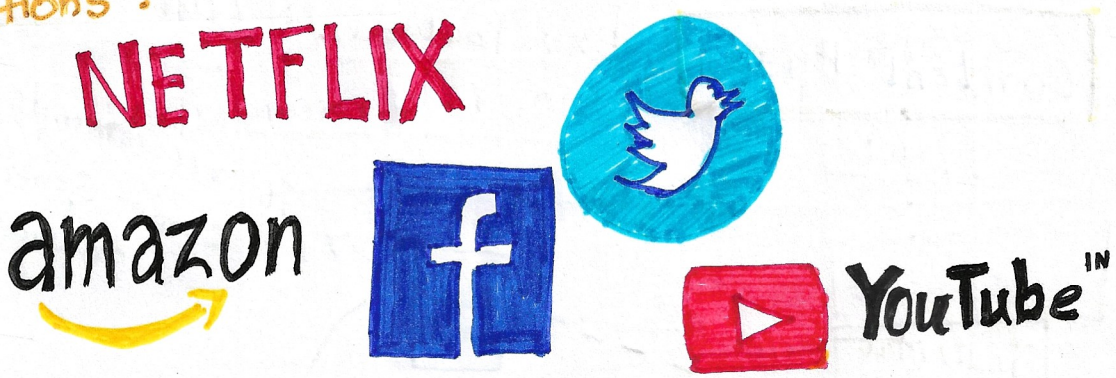
* Recommendation Systems

↓

Recommendation systems Capture The patterns of People's behavior and use it To Predict what else They might Want or like.

Q: Where we can See "These recommendation System"?

Applications :



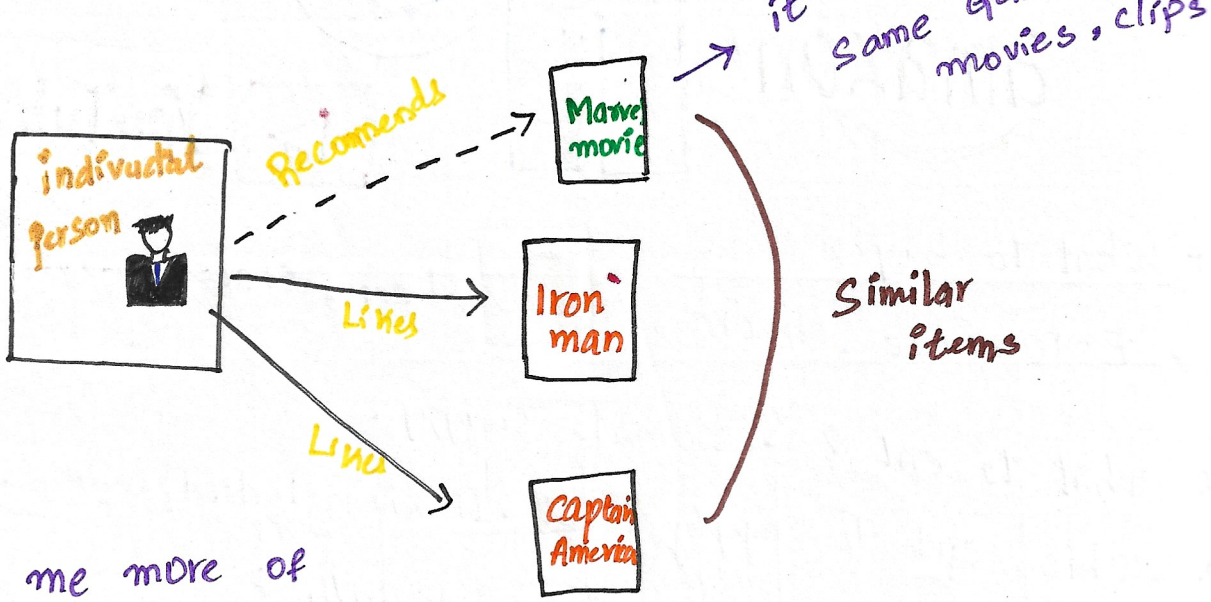
* What to buy ?
  - E-Commerce, books, movies, beer, shoes .....
* what to eat ? - Zomato, swiggy .....
* which job To apply To ? - linkdin, Indeed, Naukri ....
* who you should be friends with ?
  - Linkdin, Facebook .....
* Personalize Your Experience on The web
  - News platforms, News personalizations.

* Advantages of recommender systems

1. Broader exposure → Suggestion of broader (viewing range. time)
2. Possibility of Continual usuage
   (or)
   Purchase of products
3. Provide better Experience.

