* 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 :

NETFLIX

amazon

YouTube[IN]

* 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.

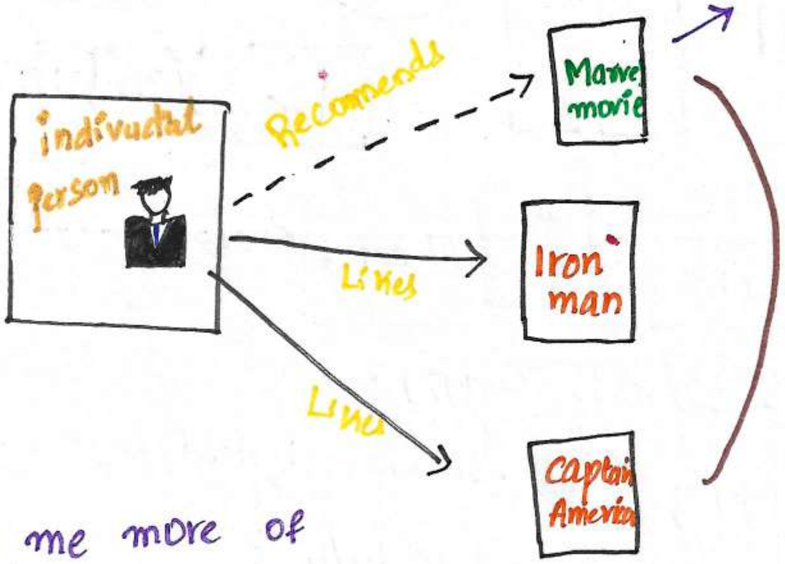
* Advantages of recommender systems *

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

* Types of recommendations *

1. Content Based   Ex: Youtube, Netflix

it recommends same Genere of movies, clips

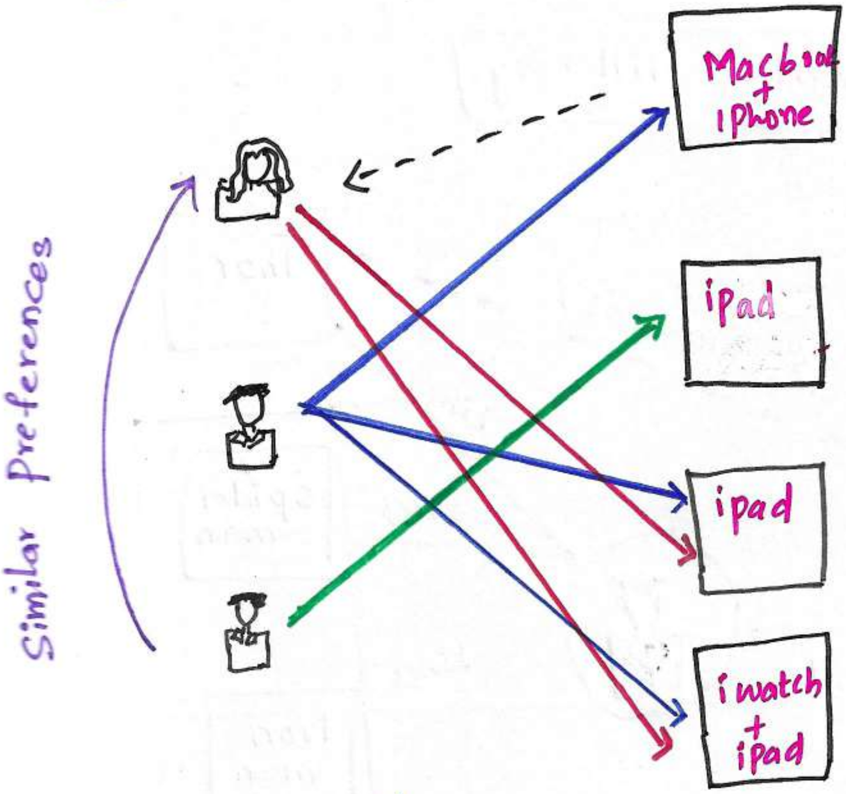


Similar items

" 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-bours, I also might like it " → neighbours

