# UNIT-5

**Recommendation Systems:** Introduction, A Model for Recommendation Systems, Collaborative Filtering System and Content Based Recommendations.

**Q) What is Recommender System? How it is useful to its users?**

Recommender systems are the systems that are designed to recommend things to the user based on various factors. It takes the user model consisting of ratings, preferences, demographics, etc. and items with its descriptions as input, finds relevant score which is used for ranking, and finally recommends items that are relevant to the user.

Recommendations are commonly seen in e-commerce systems, LinkedIn, friend recommendation on Facebook, song recommendation at FM, news recommendations at Forbes.com, etc.. Companies like Netflix, Amazon, etc. use recommender systems to help their users to identify the correct product or movies for them.

The recommender system deals with a large volume of information present by filtering the most important information based on the data provided by a user and other factors that take care of the user's preference and interest. It finds out the match between user and item and imputes the similarities between users and items for recommendation.

**Q) Explain a model for Recommendation System.**

Recommendation system is a facility that involves predicting user responses to options in web applications.

Generally we see the following recommendations:
1. "You may also like these…", "People who liked this also liked…".
2. If you download presentations from slideshare, it says "similar content you can save and browselater".

**Use-Cases Of Recommendation System**
There are many use-cases of it. Some are
A. **Personalized Content:** Helps to Improve the on-site experience by creating dynamic recommendations for different kinds of audiences like Netflix does.
B. **Better Product search experience**: Helps to categories the product based on their features. Eg: Material, Season, etc.

These suggestions are from the recommender system. The models used are as follows:

1. **Collaborative-filtering system:** It uses community data from peer groups for recommendations. This exhibits all those things that are popular among the peers.
Collaborative filtering systems recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users (community data).
**Eg.**
When we shop on Amazon it recommends new products saying "Customer who brought this also brought".

### 1.1 User-Based Collaborative Filtering

It is based on the notion of users' similarity.
Eg. On the left side, you can see a picture where 3 children named A, B, C, and 4 fruits i.e, grapes, strawberry, watermelon, and orange respectively.

Based on the image let assume A purchased all 4 fruits, B purchased only strawberry and C purchased strawberry as well as watermelon. **Here A & C are similar kinds of users because of this C will be recommended Grapes and Orange as shown in dotted line.**

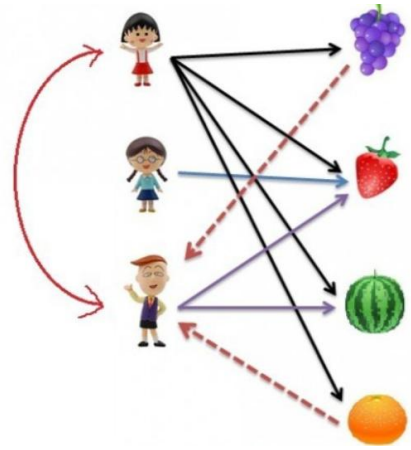


Fig. User Based Filtering

### 1.2 Item-Based Collaborative Filtering

It is based on the notion of item similarity.

Eg. Here the only difference is that we see similar items, not similar users like if you see grapes and watermelon you will realize that **watermelon is purchased by all of them but grapes are purchased by Children A& B. Hence Children C is being recommended grapes.**

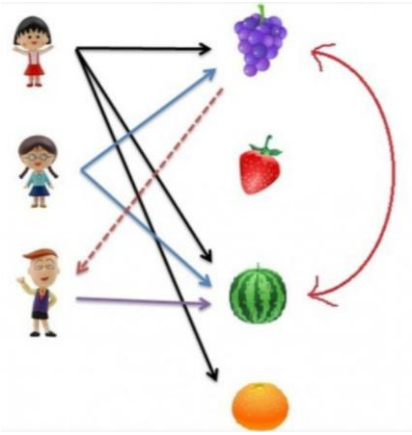


Fig. Item Based Filtering

If number of items is greater than number of users go with user-based collaborative filtering as it will reduce the computation power and If number of users is greater than number of items go with item-based collaborative filtering.

