Abdallah Maarouf (611063), Sezen Kilicarslan (610164), Winnie Leung (610175), Ecem Selen Seltuk (610402)

Machine Learning in Marketing

WS 20/21 Final Class Project “Coupon Optimization”

By Group 1

1. Introduction

To help Dr. S to continuously improve his grocery store performances by driving up sales and revenues, we explore the use of personalized coupons for individual shoppers to capture additional purchases that are not likely to take place had there been no coupons. Ideally, we would select the price sensitive products for each shopper and trigger the purchase by offering coupons and reducing the price.

The marketing mix tools around coupon strategies have been evolving over time as the objectives of retailers pivoted. Decades ago there has been limited personalization in terms of coupon strategies. Retailers offered coupons with the same discounts on the same set of products to all customers, normally printed on brochures or leaflets sent by mail to households or placed at the entrance of supermarkets. Shoppers would spend time reading through all discounted products, create their shopping list and present the physical coupon at checkout to redeem discounts. The most common personalization applied is geo-customization, where different promotion campaigns are targeted at consumers living in different areas. Nowadays, enabled by vast development of big data and smartphone technologies, retailers are offering us more digital ways to redeem coupons with some degree of personalization in an exchange for our shopping data. Commonly, marketers segment customers according to their demographic data, purchase histories or their stage of customer journeys into different subgroups and target them with different promotional campaigns to drive sales. Customers will be targeted with different discounted product offerings through channels including mail, kiosk machines, mobile apps, email, and much more. While all these factors affect coupon strategy efficiency, in this paper, we will focus on finding the perfect product and price combination for each customer.

When solving this optimization problem, we focused on tackling some issues that marketing managers might still not be aware of. In some cases, some are overly focused on measuring coupon redemption rate as a success measure, while neglecting the fact that they might not be offering coupons to only the customers who wouldn't buy the products with the original price, leading to a revenue loss because of the discounts offered. In other cases, marketing managers only leveraged pre-defined product information such as product aisle or product category information for their analysis and overlooked other latent relationships between products that may characterize a shopper, such as when shoppers only buy baby products or organic products. We will tackle these problems by applying an optimization model that selects products with the highest sales uplift after coupon application, and uncover latent relationship between products by applying neural network models.

Thanks to Dr. S distributing random discounts to random products for random shoppers in the past 90 weeks, we have gathered enough data to take on an all rounded approach to take on this optimization problem.

2. Approach

2. 1. Merging information from different datasets and labelling target variable

We are given two datasets, namely baskets and coupons, which includes purchase information of shoppers from week 0 to week 89, and coupon information which tells which products were purchased by using coupons. We have started to work by merging these two datasets. The problem with the resulting set is that it only consists of the purchase records. That is, there is no record where the target variable would be 0. Therefore, it is a must to generate some records which have a value of 0 in the target variable column; otherwise the model will always predict 1.

Our approach to the creation of such "artificial" records is as follows:

First, we have found the list of all of the different products that a customer purchased during 89 weeks. We assume that a customer purchases among those products when she/he visits the shop. Therefore for each week the set of products that a customer can buy is restricted to the list described above. As a second step, we have created 90 copies of the resulting shopper product combinations to represent the customer's available product options for each week. And then we merged the resulting dataframe with the original dataset we are given with so that the products that s/he purchased will have 1 in the target variable and 0 if not. Another approach of creating such "artificial" records could be to assume that there are 250 products that a customer can purchase every week. However this would result in creating a very huge dataset which mostly has "0" in the target variable column because the number of products that a customer purchases each week is around 10 products on average, which means that the number of records which belong to class 0 (purchased =0) would be 25 times more than the number of records that belong to class 1 (purchased =1).

In the creation of these artificial records, we have assumed that there are no discounts. Therefore, we have set the prices of those products to original prices (no discount) of the products. The original prices are the mode of the prices for every product, assuming that the number of times a product offered with a discount must be less than without a coupon.

