# Big Data Analytics
# Assignment 3

Due: Monday April 14th @ 11:59PM

## Introduction

You will work with a large-scale dataset: the **McAuley-Lab/Amazon-Reviews-2023** dataset, hosted on Hugging Face at:

https://huggingface.co/datasets/McAuley-Lab/Amazon-Reviews-2023

The uncompressed size is roughly 200GB across 34 categories. You will complete a series of tasks spanning data ingestion, cleaning, exploratory data analysis (EDA), sentiment classification, recommender systems, and clustering.

All methods and algorithms are **fixed** (no substitutions), but you may implement these steps in any environment (local machine, cluster, or cloud). Work in groups of 3–4, dividing tasks as you see fit.

**Dataset Fields:**

**For User Reviews** ("review" data):

- **rating (float)**: User's rating of the product (1.0–5.0)
- **title (str)**: Title of the user review
- **text (str)**: Text body of the user review
- **images (list)**: URLs or metadata of images in different sizes
- **asin (str)**: ID of the product
- **parent_asin (str)**: Parent ID of the product (products with different colors/sizes can share the same parent)
- **user_id (str)**: ID of the reviewer
- **timestamp (int)**: Time of the review (Unix time)
- **verified_purchase (bool)**: Whether the purchase was verified

- **helpful_vote (int)**: Number of helpful votes the review received

**For Item Metadata** ("meta" data):

- **main_category (str)**: The domain or top-level category
- **title (str)**: Name of the product
- **average_rating (float)**: Rating shown on the product page
- **rating_number (int)**: Number of ratings for the product
- **features (list)**: Bullet-point features of the product
- **description (list)**: Descriptions of the product (often multi-line)
- **price (float)**: Price in US dollars (at time of crawling)
- **images (list)**: Product images (thumb, large, hi_res)
- **videos (list)**: Videos of the product (with title, URL)
- **store (str)**: Store name (where the product is sold)
- **categories (list)**: Hierarchical categories
- **details (dict)**: Product details (materials, brand, sizes, etc.)
- **parent_asin (str)**: Parent ID of the product
- **bought_together (list)**: Bundles or recommended items often bought together

*Note:* To link a user review with its product metadata, you typically use `parent_asin` as the join key. Some of the additional fields (like `helpful_vote`, `verified_purchase`, or `price`) can be incorporated into your analysis or EDA if you wish; however, the core tasks below are mandatory.

# Assignment Specification

## 1. Data Acquisition

a) **Obtain the Entire Dataset**
You must download or otherwise access *all categories* (34) from **McAuley-Lab/Amazon-Reviews-2023**. The direct link is: https://huggingface.co/datasets/McAuley-Lab/Amazon-Reviews-2023. Use either:

- `load_dataset("McAuley-Lab/Amazon-Reviews-2023", ...)`
- The provided `download_all_amazon_reviews` function
- Or any other method, as long as you include *all categories*.

b) **Deliverable Proof**
Provide logs or screenshots verifying the successful acquisition of all categories (compressed or uncompressed).

c) **Deliverables (Data Acquisition)**:

- *Report*: Evidence (screenshots, console output) showing you obtained all categories.
- *Code*: Scripts/notebooks demonstrating how you downloaded and organized the data.

## 2. Data Cleaning & Preprocessing

Perform these steps **exactly**:

a) **Merge on parent_asin**
Join review data with its matching metadata on `parent_asin` so each record combines user ID, product ID (`asin`), star rating, review text, verified purchase, helpful votes, brand/category info, etc.

b) **Handle Invalid / Missing Values**

- Drop rows where *star_rating* is missing or not in [1–5].
- Drop rows if *text* (the review body) is empty.
- If brand cannot be found in the metadata (e.g., missing in `details` or `store`), set brand = "Unknown".

c) **Remove Duplicates**
If $(\text{user\_id}, \text{product\_id}, \text{review\_text})$ repeats, keep only the first occurrence.

d) **Derived Columns**:

- **Review Length** = integer count of tokens in the `text` field (split on whitespace/punctuation).
- **Year** = extracted from `timestamp` if available (if there's no timestamp, you may skip time-based EDA).

e) **Unified Output**
Consolidate all categories into a single large dataset (or partitioned store) for further tasks.

## 3. Exploratory Data Analysis (EDA)

Perform the following analyses on the *entire cleaned dataset* (all categories combined):

a) **Star Rating Histogram**: Show a histogram for ratings 1–5.

b) **Top 10 Categories**: Bar chart of categories by total review count (use the final `brand` or `main_category` if it is clearly identified).

c) **Top 10 Brands**: Bar chart of brand by total review count (exclude "Unknown" from the top 10).

d) **Time-Based Trend**: a line chart of average star rating per year.

e) **Correlation**: Compute Pearson correlation between *review_length* and *star_rating*; state the numeric result and interpret briefly.

f) **(Optional) Additional EDA Ideas**:

- Distribution of `helpful_vote` counts.
- Relationship between `verified_purchase` and star rating.

(These are not strictly required but may enrich your insights.)

