[ReelGood]

[G4]

Data Science Capstone Project
Data Acquisition and Pre-Processing Report

Date:

[2/8/2025]

Team Members:

Name: Jaz Zhou

Name: Precious Orekha

Name: Alireza Hatami

Name: Caitlin Dunne

Identifying Data

Data Sources:

The dataset used in this project was downloaded from Kaggle and includes metadata for 45,000 movies, along with user ratings for these films. All movies in the dataset were released on or before July 2017. The data points cover a variety of details, such as cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, and countries. This dataset was chosen because most available movie datasets typically only provide user ratings and lack comprehensive metadata. According to the Kaggle page where the data was sourced, the information was originally collected from TMDB and GroupLens. This dataset is currently the most detailed collection of movie information that could be found.

Acquisition Process:

The dataset was directly downloaded from the provided link* and we did not need to write any code to acquire it. The file is publicly accessible as a zip file containing multiple CSV files. This dataset includes the following files:

- movies_metadata.csv: This file contains features on 45,000 movies. Features include title, genre, budget, revenue, release dates, languages, production companies, etc.
- keywords.csv: This file contains the movie plot keywords for movies.
- credits.csv: This file contains the cast and crew information for all of our movies.
- links.csv: This file contains the TMDB and IMDB IDs of all the movies.
- ratings.csv: This file contains 26 million ratings from 270,000 users for all 45,000 movies.
- ratings_small.csv: This file contains a subset of 100,000 ratings from 700 users on 9,000 movies.

Link:

* https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset/data

Issues:

There were no specific issues related to the data acquisition process.

Data-Processing

A) Relative Feature Selection

The algorithms we plan to use are purely Collaborative Filtering (KNN, SVD++, MLP), which primarily relies on user ratings, while metadata serves as auxiliary information. The following features from each file have been selected:

- movies_metadata.csv
  - **id**: Primary key for joining tables.
  - **imdbId**: Key for retrieving missing data using the IMDB API.
  - **genre**: highly relevant for recommendations, as different users prefer different genres. It can also be used as a filtering criterion in the recommender system.
  - **release_date**: highly relevant for recommendations, especially for capturing temporal behavior and trends in user preferences. It can also be used as a filtering criterion.
  - **original_language**: highly relevant for recommendations, as users often prefer movies in languages they understand. It can also be used as a filtering criterion.
  - **title:** Essential for interpretability of the recommender system. It helps in presenting recommendations in a meaningful way.
- credits.csv
  - **cast**: highly relevant for recommendations, as users may favor movies featuring specific actors.
  - **crew**: crew member's role and name, only director will be kept in the cleaned version as it is the most relevant crew member for recommendation purposes.
- ratings.csv

  Since we focus on CF algorithms, **ratings** naturally become the main feature.

B) Duplicate Removal

Duplicate entries are identified and removed to ensure data integrity.

C) Handling Missing Values

Missing values exist in both `movies_metadata.csv` and `credits.csv,` we retrieve missing values using `imdbId` as a key from the IMDB API.

While a large portion of missing data is recovered, a few values are unavailable on IMDB. The still missing data are minimal and therefore dropped without significant impact on the dataset.

D) Feature Cleaning

Raw feature data is cleaned for improved usability in feature engineering.

- Example:
  - genre: Convert a nested list of dictionaries into a list of genre names.

    `"[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}]"` → `['Animation', 'Comedy']`.

For a detailed breakdown of cleaning methods for each column, refer to the **Appendix (Column-Specific Cleaning Methods)**.

E) Rating Matrix Downsizing

Due to the high sparsity and large size of the rating matrix, computational constraints necessitate downsizing. Downsizing is performed by:

- Removing movies that are missing in the final cleaned metadata.
- Retaining only users who have rated at least 20 movies.

F) Rating Train-Test Split

A **stratified split** is applied to capture temporal changes in user behavior, the split is based on the `timestamp` feature to maintain chronological order:

- The last rating of each user is placed in the test set.
- The second-to-last rating is placed in the validation set.
- The remaining ratings are placed in the training set.

Appendix

A) Raw Data:

1) movies.csv

The movie's metadata file has 24 columns and 45466 entries. We want to use only Collaborative Filtering algorithms, so we only need a few columns as auxiliary information. Specifically, we will keep ['id', 'imdbId', 'title', 'genre', 'original_language', 'release_date'] .

