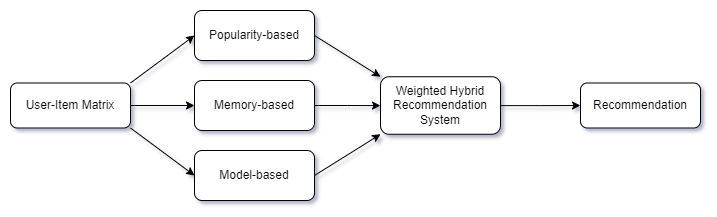
**RECOMMENDATION PROJECT**

**PROJECT OVERVIEW:**



**Project Overview – Hybrid Recommendation System for Santander Bank**

**What's the goal?**

The aim is to build a **smart recommendation system** to help Santander Bank suggest the right products to each customer based on what they already have or might need.

**How does it work?**

1. **Start with data:**  
   You begin with a **User-Item Matrix** – basically a table that shows which customer owns which products as of **May 28, 2015**.
2. **Build 3 types of recommendation models:**
   * **Popularity-based:** Recommends products that are popular with many users.
   * **Memory-based:** Looks for users with similar tastes and recommends what they like (also called collaborative filtering).
   * **Model-based:** Uses machine learning to find patterns and make recommendations.
3. **Combine them smartly:**
   * These three models are merged into one powerful system called a **Weighted Hybrid Recommendation System**. Each model contributes its recommendations with a certain weight (importance).
4. **Final Output:**
   * The hybrid system gives the **final list of recommended products** for each customer.

**How will success be measured?**

* The system's performance will be evaluated using a method called **Average Precision** – which checks how accurate the recommendations are.

**Why is this useful?**

It helps Santander Bank **personalize** product suggestions, improving customer satisfaction and possibly increasing sales.

**USER ITEM MATRIX:**

**What is a User-Item Matrix?**

Imagine you have a table where:

* **Rows = Users**
* **Columns = Items (like movies, products, songs, etc.)**

Each **cell** in the table shows:

* How much a user liked an item
* OR whether a user interacted with an item (like viewed, bought, rated)

This table is called the **User-Item Matrix**.

**Why do we use it?**

It helps us **understand user preferences**  
It’s the **foundation of Collaborative Filtering**, especially memory-based methods  
We can find:

* Similar users (User-based CF)
* Similar items (Item-based CF)
* What items to recommend to a user based on others like them

**Simple Example:**

|  | **Pizza** | **Movie** | **Game** |
| --- | --- | --- | --- |
| User1 | 5 | 0 | 3 |
| User2 | 0 | 4 | 2 |
| User3 | 5 | 4 | 0 |

* 5 means the user liked it a lot
* 0 means no interaction

**What can we do with it?**

1. **Find Similar Users:**
   * User1 and User3 both love Pizza → maybe User1 will also like Movie, just like User3
2. **Fill in the blanks:**
   * If User2 hasn’t tried Pizza, but similar users love it → we can recommend it!

**POPULARITY BASED:**

**What is Popularity-Based Recommendation?**

The **Popularity-Based model** recommends the most **frequently interacted or purchased products**, regardless of the user.  
It's like saying:

***"Let’s show everyone what’s trending or popular right now."***

**Where is it used?**

Imagine you’re using a music app like Spotify. Even if the app knows nothing about you, it can still recommend the "Top 10 Global Hits" — that's popularity-based filtering.

**How it works (Step-by-step):**

**Step 1: Count product usage**

For each product (like ind\_cco\_fin\_ult1, ind\_ecue\_fin\_ult1, etc.), count how many users have it marked as "used" (i.e., the value is 1).

