Item-based Collaborative Filtering for Product Recommendation

Developed by **NICHOLAS DOMINIC** (2024)

- Email: nicholas.dominic@binus.ac.id / dominicnick4@gmail.com
- Portfolio: https://linktr.ee/ndominic

In [1]:
```
import numpy as np
from pandas import read_csv as rcsv, merge, pivot_table, DataFrame as df
from scipy.spatial.distance import pdist, squareform
from warnings import filterwarnings as fw; fw("ignore")
from sklearn.metrics.pairwise import cosine_similarity
```

Load Dataset

In [2]:
```
ph = rcsv("dataset/purchase_history.csv", delimiter=";")
ph
```

Out[2]:

	customer_id	product_id	purchase_date
0	1	101	2023-01-01
1	1	105	2023-01-05
2	2	102	2023-01-02
3	3	103	2023-01-03
4	4	104	2023-01-04
5	5	101	2023-01-05
6	3	102	2023-01-09
7	2	104	2023-01-03

In [3]:
```
prd = rcsv("dataset/product_details.csv", delimiter=";")
prd
```

Out[3]:

	product_id	category	price	ratings
0	101	Electronics	500	4.5
1	102	Clothing	50	3.8
2	103	Home & Kitchen	200	4.2
3	104	Beauty	30	4.0
4	105	Electronics	800	4.8
5	106	Beauty	50	4.3
6	107	Clothing	39	4.0
7	108	Clothing	55	3.9

In [4]:

```
ci = rcsv("dataset/customer_interactions.csv")
ci
```

Out[4]:

	customer_id	page_views	time_spent
0	1	25	120
1	2	20	90
2	3	30	150
3	4	15	80
4	5	22	110

Data Preparation

In [5]:

```
# join multiple datasets
merged_data = merge(ci, ph, on="customer_id")
merged_data = merge(merged_data, prd, on="product_id")

# create a distance matrix based on ratings
user_item_matrix = pivot_table(
    merged_data, index="customer_id", columns="product_id", values="ratings", fill_
)

# handle missing values (e.g., impute with average rating)
user_item_matrix.fillna(user_item_matrix.mean(), inplace=True)
prd_ids = user_item_matrix.columns.to_list()
cust_ids = user_item_matrix.index.to_list()
user_item_matrix
```

Out[5]:

product_id	101	102	103	104	105
customer_id					
1	4.5	0.0	0.0	0.0	4.8
2	0.0	3.8	0.0	4.0	0.0
3	0.0	3.8	4.2	0.0	0.0
4	0.0	0.0	0.0	4.0	0.0
5	4.5	0.0	0.0	0.0	0.0

Get Similarities between Products

In [6]:
```
item_sim_df = df(cosine_similarity(user_item_matrix, user_item_matrix), index=prd_i
item_sim_df.head()
```

Out[6]:

	101	102	103	104	105
101	1.000000	0.000000	0.000000	0.000000	0.683941
102	0.000000	1.000000	0.462091	0.724999	0.000000
103	0.000000	0.462091	1.000000	0.000000	0.000000
104	0.000000	0.724999	0.000000	1.000000	0.000000
105	0.683941	0.000000	0.000000	0.000000	1.000000

In [7]:
```
def get_similar_product(prd_id):
    if prd_id not in prd_ids:
        return None, None
    else:
        sim_cust = item_sim_df.sort_values(by=prd_id, ascending=False).index[1:]
        sim_score = item_sim_df.sort_values(by=prd_id, ascending=False).loc[:, prd_
        return sim_cust, sim_score
```

In [8]:
```
selected_prd_id = 102
_prd, _score = get_similar_product(selected_prd_id)

print("Product ID = {} has similarities with:".format(selected_prd_id))
for p, s in zip(_prd, _score):
    print("  - Product ID = {}, with similarity score of {:.3f}".format(p, s))
```

```
Product ID = 102 has similarities with:
  - Product ID = 104, with similarity score of 0.725
  - Product ID = 103, with similarity score of 0.462
  - Product ID = 101, with similarity score of 0.000
  - Product ID = 105, with similarity score of 0.000
```

