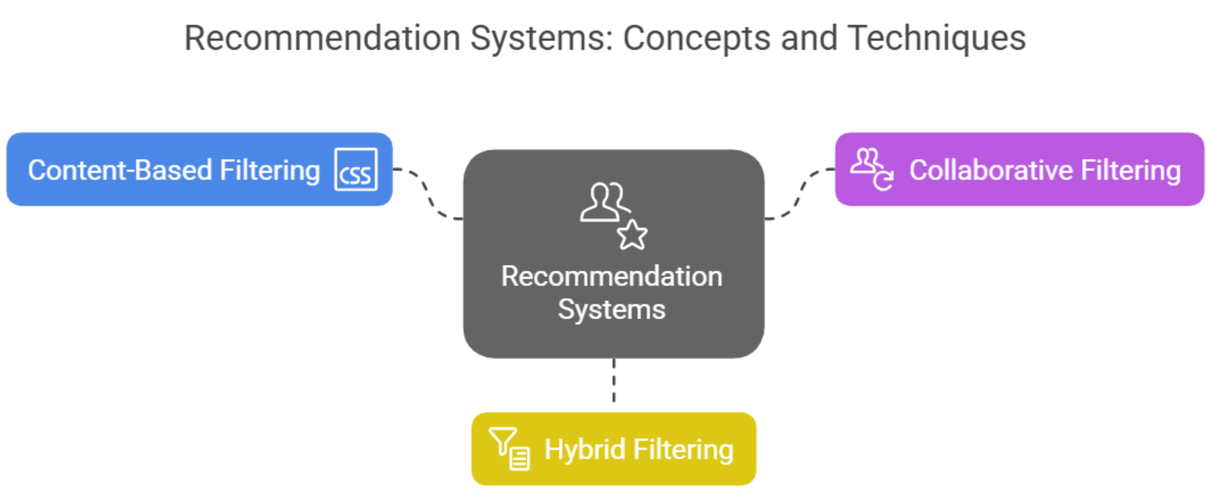
RECOMMENDATION SYSTEMS – KEY CONCEPTS



1️⃣ Content-Based Filtering

> **Instagram Analogy**: Imagine you liked a lot of **travel reels** and followed **travel bloggers**. Even if no one else did, Instagram would recommend **similar travel posts** because it knows **you like that type of content**.

Logic:

- Based on **what the item is** (title, author, genre, etc.)
- It doesn't care what others liked.
- Uses content features to recommend **similar items** to what you already liked.

📚 In Book Recommender:

If you liked "Harry Potter", we look at its **author, title, keywords**, and find **similar books** using text similarity.

2️⃣ Collaborative Filtering

> **Instagram Analogy**: Let's say 10 people who follow **Tech Creators** also like **Gaming pages**. Now you also follow Tech Creators, so Instagram starts suggesting **Gaming posts**, even if you never showed interest.

Logic:

- Based on **what users liked in common**.
- Doesn't care what the book is *about*.
- If people with **similar taste** to you liked something, you'll likely like it too.

🛠️ Matrix Form:

- Create a matrix of `User x Item (Book)` with ratings.
- Use **SVD** to learn patterns and predict missing ratings.

3️⃣ Hybrid Filtering

Instagram Analogy: You love travel and follow many travel influencers. Others who follow the same also love food vlogs. Now Insta combines **your travel interests** + **what others similar to you like** → shows you **food + travel vlogs**!

🧠 **Logic:**

- Combines both content-based and collaborative filtering.
- More personalized and effective.

📏 Key Techniques Explained

✅ Cosine Similarity

Analogy: Think of each book as a point in space made of words. The more similar their words (angle between vectors), the closer they are.