***Advantages***

- It works well even if the data is small.
- This model helps the users to discover a new interest in a given item but the model might still recommend it because similar users are interested in that item.
- No need for Domain Knowledge

***Disadvantages***

- It cannot handle new items because the model doesn't get trained on the newly added items in the database. This problem is known as **Cold Start Problem.**
- The **sparsity problem** occurs when there is too little information to work with, in order to provide the user base with decent approximations as to which products they would likely prefer.
- A **gray sheep** is a user which has no obvious closest neighbour and seems to rate content in an unpredictable pattern unlike that of any other user.

2. **Content-based systems:** In content based recommendation system, relevant items are shown using the content of the previously searched items by the users. **Here content refers to the attribute/tag of the product that the user likes**.

Eg. 1. If a user has watched many "scientific fiction" movies, then the recommender system will recommend movie classified in the database as having the "scientific fiction" genre.

2. If a user has read a book, then the recommender system will recommend other similar books.

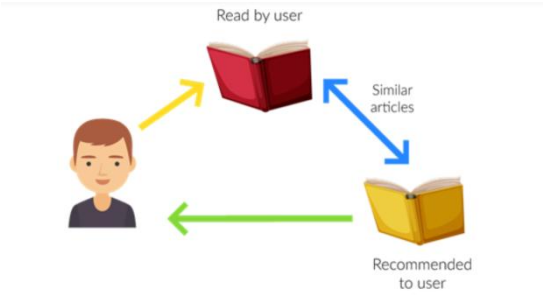


Fig. Content Based Filtering

***Advantages***

- Model doesn't need data of other users since recommendations are specific to a single user.
- It makes it easier to scale to a large number of users.
- The model can capture the specific Interests of the user and can recommend items that very few other users are interested in.

***Disadvantages***

- Feature representation of items is hand-engineered to some extent; this tech requires a lot of domain knowledge.
- The model can only make recommendations based on the existing interest of a user.

Recommendation System can be implemented in 2 ways:

1. Memory based: We use entire user-item data set to generate a Recommendation System.
   It uses statistical techniques to approximate users or items.
   Eg. Cosine similarity, Pearson Coefficient, Euclidian distance etc.

2. Model based: A model of users is developed in an attempt to learn their preferences.
   Models can be created using ML techniques like regression, clustering, classification etc.

**Q) Explain NearestNeighbor Technique for Collaborative filtering with a suitable example.**

**Nearest Neighbor Technique**

The idea is to predict the rating an "Active User" would give for an unseen item. The first step is to select a set of users (peers) who liked the same items as the "Active User" in the past and rated them too.

In order to predict the Active User's rating for the unseen item, we can use the average rating given by the peers for that unseen item.

**i. Steps for User-Based Collaborative Filtering:**

1. **Finding the similarity of users to the target user U**

Using Pearson's Correlation in peer-based collaborative filtering, the formula turns out to be:

$$Sim(a,b) = \frac{\sum_p (r_{ap} - \bar{r}_a)(r_{ab} - \bar{r}_b)}{\sqrt{\sum (r_{ap} - \bar{r}_a)^2} \sqrt{\sum (r_{bp} - \bar{r}_b)^2}}$$

$$r_{up} : rating\ of\ user\ u\ against\ item\ p$$

$$p : items$$

where "*a*" and "*b*" are users; $r_a$, *p* is the rating of user "*a*" for item "*p*"; "*P*" is the set of items that are rated by both "*a*" and "*b*".