**Step 2: Rank products**

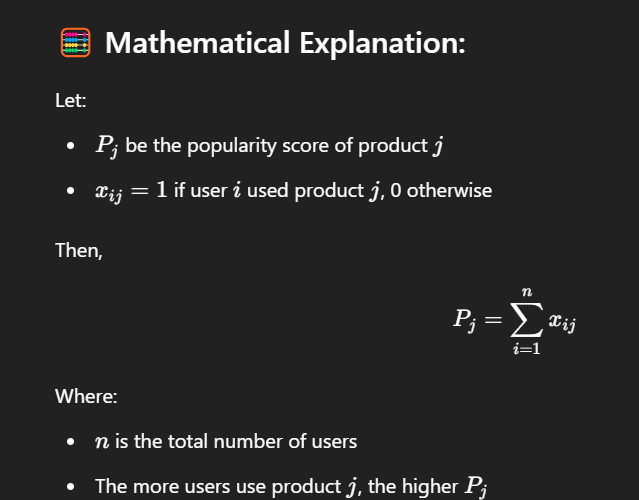
Sort the products by their total count — most used at the top.

**Step 3: Recommend top N**

Choose the top N products (like top 5 or top 10) and recommend them to every user.

**Mathematical Explanation:**

Let:



**Example:**

Suppose we have a tiny dataset:

| **User** | **Product A** | **Product B** | **Product C** |
| --- | --- | --- | --- |
| 1 | 1 | 0 | 1 |
| 2 | 1 | 1 | 0 |
| 3 | 1 | 0 | 0 |

Popularity scores:

* Product A: 1+1+1=31 + 1 + 1 = 31+1+1=3
* Product B: 0+1+0=10 + 1 + 0 = 10+1+0=1
* Product C: 1+0+0=11 + 0 + 0 = 11+0+0=1

**Result:**

* Recommend: Product A first (most popular)
* Then Product B or C (tied)

**MEMORY BASED:**

**What is Memory-Based Collaborative Filtering?**

It's a technique that recommends products based on **similarities between users** (or items) using their historical interaction patterns.

In our case:

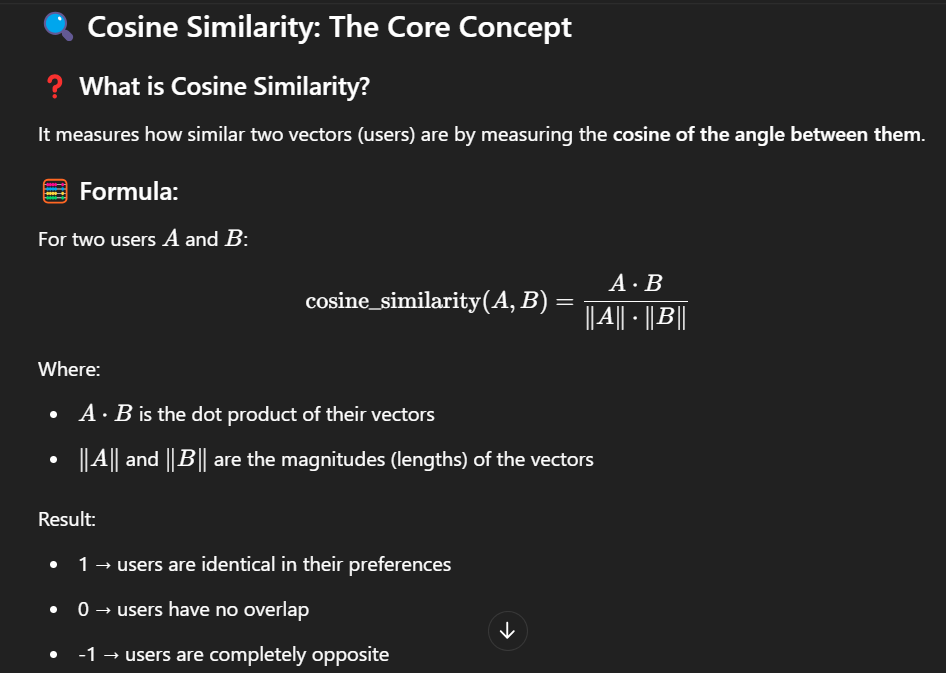
* Users are rows
* Products are columns
* Values are typically 1 if the user used that product, 0 otherwise