- **genres:** A list of genres in a JSON-like format (e.g., [{ "id": 28, "name": "Action" }]).
- **id:** A unique movie ID assigned to the movie (needs to match movieId in the ratings dataset).
- **imdb_id:** The IMDb ID for the movie (e.g., "tt0114709" for Toy Story).
- **original_language:** The primary language of the movie (e.g., "en" for English, "fr" for French).
- **title:** The official title of the movie (may differ from original_title).
- **release_date:** The date of movie release.

**The raw data format and missing value percentage can be found below.*

	adult	belongs_to_collection	budget	genres	homepage
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...	http://toystory.disney.com/toy-story
1	False	NaN	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...	NaN
2	False	{'id': 119050, 'name': 'Grumpy Old Men Collect...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, ...	NaN
3	False	NaN	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'nam...	NaN
4	False	{'id': 96871, 'name': 'Father of the Bride Col...	0	[{'id': 35, 'name': 'Comedy'}]	NaN

	id	imdb_id	original_language	original_title	overview	popularity	poster_path
0	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	21.946943	/rhIRbceoE9lR4veEXuwCC2wARtG.jpg
1	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	17.015539	/vzmL6fP7aPKNKPRTFnZmiUfciyV.jpg
2	15602	tt0113228	en	Grumpier Old Men	A family wedding reignites the ancient feud be...	11.7129	/6ksm1sjKMFLbO7UY2i6G1ju9SML.jpg
3	31357	tt0114885	en	Waiting to Exhale	Cheated on, mistreated and stepped on, the wom...	3.859495	/16XOMpEaLWkrcPqSQqhTmeJuqQl.jpg
4	11862	tt0113041	en	Father of the Bride Part II	Just when George Banks has recovered from his ...	8.387519	/e64sOI48hQXyru7naBFyssKFxVd.jpg

	production_companies	production_countries	release_date	revenue	runtime	spoken_languages
0	[{'name': 'Pixar Animation Studios', 'id': 3}]	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-10-30	373554033.0	81.0	[{'iso_639_1': 'en', 'name': 'English'}]
1	[{'name': 'TriStar Pictures', 'id': 559}, {'na...	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-12-15	262797249.0	104.0	[{'iso_639_1': 'en', 'name': 'English'}, {'iso...
2	[{'name': 'Warner Bros.', 'id': 6194}, {'name'...	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-12-22	0.0	101.0	[{'iso_639_1': 'en', 'name': 'English'}]
3	[{'name': 'Twentieth Century Fox Film Corporat...	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-12-22	81452156.0	127.0	[{'iso_639_1': 'en', 'name': 'English'}]
4	[{'name': 'Sandollar Productions', 'id': 5842}...	[{'iso_3166_1': 'US', 'name': 'United States o...	1995-02-10	76578911.0	106.0	[{'iso_639_1': 'en', 'name': 'English'}]

	status	tagline	title	video	vote_average	vote_count
0	Released	NaN	Toy Story	False	7.7	5415.0
1	Released	Roll the dice and unleash the excitement!	Jumanji	False	6.9	2413.0
2	Released	Still Yelling. Still Fighting. Still Ready for...	Grumpier Old Men	False	6.5	92.0
3	Released	Friends are the people who let you be yourself...	Waiting to Exhale	False	6.1	34.0
4	Released	Just When His World Is Back To Normal... He's ...	Father of the Bride Part II	False	5.7	173.0

```
print_missing_values(filtered_movies_metadata_df)
```

```
Checking column: genres
NaN values count: 0 (0.00%)
Empty lists count: 2442 (5.37%)
--------------------------------------------------
Checking column: id
NaN values count: 0 (0.00%)
Empty lists count: 0 (0.00%)
--------------------------------------------------
Checking column: imdb_id
NaN values count: 17 (0.04%)
Empty lists count: 0 (0.00%)
--------------------------------------------------
Checking column: original_language
NaN values count: 11 (0.02%)
Empty lists count: 0 (0.00%)
--------------------------------------------------
Checking column: title
NaN values count: 6 (0.01%)
Empty lists count: 0 (0.00%)
--------------------------------------------------
Checking column: release_date
NaN values count: 87 (0.19%)
Empty lists count: 0 (0.00%)
--------------------------------------------------
```

2) credits.csv

In the credits data frame, we have 45476 entries and two columns for the cast and crew members of the movies.

