary stats for numeric columns
print(df.isnull().sum()) # Count missing values per column
```

- **Purpose:**
  - Identify missing values.
  - Check data types.
  - Detect anomalies in the dataset.

Step 3: Handle Missing Values

```
df.dropna()
df.fillna(0)
```

- **dropna()** removes rows with missing values.
- **fillna(0)** replaces missing values with 0.

- In this case, the dataset preview was shown using these functions, but final decisions on missing values can vary (e.g., replacing with "Unknown" instead of 0).

Step 4: Remove Duplicate Records

Duplicate rows can distort insights, so they were removed.

```
before = df.shape[0]
df = df.drop_duplicates()
after = df.shape[0]
print(f"\nRemoved {before - after} duplicate rows.")
```

- **Purpose:** Ensures each Netflix title entry is unique.
- **Outcome:** Cleaner dataset without redundancy.

Step 5: Remove Extra Whitespace

Some categorical values had **leading/trailing spaces**. These were stripped.

```
df['type'] = df['type'].str.strip()
df['rating'] = df['rating'].str.strip()
```

- **Example:** " Movie " → "Movie"
- **Outcome:** Uniform categorical values.

Step 6: Standardize Country & Director Column

- Cleaned **country names** by removing unwanted symbols.
- Converted **director names** to lowercase for consistency.

```
df['country'] = df['country'].str.strip()
df['directory'] = df['director'].str.lower()
df['country'] = df['country'].str.replace(r'[^a-z  A-Z]', '', regex=True)
```

- **Purpose:**
  - Standardize country field (removing punctuation, symbols).
  - Normalize director names to lowercase for easier grouping.

Step 7: Save Cleaned Dataset

The cleaned dataset was saved for further use in normalization.

```
df.to_csv(
"/Volumes/workspace/default/netflix_analysis/netflix_titles/cleaned_netflix_titl
es.csv",
index=False
)
```

- **Purpose:** Store the cleaned dataset for downstream processes.
- **Output File:** cleaned_netflix_titles.csv

Netflix Dataset – Normalization

1. Introduction

After cleaning the Netflix dataset, the next step is **Normalization**. Normalization prepares the dataset for analysis and machine learning by converting categorical data into numeric representations and scaling numerical features.

The goal of normalization is to:

- Encode categorical values consistently.
- Scale numerical values to a standard range.
- Reduce bias caused by different feature scales.
- Make the dataset suitable for modeling and EDA.

This section explains the transformations applied to the cleaned Netflix dataset.

2. Normalization

Step 1: Import Libraries and Load Cleaned Data

The dataset cleaned in the previous step was imported for normalization.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, MinMaxScaler

df = pd.read_csv("/Volumes/workspace/default/netflix_analysis/netflix_titles/cleaned_netflix_titles.csv")
```

- **Purpose:** Bring the cleaned dataset into memory for encoding and scaling.

Step 2: Label Encoding – *Rating Column*

The rating column (e.g., *PG, R, TV-MA*) is categorical. Using **Label Encoding**, each unique rating was mapped to an integer.

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['rating_encoded'] = label_encoder.fit_transform(df['rating'].astype(str))
```

- **Example:**
  - PG → 2
  - R → 3
  - TV-MA → 5
- **Purpose:** Converts text labels into machine-readable numeric values.

Step 3: One-Hot Encoding – *Type Column*

The type column (Movie / TV Show) was transformed into a **binary indicator variable**.

```
df["is_movie"] = df["type"].apply(lambda x: 1 if x.strip().lower() == "movie" else 0)
```

- **Example:**
  - Movie → 1
  - TV Show → 0

- **Purpose:** Allows easy filtering and statistical analysis between movies and TV shows.

Step 4: Frequency Encoding – *Country Column*

The country column contains many unique values (high cardinality). Instead of one-hot encoding, **frequency encoding** was used.

```
freq_encoding = df['country'].value_counts().to_dict()
df['country_encoded'] = df['country'].map(freq_encoding)
```

- **Example:**
  - United States (most frequent) → High number
  - Smaller countries → Lower numbers
- **Purpose:** Keeps dataset compact and captures distributional information about countries.

Step 5: Scaling Numerical Features

Numerical columns (release_year and duration_minutes) were scaled to a range **[0,1]** using **MinMaxScaler**.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df['release_year_scaled'] = scaler.fit_transform(df[['release_year']])
df['duration_norm'] = scaler.fit_transform(df[['duration_minutes']])
```

- **Example:**
  - Oldest release year (1942) → 0
  - Newest release year (2021) → 1
  - Movie duration 90 minutes → normalized to range between 0 and 1.
- **Purpose:** Ensures that features on different scales (years vs minutes) are comparable for ML models.

Step 6: Save Normalized Dataset

The final normalized dataset was stored for downstream tasks like EDA or predictive modeling.

```
df.to_csv(
"/Volumes/workspace/default/netflix_analysis/netflix_normalization.csv",
```

```
index=False
)
```

- **Output File:** netflix_normalization.csv