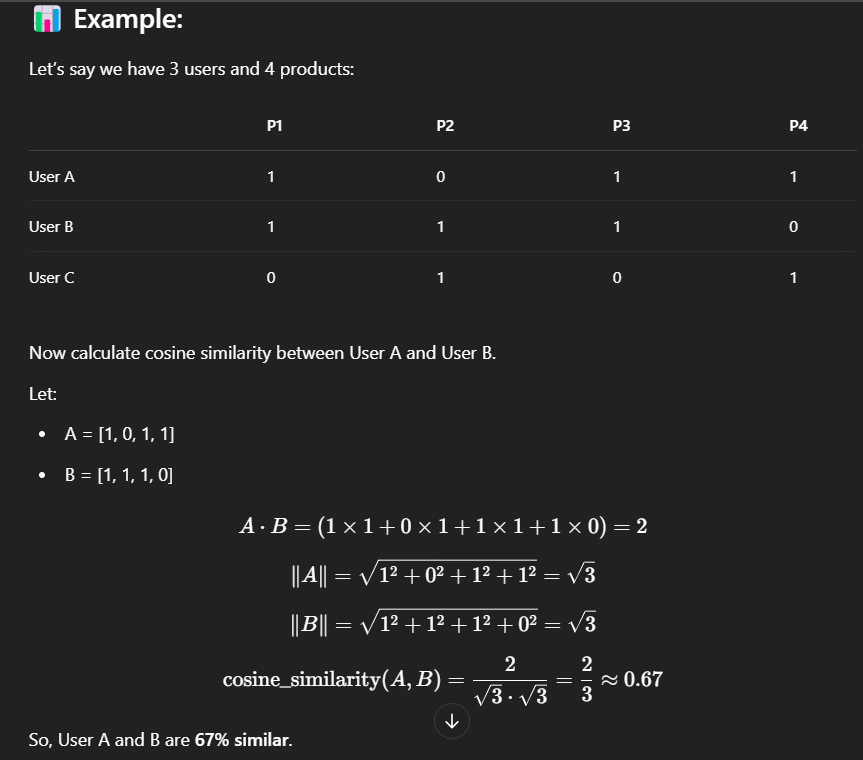
**Goal:**

Find **users who are similar to each other** using **cosine similarity** and recommend products that similar users have used.

**Why "Memory-Based"?**

It relies on the **entire user-item interaction matrix in memory** to compute similarities — hence, "memory-based".





**MODEL BASED:**

**What is Model-Based Collaborative Filtering?**

Unlike memory-based methods (which use the entire user-item matrix directly), **model-based** methods **learn a predictive model** from the data.  
We train a machine learning algorithm to predict **which product a user will use**, based on their historical data and other user patterns.

**Why Decision Trees?**

Decision Trees are:

* Easy to understand
* Handle categorical data well (like products used or not used)
* Can capture non-linear relationships between user preferences and product choices

**How It Works:**

We treat the recommendation task as a **classification problem**.

**Goal**: Predict whether a user will use a specific product (label = 1 or 0), based on their interaction with other products.

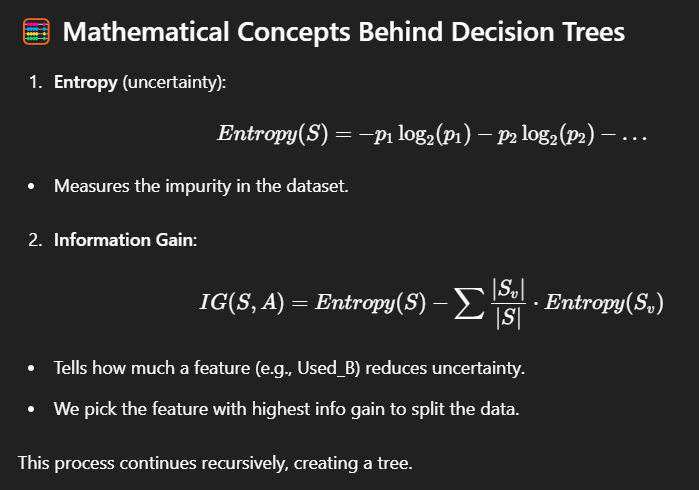
**Example:**

Let’s say we want to **predict whether a user will use Product A**.