- Cast: actors' names
- crew: other crew members, can be used to extract directors' names.

**The raw data format and missing value percentage can be found below.*

	cast	crew	id
0	[{'cast_id': 14, 'character': 'Woody (voice)',...	[{'credit_id': '52fe4284c3a36847f8024f49', 'de...	862
1	[{'cast_id': 1, 'character': 'Alan Parrish', '...	[{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de...	8844
2	[{'cast_id': 2, 'character': 'Max Goldman', 'c...	[{'credit_id': '52fe466a9251416c75077a89', 'de...	15602
3	[{'cast_id': 1, 'character': "Savannah 'Vannah...	[{'credit_id': '52fe44779251416c91011acb', 'de...	31357
4	[{'cast_id': 1, 'character': 'George Banks', '...	[{'credit_id': '52fe44959251416c75039ed7', 'de...	11862

```
print_missing_values(credits_df)
```

```
Checking column: cast
NaN values count: 0 (0.00%)
Empty lists count: 2418 (5.32%)
--------------------------------------------------
Checking column: crew
NaN values count: 0 (0.00%)
Empty lists count: 771 (1.70%)
--------------------------------------------------
Checking column: id
NaN values count: 0 (0.00%)
Empty lists count: 0 (0.00%)
--------------------------------------------------
```

3) ratings.csv

It includes the user ID, movie ID, rating, and timestamps.

	userId	movieId	rating	timestamp
0	1	110	1.0	1425941529
1	1	147	4.5	1425942435
2	1	858	5.0	1425941523
3	1	1221	5.0	1425941546
4	1	1246	5.0	1425941556

B) PsuedoCode for preprocessing:

1) movies_metadata.csv and credits.csv:

```
def preprocess_meta_data(movies_df, credits_df):

    """
    Input: Raw datasets movies.df and credits.df
    Output: Processed dataset with cleaned and imputed metadata
    """

    # Step 1: Select the relevant columns from movies.csv
    col_to_keep_movies = ['id', 'imdbId', 'title', 'genre',
    'original_language', 'release_date']
    movies_df = movies_df[col_to_keep_movies]

    # Step 2: Select the relevant columns from credits.csv
    col_to_keep_credits = ['id', 'cast', 'crew']
    credits_df = credits_df[col_to_keep_credits]

    # Step 3: Merge datasets using an outer join on 'id'
    # ** Why we merge tables before cleaning? Please see below. **
    meta_df = pd.merge(movies_df, credits_df, on='id', how='outer')

    # Step 4-6: Clean, identify missing values, and retrieve from IMDb
    # ** Why we first clean then handle missing values?
    for col in meta_df.columns:
        # ** Column-specific cleaning methods please see below **
        meta_df[col] = clean_column(meta_df[col])
        missing_value_ids = identify_missing_values(meta_df[col])
        meta_df[col] = retrieve_from_imdb(meta_df[col], missing_value_ids)

    return meta_df
```

Footnote:

*** Why we merge tables before cleaning?*

1. *imdbId is only available in movies.csv and is required for API retrieval.*
2. *Genre in movies.csv is needed for handling missing cast values in credits.csv. If genre is Documentary, an empty cast is valid; otherwise, missing cast should be retrieved via IMDb API.*

*** Why we first clean then handle missing values?*

Reasons:

1. *Raw data often contains inconsistencies (e.g., NaN, "[]", None, empty strings).*
2. *Cleaning ensures all missing values follow a uniform format, making it easier to detect and process them.*

Example:

```
def extract_genres(genres):
  try:
    genres_list = ast.literal_eval(genres)
    return [genre['name'] for genre in genres_list]
  except (ValueError, TypeError):
    return []
```

With the cleaning, we can ensure that:

1. *Any **valid** genre data is extracted cleanly.*
2. *Any **invalid/missing** data (NaN, None, empty string, malformed JSON) gets converted to [ ].*
3. *This ensures all missing values are standardized to an **empty list**, making it trivial to detect missing genres later (`df['genres'].apply(lambda x: x == [])`).*