* Types of recommendations

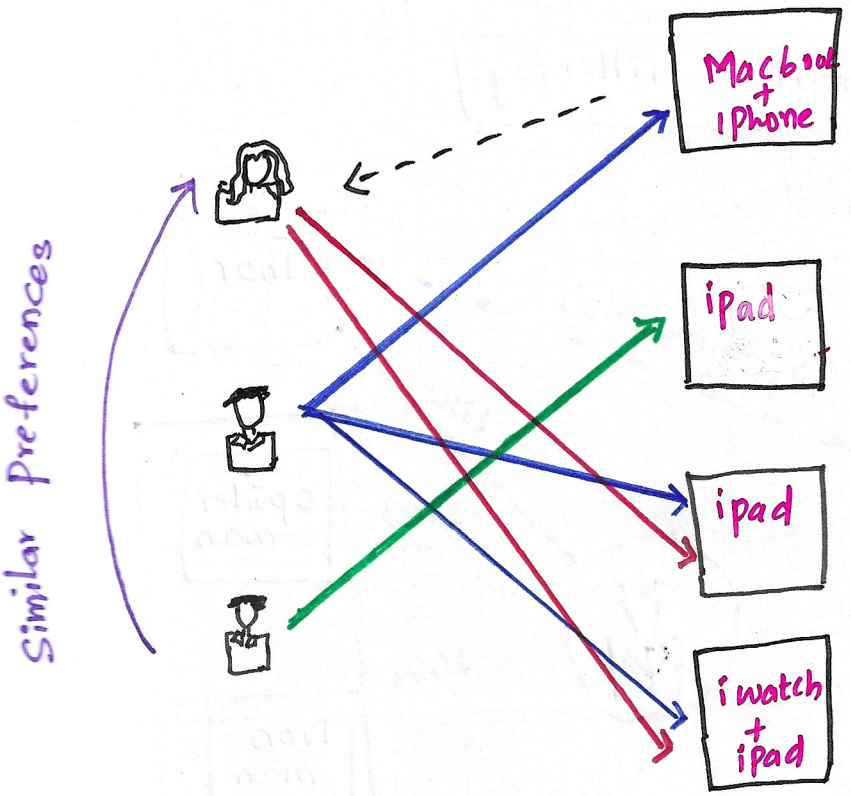
1. Content Based Ex: Youtube, Netflix



"Show me more of The same of what I've liked before."

more people joined together

2. **Collaborative filtering** : Ex :- Amazon , Flipkart



* User-based collaborative filter
  - based on user's neighborhood

* item-based collaborative filter
  - Based on item's similarity.

" Tell me what's Popular among my neigh → neighbours -bours , I also might like it "

Implementing Recommender Systems

1. Memory based :- item based → it use statistical Techniq Ex :- Pearson correlation, cosine similarity, Euclidean distance

