Recommender System in Python

Recommender System

- a system that helps customers/ users discover products or services they may like
- an application of machine learning that predicts the future preference of a set of products for a users and provides personalised suggestions
- act as information filtering system → filter out irrelevant content and provide users with a personalised experienced of whatever service
- used to predict a user's liking or preference for a particular product or service
- type of recommender system
  - content-based filtering
  - collaborative filtering

Content-based filtering

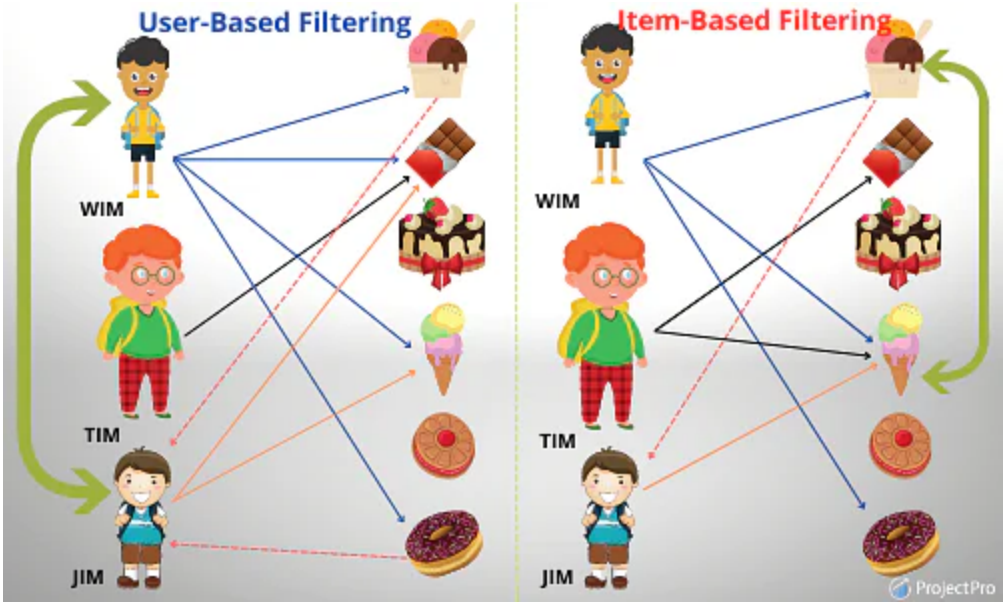
- suggest items based on a particular item
- work more solely closely with item features or attributes rather than user data
- predict the behaviour of a user based on the items they react to
- benefits:
  - the machine learning model no need the information of the user to make recommendations
  - only need to know the interests of the users which makes it easier to scale the model
  - does not have cold-start problem → it recommends movies based on their metadata and would not weigh them according to their popularity
- limitations:
  - the quality of the recommendation is dependent on the hand-engineering features (overview, genre, keywords) → requires a good amount of domain

knowledge

    - the model has absolutely no way to recommend the user something outside their range of interests
        - it is not capable of capturing tastes and generating recommendations across genres

Collaborative filtering

- depends on finding similar users to target user to make personalised recommendations
- do not require item metadata like content-based recommendations systems
- solely rely on past user-item interactions to render new recommendations
- the recommendations are totally based on a particular item
    - Exp: 2 users with very different tastes except for one common item of interest will get the same results
- solely depends on the historical preferences of a user and the interaction between a user and an item
- user preferences is studies in 2 ways
    - explicitly - this is the rating or feedback given to an item by the user explicitly
    - implicitly - the user preferences is noted in terms of clicks, views, hovering time, etc
- types
    - memory-based collaborative filtering (neighbourhood-based filtering)
        - past interactions between a user and item are stored in user-items interaction matrix
        - the recommendations are made either in a user-based or item-based fashion



- types
  - user-based collaborative filtering
    - aims at finding similar users and recommendations are made to a user based on what a similar user has liked
    - the similarity between users is calculated using either Cosine Similarity or Pearson Correlation

