

Final Class Project

Coupon Optimization

by

Group 6

Sophie Bonczyk (561846)

Anna Franziska Bothe (576309)

Christopher Gerling (598370)

Asmir Muminovic (600582)

Arash Moussavi Tasouj (600651)



Humboldt University of Berlin
School of Business and Economics
Chair of Marketing
Dr. Sebastian Gabel

in fulfillment of the requirements
for Machine Learning in Marketing

March 21, 2021

Contents

List of Figures	ii
List of Tables	iii
1 Introduction	1
2 Data	2
2.1 Data Set	2
2.2 Pre-processing	3
2.2.1 Product categories	3
2.2.2 Negative sampling	3
2.3 Feature Engineering	3
2.4 Train-Test-Splitting	4
3 Methodology	5
3.1 Proposed LightGBM model	5
3.1.1 Hyperparameter Tuning Approach	5
3.1.2 Final Model Selection	6
3.2 Evaluation of the prediction	7
3.2.1 Description of baseline models	7
3.2.2 LightGBM versus XGBoost	7
3.3 Coupon assignment and revenue maximization	8
4 Discussion	9
5 Conclusions and further research	10
References	11
Appendix	12

List of Figures

1	EDA of no_unique_products and no_products_bought	12
2	Product Categories with k=25 clusters	12
3	The AUC of the training and testing set of the final LightGBM after 300 iterations	14
4	The binary log loss of the training and testing set of the final LightGBM after 300 iterations	14

List of Tables

1	Features Engineering	13
---	--------------------------------	----

1 Introduction

In 2020, U.S. retailers distributed over 470 billion coupons for packaged consumer goods. At the same time, less than 1 percent of the coupons issued were redeemed (Statista, 2021). Yet, 90 percent of all consumers use coupons, 43 percent of them regularly (very often) (Valassis, 2019). Nevertheless, half of consumers would redeem more coupons if it was easier to find coupons for the items they really need. They also judge the search effort of coupons for certain items to be too high, which leads to a low redemption rate (Valassis, 2019).

To increase coupon profitability, retailers adopt solutions to provide personalized coupons to individual customers. Yet, many supermarkets and retailers rely very often on manual processes based on intuition and past experience to decide price promotions (Cohen and Perakis, 2020) since implementing models to predict purchase behavior across the entire assortment can be challenging (Gabel and Timoshenko, 2020). In the era of big data and artificial intelligence, advances in digital technologies have however dramatically increased the amount of information the sellers can gather about their customers (Gabel and Timoshenko, 2020). By using appropriate machine learning models, retailers can determine consumer preferences and provide tailored coupons to consumers that they need.

The overarching goal of this paper is to extract information from a large set of historical transaction data from a grocery retailer to improve its promotional decision making. Our “real-life” data set comprises weekly purchases by 2000 customers of 250 products in 90 weeks and randomly distributed coupons. Further objectives of this paper are: (1) create a high dimensional feature set and come up with sophisticated train-test splitting methods to account for the time-series aspects in the data. (2) Design and optimize state-of-the-art machine learning models in order to make purchase predictions and compare their average performances. We choose two recent and most widely used tree-based models, XGBoost and LightGBM, and one heuristic model. Among these, our final model for purchase predictions is LightGBM because it outperforms the other models. (3) Assign and choose 5 top ranked coupons to all

customers for the following week that yield the highest revenue uplift for the retailer.

The mathematical formulation of the learning problem we try to solve is:

$$D_{it}^* = \underset{D=[d_1, \dots, d_J]}{\operatorname{argmax}} \sum_{j=0}^J price_j * [(1 - d_j) * p_{itj}(D) - p_{itj}(0)] \quad (1)$$

where the last term $p_{itj}(D) - p_{itj}(0)$ is the predicted purchase probabilities for shopper i and product j at time t . The first probability $p_{itj}(D)$ is the probability of purchase with coupon assignment D and the second probability is the predicted purchase without coupon assignment. The final predictions for the next week (week 90) are based on our proposed LGBM model and our feature engineering with multiple dimensions. We add a logical layer that uses these predictions and calculates the expected revenue for each discount to provide the retailer with the top 5 coupons that maximizes its revenue.

The following section 2 describes the pre-processing steps that we took to get to our final dataset for prediction. In Section 3 we describe our methodology for solving the mathematical formula we have established. At first we explain our final LightGBM model selection and evaluate the performance compared to baseline models. Then, in our logical layer, we assign coupons to the prediction and maximize the revenue. We discuss our results in part 4 and conclude the paper in section 5.