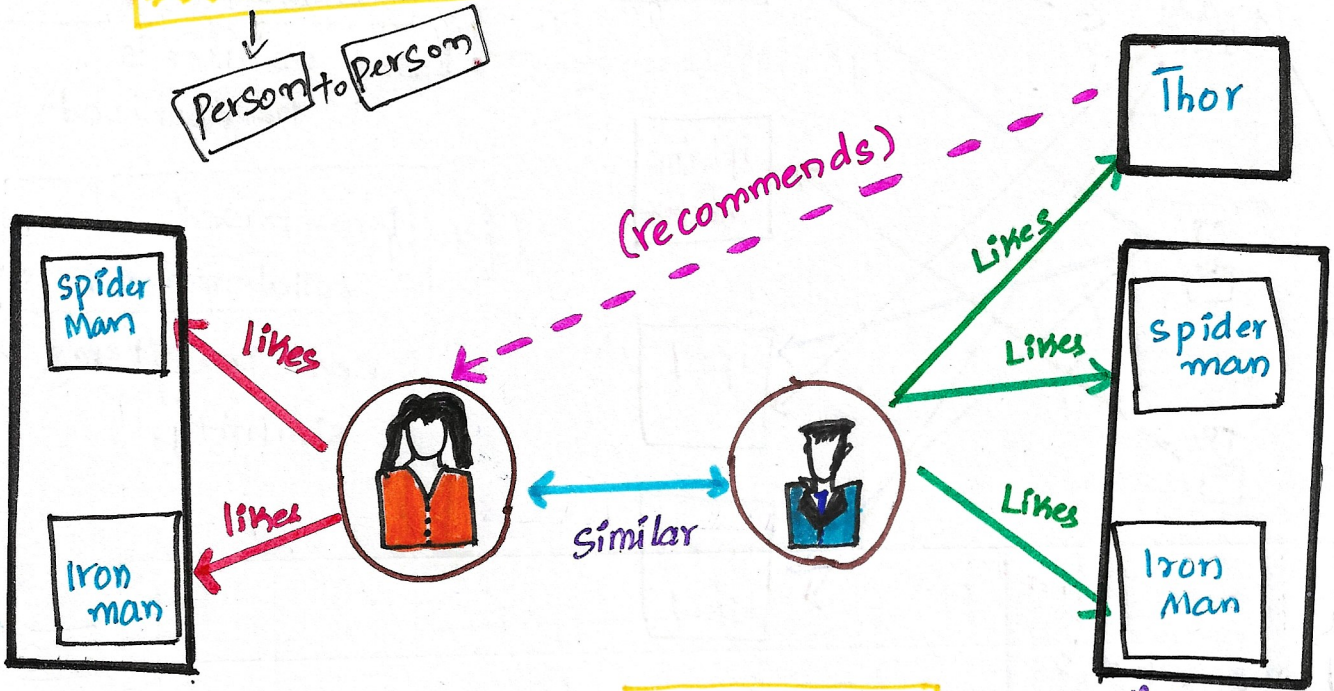
2. Model based :- Person based

models can created by using machine learning Techniques like
- regression
- classification
- clustering

* Collaborative filtering :-

1. User based collaborative filtering.

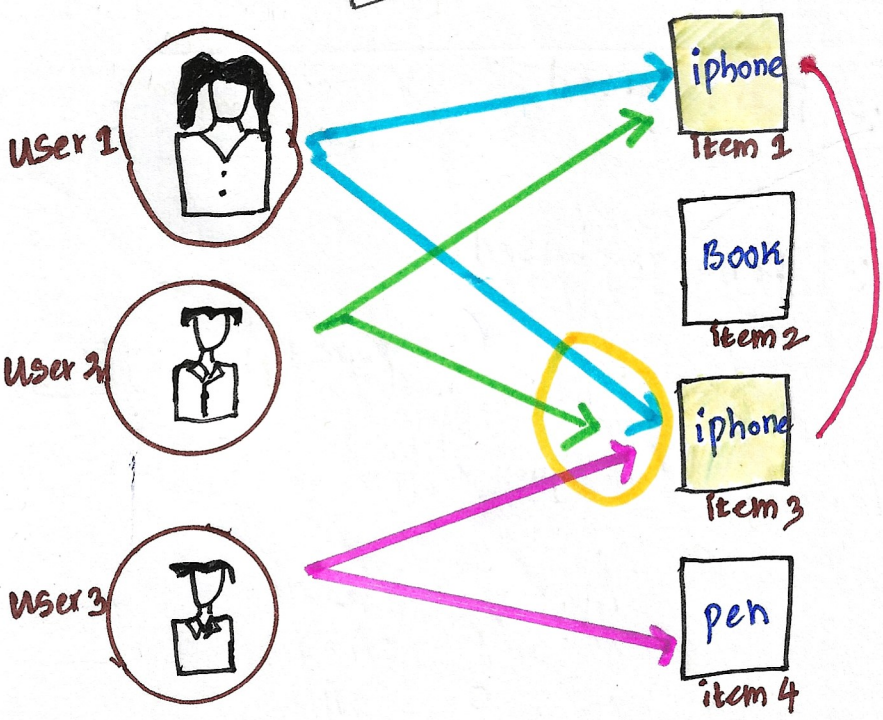
Person to Person



* it identify the similar user, with similar taste and recommend items, Movies....

2. item based collaborative filtering

item to item

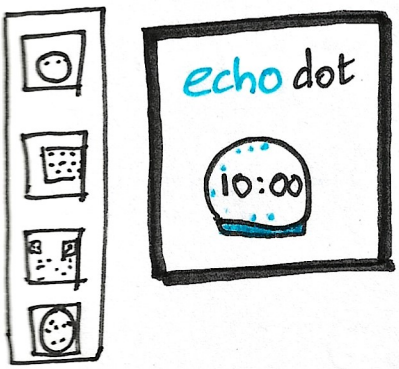


* Here it indentify "Similar items" and recommends it

* Detailed Example : * User Based collaborative

amazon.in

All

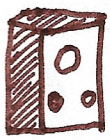


Echo dot (4th Gen. 2020 release)
generation Smart Speaker

★★★★★

₹4,999.00

Customers also bought items from Amazon Devices


Amazon smart plug
₹1,999


Echo Dot
₹3,999


Echo DOT
₹6,999


Echo show
₹24,999

* item based collaborative

amazon.in

All

Dell Vostro 3400 Laptop

★★★★☆

₹36,699.00

Featured item You may like


Dell Laptop
₹42,000.00


Dell laptop
₹60,890.00


Keyboard Protector
₹373.00


Hard drive case
₹229.00

code :- Movie Recommendation Engine.

```
import Pandas as pd
import numpy as np
```