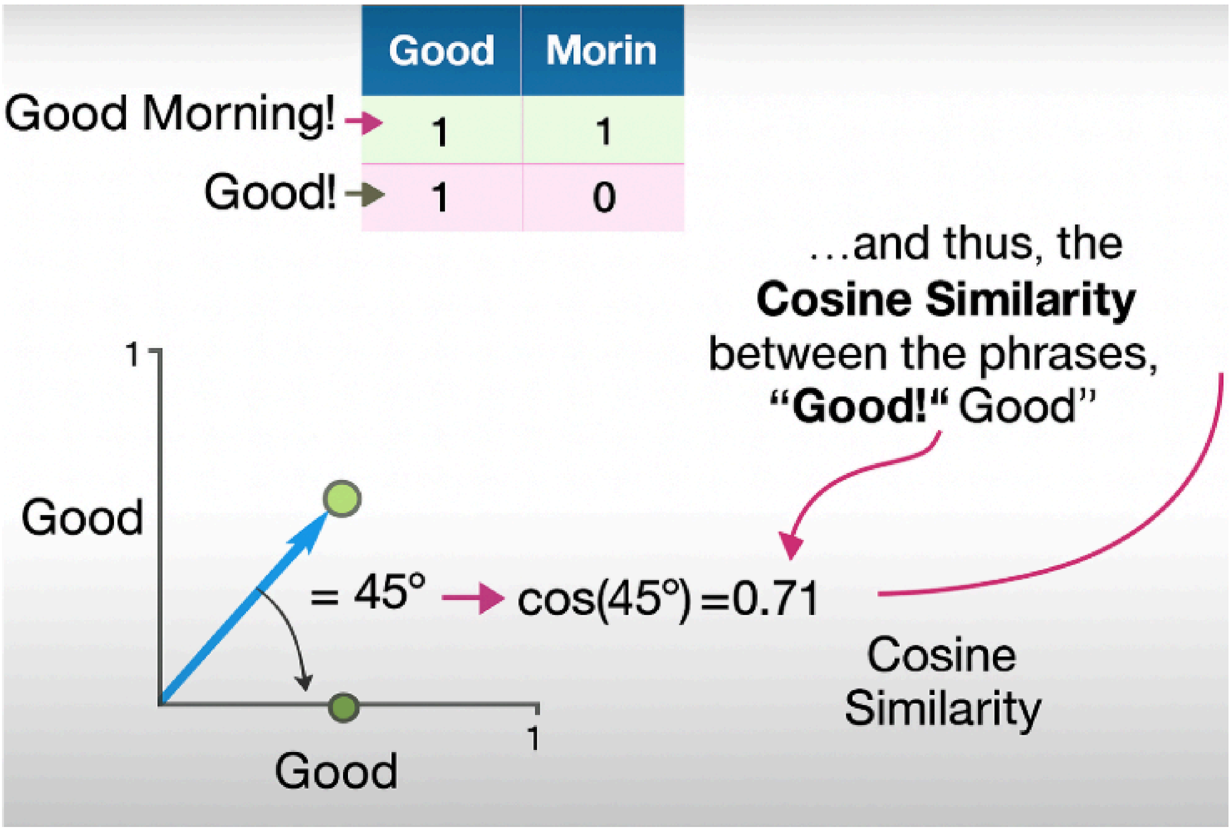
The movie is really good - positive

The movie was not bad, it was ok - positive

Cos 0 - 1

Cos 45 - 0.71

Cos 90- 0



Formula:
$\text{similarity}=||A|| \cdot ||B| \ / \ |A \cdot B$

Used in: Content-Based Filtering

✅ Matrix Factorization (Collaborative Filtering)

> **Analogy**: Imagine a big table with users and books — but it's full of blanks. Matrix Factorization learns patterns (hidden features) and fills in the blanks.

Example:

- User A likes fantasy + romance.
- Book X = fantasy.
- SVD learns this hidden feature and predicts user A will like Book X.

✅ SVD (Singular Value Decomposition)

> **Analogy**: Like breaking a song into vocals, drums, and bass — SVD breaks the user-book rating matrix into patterns.

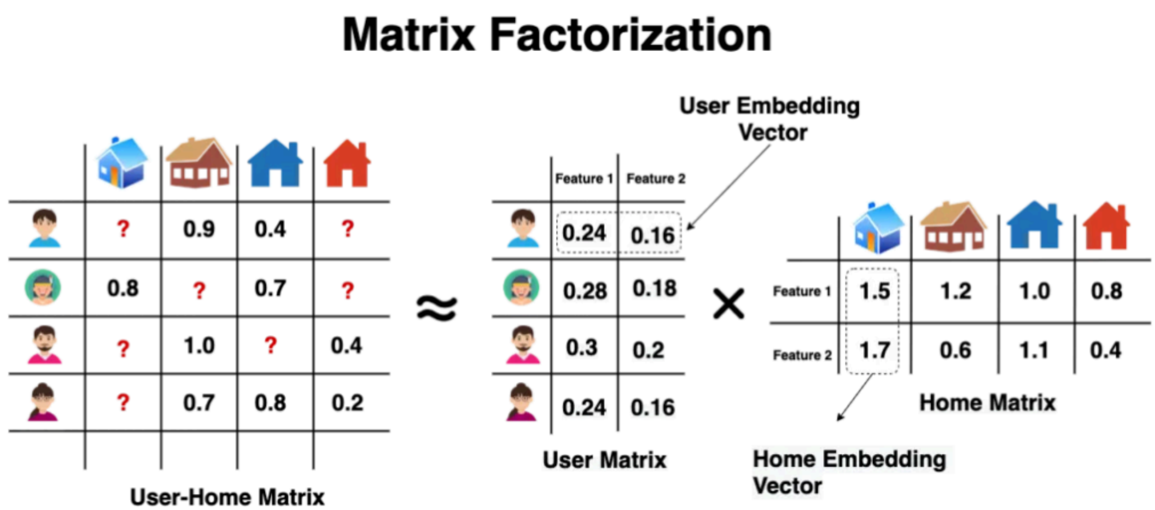
R=U⋅Σ⋅VTR = U \cdot \Sigma \cdot V^TR=U⋅Σ⋅VT

- U: user features
- Σ: importance (weights)
- V^T: item (book) features

Used in: `Surprise` library for collaborative filtering.

Matrix Factorization

This is a technique often used in **recommendation systems** — like how Netflix recommends movies or how Amazon recommends products.



Left Side: User-Home Matrix

- Rows = Users (4 users in this case)
- Columns = Homes (4 homes)
- The numbers in the matrix represent **how much a user likes a home** — kind of like a rating.
- The **question marks (?)** mean we don't know how much a user likes that home — our goal is to **predict those missing values**.

Middle & Right Side: Breaking It Down into Simpler Parts

To predict the missing values, we break this big matrix into two smaller matrices:

1. **User Matrix**:

    - Each user is represented by a **vector** of features (e.g., what they like in a home: maybe size, color, location, etc.)
    - Example: The first user has features `[0.24, 0.16]`
2. **Home Matrix**:
    - Each home is also represented by a **vector** of features.
    - Example: The first home has features `[1.5, 1.7]`

Multiply Them Together

To **reconstruct the original matrix** (and fill in the missing values), we multiply:

User Matrix × Home Matrix → Gives us an estimate of the User-Home preferences.

Why Do This?

Instead of directly guessing the missing numbers, matrix factorization helps us:

- Learn patterns in user preferences.
- Learn characteristics of homes.
- Make smart predictions about the question marks.

SVD→ singular Value Decomposition