2. **Prediction of missing rating of an item using Nearest Neighbor technique:**

Similarity measure lies in the range of −1 and 1, where −1 indicates very dissimilar and 1 indicates perfect similarity. A common prediction function using the above similarity function is

$$pred(a,p) = \bar{r}_a + \frac{\sum_{b \in N} sim(a,b) \times (r_{b,p} - \bar{r}_b)}{\sum_{b \in N} sim(a,b)}$$

Using this, we can calculate whether the neighbors' ratings for the unseen item *i* are higher or lower than their average, then combine the rating differences using the similarity as a weight.
Finally the neighbor's bias is added to or subtracted from the Active User's average and used as a prediction.
**Eg.**

Consider a matrix which shows four users *Alice, U1, U2 and U3* rating on different news apps. The rating range is from 1 to 5 on the basis of users likability of the news app. The *'?'* indicates that the app has not been rated by the user.

| Name | Inshorts(I1) | HT(I2) | NYT(I3) | TOI(I4) | BBC(I5) |
|---|---|---|---|---|---|
| Alice | 5 | 4 | 1 | 4 | ? |
| U1 | 3 | 1 | 2 | 3 | 3 |
| U2 | 4 | 3 | 4 | 3 | 5 |
| U3 | 3 | 3 | 1 | 5 | 4 |

Fig. Dataset

**Calculating the similarity between Alice and all the other users**
At first we calculate the averages of the ratings of all the user excluding I5 as it is not rated by Alice. Therefore, we calculate the average as:

$$\bar{r}_i = \frac{\sum_p r_{ip}}{\sum p}$$

Therefore, we have

$$\bar{r}_{Alice} = 3.5$$
$$\bar{r}_{U1} = 2.25$$
$$\bar{r}_{U2} = 3.5$$
$$\bar{r}_{U3} = 3$$

and calculate the new ratings as,

$$r'_{ip} = r_{ip} - \bar{r}_i$$

Hence we get the following matrix:

| Name | Inshorts(I1) | HT(I2) | NYT(I3) | TOI(I4) |
|---|---|---|---|---|
| Alice | 1.5 | 0.5 | -2.5 | 0.5 |
| U1 | 0.75 | -1.25 | -0.25 | 0.75 |
| U2 | 0.5 | -0.5 | 0.5 | -0.5 |
| U3 | 0 | 0 | -2 | 2 |

Now, we calculate the **similarity between Alice and all the other users**:

$$Sim(Alice, U1) = \frac{((1.5*0.75)+(0.5*-1.25)+(-2.5*-0.25)+(.5*0.75))}{\sqrt{(1.5^2+0.5^2+2.5^2+0.5^2)}\sqrt{(0.75^2+1.25^2+0.25^2+0.75^2)}} = 0.301$$

$$Sim(Alice, U2) = \frac{((1.5*0.25)+(0.5*-0.5)+(-2.5*0.5)+(.5*-0.5))}{\sqrt{(1.5^2+0.5^2+2.5^2+0.5^2)}\sqrt{(0.5^2+0.5^2+0.5^2+0.5^2)}} = -0.33$$

$$Sim(Alice, U3) = \frac{((1.5*0)+(0.5*0)+(-2.5*-2)+(.5*2))}{\sqrt{(1.5^2+0.5^2+2.5^2+0.5^2)}\sqrt{(0^2+0^2+2^2+2^2)}} = 0.707$$

**Predicting the rating of I5 by Alice:**

$$r_{(Alice,I5)} = \bar{r}_{Alice} + \frac{(sim(Alice,U1)*(r_{U1,I5}-\bar{r}_{U1}))+(sim(Alice,U2)*(r_{U2,I5}-\bar{r}_{U2}))+(sim(Alice,U3)*(r_{U3,I5}-\bar{r}_{U3})}{|r_{U1,I5}|+|r_{U2,I5}|+|r_{U3,I5}|}$$

$$r_{(Alice,I5)} = 3.5 + \frac{(0.301*0.75)+(-0.33*1.5)+(0.707*1)}{|0.301|+|-0.33|+|0.707|} = 3.83$$

**If we assume the threshold rating as 3.5, we recommend I5 to Alice as the predicted rating is 3.83.**

But the problem arises when it is a cold start. How to recommend new items? What to recommend to the new users? In the initial phase, the user can be requested to rate a set of items. Otherwise either demographic data or non-personalized data can be used.

**ii. Steps for Item-Based Collaborative Filtering:**

1. To build the model by finding similarity between all the item pairs. The similarity between item pairs can be found in different ways. One of the most common methods is to use cosine similarity.

