UL FRI Data Science - Introduction to Data Science, 2021-2022
Interim report

Product recommendation system

David Ocepek[1], Klementina Pirc[2] and Matej Miočić[3]

Abstract

For our project we have evaluated two models FM and deepFM using seen and unseen products. The best performing model was FM with a Top-5 hit rate of 0.64 outperforming the baseline of 0.44, while deepFM scored 0.24. On unseen products FM had a Top-5 hit rate of 0.23, in comparison to the baseline of 0.20, while we did not evaluate deepFM due to its poor performance. Our model enables Siemens to feature ads that match the interests of 2-in-3 customers for old products. and 1-in-4 for newly launched products.

Keywords

recommendation system, products, RankFM, DeepFM

[1] *do8572@student.uni-lj.si, 63160248*
[2] *kp3437@student.uni-lj.si, 63210492*
[3] *mm9520@student.uni-lj.si, 63180206*

Introduction

Sales performance is one of the most important aspects of businesses therefore a lot of effort goes into its advancement. Suggesting new popular products to customers or products adjusted to their needs contributes to mentioned goals and this is where recommendation systems are utilised. Our goal was to create a recommendation system for Siemens which provided us with information about their customers purchases. For the task we resorted to different models and we present our results below.

Data analysis

Our data set consists of 1249 consecutive days for which we have information about sales for each customer. Starting date is unknown and purchase amounts are scaled down. *5HB8_4N50_1I6D* is an example of a product label, where *5HB8* is the family, *4N50* the product and *1I6D* product's variation. Data set contains 2762 customers, 126 families, 2013 products and 12304 variations.

We show the dominance of the most popular family of products in Figure 1, by comparing amount of sales to the second and third most popular family of products. We observe that the most popular family has approximately three times as much sales as the second best. Meanwhile sales of second and third best are more similar. We also observe a seasonality break that recurs every year.

To better understand the diversity of purchases we explore the number of unique variations, products and families for each customer. Results are shown in Figure 2 and we see that most customers bought only a small number of different products.

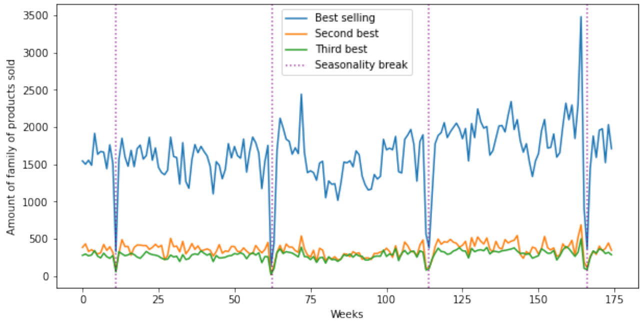

Figure 1. Amount of **top 3 families of products** sold by weeks and a recurring seasonality break of drop in sales.

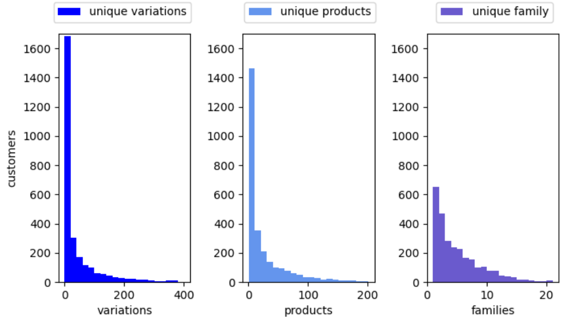

Figure 2. Distribution of unique families, products and variations per customer.

Methods

Walk-forward cross-validation

We are working with a time series data and the model must not learn from the future data, so the more general cross-validation techniques are not correct. Therefore we decided to use walk-forward cross-validation which is a technique that splits the data into folds but still preserves the data order. We decided to use 500 days as train data and 125 days as test data as it allows us to go through all the data in 6 folds (Figure 3).

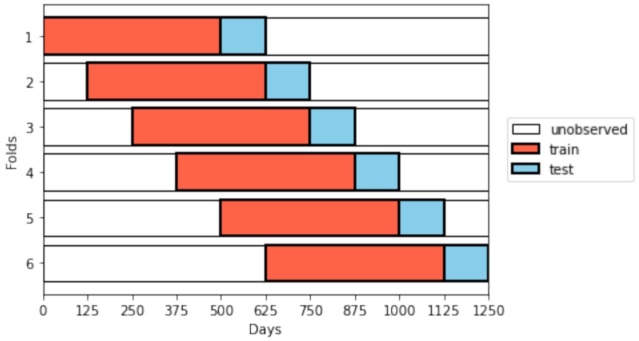

Figure 3. Walk-forward cross-validation. We show how data is distributed into train and test data in consecutive folds.

Baseline

Baseline model is important for understanding the true performance of our models. Baseline values for predictions are calculated by picking the most frequent family for each customer, then predicting top 5 variations or products from this family and finally checking if the customer bought any of them in the future. Family prediction is done independent of customers by always using top 5 families from the train set.

RankFM

Factorization Machines (FM) are generic supervised learning models that map arbitrary real-valued features into a low-dimensional latent factor space and can be applied naturally to a wide variety of prediction tasks including regression, classification, and ranking. FMs can estimate model parameters accurately under very sparse data and train with linear complexity, allowing them to scale to very large data sets — these characteristics make FMs an ideal fit for real-world recommendation problems. [1]

RankFM[1] is a python implementation of the general Factorization Machines model class adapted for collaborative filtering recommendation/ranking problems with implicit feedback user/item interaction data. This means we are not able to utilize explicit data such as the exact day and the amount of items purchased.

DeepFM

Deep Factorization Machines (DeepFM) is a family of supervised machine learning models combining neural networks (NN) and FM. The main benefit DeepFM has over FM is its ability to model higher-degree non-linear interactions. Additionaly, deepFM can use any type of NN such as LSTM or CNN [2].

DeepFM[2] is a pytorch library implementing implicit and explicit models. For our task we used LSTM-FM with implicit negative sampling. We tuned our models hyperparameters using Bayesian Optimization.

Results

	Baseline	RankFM	DeepFM
Variation	0.44	0.64	0.23
Product	0.53	0.74	0.59
Family	0.94	0.97	-

Table 1. Table showing top 5 hit rate for predicting **any** variations, products and families.

	Baseline	RankFM	DeepFM
Variation	0.20	0.23	-
Product	0.22	0.31	-
Family	0.24	0.26	-

Table 2. Table showing top 5 hit rate for predicting **only new** variations, products and families.

From our results we conclude that Siemens can recommend ads users are interesting with fairly high likelihood. The prior is partially due to Siemens customer tendencie of buying the same goods over-and-over again.

Discussion

DeepFM performed very poorly and further analysis is neccesary to conclude whether this failure was because of a coding error or a specific failure of the deepFM model.

References

[1] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[2] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction, 2017.

[1] https://github.com/etlundquist/rankfm

[2] https://github.com/maciejkula/spotlight