**Netflix Data Processing**

**– Kshitij Thorat**

**Executive Summary**

This document explains a real data processing pipeline that transformed 4,103 Netflix titles (movies and TV shows) into a format ready for machine learning analysis. The pipeline ran on Databricks Serverless using Apache Spark 4.0.0, converting messy text data into 78 structured columns of numerical features.

**Dataset:** Netflix catalog with 4,103 titles and 16 original columns   
**Output:** 78 columns including encoded features, normalized values, and standardized metrics   
**Technology:** Databricks Serverless + Apache Spark 4.0.0 + PySpark

**Dataset Overview**

**Source File:** netflix\_clean\_dataset\_kshitij\_.csv   
**Total Records:** 4,103 Netflix titles   
**Original Columns:** 16 attributes per title

**Data Processing**

**Stage 1: Environment Setup & Data Loading**

Loading verified the data is accessible and correctly formatted before investing time in processing.

**Stage 2: Column Type Identification**

Different column types need different processing approaches. This classification guides the encoding strategy.

**String Columns (12):** Text data requiring encoding

show\_id, type, title, director, cast, country, date\_added, rating, duration, listed\_in, description, duration\_type

**Numeric Columns (4):** Already in number format

release\_year, duration\_int, added\_year, added\_month

**Stage 3: Handling Missing Values**

**Columns Processed:** type, rating, duration\_type, country, director, cast, listed\_in

**Result:** All 4,103 records now have complete data across all columns.

**Data Encoding: Converting Text to Numbers**

Machine learning algorithms require numerical input. The following sections explain how text and categories were transformed into numbers while preserving their meaning.

**Stage 4: Label Encoding**

Convert ordered categories into sequential numbers. Dense rank function assigns consecutive integers based on alphabetical order.

**Columns Encoded:**

1. **type** (Movie vs TV Show)
2. **rating** (G, PG, PG-13, R, TV-MA, etc.)
3. **duration\_type** (minutes vs seasons)

|  |  |
| --- | --- |
| **Original Type** | **Encoded Value** |
| **movie** | **0** |
| **tv show** | **1** |

**Stage 5: One-Hot Encoding**

Create binary flags for categories without natural ordering.

**Technical Process:**

1. Identify unique country values
2. Create one column per country
3. Set value to 1 if match, 0 if not match

**Stage 6: Frequency Encoding**

Capture how common or rare each category value is.

**Columns Encoded:**

1. **director** - How prolific is this director?
2. **cast** - How frequently do these actors appear?
3. **listed\_in** - How common is this genre combination?

**Formula:**

Frequency = (Count of specific value) ÷ (Total records)

**Process:**

1. Count occurrences of each unique value
2. Divide by total dataset size (4,103)
3. Join frequency back to original data

**Data Scaling: Making Numbers Comparable**

Numbers in the dataset have wildly different ranges:

Release years: 1925 to 2021 (range of 96)

Duration: 1 to 237 minutes (range of 236)

Added year: 2015 to 2021 (range of 6)

**Stage 7: Min-Max Normalization**

Squeeze all values into a 0 to 1 range.

**Columns Normalized:**

1. release\_year
2. duration\_int
3. added\_year

**Formula:**

Normalized = (Value - Minimum) / (Maximum - Minimum)

**Process:**

1. Find minimum and maximum values for each column
2. Apply formula to every record
3. Result: 0 = minimum value, 1 = maximum value, everything else between

**Release Year Normalization:**

Minimum: 1925

Maximum: 2021

Range: 96 years

Examples:

1925 → (1925 - 1925) / 96 = 0.00 (oldest content)

2005 → (2005 - 1925) / 96 = 0.83 (relatively recent)

2021 → (2021 - 1925) / 96 = 1.00 (newest content)

**Added Year Normalization:**

Minimum: 2015

Maximum: 2021

Range: 6 years

Examples:

2018 → 0.77

2020 → 0.92

2021 → 1.00 (most recent additions)

**Interpretation Guide after normalization:**

**0.0 - 0.3:** Very low/old/short

**0.3 - 0.7:** Medium/average

**0.7 - 1.0:** High/new/long

**Stage 8: Standardization (Z-Score)**

Transform data to show how unusual each value is compared to the average.

**Columns Standardized:**

* release\_year
* duration\_int
* added\_year

**Formula:**

Z-Score = (Value - Mean) / Standard Deviation

**Process:**

1. Calculate mean (average) for each column
2. Calculate standard deviation (measure of spread)
3. Apply formula to every record

**Real Statistics Calculated:**

**Release Year:**

Mean: 2013.24

Standard Deviation: 8.89 years

**Duration:**

Mean: 99.21 minutes

Standard Deviation: 23.53 minutes

**Added Year:**

Mean: 2019.65

Standard Deviation: 1.59 years

**Real-World Examples:**

**Apocalypse Now (1979):**

* Release year Z-score: -3.96 → Extraordinarily old for Netflix catalog
* Duration Z-score: +3.27 → One of the longest films
* This is a rare, classic, epic-length film

**Final Dataset Summary**

**Input:**

1. 4,103 records
2. 16 columns
3. Mix of text and numbers

**Output:**

1. 4,103 records (all preserved)
2. 78 columns total
3. All numerical features
4. No missing values

**Original Columns:** 16 (preserved)

show\_id, type, title, director, cast, country, date\_added, release\_year, rating, duration, listed\_in, description, duration\_int, duration\_type, added\_year, added\_month

**New Encoded Columns:** 62

**Label Encoded:** 3 columns

1. type\_label
2. rating\_label
3. duration\_type\_label

**One-Hot Encoded:** 50 columns

1. country\_onehot\_United\_States
2. country\_onehot\_India
3. country\_onehot\_South\_Korea
4. ... (47 more country columns)

**Frequency Encoded:** 3 columns

1. director\_freq
2. cast\_freq
3. listed\_in\_freq

**Normalized:** 3 columns

1. release\_year\_normalized
2. duration\_int\_normalized
3. added\_year\_normalized

**Standardized:** 3 columns

1. release\_year\_standardized
2. duration\_int\_standardized
3. added\_year\_standardized

**Total:** 78 columns

**Stage 9: Saving Processed Data**

/Volumes/workspace/default/netflix/netflix\_processed

**Storage Format:** Delta Lake

**Delta Lake Benefits: ACID Transactions:**

1. Atomicity: All-or-nothing writes
2. Consistency: Data integrity maintained
3. Isolation: Concurrent operations don't interfere
4. Durability: Once saved, data persists