   Formula for Cosine Similarity:

$$Similarity(\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{||\vec{A}|| * ||\vec{B}||}$$

**2.** Executing a recommendation system. It uses the items (already rated by the user) that are most similar to the missing item to generate rating.

$$rating(U, I_i) = \frac{\sum_j rating(U,I_j)*s_{ij}}{\sum_j s_{ij}}$$

**Eg.**

Given below is a set table that contains some items and the user who have rated those items. The rating is explicit and is on a scale of 1 to 5. Each entry in the table denotes the rating given by ai$^{th}$ User to a j$^{th}$ Item. We need to find the missing ratings for the respective user.

| User/Item | Item_1 | Item_2 | Item_3 |
|---|---|---|---|
| User_1 | 2 | - | 3 |
| User_2 | 5 | 2 | - |
| User_3 | 3 | 3 | 1 |
| User_4 | - | 2 | 2 |

Fig. Dataset

1. **Finding similarities of all the item pairs.**

Sim(Item1, Item2):

In the table, we can see only User_2 and User_3 have rated for both items 1 and 2.

Thus, let I1 be vector for Item_1 and I2 be for Item_2. Then,

I1 = 5U2 + 3U3 and,

I2 = 2U2 + 3U3

$$Similarity(I1, I2) = \frac{(5*2)+(3*3)}{\sqrt{5^2+3^2}\sqrt{2^2+3^2}} = 0.90$$

Sim(Item2, Item3):

In the table we can see only User_3 and User_4 have rated for both the items 1 and 2.

Thus, let I2 be vector for Item_2 and I3 be for Item_3. Then,

I2 = 3U3 + 2U4 and,

I3 = 1U3 + 2U4

$$Similarity(I2, I3) = \frac{(3*1)+(2*2)}{\sqrt{3^2+2^2}\sqrt{1^2+2^2}} = 0.869$$

Sim(Item1, Item3):

In the table we can see only User_1 and User_3 have rated for both the items 1 and 2.

Thus, let I1 be vector for Item_1 and I3 be for Item_3. Then,

I1 = 2U1 + 3U3 and,

I3 = 3U1 + 1U3

$$Similarity(I1, I3) = \frac{(2*3)+(3*1)}{\sqrt{2^2+3^2}\sqrt{3^2+1^2}} = 0.789$$

2. **Generating the missing ratings in the table**

$$r(U_1, I_2) = \frac{r(U_1,I_1)*s_{I_1I_2}+r(U_1,I_3)*s_{I_3I_2}}{s_{I_1I_2}+s_{I_3I_2}} = \frac{(2*0.9)+(3*0.869)}{(0.9+0.869)} = 2.49$$

$$r(U_2, I_3) = \frac{r(U_2,I_1)*s_{I_1I_3}+r(U_2,I_2)*s_{I_2I_3}}{s_{I_1I_3}+s_{I_2I_3}} = \frac{(5*0.789)+(2*0.869)}{(0.789+0.869)} = 3.43$$

$$r(U_4, I_1) = \frac{r(U_4,I_2)*s_{I_1I_2}+r(U_4,I_3)*s_{I_1I_3}}{s_{I_1I_2}+s_{I_1I_3}} = \frac{(2*0.9)+(2*0.789)}{(0.9+0.789)} = 2.0$$

**If we assume the threshold rating as 3 then we only recommend I3 to U2.**

**Q) Briefly explain about Content based Recommendation techniques.**

**i. Discovering Features of Documents:**

Document collections and images are other classes of items where the values of features are not immediately apparent. There are many kinds of documents for which a recommendation system can prove to be useful.

Eg. Web pages are also a collection of documents. For example, there are many news articles published each day, and all cannot be read. A

recommendation system can suggest articles on topics a user is interested in.