Implementing Recommender Systems

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

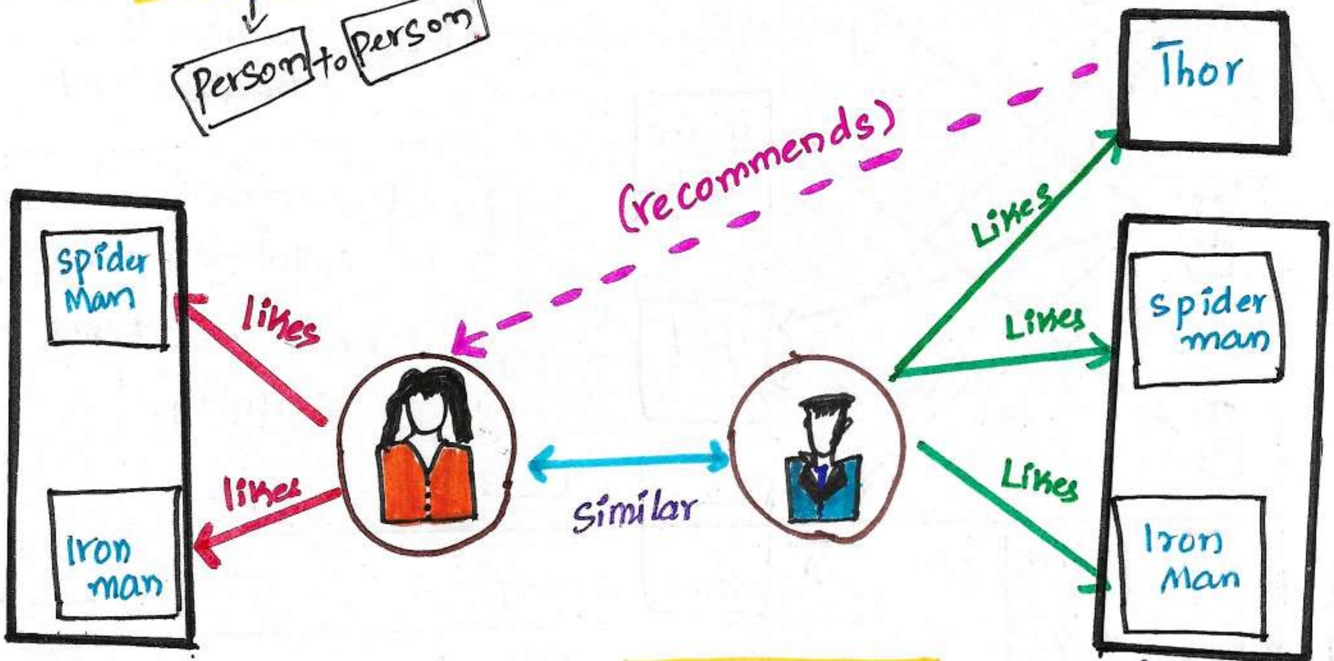
2. Model based :- person based → models can created by using machine learning Techniques like
   * regression
   * classification
   * clustering

* Collaborative filtering :-

1. **User based** collaborative filtering.

