Oli Tailby
24th November 2024

Jollyes DS Task

“Recommend three products Jollyes should up-sell for each SKU. For example, if I buy a lead, you may recommend a collar, poo bags, and a treat.”

Data Preparation

I made a couple of choices to clean the data given the context of the problem:

- Ignoring transactions with negative quantity (presumably these are refunds)
- I had explored using quantity as an indicator of preference (ie. buying two of a product would suggest higher interest than one) but have avoided this as some SKUs are often purchased in bulk or have non-integer quantities.
- Removing “15% Off Voucher” and “Delivery” SKUs, as these are not stocked purchasable items in the context of this brief.

Modelling

I think the most challenging part of this problem is recommending SKUs that are similar enough that they logically share a basket but not so similar that they fulfil the same purpose (customers are unlikely to buy two dog leads of different brands at the same time!). To tackle this, I initially approached this problem with a version of a Collaborative Filtering algorithm. This approach finds similar transactions to an input using a k-nearest-neighbours algorithm. We can then aggregate these similar transactions to generate recommendations for each SKU based on the most frequent co-occurrences. I felt this would be more representative of consumer behaviour than using a content-based method as it is directly based on actual transaction data. However, with the limited transaction data here (most SKUs only appear in only a small handful of multi-item transactions and some appear in none) this approach over-fits so is not performant on the dataset given.

I therefore expanded to a hybrid Collaborative Filtering *and* Content-Based approach. The scores from the approach above are combined with a product similarity score based on the product features available to reach an overall recommendation score. This option aims to maintain the benefit of being at least partially informed by prior transaction behaviour, whilst also being applicable for low-volume or new SKUs

Benefits

1. This approach is partially informed by past customer behaviour, so considers the data products that customers *actually* buy together, rather than assuming that product similarity is the only factor.
2. The approach can scale beyond recommending cross-sell opportunities for a single SKU. As collaborative filtering scores are calculated based on transaction-level vectors and product similarity scores can be aggregated across multiple products, we can expand this to recommend cross-sell opportunities for a basket containing multiple items. The same approach could

also be modified to recommend at *customer*-level based on all past purchases, on the basis of what similar customers have bought.

Drawbacks

1. As discussed above, the collaborative filtering part of the approach relies on a large set of transactional data. With real data I anticipate the ratio of transactions to SKUs to be much higher, improving recommendations, but the risk would remain for small-volume and new SKUs. Likewise, the product similarity scores may give high precedence to items that are *too* similar for customers to realistically buy together.
2. Collaborative Filtering relies on a nearest neighbour algorithm on a large transactional dataset, which may raise time and computational costs at scale.

Improvements and Next Steps

1. Currently I have assumed a basic additive combination of the two approaches, leaning more heavily on product similarity due to the small set of transaction data. Alternative weighting solutions could help to capture the benefits and minimise the risks of both approaches.
2. The Collaborative Filtering method picks fixed k nearest neighbours and aggregates them. A better approach could be to pick all (or a sample of) neighbours based on distance thresholds, avoiding picking neighbours that are too far away for low-volume SKUs
3. There are multiple further parameters in this approach we can tune - eg. weighting, distance metric and number of nearest neighbours in collaborative filtering or feature choice for product similarity.
4. Currently, the product similarity method uses tf-idf vectors with equal weighting to all words contained in its features. This leads to some undesirable cases (eg. Cat Woodshavings is highly similar to Small Animal Woodshavings). To improve this algorithm, exploring heavier weighting on key features such as Department may be helpful.

Performance and Monitoring

1. Theoretical performance: With a large enough set of transactions it would be possible to create separate train and test sets, in which we confirm whether the scores for each SKU align with the frequency that SKU is purchased with other items in the test set. However, the sample data is not large enough to do this meaningfully.
2. Practical performance: In reality, the fact that a customer has purchased two SKUs together doesn't necessarily mean that would have been a good upsell opportunity so the theoretical performance may not entirely match the practical performance. Successful up-sell is likely to be hard to measure, but depending on the implementation it could be beneficial to run A/B testing on outcomes frequency of recommended SKU combinations to confirm the model is a useful tool.