To measure the similarity of the two documents, there are several natural distance measures we use:

1. The Jaccard distance between the sets of words.

$$J(\mathrm{x},\mathrm{y}) = \frac{|\mathrm{x} \cap \mathrm{y}|}{|\mathrm{x} \cup \mathrm{y}|}$$

**Eg.**
doc_1 = "Data is the new oil of the digital economy"
doc_2 = "Data is a new oil"

Let's get the set of unique words for each document.

words_doc1 = {'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'}
words_doc2 = {'data', 'is', 'a', 'new', 'oil'}

Now, we will calculate the intersection and union of these two sets of words and measure the Jaccard Similarity between doc_1 and doc_2.

$$J(doc_1, doc_2) = \frac{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cap \{'data', 'is', 'a', 'new', 'oil'\}}{\{'data', 'is', 'the', 'new', 'oil', 'of', 'digital', 'economy'\} \cup \{'data', 'is', 'a', 'new', 'oil'\}}$$

$$= \frac{\{'data', 'is', 'new', 'oil'\}}{\{'data', 'a', 'of', 'is', 'economy', 'the', 'new', 'digital', 'oil'\}}$$

$$= \frac{4}{9} = 0.444$$

**2.** The cosine distance between the sets, treated as vectors.

**Cosine Similarity:**

Suppose that $\boldsymbol{x}$ and $\boldsymbol{y}$ aretwo term-frequency vectors:

$$sim(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}||||\boldsymbol{y}||},$$

where $||\boldsymbol{x}||$ is the Euclidean norm of vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$, defined as $\sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}$. Conceptually, it is the length of the vector. Similarly, $||\boldsymbol{y}||$ is the Euclidean norm of vector $\boldsymbol{y}$. The

**Eg.**
doc_1 = "Data is the oil of the digital economy"
doc_2 = "Data is a new oil"

# Vector representation of the document
doc_1_vector = [1, 1, 1, 1, 0, 1, 1, 2]
doc_2_vector = [1, 0, 0, 1, 1, 0, 1, 0]

| | data | digital | economy | is | new | of | oil | the |
|---|---|---|---|---|---|---|---|---|
| doc_1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 2 |
| doc_2 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$

$$= (1 * 1) + (1 * 0) + (1 * 0) + (1 * 1) + (0 * 1) + (1 * 0) + (1 * 1) + (2 * 0)$$

$$= 3$$

$$\sqrt{\sum_{i=1}^{n} A_i^2} = \sqrt{1 + 1 + 1 + 1 + 0 + 1 + 1 + 4} = \sqrt{10}$$

$$\sqrt{\sum_{i=1}^{n} B_i^2} = \sqrt{1 + 0 + 0 + 1 + 1 + 0 + 1 + 0} = \sqrt{4}$$

$$cosine\ similarity = cos\theta = \frac{A \cdot B}{|A||B|} = \frac{3}{\sqrt{10} * \sqrt{4}} = 0.4743$$

The following two kinds of document similarity exist:
**1. Lexical similarity:** Documents are similar if they contain large, identical sequences of characters.
**2. Semantic similarity:** Semantic similarity is defined over a set of documents or terms, where the idea of distance between them is based on the likeness of their meaning or semantic content.

Semantic similarity can be estimated by defining a topological similarity using ontologies to define the distance between terms/concepts. For example, a partially ordered set, represented as nodes of a directed acyclic graph, is used for the comparison of concepts ordered. Based on text analyses, semantic relatedness between units would be the shortest path linking the two concept nodes. This can also be estimated using statistical means, such as a vector space model, to correlate words and textual contexts from a suitable text corpus.

For recommendation systems, the notion of similarity is different. We are interested only in the occurrences of many important words in both documents, even if there is little lexical similarity between the documents.

**Item Profile:**
In a content-based system, each item is a profile, which is a record or a collection of records representing important characteristics of that item, is first constructed. For example, for a movie recommendation system, the important characteristics are:
**1.** The set of actors of the movie.
**2.** The director.
**3.** The year in which the movie was made.
**4.** The genre or general type of movie, and so on.

The objective of content-based recommendation systems is to find and rank things (documents) according to the user preferences.

To find similarity between the items, Dice co-efficient given below is used. Let $b1$, $b2$ be two items. Similarity between the two is given by

$$\text{sim}(b_1, b_2) = (2*((\text{keywords}(b_1) \cap (\text{keywords}(b_2))) / ((\text{keywords}(b_1) + (\text{keywords}(b_2))$$

This approach has the basic assumptions that all keywords are of equal importance.