↓

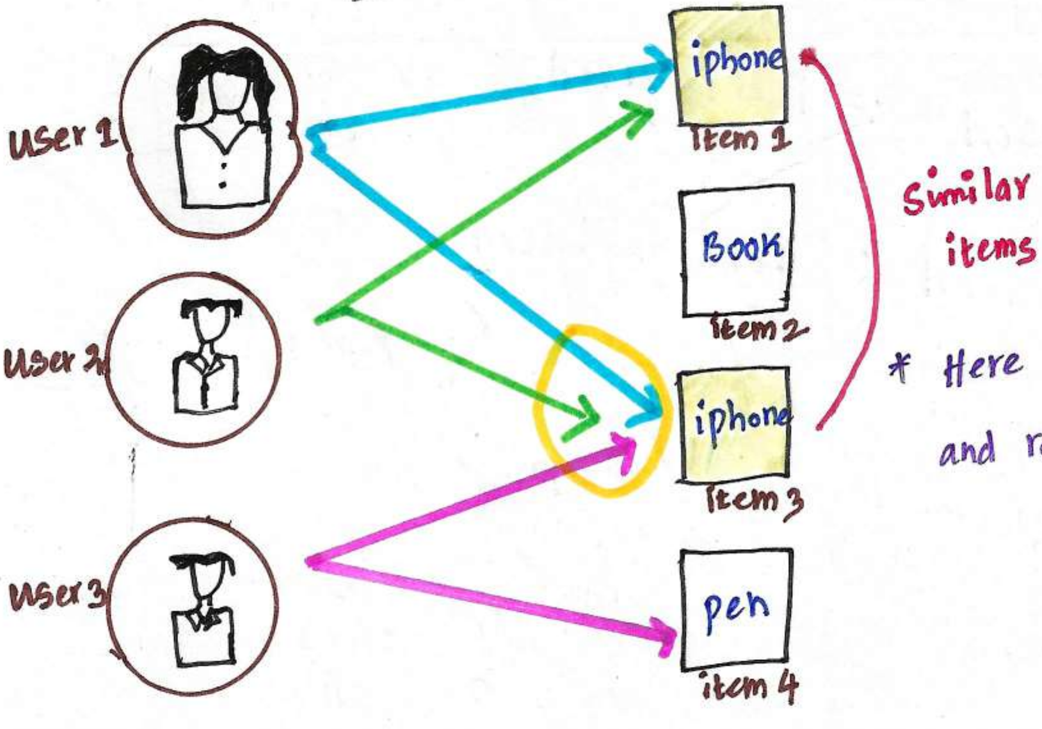
Person to Person



* it identify the Similar user, with Similar Taste and recommend items, Movies....

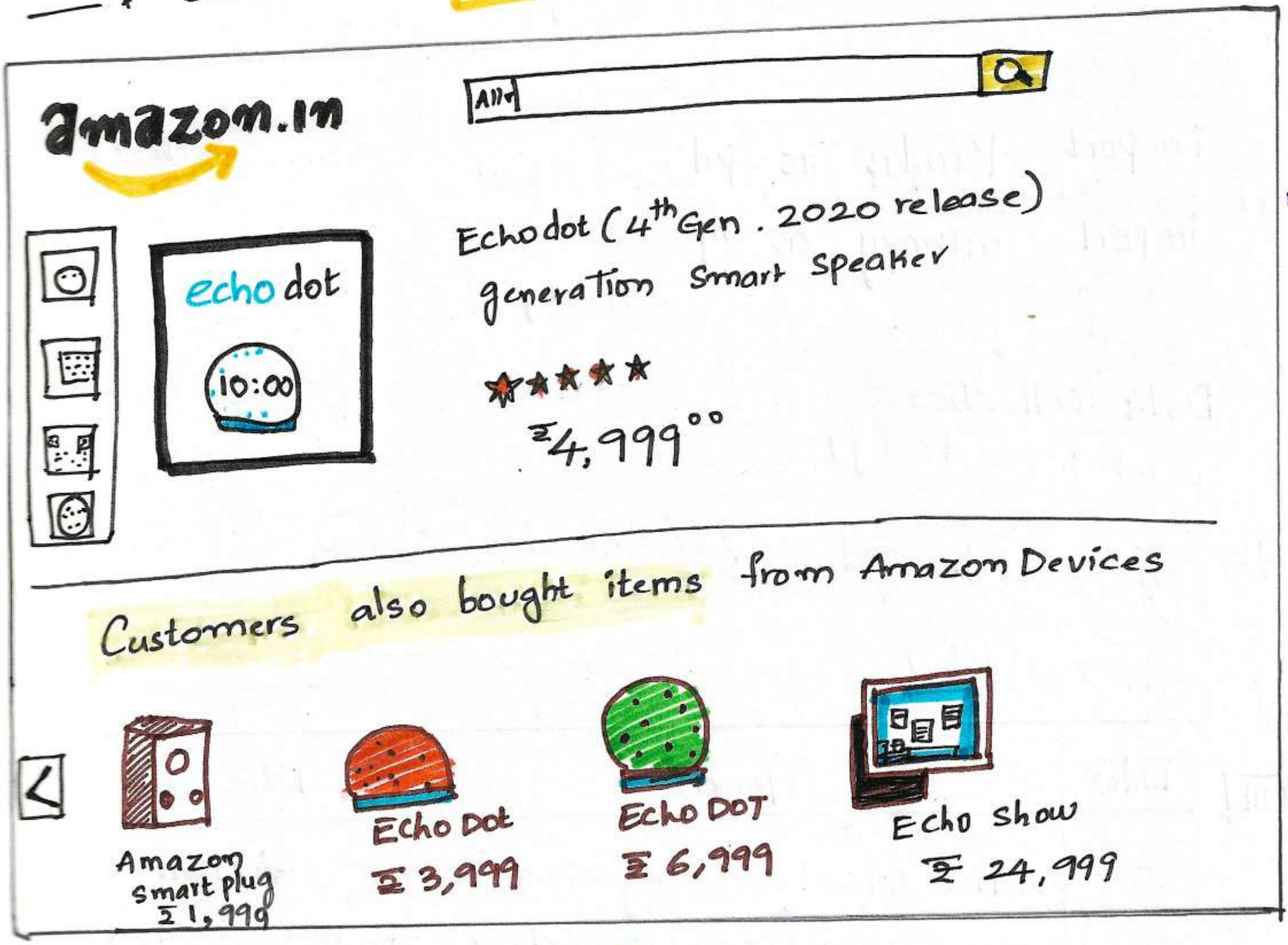
2. **item based** collaborative filtering