Data collection

```
# df = pd.read_csv ("movies.csv")
# df.head ()
```

Out

Index	Genres	title
0	Action Adventure Fantasy Science Fiction	Avatar
1	Adventure Fantasy Action	Pirates of caribbean
2	Action Adventure Crime	spectre
3	Action Crime Drama Thriller	The Dark Knight rises
4	Action Adventure Science fiction	John Carter

```
# df.shape
```

Out : (4803, 3)

```
# df.isnull().sum()
```

Out :

```
index     0
Genres   28
title     0
```

Data Preprocessing

```
# replacing the null values with null (" ") string.

# df["generes"] = df["generes"].fillna("")

Out: filled with (" ")

# df.isnull().sum()

Out: index   0
     genres  0
     titles  0
```

20/6/22 5:00 pm

```
# Converting the text Data to Feature Vectors [NLP - vectorization]

from sklearn.feature_extraction.text import TfidfVectorizer

# Vectorizer = TfidfVectorizer()

# feature_vectors = vectorizer.fit_transform(df["generes"])

# print(feature_vectors)

Out:  (0,9)   0.4717
      (0,17)  0.4717
      (0,8)   0.5066
        :       :
      (4800,6) : 0.15838
      (4800,15)  1.0
```

Cosine Similarity

#Getting the Similarity Scores using Cosine Similarity

```
from sklearn.metrics.pairwise import cosine_similarity

# similarity = cosine_similarity ( feature_vectors )

# print ( similarity)
```

Out: [[1, 0.74495 , 0.42850 , 0, 0,]
[0.70428 , 1 , 0.595 , 0, 0]
[0, 0, 0, 0]
. . . . . .
[0, 0, 0, 0, 0. 1]
[0, 0. 0, 0, 0., 0.]]

```
# Print (similarity.shape)
```

Out: (4803, 4803)

Cosine similarity : $\frac{A.B}{\|A\| * \|B\|}$

$$\cos\theta = \frac{A.B}{\|A\| \times \|B\|}$$

Ex:- A [5, 1, 3, 2]
B [0, 1, 2, 3]

$$\cos\theta = \frac{(5\times0)+(1\times1)+(3\times2)(2\times3)}{\sqrt{5^2+1^2+3^2+2^2} * \sqrt{0^2+1^2+2^2\times3^2}}$$

= 0.7 3

↳ Similarity between two Vectors

cosine_distance = 1 - Cosin_similarity

Getting the movie name from the User

```
# movie_name = input (" Enter the movie name ")
```

Out: Enter the movie name : Iron man

creating a List with all the movie names given in dataset

```
# list_of_all_titles = df["title"].to_list()

# print(list_of_all_titles)
```

"Ironman" is "Ironman" in list / not we are checking

Out: ["Avatar", "Pirates of caribean", "Spectre", "Dark night Rises", ...]

finding the close match for the movie given by User

→ df[df["title"] == "ironman"] → another way

→ difference library

Given by

```
import difflib

# find_close_match = difflib.get_close_matches(movie_name,
                                                list_of_all_titles)

# print("find_close_match)
```

Out: ["IronMan", "IronMan2", "IronMan3"]

Extract 'Ironman' from above options

```
# close_match = find_close_match[0]

# print(Close_mateh)
```

Out: Ironman

finding the index of the movie with title

```
# index_of_movie = df[df.titles == close_match]["index"].
                                                Values[0]

# print(index_of_movie)
```

Out: 68

getting a list of Similar movies

```
# Similar = list (enumerate (Similarity [index_of_movie]))
```

gives numbering of Each Similar words (vectors)

```
# print (similar)
```

Out: [[(0.0.86), (1, 0.467), (2,0.497), (3,0.209)

```
# Len (similar)
```

Out: 4803

Sorting the movies based on their Similarity_score

```
# sorted_movies = sorted (similar, key = lambda x : x[1],
                          reverse = True)
# print (sorted_movies)
```

Out: [4, 1.0], [7,1.0] (16,1.0) ..(26,1.0) .....

Print the names of similary movies According to Similarity Score

```
# Print ("movies suggest for you : \n")
# i=1
for cinema in sorted_movies :
    index = cinema[0]
    index = df[a == index] ["titles"].values[0]
    if (i<=10):
        Print (i, ".", index)
        i+ = 1 → i+1 =1
```

Out: Johncarter