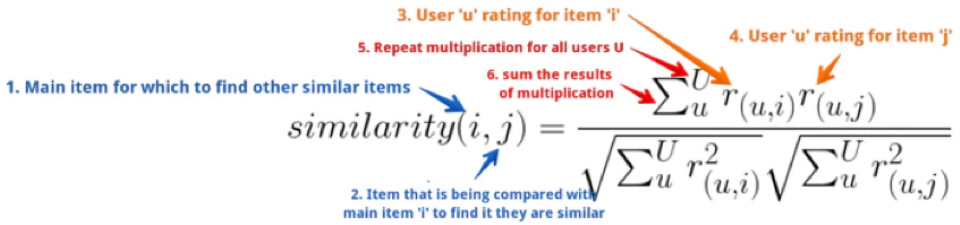
	Product A	Product B	Product C	Product D	Similarity(i, B)
User A	4.		2.	5.	0.75
User B	5.		4.	4.	1.
User C		1.	4.		0.
User D		3.			MA.
User E	4.	5.	4.	4.	0.

User B is most similar to User E so the rating that User B would give to Product B would be close to 5

- item based collaborative filtering
  - help to avoid the dynamic preference problem posed by user-based filtering

	Product A	Product B	Product C	Product D	Product E
User A	4.		2.	5.	
User B	5.	3.	4.	4.	
User C			4.		
User D	4.	3.			
User E	4.	5.	4.	4.	
•					

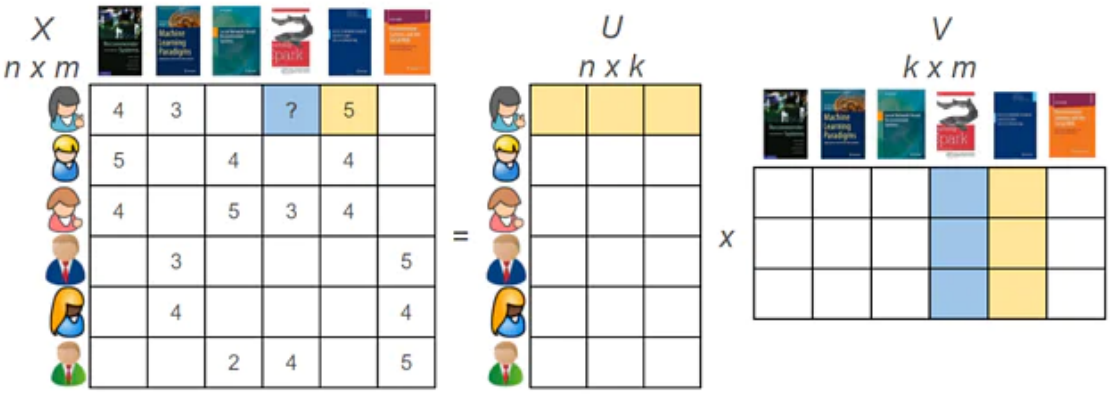
  - steps to calculate the similarity between Product A and Product B
    - multiply all the ratings the users have given to these products and sum of the results
    - square the ratings of Product A and B separately and take the sum of these squares
    - other products are removed since they are not related
    - take the square roots of the sum of these squares for A and B and multiply them



> 💡 The similarity between Product A and B is 0.83

- limitations
  - do not scale easily, even with 24k distinct users it took a while to find similar users to a single user
  - when using huge datasets, the memory-based collaborative filtering technique is close to impractical
  - has the “cold-start” problem, it’s hard to get recommendations for new users or recommend new products

- model-based collaborative filtering
  - the user-item interaction matrix is very sparse, to use it more efficiently → reduce or compress the user-items interaction matrix into 2 matrices using a model which are user-factor matrix (has user representations) and item-factor matrix (has item representation) by using Matrix Factorisation Techniques for Recommender Systems



- latent factor (represent a property or concept that a user or an item might have) captures the similarity between users and items
- do not hand engineer these features → let the model discover these hidden representations of user interests and item attributes
- advantages
    - easy to scale and can be used to work on super large datasets
    - overfitting can be avoided if the data on which we have trained is representative of the general population
    - the prediction speed is much faster than memory-based models- since you only query the model, not the whole dataset
- disadvantages
    - the quality of predictions is solely dependent on the quality of the model built
    - it is not very intuitive especially if you use Deep Learning models.

Term frequency - Inverse Document Frequency

- term frequency - it is the relative frequency of any word in a document and is given by dividing term instances with total instances
- inverse document frequency - is the relative count of documents containing the term and is given as a log (number of documents/documents with the term)
- useful in reducing the importance of words that occur frequently in our soup of movie description, genre, and keyword and would, in turn, reduce their significance in computing the final similarity score