item to item

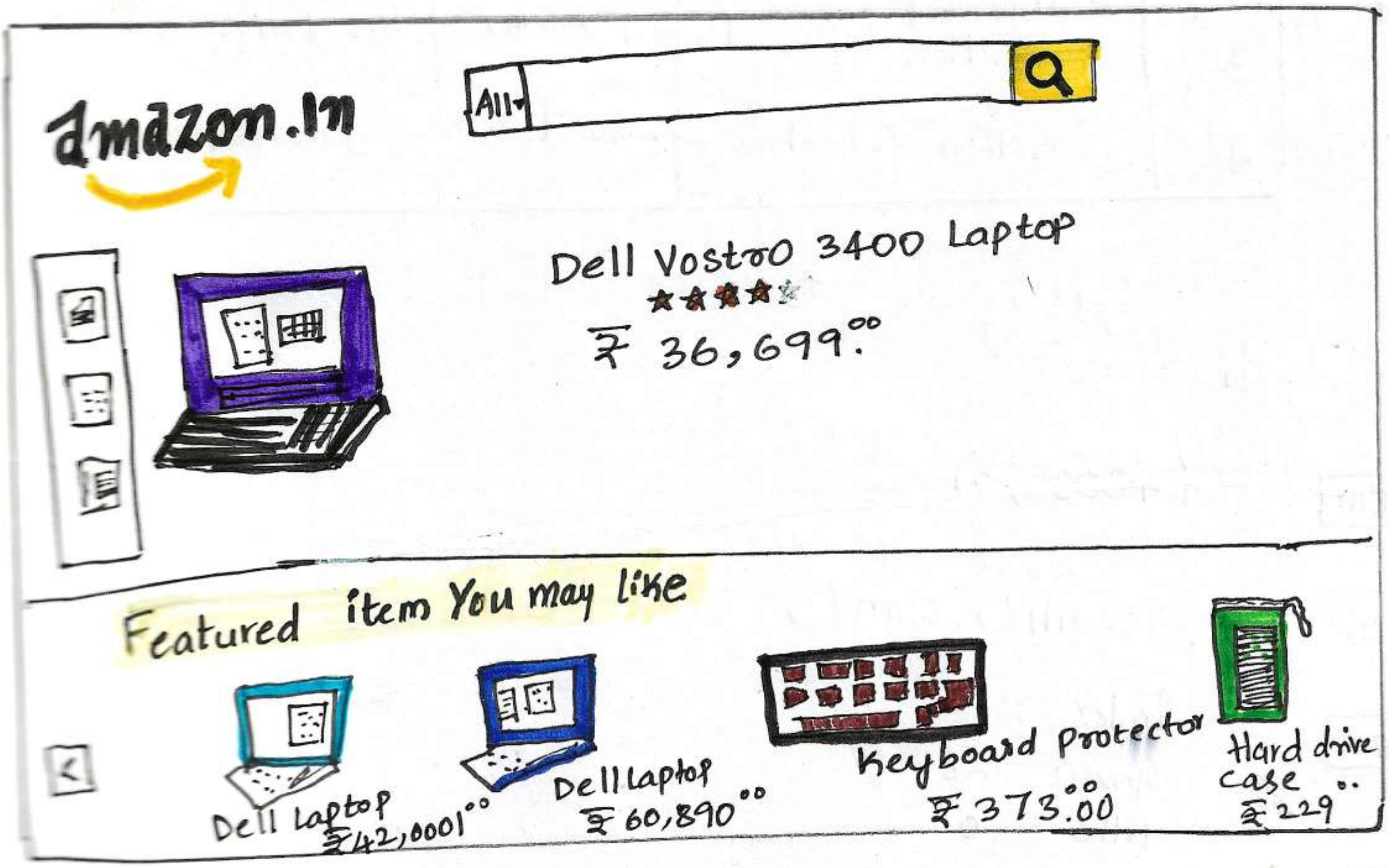


* Here it indentify "Similar items" and recommends it

* Detailed Example : * User Based collaborative



* item based collaborative



code :- Movie Recommendation Engine.

