Data Mining Assignment - Recommender Systems

Nick Wils

June 7, 2022

Abstract

You can find the code of my project with the following link: github_link.
The dataset.csv and recommend.csv files should be in the same directory as main.py. Run main.py to run the the project.

1 Introduction

In this report I will discuss how I implemented a way to find a baseline / individual recommender. To achieve this I assumed that the goal of the recommendations is to make as much sales as possible. When that goal changes, the recommender will probably not give an optimal solution. For example if the goal would be to make as much profit as possible, you would need to keep the price of the products in mind and favor more expensive products.

2 Baseline

2.1 Implementation

To implement a baseline recommender, that recommends the same 10 products to all users, I needed to find the 10 most popular items. To find these 10 products, I implemented a scoring system. The scoring of a product is based on the amount and type of interactions users had with that product.

There are 4 types of interactions possible and they have the following amount of points:

- view: 1 point. Viewing a product means there is some interest.
- cart: 10 points. Adding a product to the cart means that there is some interest to buy the item.
- remove_from_cart: -5 points. Removing a product from the cart means that there is possibly a problem with the product or there might be a better alternative.
- purchase: 25 points. Purchasing a product is exactly the type of action we want to promote.

As you can see, the scoring system heavily favors purchased items. This is logical as we want to make as much sales as possible with the recommendation system. The other events are added mostly to rearrange the positioning slightly: a product that is bought 10 times, viewed 10 times and added in 10 carts is probably less interesting than an item that is bought 9 times, viewed 30 times and added in 20 carts. With this scoring system I counted all the points for each product, ordered them and took the 10 products with the highest score.

To make sure that the products make as much sales as possible, I ensured the following conditions were satisfied:

- The products are spread over different categories: If all 10 products would be in the same category, it is possible that these products target the same group. It's probably better to promote a laptop and a lamp instead of 2 laptops. Because you want to entice as many users as possible.
- The products are bought by enough users: If there is only a small group of people that bought these items (multiple times), it doesn't make sense to advertise these items to everybody. These users that buy it already will probably keep buying it regardless.

There is only one category that overlaps and of the 1282 products only 37 are bought by somebody that already bought another item in the top10 list. So these conditions are sufficiently met.

2.2 Performance

To estimate an upper and lower limit on how often these recommendations will be bought, I looked at the past. So in my estimation I will assume that the provided dataset gives a sufficiently precise idea on future sales. My estimations will go over the same timeframe as the dataset which is 2 months. The interactions in the dataset took place around the December holidays. This will most likely influence the sales. If the recommendations are not tested in a similar setting the results will most likely be lower than in the dataset.

For a lower limit I would estimate that there would be about the same amount of sales as in the dataset. I assumed this, because in the dataset there wasn't a recommender. Even a very bad recommender would probably not give a negative impact on sales, meaning the sales wouldn't change. In the dataset there were 25257 sales in total, 1282 of these sales were top10 products.

For an upper limit I reasoned that people, that bought a product in the same category as a top10 product, might be persuaded to buy the top10 product instead. 5117 products were sold in categories from top10 products. This does not take all benefits of recommendations into account. For example people, that are not looking for a new lamp, might see a beautiful lamp in the recommendations and buy it. I did not find an accurate way to take factors like this into account.

With the previous calculations, my best guess would be that the recommended items will be sold between 1282 and 5117 times.

3 individual recommendations

To make the recommendations more personal I used association rule mining. The mining algorithm I used in the previous assignment was Apriori. This is however quite slow and not practical in real world applications. Instead I used FP growth, which is a more efficient version of Apriori. I used a library "mlxtend" that already implemented this.

But before I could use the FP growth algorithm, I needed to preprocess the data. I filtered out "remove from cart" events, as they do not reflect positive information. I then grouped the data by user id and transformed this dataset with the TransactionEncoder from the "mlxtend" library. This encoder transforms the data into a matrix where every cell represents a boolean whether or not a user bought a certain product.

The FP growth algorithm will then give back itemsets that are frequently bought together. The "mlxtend" library has a function that uses this information to form association rules based on a minimum threshold on a certain metric. This metric can be confidence or lift. I used a minimum of 0.1 as a threshold for confidence. This made sense because I have 10 items that I can show as recommendations. I filter these rules one more time to make sure the lift is high enough. Rules with a lift lower than 100 are filtered away.

Now the rules are determined, I just need to use them for each user. I did this by going through the following process for each user. First I filtered "remove from cart" events from the dataset. Then I got a list, with all the products a user interacted with, from the "recommend.csv" dataset and removed duplicate entries.

The next step is to go over all the items in that list and get the rules associated with those items. I store all the items that those rules point to, together with the lift of the rules. If an item is a consequence of multiple rules, it means that there is a high probability that it is a good item to recommend. Because of this, I add the lift of both occurrences. When I have the full list of items, I order them by their lift. Three things can happen now:

- The list contains more than 10 items: I only keep the 10 items with the highest lift.
- The list contains less than 10 items: I add items from the baseline to fill it.
- The list already contains 10 items: I keep the list as it is.

This list now contains my 10 recommendations for each user.

3.1 Performance

To estimate an upper and lower limit I would again look at the past. In fact, my answer is exactly the same as with the performance of the baseline. The lower limit would be the sales of the recommended items in the past. The upper limit would be the total sales of the categories of these products. This is still assuming that the past accurately describes the future, the test time is 2 months and the test is performed in a similar setting as the December holidays.

With the addition of the "recommend.csv" dataset I am able to test, to an extent, how the personalized recommender performed against the baseline. I did two slightly different tests.

First I dropped "remove from cart" events from the recommend dataset. then I splitted this dataset in 2 parts:

- past dataset: view/cart events
- solution dataset: the purchase events

With the past dataset I created personalized recommendations for all the users in this dataset. It is now possible to test the baseline recommendations against the personalized recommendations. To do this, I went over all the users that are in both the past and solution datasets. For each user I checked if the purchased product (in the solution dataset) was predicted or not. As a result I found that the personalized recommender performed almost twice as good:

- The baseline recommendations were purchased 215 times of the possible 6278.
- The personalized recommendations were purchased 519 times of the possible 6278.

However, there are some problems with this test. I removed the most important information from when making the personalized recommendations. There is also the problem that if a person wants to buy product A, then product A has to be first placed in the cart and is probably viewed as well. Meaning that there is a big overlap in the datasets past and solution. To try and solve this problem I made another test that can be performed by uncommenting a line in the test function. In this test I removed the view and cart events from the past dataset, if a purchase event occurs with that same item and user in the solution dataset. This test has a better separation of data, but made the problem of missing data even worse. The result here is again that the personalized recommender preformed twice as good:

- The baseline recommendations were purchased 149 times of the possible 4411.
- The personalized recommendations were purchased 346 times of the possible 4411.

Although these test are not perfect for testing the personalized recommender, it gives an insight on how it preforms against the baseline recommender.

In conclusion: I believe that my personalized recommender will perform better than the baseline recommender. Both the personalized and the baseline recommender have the same lower and upper limits. The lower limit being the sales of the recommended products in the past and the upper limit being the sales of the categories of these products in the past.