### ii. Term Frequency–Inverse Document Frequency

**TF-IDF** stands for **"*Term Frequency – Inverse Document Frequency*"**. TF-IDF is a numerical statistic which measures the importance of the word in a document.

- **Term Frequency:** Number of time a word appears in a text document.
- **Inverse Document Frequency:** Measure the word is a rare word or common word in a document.

**tf(t,d) = (Number of times term t appears in a document) / (Total number of terms in the document)**

Where,
tf(t,d) - Term Frequency, t = term, d = document

**idf(t) = log [ n / df(t) ] + 1**

where,
idf(t) - Inverse Document Frequency
n - Total number of documents
df(t) is the document frequency of term t;

**tf-idf(t, d) = tf(t, d) * idf(t)**

**Eg.**

Consider a document which has a total of 100 words and the word **"book"** has occurred 5 times in a document.

**Term frequency (tf)** = 5 / 100 = 0.05

Let's assume we have 10,000 documents and the word **"book"** has occurred in 1000 of these. Then idf is:

**Inverse Document Frequency(IDF)** = log[10000/1000] + 1 = 2

**TF-IDF** = 0.05 * 2 = 0.1

### iii. *Obtaining Item Features from Tags*

Consider that images of features have been obtained for items. The problem with images is that their data, which is an array of pixels, does not tell us anything useful about their features. In order to obtain information about features of items, request users to tag the items by entering words or phrases that describe the item.

The problem with tagging as an approach to feature discovery is that the process only works if users are willing to take the trouble to create the tags, and the number of erroneous tags are minimal/negligible when compared to total number of tags.

**Eg.**
To compute the cosine distance between vectors, consider the movie recommendation system. Suppose the only features of movies are the set of actors and the average rating. Consider two movies with five actors each. Two of the actors are in both movies. Also, one movie has an average rating of 3 andthe other has an average of 4. The vectors look something like
0 1 1 0 1 1 0 1 3α
1 1 0 1 0 1 1 0 4α

The last component represents the average rating with an unknown scaling factor α. Computing in terms of α, the cosine of the angle between the vectors is

$$\frac{2+12\alpha^2}{\sqrt{25 + 125\alpha^2 + 144\alpha^4}}$$

where the dot product is $2 + 12\alpha^2$ and the length of the vectors are $\sqrt{5+9\alpha^2}$ and $\sqrt{5+16\alpha^2}$.

For α = 1, the cosine is 0.816.
For α = 2, the cosine is 0.940, the vectors are closer in direction than if we use α = 1.

### iv. *User Profiles*

To consider the user profiles, vectors with the same components that describe user's preferences are created.
The utility matrix represents the connection between the users and the items. The entries in the utility matrix to represent user purchases or a similar connection could be just 1s, or they could be any number representing a rating or liking that the user has for the item.

This information can be used to find the best estimate regarding which items the user likes. This is taken as aggregation of the profiles of those items.

If the utility matrix has only 1s, then the natural aggregate is the average of the components of the vectors representing the item profiles for the items in which the utility matrix has 1 for that user.
**Eg.**
Suppose items are movies, represented by Boolean profiles with components corresponding to actors.

If 25% of the movies that user U likes have Tom Hanks as one of the actors, then the user profile for U will have 0.25 in the component for Tom Hanks.

If the utility matrix has ratings 1–5, then we can weigh the vectors representing the profiles of items by the utility value. The utilities are normalized by subtracting the average value for a user.
**Eg.**
User U gives rating for three movies as 1, 3 and 4. The user profile for U has, in the component will be

Average(3) of 1 − 3, 3 − 3 and 4 − 3, that is, the value −1/3. Therefore, items with a below-average rating get negative weights, and items with above-average ratings get positive weights.

