Recommendation System Algorithms

- Content based filtering
- Collaborative filtering
  - User based filtering
  - Item based filtering

Content based filtering

Item Vector (I)

	Action	Animation	Children
Mission Impossibl	1	0	0
Star Wars	1	1	0
Toy Story	0	1	1

Movie, books, etc

Eg User Review

	Review	Rating
Mission Impossible	1	Good
Star Wars	1	Good
Toy Story	1	Bad

User Vector (U)

	Action	Animation	Children
Nikhil	9	2	-6
Baby Tom	-6	2	9

Top Recommendations

Nikhil - Star Wars
Baby Tom - Toy Story

I x U

	Nikhil	Baby Tom
Mision Impuso	9	-6
Star Wars	11	-4
Toy Story	-6	9

User-Based Collaborative Filtering

what a user is more likely to prefer is highly correlated to what the other users similar to him/her have liked in the past. Used in e-commerce, product and services

Predict rating of Nikhil for each movie he has not seen

User	Fast 7 Furious	Star Wars	Toy Story	Pulp Fiction
Vivek	5	5	1	4
Nikhil	-	4	2	4.5
Abhirami	1	2	-	-
Reetesh	-	-	5	-

User vector

pred(Nikhil, Fast & Furious)
= (5(0.91) +1(0.70))/(0.91 + 0.7)
= 3.26

To summarise the algorithm of user-based filters:

1. Find users similar to the user u (called the peer users) for whom predictions are to be made using any similarity measure like the **correlation coefficient**.
2. For each movie m that the user has not seen, calculate the **weighted average** of the ratings given to m by the peer users.
3. Recommend the top n movies to the user u.

Pearson correlation coefficient

Ai - ratings of Nikhil
Bi - ratings of Vivek

$$\text{similarity}(A,B) = \frac{\sum (A_i - \bar{A})(B_i - B)}{\sqrt{\sum (A_i - \bar{A})^2} \cdot \sqrt{\sum (B_i - \bar{B})^2}}$$

→ Nikhil ↔ Vivek -> 0.91
Nikhil ↔ Abhirami -> 0.7

Item-Based Collaborative Filtering

Collaborative based filtering depends on rating given by users to product movies

Predict rating Nikhil will give for each movie he has not seen

User	Fast 7 Furious	Star Wars	Toy Story	Pulp Fiction
Vivek	5	5	1	4
Nikhil	-	4	2	4.5
Abhirami	1	2	-	-
Reetesh	-	-	5	

Item Vector

pred(Nikhil, Fast & Furious) =

$$\frac{4 * 0.98 + 4.5 * 0.99}{0.98 + 0.99} = 4.25$$

To summarise the algorithm of Item-based filters:

1. Find items similar to the movie m (often called peer group of items) using a similarity measure like cosine.
2. Calculate the rating that the user will give to the movie m using the weighted average of the ratings given to the nearest movies by the user.
3. Recommend the top-n movies to the user.

Consine Similarities

Ai - ratings of Nikhil
Bi - ratings of Vivek

$$\text{similarity}(A, B) = \frac{\sum A_i B_i}{\sqrt{\sum A_i^2} \cdot \sqrt{\sum B_i^2}}$$

Fast and Furious ↔ Star Wars -> 0.99 Fast and Furious ↔ Pulp Fiction -> 0.98

Develop Collaborative Recommendation System

Product dataset (train)

id
name
reviews_rating
reviews_username

```
df_pivot =  pd.pivot_table(train,index= reviews_username, columns = id, values = reviews_rating)
```

df_pivot(70x6)

id	AV14LG0R-jtxr-f38QfS	AV16khLE-jtxr-f38VFn	AV1d76w7vKc47QAVhCqn	AV1h6Gu0glJLPUi8IjA_	AV1h6gSl-jtxr-f31p40	AV1l8zRZvKc47QAVhnAv
reviews_username						
5742870423	0.0	0.0	0.0	0.0	0.0	3.0
5alarm	0.0	0.0	5.0	0.0	0.0	0.0
allycat	0.0	0.0	0.0	0.0	0.0	3.0
alnscoob97	0.0	0.0	0.0	0.0	0.0	1.0
amanda	0.0	5.0	0.0	0.0	0.0	0.0
anonymous8589	0.0	0.0	0.0	5.0	0.0	0.0
ashley a	0.0	5.0	0.0	0.0	0.0	0.0
beccagrl532	0.0	1.0	0.0	0.0	0.0	0.0
bre234	0.0	1.0	0.0	0.0	0.0	0.0
browns fan	0.0	3.0	0.0	0.0	0.0	0.0

```
dummy_train = train.copy()

# The products not rated by user is marked as 1 for prediction.
dummy_train[value_column] = dummy_train[value_column].apply(lambda x: 0 if x>=1 else 1)

dummy_train = pd.pivot_table(dummy_train,index=user_column, columns = product_column, values = value_column)
```

dummy_train(70x6)

id	AV14LG0R-jtxr-f38QfS	AV16khLE-jtxr-f38VFn	AV1d76w7vKc47QAVhCqn	AV1h6Gu0glJLPUi8IjA_	AV1h6gSl-jtxr-f31p40	AV1l8zRZvKc47QAVhnAv
reviews_username						
5742870423	1.0	1.0	1.0	1.0	1.0	0.0
5alarm	1.0	1.0	0.0	1.0	1.0	1.0
allycat	1.0	1.0	1.0	1.0	1.0	0.0
alnscoob97	1.0	1.0	1.0	1.0	1.0	0.0
amanda	1.0	0.0	1.0	1.0	1.0	1.0
anonymous8589	1.0	1.0	1.0	0.0	1.0	1.0
ashley a	1.0	0.0	1.0	1.0	1.0	1.0
beccagrl532	1.0	0.0	1.0	1.0	1.0	1.0
bre234	1.0	0.0	1.0	1.0	1.0	1.0
browns fan	1.0	0.0	1.0	1.0	1.0	1.0