2 Data

2.1 Data Set

We employ data that matches the consumption of 100,000 customers from 250 products across 90 weeks. This study intends to provide the first 2000 shoppers with 5 coupons each, so that they optimize the generated revenue.

In terms of basket sizes, shoppers bought on average 71 unique products, which generated 704 transactions throughout the period. To show the distribution, Figure 1 represents how shoppers vary in their shopping patterns. Although weekly baskets value 48 euros on average, only 1% of baskets include discounted products on average. According to the dataset, our shopper with the largest product variety bought about

145 different products in 1189 transactions whereas the shopper with the smallest variety bought only 30 items in 289 shopping transactions.

2.2 Pre-processing

2.2.1 Product categories

The most important aspect in predicting potential future purchases is to understand how the shoppers select the items for their shopping baskets. With help of the baskets data, it is possible to divide products into different categories using a Word2Vec model. Once trained, this model can detect similarities between products, which are then represented as 30-dimensional vectors. As figure 2 in the appendix shows, we are able to distinguish offered products into 25 different categories in different dimensions.

2.2.2 Negative sampling

Shoppers tend to buy only a fraction of the whole product assortment. They have individual preferences and at each shopping event they select only a fraction from those. We use negative sampling to enrich our data and compute a more robust model. In addition to the coupon dataset, we have added further negative samples to our dataset so that our final model learns to differentiate between purchases and non-purchases. The introduced negative samples stem from a preference list of every shopper which contains products that are bought at least 3 times throughout the 90 weeks period. The number of samples added corresponds to the size of the shopping basket for each individual shopper in the corresponding week and are randomly selected.

2.3 Feature Engineering

Several features have been engineered to improve the predictive power of the models (Appendix, table 1). The engineered features can be divided into the following four major categories:

- *Customer related features*: these features capture customer’s shopping behavior and compute for example the average basket size or the generated revenue of each

customer.

- *Product related features*: these features contain information about popularity of products among customers and how many times a specific product is sold.
- *Customer-Time related features*: these features help with the understanding of shopping patterns and shopping frequency for each product by each customer. These features obtain necessary information about time intervals and recurrences of a product purchase for a specific shopper.
- *Customer-Product related features*: with these features we learn for instance the proportion of discounted products in every shopper's basket, check for average price of products that the customer buys and calculate the share of a product in comparison to all others that were bought by that specific shopper.

2.4 Train-Test-Splitting

For the train-test splitting, we decided on an approach that takes place in 2 steps (with the modules *module_generate_datasets* and *module_train_test_splitting*). The function *generate_dataset* executes our feature engineering and outputs a train and a test set. The main part of the feature engineering is calculated separately, over a different span of weeks. The other part of the feature engineering is the same for both data sets (load, merge coupon basket datasets, creation of the target variable, adding temporal features and imputing missing values that were created by the merging). At first, we generate a train set for weeks [0-88] and a second data set with weeks [1-89]. The two data sets differ slightly and we can therefore more easily see whether our prediction models are overfitting or not. The split in train and test set before feature engineering is essential since bias in the features are reduced and we do this to ensure a better generalization of our models. The bigger the difference in time spans, the greater the impact. The time duration for train and test sets for the final predictions (after the feature engineering part) are separate from each other and do not overlap. For each prediction model, all the training was done on a train dataset while validation and evaluation were made on a test dataset. The prediction datasets are comprised of the

weeks [0-88] for the train set, weeks [89] for the test set. While tuning the models we also introduced an evaluation set; however, for the final model we excluded it since the model is best at predicting only one week in the future.

Our final datasets (train, test) for prediction comprises 26 features after train-test-splitting occurred. We dropped from the first generated dataset 9 features due to several reasons. First of all, we dropped our target variable 'product_bought'. Also, the features 'shopper' and 'product' were removed. The 'shopper' feature represents unique Ids which do not add additional information to the dataset. The information of products is stored inside the "category" column, based on the Word2Vec model. Hence, we reduced the 250 product dimensions and kept the category level feature. Moreover, we removed four variables due to target leakage: 'discount_effect', 'discount_offered', 'purchase_w/o_discount' and 'no_purchase_with_discount'. Finally, two features were dropped because of non-reproducibility for week 90 (our final prediction week). 'week_basket_size' and 'week_basket_value' are features that calculate the basket size and respective value for each week individually. Since we do not have data for week 90, we dropped those variables from our feature set.

3 Methodology

3.1 Proposed LightGBM model

3.1.1 Hyperparameter Tuning Approach

Tree-based models are competent for handling multidimensional and complex data structures. Moreover, gradient tree boosting models are well-known to give good performance and have been widely used in supervised machine learning tasks (Qadeer et al., 2020). In order to benefit from mentioned strengths, our proposed machine learning model for the prediction task is Light Gradient Boosting Machine (LightGBM).