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

Data collection

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

# df.head ()
```

out

Index	Geners	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
Generes	28
title	0

Data pre processing

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

(4800,5) 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]

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

= 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)

is "Iron man" in list/not we are checking

out: ["Avatar", "pirates of caribean", "Spectre", "Dark night Rises",
-]

df[df["title"] == "Iron man"] → Another way

\# finding the close match for the movie given by User

difference library

Given by

import difflib

\# find_close_match = difflib.get_close_matches (movie_name, list_of_all_titles)

\# print ("find_close_match)

Out: ["Iron Man, Iron Man 2, Iron Man 3]

\# Extract "Iron man" from above options

\# close_match = find_close_match [0]

\# print (Close_match)

Out: Iron man

\# 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 [o]
        index = df [a == index] ["titles"]. Values [o]
        if (i<=10):
          Print ( i, "." , index)
        i+ = 1
```

: 10_movies

1+1+1

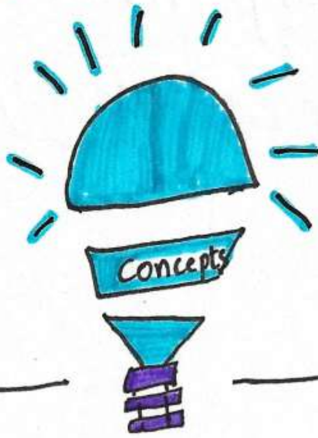
out: Johncarter

Dt:- 18/5/22
10:30 PM

Association Rule

→ connection, relationship, correlation of things.

* Basic Concepts



Given a database of transactions, Each transaction (bill) is considered as a list of items

Finds all rules that correlate presence of one set of items with that of another set of items

identifies frequent patterns

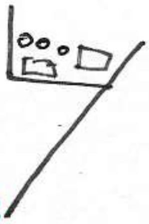
Most commonly used for "Market basket Analysis"

* Where we use association rules?

- Super Markets
- E-commerce website
  - Amazon
  - flipkart
  - bigbasket

MBA → "Market Basket Analysis"