2. 2. Training Dataset Size

Feeding a large dataset to train would provide our models more information and eventually yield better targeting performance, but it can also be very costly, time-consuming or even infeasible. With an original dataset that contains 100,000 shoppers' shopping information over 90 weeks, we used only 2,000 shopper's data for model training which accounts for only 2% of the original dataset. Since Dr. S. would evaluate the result on 2,000 shoppers, it was our practical choice for our first model training. Later we iterated through a few testings with different dataset sizes, when we increased the dataset size from 2,000 shoppers to 5,000 shoppers, the training time almost doubled (30min vs 50min) taking up 96Gb ram while our measured accuracy only increased 0.5%. We believe that 2,000 shoppers is a practical and reasonable training set size in our case which also provides good training performance.

While we reduced the dataset size by reducing shopper dimension, we decided to keep all 90 weeks because of the invaluable shopper habits and buying cycle information that we can generate by including time effects in our feature engineering.

2. 3. Finding structures between products in the same shopping basket

In order to find out existing market structures of our products and analyse their within-category substitution or complementarity effects, we adapted a pioneering approach proposed recently where researchers leveraged neural network language model to derive and visualize latent product attributes (Gabel et al., 2019) on a two-dimensional product2vec-map. With the results, we grouped the products by their similarities and created features for our model to learn about how the products compete with each other to enter a shopping basket. Since this model leverages and analyses the co-occurrences of products in shopping baskets, we compared the results with a traditional co-occurrence matrix as in Figure 1., with the product2vec-map showed on the left and a map of product co-occurrences within the same basket on the right.

The graph on the left visualizes the clustering output of a Word2Vec model, where similar products are located close to each other and are grouped into a single cluster which we would use as our product category information. The co-occurrence matrix on the right applies colors to product pairs by the frequency that they occur together in the same basket. By comparing the two approaches, we found that products grouped into the same category by their similarities are never purchased together. From the two graphs, we also observed that the product ids in our dataset are perfectly ordered by their categories, namely the first 10 products (0,...,9) are never purchased together, same as every other next 10 products (10,...,19). Both maps confirm that there are 25 categories to be identified based on product similarity and co-occurrences, we can derive the same results by applying k-means clustering to the neural network output or assigning every 10 products a new category manually.

With this product category information, we can generate new features that will not only account for the products substitutional or complementarity effect, namely how the purchase of one product affects others that are within or not within the same category, but more importantly how distributing a coupon to the product will not only affect its purchase probability but also create side effect to other products. We will give more details about how we leveraged category information to generate useful features for our training set in section 2.5.

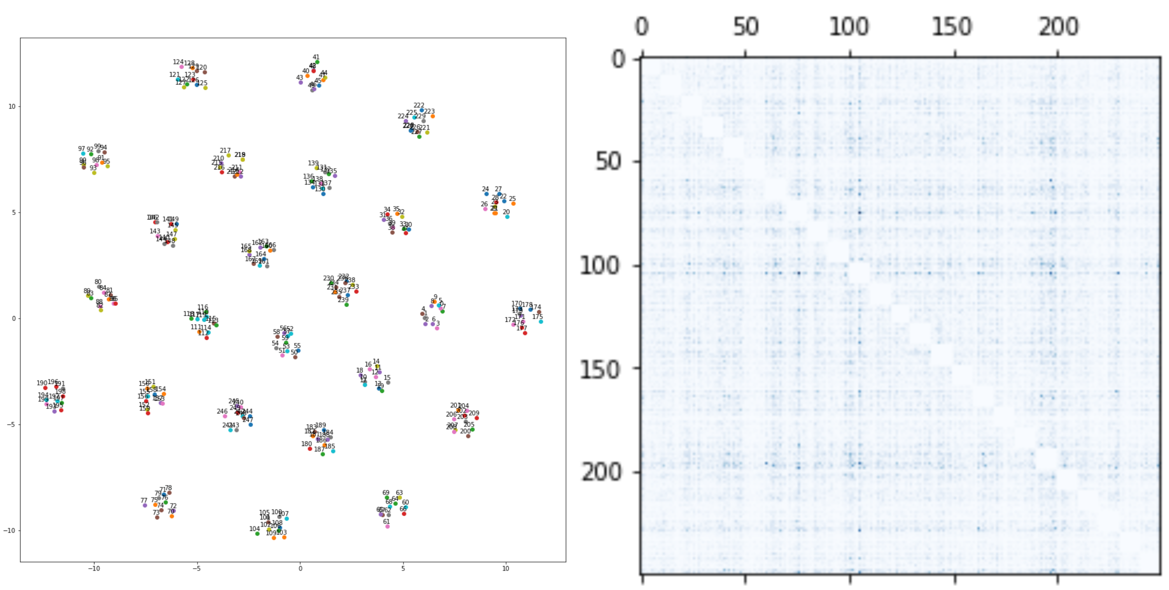
Figure 1: P2V-MAP(left) and product co-occurrence matrix(right)