Predict User Rating

user_correlation (75x75) = consine_similarity(df_pivot(75x6))

```
[[1. 0. 1. ... 0. 0. 1.]
 [0. 1. 0. ... 0. 0. 0.]
 [1. 0. 1. ... 0. 0. 1.]
 ...
 [0. 0. 0. ... 1. 1. 0.]
 [0. 0. 0. ... 1. 1. 0.]
 [1. 0. 1. ... 0. 0. 1.]]
(75, 75)
```

user_predicted_ratings(75x6) = np.dot(user_correlation, df_pivot)

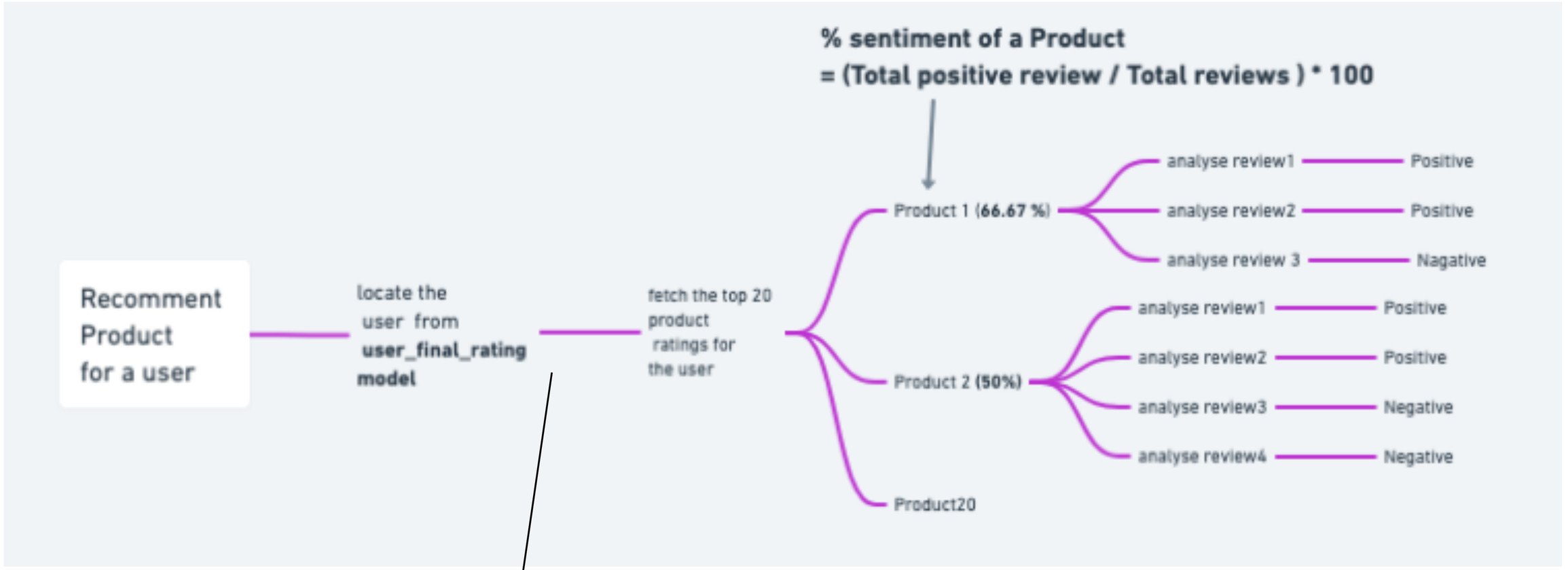
#since we are interested in products that are not rated by the user, we multiply with dummy train to make it zero

user_final_rating = np.multiply(user_predicted_ratings,dummy_train)

id	AV13O1A8GV-KLJ3akUyj	AV14LG0R-jtxr-f38QfS	AV16khLE-jtxr-f38VFn	AV1YGDqsGV-KLJ3adc-O	AV1Ylch7GV-KLJ3addeG	AV1YlENIglJLPUi8IHsX	AV1YmBrdGV-KLJ3adewb
reviews_username							
00sab00	0.0	0.0	1.21	13.58	0.0	0.0	0.0
01impala	0.0	0.0	3.12	15.58	0.0	0.0	0.0
02dakota	0.0	0.0	3.12	15.58	0.0	0.0	0.0
02deuce	0.0	0.0	3.12	15.58	0.0	0.0	0.0
0325home	0.0	0.0	0.00	11.34	0.0	0.0	0.0

user_final_rating

Sentiment Based Product Recommendation




```
user_final_rating.loc[user_input].sort_values(ascending=False)[0:20]
```