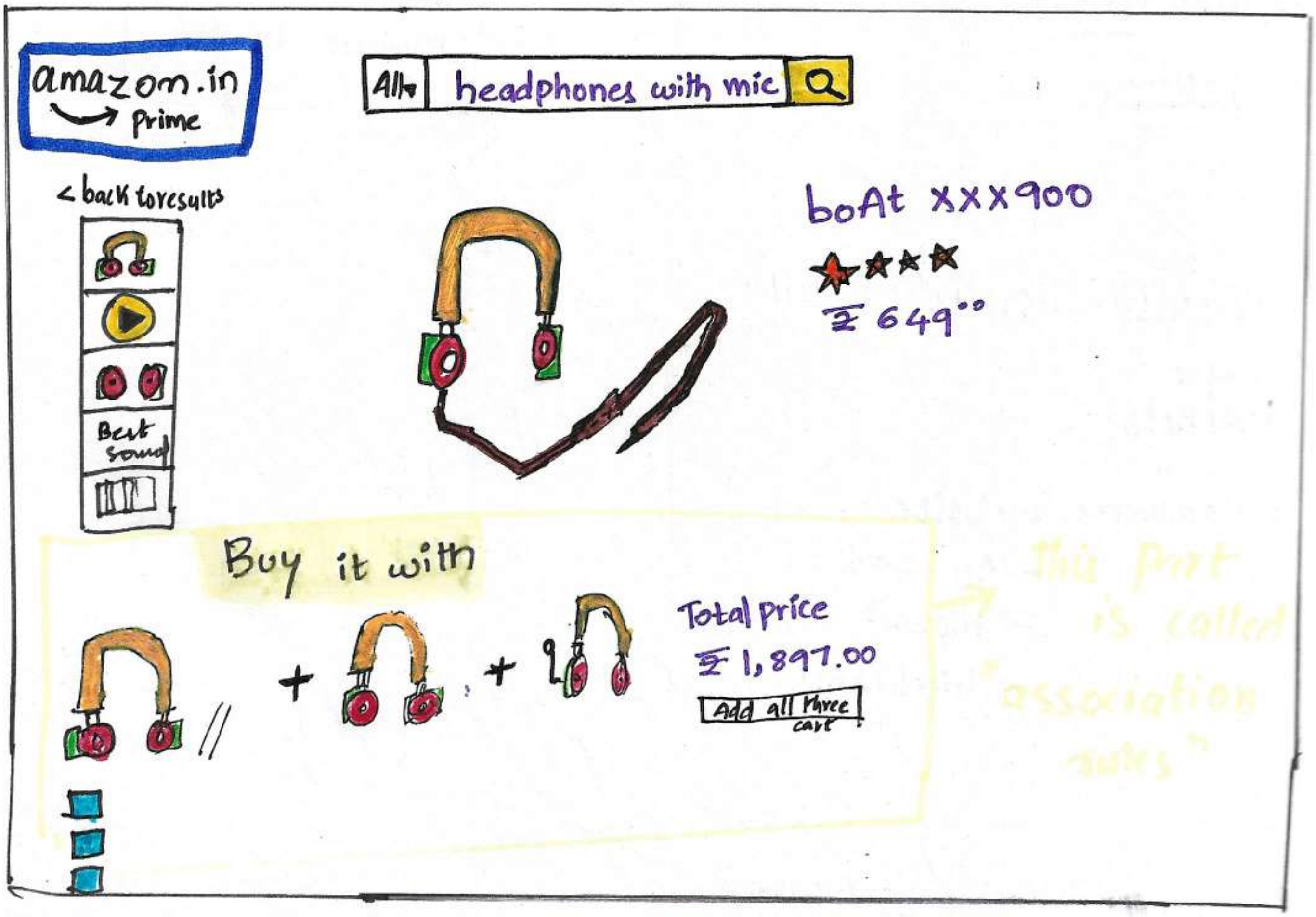
Set of items



"Market Basket Analysis is one of the Key Techniques used by large retailers to Uncover associations between items.

Ex:- brush + paste
Milk + bread
Jam + bread
Laptop + bag

Example:



This Part is called "association rules"

Date:- 13/06/22.

Association analysis :-

it's discovers the Probability of occurence of items in a collection. Helps in discovering some intersting relationships in large datasets.

A dataset Contains data objects and Each data object Contains a set of attributes. an attributes is also called as "dimension" or "feature" or "variable" which represents the characterstics feature of a data object.

Ex :- Height, qualification, colour etc...

* Association Rule Mining :-

it finds the intersting associations and relationships among large sets of data items. this rule shows how frequently a itemset occurs in a transaction.

* Association Rule Mining :-

$$\boxed{A} \Longrightarrow \boxed{B}$$

IF THEN

* Measure Association :

1. Support
2. Confidence
3. Lift

* **Support** :- $\dfrac{\text{Number of transactions Consists of } A, B}{\text{total number of transactions}}$

$$\boxed{\text{Support} \Rightarrow \frac{\text{freq}[A, B]}{N}}$$

* **Confidence** :- $\dfrac{\text{Number of transactions Consists of } A, B}{\text{Number of transaction of only } A}$

* Confidence $\Rightarrow \dfrac{freq(A,B)}{freq(A)}$

* **Lift** $= \dfrac{Support}{Support(A) \times Support(B)}$

Support

- it is a Measure of how frequently a Set of items occur in total number of transactions.

$\{Milk, Bread\} \longrightarrow \{X, Y\}$ $\{X: Milk), (y: Bread\}$

- There fore the frequency of occurence of X and Y together in total no. of transactions is support.

$\{Milk, Bread, Jam\} \longrightarrow \{X, Y\}$ $\{X: Milk\}, \{Y: Bread\ Jam\}$

- Here the Frequency of occurrance of {Bread, Jam} with {Milk} in whole transaction is "support".

$$\boxed{Support(s) = \frac{\sigma(X \cup Y)}{N}}$$

Confidence

- it is a Measure of how often items in "Y" appears in transactions that contain "X"

$\{Milk, Bread, Jam\} \rightarrow \{X, Y\}$ [X: Milk, Y: Bread, Jam]

- Therefore The Frequency of occurance of X and Y in all transactions where "X" exists.

$$\text{confidence } (c) = \frac{\sigma(X \cup Y)}{\sigma X}$$

*** Apriori Algorithm:** Association Rule Generation.

1. uses Frequency itemset to Generate association Rule
2. A Subset of a Frequent itemset must also be a Frequent itemset
3. Frequent itemset is a Set of items whose Support Value > Threshold Value.

Greater Than

Association Rule mining:

Given a Set of transactions $\boxed{T}$, The goal of association rule mining is to $\boxed{\text{find all rules}}$ having

- Support $\geq$ min Support thereshold
- Confidence $\geq$ min Confidence thereshold

Example :- Market Basket Data

(Bill) Transaction ID	items
1	{Milk, Bread, Rice, Book}
2	{Bread, Jam, Book, Pen}
3	{Jam, Milk, Bread, Rice, Eggs}
4	{Rice, Eggs, Pen, Book}
5	{Eggs, Pen, Milk, Bread, Jam}
6	{Eggs, Rice, Bread, Jam}

* Unique items :- {Milk, Bread, Jam, Rice, Eggs, Book, Pen}

Converting into "Dummies"

Transaction ID	Items						
	Milk	Bread	Jam	Rice	Eggs	Book	Pen
1	1	1	0	1	0	1	0
2	0	1	1	0	0	1	1
3	1	1	1	1	1	0	0
4	0	0	0	1	1	1	1
5	1	1	1	0	1	0	1
6	0	1	1	1	1	0	0

⇒ Frequent itemset :-

Set of Products bought frequently

→ An itemset contains "K" items, Than it is known as K itemset

* if it has only "milk" items, Than it is called "one itemset"
* {"Milk" and "Bread"} = 2 (two) itemset.
* {Milk, Bread, Jam} = 3 (Three) itemset

Ex :- Frequent item sets.