2. 4. Finding structures between product categories in the same basket

From Gabel's work, we learned that products from different categories would form larger clusters if they are complements with similar use cases and are frequently purchased together. Therefore, after assigning categories to the products, we further investigated how these product categories relate to each other. Similar to product purchase patterns within the same baskets, there might exist category purchase patterns, where some category groups or pairs never or are often purchased together.

In our P2V-MAP, we have seen how all clusters are perfectly separated from each other with the same distance in between. Although we can identify that some categories are more because they are closer to each other, we are not able to make a clear differentiation or create clear groupings between these categories. It is hard for us to identify any larger clusters based on their similarity. We used the same approach to create category2vector-map and visualized categories based on their similarity as in Figure 2 and found exactly the same structure as before.

We continued with creating a co-occurrence matrix of categories in order to see which of them appear more or less often together in the same basket. By clustering the co-occurrences, 3 main groups of categories can be identified. These three groups correspond to the categories on columns 0-2, 3-10 and the rest where they are very often, rarely and are sometimes purchased together respectively. We believe that this information gives us some degree of product complementarity, and included the category cluster information in our training data so that the model can learn if one product has a high purchase probability (with or without coupons), another product from the same category cluster would also have a high purchase probability.

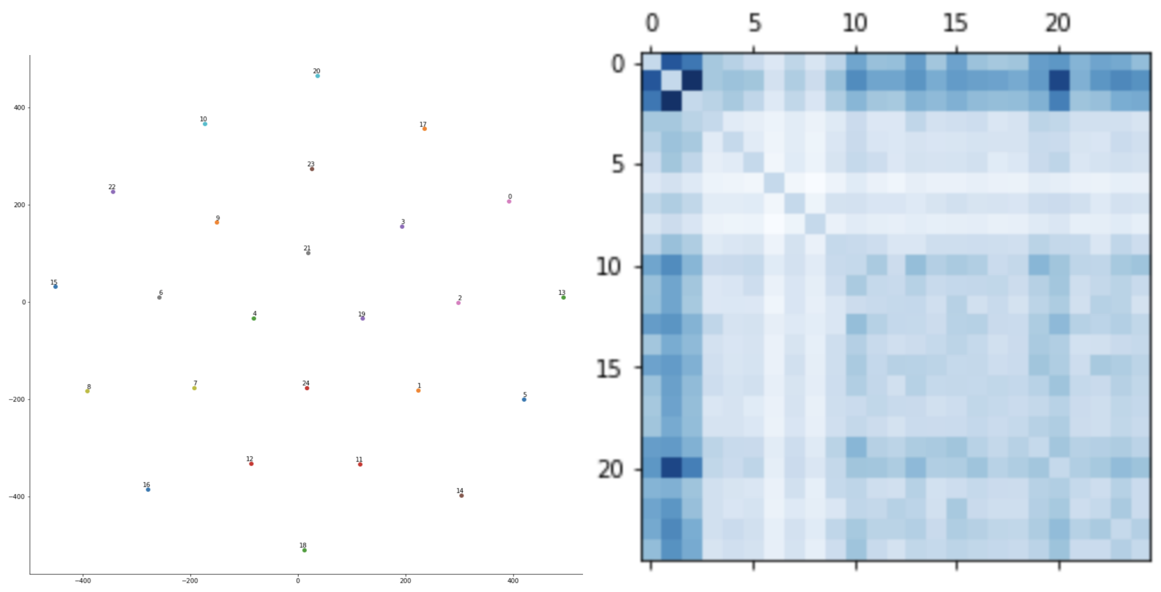

Figure 2: category2vector-map (left) and category co-occurrence matrix(right)