## 4. Binary Sentiment Prediction (Logistic Regression)

Transform *rating* into:

$$\text{Positive if rating } > 3, \quad \text{Negative if rating } \leq 3.$$

a) **Train/Test Split**: 80% train, 20% test, randomly shuffled.

b) **Text Vectorization**: *TF-IDF* on review text (lowercase, split on whitespace/punctuation), discarding tokens in fewer than 5 reviews or in over 80% of reviews.

c) **Classifier**: *Logistic Regression* (default hyperparameters).

d) **Evaluation**:

- Accuracy
- F1 Score
- Confusion Matrix (2×2: TP, FP, TN, FN)

## 5. Recommender System (ALS)

Create a **collaborative filtering** model using **Alternating Least Squares (ALS)**:

a) **Data Setup**: Retain $(\text{user\_id}, \text{product\_id}, \text{rating})$. Drop users with fewer than 5 total reviews.

b) **Split**: 80% train, 20% test.

c) **ALS**: Use any library that supports ALS (Spark MLlib, etc.). Minimal tuning is acceptable.

d) **Evaluation**: RMSE on the test set (predicted rating vs. actual).

e) **Demo**: Show top 5 recommendations for 3 random users in the test set, including predicted ratings.

## 6. Clustering / Segmentation (k-means)

Segment *products* using **k-means** with $k = 5$:

a) **Features** (per product):

$$(\text{mean\_rating},\ \text{total\_reviews},\ \text{brand\_id},\ \text{category\_id})$$

- `mean_rating`: Average user rating per product *(based on your merged data, not necessarily `average_rating` from metadata, though you could compare them.)*
- `total_reviews`: Count of all reviews for that product
- `brand_id`: Map each distinct brand string to an integer
- `category_id`: Map each main category or top-level category string to an integer

b) **k-means**: Exactly $k = 5$, default initialization, until convergence.

c) **Cluster Analysis**: For each cluster, report:

- **Size**: number of products in the cluster
- **Average mean_rating**, **average total_reviews**
- **Average brand_id** and **category_id**
- A short interpretation (e.g., high-rating electronics, unknown-brand items, etc.)

## 7. Final Report (8–15 pages)

Must have these sections in **exact** order:

a) **Introduction**

b) **Data Acquisition (Task 1)**

c) **Data Cleaning & Preprocessing (Task 2)**

d) **EDA (Task 3)**

e) **Binary Sentiment (Logistic Regression) (Task 4)**

f) **Recommender (ALS) (Task 5)**

g) **Clustering (k-means) (Task 6)**

h) **Conclusion (mention hardware/resources used, challenges, etc.)**

**Code Repository**: Organize notebooks/scripts clearly.

## Grading Breakdown

| **Section** | **Weight** |
|---|---|
| Data Acquisition (Task 1) | 5% |
| Data Cleaning & Preprocessing (Task 2) | 15% |
| EDA (Task 3) | 10% |
| Binary Sentiment (Logistic Regression) (Task 4) | 20% |
| Recommender (ALS) (Task 5) | 20% |
| Clustering (k-means) (Task 6) | 15% |
| **Final Report (Task 7)** | 15% |
| **Total** | **100%** |

**Notes:**

- **All Categories**: Ensure you eventually use all categories (∼200GB total).
- **Resource Planning**: At least 16GB RAM is recommended and enough disk space or a streaming or distributed setup to handle large data.

# Example Data Acquisition Workflows

These examples illustrate different ways to download/load data from McAuley-Lab/Amazon-Reviews-2023. Feel free to mix/modify to fit your environment.

## Option A: Directly Use `load_dataset`

```
from datasets import load_dataset

# Example: Load "Automotive" reviews and metadata
auto_review = load_dataset(
    "McAuley-Lab/Amazon-Reviews-2023",
    "raw_review_Automotive",
    trust_remote_code=True
)
auto_meta = load_dataset(
    "McAuley-Lab/Amazon-Reviews-2023",
    "raw_meta_Automotive",
    trust_remote_code=True
)
```

**Pros**: Simple for smaller subsets or prototyping.
**Cons**: Repeated loads can be slow for the entire 200GB.

## Option B: `save_to_disk` / `load_from_disk`

```
from datasets import load_dataset, load_from_disk
from pathlib import Path

# Download + save
auto_review = load_dataset(
    "McAuley-Lab/Amazon-Reviews-2023",
    "raw_review_Automotive",
    trust_remote_code=True
)
save_path = Path("/path/to/data/raw_review_Automotive")
auto_review.save_to_disk(str(save_path))

# Reload later
loaded_review = load_from_disk(str(save_path))
print(loaded_review)
```

**Pros**: One-time download, faster local reloads.
**Cons**: Requires storing all uncompressed data locally.

### Option C: `download_all_amazon_reviews` (No Compression)

```
from bigdata_a3_utils import download_all_amazon_reviews

download_all_amazon_reviews(
    base_save_path="/path/to/data",
    categories=None,   # all categories
    compress=False
)
# Creates folders: raw_review_<CATEGORY>/, raw_meta_<CATEGORY>/, etc.
```

Then you can use `load_from_disk` on the created folders.

### Option D: `download_all_amazon_reviews` (With Compression)

```
from bigdata_a3_utils import download_all_amazon_reviews, load_compressed_dataset

download_all_amazon_reviews(
    base_save_path="/path/to/data_compressed",
    categories=None,    # all
    compress=True,
    compression_format="gz",  # or "bz2", "xz"
    compression_level=6
)

# After downloading, you'll have .tar.gz files like raw_review_Automotive.tar.gz
dataset = load_compressed_dataset("/path/to/data_compressed/raw_review_Automotive.tar.gz
```

**Pros**: Saves disk space.
**Cons**: More CPU/time overhead for compress/decompress.

***Note:*** If using options **A** or **B**, `load_dataset()`, data is stored in your HuggingFace cache directory, so you do not need to redownload it each time you call `load_dataset()`. However, this directory is not cleared automatically. Use functions from `bigdata_a3_utils` to locate and/or clear your HuggingFace cache when you are finished with the data.