. Two itemsets : {Milk, Bread}, {Bread, Jam}, {Rice, Eggs}
{Books, Pen}

. Three itemsets : {Milk, pen, Book}, {Rice, Bread, Eggs}
{Book, Eggs, Pen}

. Four itemsets : {Milk, Bread, Rice, Eggs} Etc.....

13/06/22

Dt :- 14/06/22

Example :- 01

Suppose,

min Sup (min. support) = 0.3

min conf (min. confidence) = 0.6

Taken Randomly

We can change by Hyper parameter Tunning

TID	items
1	{Milk, Bread, Rice, Book}
2	{Bread, Jam, Book, Pen}
3	{Jam, Milk, Bread, Rice, Eggs}
4	{Rice, Eggs, Pen, Book}
5	{Eggs, pen, Milk, Bread, Jam}
6	{Eggs, Rice, Bread, Jam}

Consider.

$\{Rice, Eggs\} \rightarrow \{x, y\}$

$$\boxed{Support\ (s) = \frac{\sigma(x \cup y)}{N}}$$

it checks How many no. of Transactions consist of "Rice", "Eggs"

$$Support\ (s) = \frac{1+1+1}{6} = \frac{3}{6} = 0.5$$

Total no. of

$$\boxed{\text{Confidence (c)} = \frac{\sigma(X \cup Y)}{\sigma X}} \rightarrow \frac{\text{No. of items present in } X, Y}{\text{No. of transaction in } X}$$

$$\text{Confidence (c)} = \frac{\overset{\text{Eggs + Rice}}{3}}{\underset{\text{Rice}}{4}} = 0.75$$

* Association Rule Mining :-

if we got 0.2, we Take This combination.

Here, Support (s) : $\boxed{0.5} \geq$ min Sup $\boxed{(0.3)}$

Confidence (c) : $\boxed{0.75} \geq$ min Conf $\boxed{(0.6)}$

Therefore, We can Mine

{Rice, Eggs} as a rule.

Example : 2 Suppose, we Have Three (3) items

Suppose,

min Sup = 0.3

min Conf = 0.6

Consider,

{Milk, Bread, Jam} → [X, Y]

TID	items
1	{Milk, Bread, Rice, book}
2	{Bread, Jam, Book, Pen}
3	{Jam, Milk, Bread, Rice, Eggs}
4	{Rice, Eggs, Pen, Book}
5	{Eggs, pen, Milk, Bread, Jam}
6	{Eggs, Rice, Bread, Jam}

$$\boxed{\text{Support}(s) = \frac{\sigma(X \cup Y)}{N}}$$

$$\text{Support}(s) = \frac{1+1}{6} = \frac{2}{6} = 0.333$$

and,

$$\boxed{\text{Confidence}(c) = \frac{\sigma(x \cup Y)}{\sigma x}}$$

milk + bread, Jam

$$\text{Confidence}(c) = \frac{2}{3} = 0.667$$

If a person bought → {milk + Bread} → what is the probability He's going to by {"Jam"}

* Association Rule mining :

Support (s) : 0.333 ≥ minsup (0.3)

Confidence (c) : 0.667 ≥ min conf (0.6)

There fore, we can Mine

{Milk, Bread, Jam} as a rule.

Code :-

Here, we Take The same Example in Python coding

```
#  Data = [["milk", "bread", "rice", "book"],
           ["bread", "Jam", "book", "Pen"],
           [ "Jam", "Milk", "bread", "rice", "eggs"],
           ["rice", "Eggs", "pen", "book"],
           ["Eggs", "pen", "milk", "bread", "Jam"],
           [ "Eggs", "rice", "bread", "Jam"]]
```

install : pip install mlxtend

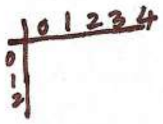
from mlxtend.preprocessing import TransactionEncoder

↳ it converts the data into array

```
#  te = TransactionEncoder()
#  te_array = te.fit_transform(Data)

import pandas as pd

#  df = pd.DataFrame(te_array, columns = te.columns_)
```