2. 5. Feature engineering

We created multiple sets of features to cover different effects on a product's purchase probability. First, we have the effect of recent category purchases that would model how often a user buys a product or from a specific category. Then, we have the coupon effect that accounts for how applying discounts affect the purchase probability. Last but not the least, we have promotion effects that differ for same category products (substitute effect) or for products within the same category grouping (complementarity effect).

In the first set of features that models for the effect of recent category purchases, we account for shoppers' different purchase behavior for different products over time. First, we calculated 'purchase_history_n' with n $\in$ {3,4,5} indicating how many times a customer has purchased a particular product in the last n visits. Gabel et al. (2020) argues that recent purchases are more relevant for the purpose of modelling purchase histories and n=∞ can be used as a summary of entire purchase history. Therefore, we filtered by time to get a sparse purchase history of every shopper x product combinations as well as added another variable to account for older purchases. Then, we attained purchase cycles of all product and shopper combinations which translates to how many weeks has passed by as a percentage of the shopper's averaged purchase frequency for the product. If the shopper normally buys a product every i weeks, and j weeks has passed by since the last purchase, then it would have a purchase cycle of j/i. For the purchase frequency i, probability of buying the same product one week after the last purchase becomes 1/t and it increases as time goes by.

In the second set of features about product discount effects, we accounted for how applying coupons to a product changes its purchase probability. First, we created a variable called 'indiff_to_price' which indicates whether a shopper has paid more than one unique price for a product. If this is the case we can say that the shopper is indifferent to price changes, where the product was purchased with and without discounts and sometimes with different sizes of discounts applied. To optimise coupon strategies, it would be desirable to avoid offering coupons to products where their shoppers are indifferent to price, because instead of increasing revenue with the help of coupons, we are not necessarily able to increase the product's purchase probability while decreasing our potential revenue by reducing the price of a product that the shopper would also buy without coupons. Additionally, we created variable 'only_with_coup', which is also a Boolean showing if the shopper has only purchased the product with a coupon and never without discount. To create the variable, we searched wherever the product is discounted and the shopper is not indifferent to price changes which means she paid only discounted prices for the product.

Last, we created variables based on categories created from product2vector and co-occurrenc analysis to account for promotion effects within and between the same category. We were well aware that the same category and substitute products would never appear in the same basket, and at most one product from a category would appear in the shopping basket at the same time. Thinking that we might further leverage the variable 'only_with_coup' to reveal the average discount needed for a particular customer to buy from the category, we calculated the average size of coupons given to a shopper in each category and named it as "average_coup". To account for substitute/within-category promotion effect, we created a 'despite_coupon' variable which indicates whether the shopper was purchasing an undiscounted product from a category although she received a coupon for another product in the same category. If this is the case, the substitution effect of this product category for this shopper is very low, since the shopper is loyal to a specific product and is not likely to try out other similar products with more attractive prices. To account for promotion effects of complementary products, we created weight of evidence for the category clusters that we applied to indicate to the model that there might exist a latent relationship.

2. 6. Train-test split approach (Evaluating solutions offline)

Evaluating solutions offline is usually done with the data at hand by taking one fifth of the data randomly as a test set which consists of unseen data and the rest is used to train the model. In other words, offline evaluation is performed on a newly trained model with available historical data which is unseen by the model. According to Parunov (2019), the goal of the offline evaluation is to get the performance of a model on data points which are present in the database and select the best model up to a knowledge at the time of training. Hold-out validation, cross-validation and bootstrapping are the mechanisms of offline evaluation.

Brownlee (2019) asserts that usual methods such as k-fold cross validation do not work in time-series related data due to ignoring the temporal component of the inherent problem. Since we deal with a dynamic machine learning problem where the environment changes over time, we should use a splitting approach that is based on time in order to have a more realistic and robust evaluation. However, our problem is not an ordinary time series while having a substantial time dimension. Then we should utilise 'time-based cross validation' which is a highly favourable method in time related splitting tasks (Herman-Saffar, 2020). Time-based cross validation forms ‘sliding-window’ training which is the use of prior time steps to test the next time step (see Figure 3).

