[52]
```
def customer_recomendation(customer_id):
    if customer_id not in df_output.index:
        print('Customer not found.')
        return customer_id
    return df_output.loc[customer_id]
```

[53]
```
customer_recomendation(4)
```

```
recommendedProducts    2|1|36|13|216|61|20|33|25|157
Name: 4, dtype: object
```

[54]
```
customer_recomendation(21)
```

```
recommendedProducts    38|36|48|79|1|2|15|13|25|20
Name: 21, dtype: object
```

7. Model Evaluation

For evaluating recommendation engines, we can use the concept of precision-recall.

- RMSE (Root Mean Squared Errors)
  - Measures the error of predicted values
  - Lesser the RMSE value, better the recommendations
- Recall
  - What percentage of products that a user buys are actually recommended?
  - If a customer buys 5 products and the recommendation decided to show 3 of them, then the recall is 0.6
- Precision
  - Out of all the recommended items, how many the user actually liked?
  - If 5 products were recommended to the customer out of which he buys 4 of them, then precision is 0.8

Lets compare all the models we have built based on precision-recall characteristics:

8.1. Evaluation summary

- Based on RMSE

```
1. Popularity on purchase counts: 1.1111750034210488
2. Cosine similarity on purchase counts: 1.9230643981653215
3. Pearson similarity on purchase counts: 1.9231102838192284

4. Popularity on purchase dummy: 0.9697374361161925
5. Cosine similarity on purchase dummy: 0.9697509978436404
6. Pearson similarity on purchase dummy: 0.9697745320187097

7. Popularity on scaled purchase counts: 0.16230660626840343
8. Cosine similarity on scaled purchase counts: 0.16229800354111104
9. Pearson similarity on scaled purchase counts: 0.16229826683334026
```

- Based on Precision and Recall

```python
name = 'pearson'
target = 'purchase_count'
pear = model(train_data, name, user_id, item_id, target, users_to_recommend, n_rec, n_display)
```

```
Preparing data set.
    Data has 106868 observations with 23382 users and 300 items.
    Data prepared in: 0.154145s
Training model from provided data.
Gathering per-item and per-user statistics.
+--------------------------------+------------+
| Elapsed Time (Item Statistics) | % Complete |
+--------------------------------+------------+
| 1.484ms                        | 4.25       |
| 24.436ms                       | 100        |
+--------------------------------+------------+
Setting up lookup tables.
Processing data in one pass using dense lookup tables.
+-------------------------------------+------------------+-----------------+
| Elapsed Time (Constructing Lookups) | Total % Complete | Items Processed |
+-------------------------------------+------------------+-----------------+
| 25.339ms                            | 0                | 0               |
| 77.412ms                            | 100              | 300             |
+-------------------------------------+------------------+-----------------+
Finalizing lookup tables.
Generating candidate set for working with new users.
Finished training in 0.095617s
recommendations finished on 1000/1000 queries. users per second: 47321.6
+------------+-----------+--------------------+------+
| customerId | productId |       score        | rank |
+------------+-----------+--------------------+------+
|    1553    |    248    | 3.269303671338341  |  1   |
|    1553    |    132    | 3.172413793103448  |  2   |
|    1553    |     37    | 3.0322580645161303 |  3   |
|    1553    |     34    | 2.9947963457127105 |  4   |
|    1553    |     0     | 2.9891616797683294 |  5   |
|    1553    |     3     | 2.7993184668951274 |  6   |
|    1553    |    110    | 2.7272471324963994 |  7   |
|    1553    |     27    |  2.69852878164528  |  8   |
|    1553    |    230    | 2.673649873986517  |  9   |
|    1553    |     10    | 2.6516516516516515 |  10  |
|   20400    |    248    | 3.2727272727272725 |  1   |
|   20400    |    132    | 3.172413793103448  |  2   |
|   20400    |     37    | 3.029654354818407  |  3   |
|   20400    |     34    | 2.995901639344264  |  4   |
|   20400    |     0     | 2.992079207920795  |  5   |
|   20400    |     3     | 2.7986623836689235 |  6   |
```