Ratings Prediction

In [9]:
```
def predict_rating(cust_id, prd_id, max_neighbor=2):
    _prd, _score = get_similar_product(prd_id)
    prd_arr = np.array([x for x in _prd])
    sim_arr = np.array([x for x in _score])

    # select only the product that has already rated by user x
    filtering = user_item_matrix[prd_arr].loc[cust_id] != 0

    # calculate the predicted score
    sim_scores = sim_arr[filtering][:max_neighbor]
    closest_rating = user_item_matrix.loc[cust_id][prd_arr[filtering][:max_neighbor
    sum_sim_scores = np.sum(sim_arr[filtering][:max_neighbor])

    s = np.dot(sim_scores, closest_rating) / sum_sim_scores
    return s
```

In [10]:
```
for c in cust_ids:
    for p in prd_ids:
        print("Cust ID = {}, Product ID = {}, Pred. Rating = {:.2f}".format(c, p, p
```

```
Cust ID = 1, Product ID = 101, Pred. Rating = 4.80
Cust ID = 1, Product ID = 102, Pred. Rating = nan
Cust ID = 1, Product ID = 103, Pred. Rating = nan
Cust ID = 1, Product ID = 104, Pred. Rating = nan
Cust ID = 1, Product ID = 105, Pred. Rating = 4.50
Cust ID = 2, Product ID = 101, Pred. Rating = nan
Cust ID = 2, Product ID = 102, Pred. Rating = 4.00
Cust ID = 2, Product ID = 103, Pred. Rating = 3.80
Cust ID = 2, Product ID = 104, Pred. Rating = 3.80
Cust ID = 2, Product ID = 105, Pred. Rating = nan
Cust ID = 3, Product ID = 101, Pred. Rating = nan
Cust ID = 3, Product ID = 102, Pred. Rating = 4.20
Cust ID = 3, Product ID = 103, Pred. Rating = 3.80
Cust ID = 3, Product ID = 104, Pred. Rating = 3.80
Cust ID = 3, Product ID = 105, Pred. Rating = nan
Cust ID = 4, Product ID = 101, Pred. Rating = nan
Cust ID = 4, Product ID = 102, Pred. Rating = 4.00
Cust ID = 4, Product ID = 103, Pred. Rating = nan
Cust ID = 4, Product ID = 104, Pred. Rating = nan
Cust ID = 4, Product ID = 105, Pred. Rating = nan
Cust ID = 5, Product ID = 101, Pred. Rating = nan
Cust ID = 5, Product ID = 102, Pred. Rating = nan
Cust ID = 5, Product ID = 103, Pred. Rating = nan
Cust ID = 5, Product ID = 104, Pred. Rating = nan
Cust ID = 5, Product ID = 105, Pred. Rating = 4.50
```

Product Recommendation

In [11]:
```
def get_recommendation(cust_id, n_recommended=2):
    pred_ratings = [predict_rating(cust_id, p) for p in prd_ids]

    # do not recommend products that customer has already rated
    temp = df({'predicted' : pred_ratings, 'prd_id' : prd_ids})
    filt = (user_item_matrix.loc[cust_id] == 0.0)
    temp = temp.loc[filt.values].sort_values(by='predicted', ascending=False)

    print("Product recommendations\nfor Customer ID = {}:".format(cust_id))
    return prd[prd.product_id.isin(temp.prd_id[:n_recommended])]
```

In [12]:
```
get_recommendation(cust_id=5)
```

```
Product recommendations
for Customer ID = 5:
```

Out[12]:

	product_id	category	price	ratings
1	102	Clothing	50	3.8
4	105	Electronics	800	4.8

References

- https://www.datacamp.com/tutorial/streamlit
- https://www.kaggle.com/code/varian97/item-based-collaborative-filtering
- https://github.com/yjeong5126/movie_recommender/blob/master/item_based_collaborative