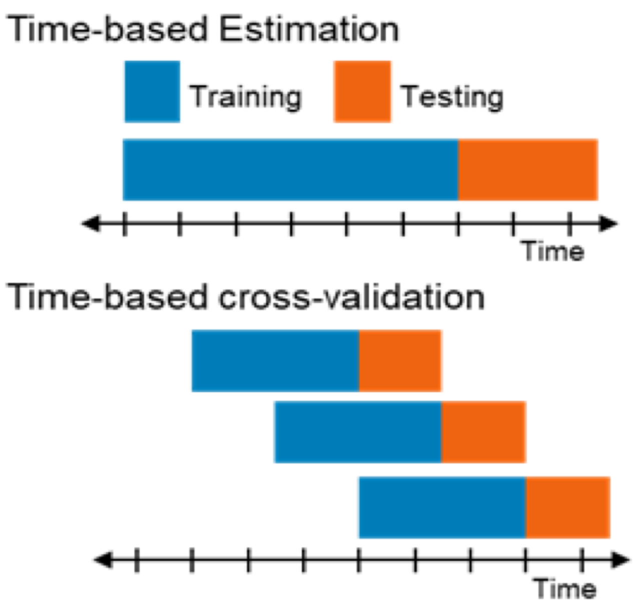

Figure 3: Time-based cross validation approach

In order to create robust and general models, we will use several splitting points in time and then apply time-based cross validation. For our purposes, we chose to train on the first 84 weeks and then test on the 85th week. In the second step, we train the model on the weeks between and including 2-85, then test on the 86th. We slide the windows up until 89th week is tested, keeping in mind that the testing period must always be the same size i.e. one week. The final test result is the weighted average over all testing windows.
We could have generated more sliding windows in other words training on fewer weeks, however it would be computationally costly.

In her article, Herman-Saffar (2020) summarizes the drawbacks of Scikit-learn’s TimeSeriesSplit method: 1) When splitting the data in train/test sets, the method doesn’t allow to specify set sizes but the number of splits. 2) It assumes that there is only one observation per date. Thus, we utilised her suggested solution on towardsdatascience.com which allows us to choose relevant window size.

Parameters of her cross validator are ‘train_period’ which is for us 1-84 and 2-85 so on, ‘test_period’ which is 84-85 and 85-86 etc. Lastly, ‘freq’ is needed in the case where time is given in date format, however it has to be defined in order to create the class object. It can also be simply deleted in our case. Moreover, the methods are ‘get_n_splits’ which returns the number of splitting iterations in the cross validator and ‘split’ which returns a list of tuples as train and test indices similar to sklearn cross-validators. The metrics that the cross-validator uses to get accuracy is R^2, log-loss and ROC-AUC.

2. 7. Model Selection

There are 3 models that we have trained to predict the probability of purchases. These models are Logistic Regression, Random Forest Classifier and XGBoost classifier. The dataset to be trained is very large; therefore, we chose the models that can be trained in a limited time window. For example, knn is expected to be slower than linear regression because of the costly calculation of the distance matrices or neural networks would take longer than logistic regression depending on the network size. In the table below auc scores Log-Loss values of the models that we tried can be seen. The best performing model on our training data with 2000 customers is XGBoost classifier with an auc score of 0.835. The 5 week cross validation lasted around 35 mins and the training with the entire dataset lasted 15 mins for XGBoost.