The vector for a user will have positive numbers for actors who appear in movies the user likes and have negative numbers for actors appearing in movies the user does not like.
**1. Case 1:** Consider a movie with many actors the user likes, and only a few or none that the user does not like. The cosine of the angle between the user's and movie's vectors will be a large positive fraction. That implies an angle close to 0, and therefore a small cosine distance between the vectors.
**2. Case 2:** Consider a movie with as many actors that the user likes as those the user does not like. In this situation, the cosine of the angle between the user and movie is around 0, and therefore the angle between the two vectors is around 90°.
**3. Case 3:** Consider a movie with mostly actors the user does not like. In that case, the cosine will be a large negative fraction, and the angle between the two vectors will be close to 180°– the maximum possible cosine distance.

### *v. Classification Algorithms*

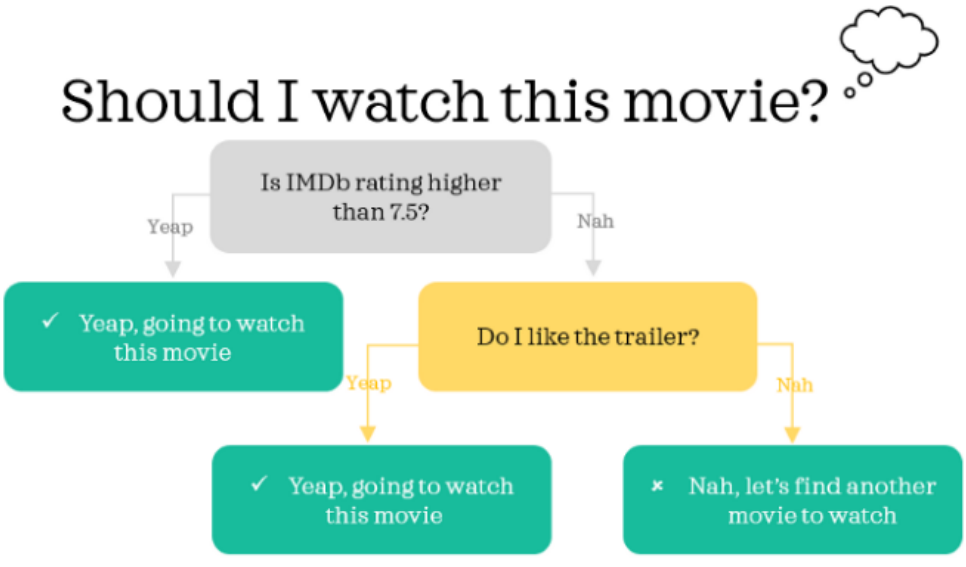
Another approach to a recommendation system is to treat this as the machine learning problem using item profiles and utility matrices.
The given data is taken as a training set, and for each user, classifier that predicts the rating of all items is built.

There are a number of different classifiers, and the most common one is the decision trees.

A decision tree is a collection of nodes, arranged as a binary tree. The leaves render decisions; in movie recommendation system, the decision would be "likes" or "does not like". Each interior node is a condition on the objects being classified; here the condition would be a predicate involving one or more features of an item.

To classify an item, starting from the root, the predicate is applied from the root to the item. If the predicate is true, the path to the left (child) is taken, and if it is false, the path to right (child) is taken. Then the same process is repeated at all the nodes visited, until a leaf is reached. It is that leaf which classifies the item as liked or not.

After selecting a predicate for a node *N*, the items are divided into two groups: one that satisfies the predicate and the other that does not. For each group, the predicate that best separates the positive and negative examples in that group is then found. These predicates are assigned to the children of *N*. This process of dividing the examples and building children can proceed to any number of levels.




Decision tree built on past experience to assess whether to watch a particular movie

**Q) Differentiate User based and Item based collaborative filtering Recommendation System.**

Collaborative filtering works by finding out similarities between two users or two items.

User based: Recommend items by finding similar users.

**Eg.** If you have seen 10 movies and 7 out of those have been seen by someone else too, that would imply that you both have similar taste and you'd be recommended movies that he watched and you haven't.

Item based: Calculate similarity between items and make recommendations.

**Eg.** If you gave high rating to a restaurant then you would be recommended other restaurants rated highly by people who rated this restaurant highly too.