recommendations finished on 1000/1000 queries. users p

customerId	productId	score	rank
1553	259	1.0	1
1553	44	1.0	2
1553	298	1.0	3
1553	45	1.0	4
1553	249	1.0	5
1553	106	1.0	6
1553	15	1.0	7
1553	3	1.0	8
1553	49	1.0	9
1553	11	1.0	10
20400	44	1.0	1
20400	298	1.0	2
20400	45	1.0	3
20400	249	1.0	4
20400	106	1.0	5
20400	15	1.0	6
20400	3	1.0	7
20400	49	1.0	8
20400	193	1.0	9
20400	11	1.0	10
19750	44	1.0	1
19750	298	1.0	2
19750	45	1.0	3
19750	249	1.0	4
19750	106	1.0	5
19750	15	1.0	6
19750	3	1.0	7
19750	49	1.0	8
19750	193	1.0	9
19750	11	1.0	10

```
# these variables will change accordingly
name = 'cosine'
target = 'purchase_count'
cos = model(train_data, name, user_id, item_id, target, users_to_recommend, n_rec, n_di
```

```
Preparing data set.
    Data has 106868 observations with 23382 users and 300 items.
    Data prepared in: 0.169002s
Training model from provided data.
Gathering per-item and per-user statistics.
+--------------------------------+------------+
| Elapsed Time (Item Statistics) | % Complete |
+--------------------------------+------------+
| 1.202ms                        | 4.25       |
| 10.97ms                        | 100        |
+--------------------------------+------------+
Setting up lookup tables.
Processing data in one pass using dense lookup tables.
+-------------------------------------+------------------+-----------------+
| Elapsed Time (Constructing Lookups) | Total % Complete | Items Processed |
+-------------------------------------+------------------+-----------------+
| 12.926ms                            | 0                | 0               |
| 49.811ms                            | 100              | 300             |
+-------------------------------------+------------------+-----------------+
Finalizing lookup tables.
Generating candidate set for working with new users.
Finished training in 1.07754s
recommendations finished on 1000/1000 queries. users per second: 33992.8
+------------+-----------+----------------------+------+
| customerId | productId |        score         | rank |
+------------+-----------+----------------------+------+
|    1553    |     2     | 0.0716024488210678l  |  1   |
|    1553    |     35    |  0.069358721375465   |  2   |
|    1553    |     1     |  0.069155260920524   |  3   |
|    1553    |     61    | 0.0625116676092147   |  4   |
|    1553    |     21    | 0.0449709445238113   |  5   |
|    1553    |     76    | 0.0421509444713592   |  6   |
|    1553    |     0     | 0.0397711694240570   |  7   |
|    1553    |    269    | 0.03771896660327911  |  8   |
|    1553    |     8     | 0.0359516739845275   |  9   |
|    1553    |     5     | 0.0334282368421554   |  10  |
|   20400    |    122    | 0.04556488990783691  |  1   |
|   20400    |    215    | 0.0400064587593078   |  2   |
|   20400    |     6     | 0.0369896292686462   |  3   |
|   20400    |     1     | 0.03644502162933349  |  4   |
|   20400    |     54    | 0.0354259610176086   |  5   |
|   20400    |     56    | 0.03466206789016724  |  6   |
|   20400    |    179    |  0.034053981304168   |  7   |
```

customerId	productId	score	rank
0	248	3.27272727272727273	1
0	132	3.1724137931034484	2
0	37	3.032258064516129	3
0	34	2.9959016393442623	4
0	0	2.992079207920792	5
0	3	2.800429184549356	6
0	110	2.7329545454545454	7
0	27	2.699248120300752	8
0	230	2.676258992805755	9
0	10	2.6516516516516515	10
1	248	3.27272727272727273	1
1	132	3.1724137931034484	2
1	37	3.032258064516129	3
1	34	2.9959016393442623	4
1	0	2.992079207920792	5
1	3	2.800429184549356	6
1	110	2.7329545454545454	7
1	27	2.699248120300752	8
1	230	2.676258992805755	9
1	10	2.6516516516516515	10
2	248	3.27272727272727273	1
2	132	3.1724137931034484	2
2	37	3.032258064516129	3
2	34	2.9959016393442623	4
2	0	2.992079207920792	5
2	3	2.800429184549356	6
2	110	2.7329545454545454	7
2	27	2.699248120300752	8
2	230	2.676258992805755	9
2	10	2.6516516516516515	10

customerId	productId	purchase_count
1750	146	2
2870	76	1
21091	4	2
9881	2	1
10807	189	1
8564	274	1
13417	1	2
25794	228	1
21384	228	2
15982	175	1

3.3. Normalize item values across users

- To do this, we normalize purchase frequency of each item across users by first creating a user-item matrix as follows

```
[14] df_matrix = pd.pivot_table(data, values='purchase_count', index='customerId', columns='productId')
     df_matrix.head()
```

productId	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27
customerId																												
0	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	3.0	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	6.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	1.0	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 300 columns

```
[6] print(customers.shape)
    customers.head()
```

```
(1000, 1)
```

	customerId
0	1553
1	20400
2	19750
3	6334
4	27773

```
[7] print(transactions.shape)
    transactions.head()
```

```
(62483, 2)
```

	customerId	products
0	0	20
1	1	2|2|23|68|68|111|29|86|107|152
2	2	111|107|29|11|11|11|33|23
3	3	164|227
4	5	2|2

User Based Collaborative Filtering

The goal of user-based collaborative filtering is to predict the rating of i_n given the previous actions of one user, u_m. In user-based collaborative filtering, we find k candidates of other users that are similar to u_m. To find similarity, we can use

Cosine Similarity:

$$cos\theta = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Pearson Correlation: $$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

There is also a method of using K-nearest-neighbors to find the most similar candidate users, defined by

$$p_{a,i} = \bar{r}_a + \frac{\sum_{u \in K}(r_u, i - \bar{r}_u) \times w_{a,u}}{\sum_{u \in K} w_{a,u}},$$

where $p_{a,1}$ is the prediction for target user u_m at item i_n, $w_{a,u}$ is the similarity between the users, and K is the neighborhood of similar users.

Item-based Collaborative Filtering

The goal of item-based collaborative filtering is to predict ratings for a specific item i_n based on ratings of similar previous items rated by all possible users.

The rating for target item i_n for active user u_m can be found using

Weighted Average

$$p_{a,i} = \frac{\sum_{j \in K} r_{a,j} w_{i,j}}{\sum_{j \in K} |W_{i,j}|}$$

where K is the neighborhood of most similar items rated by user u_m and $w(i, j)$ is the similarity between items i and j.

Adjusted Cosine Similarity

$$\mathbf{sim}_i j = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}}$$

	i_1	i_2	i_3	i_4	…	i_n
u_1	√		√			
u_2			√			√
u_3				√		
u_4		√		√		
…						
u_m			√			