	XGBoost	**Logistic Regression**	**Random Forest**
AUC	0.835	0.825	0.824
Log-Loss	0.259	0.277	0.279

Table 1: Comparison of deployed models versus metrics that were used

2. 8. Feature importance

After the model selection, the features which add the most gains to the model accuracy are selected to be used in the final training and prediction. The most important 5 features are listed below (see Table 2):

'Cumsum'	1444.02261
'purchase_cycle'	207.85005
'last_occurance’	199.15393
‘discount_x'	63.15436
'purchase__history_5'	241.45364

Table 2: The most important 5 features that lead to a gain

The most important feature is Cumsum which refers to the entire purchase history. Second most important one is the purchase history of the last 5 visits. As it is stated in Gabel et al., 2019, the recent purchase history is important besides the entire purchase histories. Purchase cycle and last time that the product was purchased were also among the top 5 features. They define the purchase pattern for a shopper and a product and they help models to make more accurate purchase probability predictions. Lastly, feature importance indicates that discount amount is also an important factor in the customer purchase decision process.

3. Results

3. 1. Selecting five products to assign coupons to

The predictions were made based on the model and features that are explained in the previous sections. In the test set, there are 2000*250*5 probabilities to be predicted - for each shopper (2000 shoppers) predict purchase probability for every product (250 products) discount (5 types of discount (0 discount also included)) combination. Given the purchase probabilities, the revenue from a single customer is calculated as in the following equation:
The total revenue is calculated as follows:

$$R = \sum_j p_j d_j (pr_j)(1 - d_j)$$

There are 5 coupons to be given to an individual customer. Coupons can have 0.15, 0.20, 0.25, and 0.30 discounts. Knowing the probability purchase of a customer for all product discount combinations, it is straightforward to select the 5 product-discount combinations that increase the revenue the most compared to not giving a discount at all. If the revenue uplift is maximized for every individual customer, then it applies that the total revenue across all customers is also maximized.
The selected model was given week 90 data, of all unique shopper-product combinations that did appear in the history of purchases, and asked to predict purchase probability 5 times on all provided data, once with no discount applied, and once for every discount to be offered, 15%, 20%, 25% and 30%. Expected revenues were calculated accordingly using the total revenue function. Revenues with no discount were subtracted from coupon-applied expected revenues to provide uplift. 5 most uplifting revenues per shopper were selected to have coupons with the maximizing discount percentage, and assigned coupons 0 to 4, from most to least.
The maximization part may not be straightforward if there were more constraints to the coupon recommendation problem. For example, it can be the case that a customer cannot receive more than one coupon with 0.3 discounts. When such constraints are also given, then it is recommended to implement integer programming.

3. 2. Limitations of our approach

Due to time constraints, we weren't able to apply the following improvements to our model. Firstly, to enhance our model, we could have added the effect of recent category purchases. That way the probabilities of buying a product from the category would be decreased in the following weeks which will lead to a more accurate prediction. Secondly, we would evaluate our results by analysing a random product through a discount in a complementary category as well as a discount in a substitute in the same category. In the third place, we couldn't deploy any other reference models than Logistic Regression and Random Forest, we believe we could have achieved a higher accuracy with a Neural Network or k-Nearest Neighbor since for NN we need larger data but it would have been slower to deploy when compared to, for example, Logistic Regression. On the other hand, the average accuracy is always better with a neural network. Another limitation that has arisen due to limited time is that we weren't able to apply hyperparameter tuning which also could have resulted in a better accuracy. Additionally, in the feature engineering process, we could have created extra variables besides cluster weight of importance to account for the complementarity effect.

Bibliography

Brownlee, J. (2019). How to BACKTEST machine learning models for time SERIES Forecasting. Retrieved March, 2021, from https://machinelearningmastery.com/backtest-machine-learning-models-time-series-forecasting/

Gabel, S., Guhl, D., & Daniel Klapper. (2019). P2V-MAP: Mapping Marketing Structures for Large Retail Assortments. *Journal of Marketing Research*, *56*(4), 557-580. sagepub. 10.1177/0022243719833631

Gabel, S., & Timoshnko, A. (2020). Product Choice with Large Assortments: A Scalable Deep-Learning Model. Electronic copy available at: https://ssrn.com/abstract=3402471

Herman-Saffar, O. (2020). Time based cross validation. Retrieved March, 2021, from https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8

Parunov, A. (2020). Fundamentals of machine learning model evaluation. Retrieved March, 2021, from https://medium.com/heyjobs-tech/fundamentals-of-machine-learning-model-evaluation-f2bae6eb5d36#_=_