Content based Method

Asal Rahbari – 99222044

The procedure of developing this recommender system is made of 5 phases. Here, I explain each of these phases and what is actually done during that phase.

1) Loading Dataset:

   In our first step, we should load our dataset and observe it carefully to figure out most of its characteristics. For reaching this goal, we can use several methods like getting the list of the columns of dataset, number of unique values in each column, type of data in each column, shape of the whole data frame, using the “describe” command and so on.
   By observing “big basket products” dataset for the first time, even before seeking all of the info mentioned above, we immediately realize that the column “index” is unnecessary and we can just drop this column and get rid of it.

   After getting familiar with the data we’re supposed to deal with for our recommendation system, we move on to the next phase.

2) Null Values:

   It’s essential to look for any null values and empty cells in your dataset, because null values are in fact the lack of information and before applying any method on our data we must make sure that this lack of info is somehow handled and won’t cause any problem in the following phases.
   There are many approaches and techniques for handling the null values of each column. The first decision to make is to fill in the

blank segments of the columns in some way or to just drop the column and remove it from our data frame. The most important factor that helps us with this decision is the percentage of null values in a column. For instance, in our problem the percentage of null values for none of the columns is that high to remove the column thoroughly from our dataset. So, we have to search for a better solution.

The next decision to make is to drop all of the rows containing null values of a feature or not. In our case, the percentage of null values in description, product and brand is too low that it’s just not worth it to try to fill the null values, so we just drop the rows. But for the “rating” the percentage of null values is significant and cannot be left on its own (it’s also not that high to drop the whole column, means that it provides us with enough amount of information hence it’s valuable for us to keep it).

We fill in the null values of “rating” using the “median” strategy, which seems to be the best available choice for this column since its numerical and outliers don’t affect median compared to mean.

3) Encoding:

Clearly, encoding categorical columns of a dataset is a crucial part of data pre-processing and it should be done carefully, because choosing the wrong method for encoding a categorical feature may lead to further problems.

Category: The number of unique values for this column is few, so we can definitely use one-hot encoding for this feature. After applying this method on our data frame, we just drop the original category column since we no longer need it.

Sub_Category: For this column we are not able to use one-hot encoding because it’s not an efficient or even possible way considering the number of its unique values. We can use label encoding for this column. Label encoder will assign an integer to each unique value of the column.

Brand: Considering the role and impact of a brand on the product’s rating, we can use the frequency encoding for this column. Assuming that the brands with same frequency will produce items with similar quality and therefore their rating patterns wouldn’t differ that much.

Type: Label encoder can be a good choice for this column, too.

Description: For this column the number of unique values is almost the same as the whole number of rows in the data frame. Also, the value type of this column is text, which can make a lot of trouble while encoding it. The best choice for these kind of features is word embedding. In this method each value is mapped to a vector.

Now that encoding process has come to an end, we can go for the recommender system.

4) Create Dictionary:

Sometimes in finding the similarity of two products, we don't use all of their available features and we only use the most effective and important ones. In our case, for being able to find the similarity of products we need to have a vector representation for each of them and then find the difference between those vectors. The more distant those vectors are, the less similar the products are. So, first we try to get those vector representations for each row(product) of our dataset and create a dictionary for our products, which the keys of the dictionary are the product's name and the value is the vector representation of the product's characteristics. Using this method, whenever we get a product's name as our input we reach out to the dictionary and find its corresponding vector, then we compare that vector with all other available vectors we have and we give the similar one's product name as an output (the threshold for similarity is set by us based on our special preferences).

5) Recommender System:

For this recommender system we’re using jaccard similarity, which is a measure to assess the similarity between two sets of items. It somehow calculates the ration of the intersection of two items to their union.
For giving recommendations to the user I decided to let the user choose the range of similarity between the items, between two options: very similar and fairly similar.
For the high similarity option, I set the threshold “0.4” for finding similar products using jaccard method and for the other option “0.3” is set.
Then our system will get the name of the product and find its corresponding vector (which represents the product’s characteristics) from the dictionary we created in the previous step and then applies the jaccard test on this vector and all of other available vectors in the dictionary. Finally, it decides whether a vector is similar to the chosen vector or not based on the threshold that is set. A list of similar products’ names is printed for the user. Obviously, the higher accuracy of similarity leads to a less long list of similar items.

The End