** *`Column`-specific cleaning methods:*

1. `'id', 'imdbId', 'title', 'original_language'`:
   - Convert to string format to ensure consistency.
2. `'genre'`:
   - Convert nested list of dictionaries into a list of genre names.
   - Example: "[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}]" → ['Animation', 'Comedy']
3. `'release_date'`:
   - Extract only the year from the full date format.
   - Example: "1994-06-15" → "1994"
4. `'cast'`:
   - Convert nested list of dictionaries into a list of actor names.
   - Keep only the first 3 actors.
   - Example: "[{'cast_id': 14, 'name': 'Tom Hanks'}, {'cast_id': 2, 'name': 'Tim Allen'}]" → ['Tom Hanks', 'Tim Allen']
5. `'crew'`:
   - Extract only the director's name from the list of crew members.
   - Example: "[{'job': 'Director', 'name': 'Joe Johnston'}, {'job': 'Producer', 'name': 'Jane Doe'}]" → ['Joe Johnston']

2) ratings.csv:

```
-----------------------------------------------------------------------------

def preprocess_ratings(ratings_df, meta_df):
      """
      Input:
      - ratings_df: Raw ratings dataset
      - meta_df: Preprocessed metadata containing valid movies
      Output:
      - Processed ratings dataset with filtered movies and active users
      """
      ratings_df = only_keep_movies_in_meta_df(ratings_df, meta_df)

      ratings_df = only_keep_users_with_enough_rating(ratings_df, threshold = 
      20)

      test_df = last_rating_of_each_user_based_on_timestamp(ratings_df)
      validation_df = 
      second_last_rating_of_each_user_based_on_timestamp(ratings_df)
      train_df = remove_rows(ratings_df, test_df + validation_df)
      train_matrix = convert_to_matrix(train_df)

      return ratings_df, train_df, validation_df, train_df, train_matrix
-----------------------------------------------------------------------------
```

C) Cleaned Data:

– Metadata

- Standardized feature format for future encoding.
- No missing values. (majority being retrieved through API, a few being dropped)

	id	title	year	genres	first_three_actors	director	original_language	imdb_id
0	862	Toy Story	1995	['Animation', 'Comedy', 'Family']	['Tom Hanks', 'Tim Allen', 'Don Rickles']	John Lasseter	en	tt0114709
1	8844	Jumanji	1995	['Adventure', 'Fantasy', 'Family']	['Robin Williams', 'Jonathan Hyde', 'Kirsten D...	Joe Johnston	en	tt0113497
2	15602	Grumpier Old Men	1995	['Romance', 'Comedy']	['Walter Matthau', 'Jack Lemmon', 'Ann-Margret']	Howard Deutch	en	tt0113228
3	31357	Waiting to Exhale	1995	['Comedy', 'Drama', 'Romance']	['Whitney Houston', 'Angela Bassett', 'Loretta...	Forest Whitaker	en	tt0114885
4	11862	Father of the Bride Part II	1995	['Comedy']	['Steve Martin', 'Diane Keaton', 'Martin Short']	Charles Shyer	en	tt0113041

– Ratings

- Ensured all movie IDs are valid and has corresponding metadata.
- Filtered users with at least 20 ratings.
- Split into training, validation, and test sets for Collaborative Filtering.
- Training sets transformed into matrix ready for training.

	userId	movieId	rating	timestamp
59	4	223	4.0	1042668576
60	4	415	4.0	1042667925
61	4	648	4.0	1042674800
66	4	1422	4.0	1042674861
68	4	1597	3.0	1042674787

userId	8	11	12	15	16	20	24	30	34	37	...	270859	270869	270871	270872	270879	270885	270887	270893	270894	270896
movieId																					
1	4.0	NaN	4.0	NaN	NaN	4.0	4.0	NaN	3.0	3.5	...	3.0	4.0	5.0	3.5	3.0	NaN	5.0	4.0	NaN	4.5
2	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	3.0	NaN	...	NaN	2.0	2.5	NaN	3.5	NaN	5.0	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	1.5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 64433 columns

```
Total elements in matrix: 899420442
Non-null entries: 9972455
Sparsity: 98.89%
```

Table of Contributions

The table below identifies contributors to various sections of this document.

	Section	Writing	Editing
1	**Data Sources & Acquisition**	Alireza Hatami & Caitlin Dunne	Precious Orekha
2	**Data-Processing**	Alireza Hatami & Jaz Zhou	Caitlin Dunne
3	**Appendix**	Alireza Hatami & Jaz Zhou	Precious Orekha

Grading

The grade is given on the basis of quality, clarity, presentation, completeness, and writing of each section in the report. This is the grade of the group. Individual grades will be assigned at the end of the term when peer reviews are collected.