We use other product interactions (Product B, C, D...) as input features, and whether they used Product A as the label.

| **Used B** | **Used C** | **Used D** | **Will Use A (label)** |
| --- | --- | --- | --- |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |

Train a decision tree classifier on this data.



**WEIGHTED HYBRID MODEL:**

**🤝 What is a Hybrid Recommendation System?**

A **hybrid model** combines two or more recommendation techniques to **leverage the strengths** of each while **minimizing their weaknesses**.

**🔢 What is a Weighted Hybrid Model?**

In a **weighted hybrid model**, we:

* Use multiple models (like popularity-based, memory-based, model-based).
* Assign each model a **weight** based on its **importance or accuracy**.
* Combine their predictions using those weights.

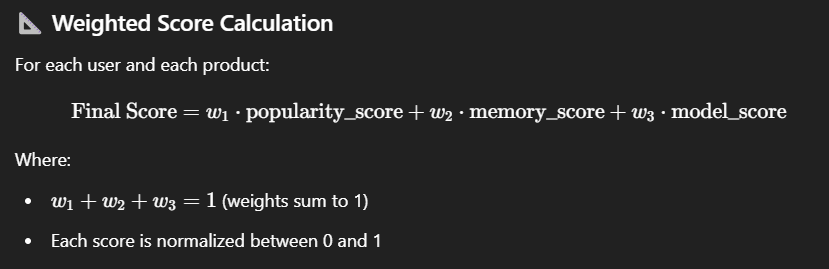
Think of it like making a smoothie 🥤: Each model is an ingredient, and you decide how much of each to add for the best flavor (i.e., recommendation quality).

**💡 Models Being Combined in Our Case:**

1. **Popularity-Based Model** (weight = w₁)
2. **Memory-Based Model** (Collaborative Filtering using Cosine Similarity) (weight = w₂)
3. **Model-Based Decision Tree Model** (weight = w₃)

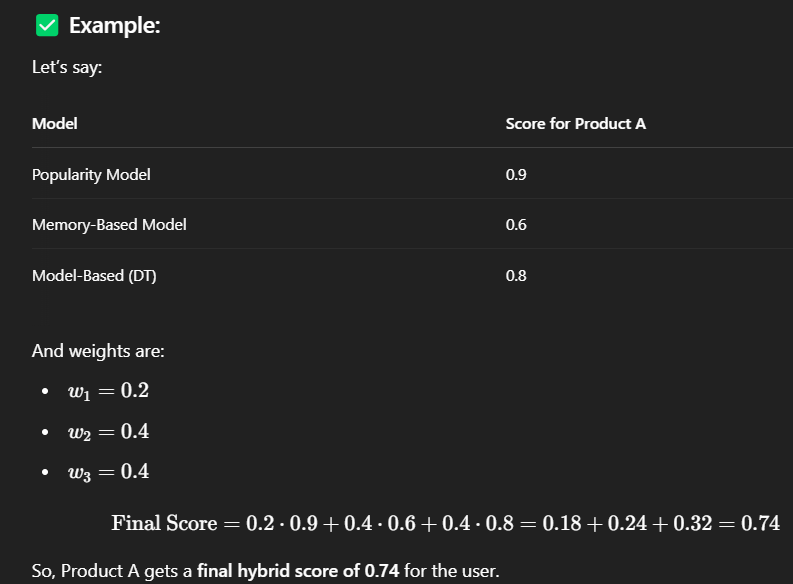
These models each give a **score or prediction** for how likely a user is to use a product.

**📐 Weighted Score Calculation**

****

**✅ Example:**

Let’s say:



**🧠 Why Use a Weighted Hybrid?**

* 📊 **Popularity Model** is fast but not personalized.
* 🤝 **Memory-Based** captures user similarity but struggles with sparse data.
* 🧠 **Model-Based** captures complex patterns but may overfit.

So, combining them: ✅ gives robustness  
✅ balances personalization and performance  
✅ improves recommendation accuracy