☆ Cache() & Persist() in Spark →

~~Cache() →~~

- Stores in-memory only.
- If dataset is reused cache it
- in Collaborative filtering, Similarity matrix is queried many Times
  → Caching avoid recomputation

Cache()

- Stores in memory only (RAM)
- good for small dataset which fit in RAM.
- Shortcut of .persist(StorageLevel.MEMORY_ONLY)

persist()

- More flexible : Can store to disk, memory & both.
- useful for data too big for RAM.
- .persist(StorageLevel.MEMORY_AND_DISK)

★ Item Based Collaborative filtering.

Recomendation technique that predict a user's prefrences for an item based on their rating of similar items

Analogy → Sia Rated Fight Club & Forrest gump similarly
So, do Ben,
Ted only watched Forrest Gump, so we can recommend Fight Club to him.

very old school technique.

Steps →

1. Self-Join on userid → get (movie1, movie2, rating1, rating2)
2. Filter out duplicates (movie1, movie2) == (movie2, movie1)
3. Compute cosine similarity

$$\cos(A, B) = \frac{\sum(r_A \cdot r_B)}{\sqrt{\sum(r_A^2)} \cdot \sqrt{\sum(r_B^2)}}$$

4. Rank & Pick Top similar movies.

why? →
Because Captures pattern of similarity even if users rate on different scales. (Normalizes Ratings)

★ System module in python →

import sys → captures cli arguments.
if (len(sys.argv) > 1) → check if movie id is passed or not
we cores → spark-submit file.py 50
sys.exit(1) & sy.exit(0)

__/__/____

☆ Code for movie similarity →

1. Core logic → Cosine similarity (def cosine_compute_Similarity(data):)

   $xx = rating1^2$, $yy = rating2^2$, $xy = rating1 * rating2$

   group by (movie1, movie2)

   ⇒ $\frac{numerator}{denominator} = \frac{\Sigma(xy)}{\sqrt{\Sigma(xx)} * \sqrt{\Sigma(yy)}} = score.$

   if denominator = 0 → score = 0

   output → (movie1, movie2, score, numpairs) → count(xy)

2. after return of dataframe cache it.

3. Threshold for good similarity →

   - Score Threshold = 0.97 → only recommend if similarity is High
   - coOccurrence Threshold = 50, if number of pairs (enough users) rated both.

4. Broadcast lookup → (def getMovieName(movies_name, movieID))
   uses movieName Dataset to Map IDs → titles.

Short flow →

load → Pair → Cosine → Cache → Filter → Sort → Print