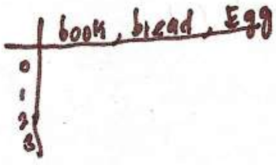
• If we don't write This one. it gives (0,1,2,3)

```
#  df
```

• it Takes in alphabtical order

book, bread, eggs, Jam, milk for

Out: dataframe



	book	bread	eggs	Jam	milk	Pen	rice
0	True	True	False	False	True	False	True
1	True	True	False	True	False	True	False
2	False	True	True	True	True	False	True
3	True	False	True	False	False	True	True
4	False	True	True	True	True	True	False
5	False	True	True	True	False	False	True

14/06/22

Date:- 20/6/22
11:00Am

```
from mlxtend.frequent_patterns import apriori

# itemset = apriori (df, min_Support = 0.3, use_colnames = True)

# itemset
```

We can Take anything (Randomly) 0.5, 0.6.. ...

Out:

Support	itemsets
0.500000	(book)
0.83333	(bread)
0.66667	eggs
0.333333	(bread, book)
0.333333	(book, rice)
0.500000	(bread, Jam, Eggs) ..... 33 results

from mlxtend.frequent_patterns import association_rules

# res = association_rules (itemset, metric = "confidence",
min_threeshold = 0.6)

res

Out[ ]: if → antecedents, Then → consequents

	antecedents	consequents	antecedent_support	consequent_support	support	confidence	lift	Leverage	Conviction
0	book	bread	0.500000	0.833333	0.33333	0.666667	0.8000000	-0.08333	0.500000
1	book	pen	0.500000	0.500000	0.33333	0.666667	1.333333	0.08333	1.500000
2	pen	book	0.500000	0.500000	0.33333	0.666667	1.333333	0.08333	1.500000
	.....	.....	.....	.....	.....	.....	.....	.....	.....
69	(bread_rice)	(jam, eggs)	0.500000	0.500000	0.33333	0.666667	1.333333	0.08333	1.5000000
70	(bread_eggs)	(rice, jam)	0.500000	0.33333	0.33333	0.666667	2.000000	0.166667	2.000000
71	(rice_jam)	(bread, Eggs)	0.33333	0.500000	0.33333	1.000000	2.000000	0.166667	inf
72	(rice_eggs)	(bread, Jam)	0.5000000	0.500000	0.33333	0.666667	1.000000	0.000000	1.0000000

⇒ To Reduce options (or) To gain accuracy, we change the $\boxed{\text{min\_Support} = 0.6,\ \text{min\_threshold} = 0.6}$ — more probability

⇒ it Gives The less item by sorting the high probability.

again

```
from mlxtend.frequent_patterns import apriori

# itemset = apriori(df, min_support = 0.6,
                    use_colnames = True)

# itemset
```

out:

	Support	itemsets
0	0.833333	bread
1	0.666667	eggs
2	0.666667	Jam
3	0.666667	rice
4	0.666667	(bread, Jam)

```
from mlxtend.frequent_patterns import association_rules

# res = association_rules(itemset, metric = "confidence",
                          min_threshold = 0.6)

# res
```

out:

if → then

	antecedes	Consequents	antecedes Support	Consequents support	Support	Confidence	lift	leverage	Conviction
0	bread	Jam	0.833333	0.666667	0.666667	0.8	1.2	0.111111	1.666667
1	Jam	bread	0.666667	0.833333	0.666667	1.0	1.2	0.111111	inf

$$\frac{\text{bread, Jam}}{\text{Total}} = \frac{4}{6} = \frac{2}{3} = 0.666667$$

We Only Consider only Few column for Gaining Result

result = res [["antecedes", "consequents", "support",
"confidence", " Lift"]]

result

If / then

Out]:

	antecedents	Consequents	Support	Confidence	Lift
0	bread	Jam	0.666667	0.8	1.2
1	Jam	bread	0.666667	1.0	1.2

→ Greater Than = 1

result [result [" confidence"] > = 1]

"Support" or Lift → we can Give our own options there

Out]:

antecedents	Consequents	Support	Confidence	Lift
Jam	bread	0.666667	1.0	1.2

→ higher the lift ⇓ higher The combination

items are Selected by Given Support & Confidence

Offers are designed by using This method in Super markets are any Grocery Stores

no. of times it repeated (or) not

it applies Concept of no. of times repeation (Frequency)

min_support, min_Confidence is not fixed.

it changes item by item, which gives high "probability"

↓

The more no. of Times it repeated

it has more association

it Gives combinations of items

For large data of items apply For loop