In contrast to many other tree algorithms that use depth-wise tree growth, LightGBM uses a leaf-wise tree growth algorithm which allows it converge much faster.

However, the parameters need to be carefully chosen since the leaf-wise growth may be overfitting if not used with the proper parameters (LGBM, 2021). To perform the classification task, we used 'LGBMClassifier' from lightgbm Python package for the implementation of the LightGBM model. Since it was necessary to find the best parameters for each model by using a customized search approach, we did not only train the LGBM model with different parameters, but also tried different stopping rounds to get better results for the tree-based model. We used gridsearch for the hyperparameter tuning. For training phases and optimizations, AUC and binary log loss were chosen as metrics, taking the prediction task of binary classification into account. Additionally, we checked whether the trained model predicts the two classes of the outcome variable using a confusion matrix. Later on, we display the performance metrics as a plot to see if the model in question is overfitting. Hence, we performed a holistic assessment to be able to evaluate different models and choose the final model for the prediction.

3.1.2 Final Model Selection

We trained 9 models and have taken special notice of fine-tuning different parameters, with all other parameters set to default. Model 1 and 2 tackle overfitting, with a focus on 'num_leaves' and 'max_depth' parameters. Model 2 additionally fine-tunes the 'learning_rate' for better accuracy. Model 3 attempts to increase the accuracy with a smaller learning rate but also a greater number of 'n_estimators'. In order to prevent overfitting the parameters 'reg_alpha', 'reg_lambda' and 'subsample' are introduced. Model 4 does a fine-tuning of the regularization parameters in combination with larger number of leaves. Model 6 tests the boosting type 'dart' instead of the default parameter which is 'gbdt'. However the number of estimators is set to a higher level, models 7 and 8 are quite similar. Lastly, we chose the three best performing models (model 3, 4 and 8) and fine-tuned a last model 9 with a combination of their best parameters. For LightGBM, the final and best parameter setting is: objective = 'binary', learning_rate = 0.01, n_estimators = 300, num_leaves = 1000, subsample = 0.5, reg_alpha = 0 and reg_lambda = 0.5, with number of early stopping rounds of 50. The reached AUC score is 0.9193 and binary log loss is 0.3580, see figure 3 and 4 in

the appendix.

3.2 Evaluation of the prediction

3.2.1 Description of baseline models

In order to evaluate the performance of our LGBM model we compare our results with other models as baseline. First a heuristic model which replaces our entire pipeline of LGBM modeling to get purchase predictions and the subsequent logical layer. It basically computes the whole task in a rule based manner without using a machine learning model. It will be evaluated in Section 3.3. We assume that a customer will buy a product with a coupon if she has done so previously. In order to measure that, we calculate the purchase rate with a coupon for each shopper. The items are then sorted in descending order by the coupon redemption rate and the top five products are selected. In the following week, here week 90, the shopper is then offered a discount for these top products. Second, we chose XGBoost as another gradient boosting model like LightGBM, but in contrast to that grows its tree depth wise. Using AUC and binary log loss, as our performance metrics, The best parameters for XGBoost model are: `num_round = 100`, `max_depth = 10`, `eta = 0.01`, and `objective = 'binary logistic'`, with all other parameters set to default. The performance metrics are the following: AUC is 0.8809, log loss is 0.4803. Besides the slightly better performance of the LightGBM, the computational time of LightGBM is much fast.

3.2.2 LightGBM versus XGBoost

In comparison, our final LightGBM model has a similiar AUC to the baseline XGBoost model. The performance of machine learning models is often measured in terms of AUC (the area under the ROC curve) and also in terms of a loss function. The log loss of the LGBM model is 0.1223 points lower than the log loss of XGBoost, which shows it's better performance. However, to assess the overall level of the log loss, it would be necessary to consider the balance of classes in the datasets used for predictions (the log loss depends on the balance of the classes, their prevalence in the observed dataset).

A log loss of 0.67 is a non-informative value, when the classes are balanced in a 60:40 way (Zammito, 2019). This figure is obtained by predicting $p = 0.5$ for any class of a binary problem. In our pre-processing part (Section 2.3), we employed negative sampling to make our model more robust. Hence, we can state that our LGBM log loss compared to the non-informative value of 0.67 is better with 0.3580. Furthermore, because of the creation of the train-test splits on different time spans, we do not have large discrepancies between predictions made on our test and evaluation sets. This also contributes to a good AUC of 0.91, which we consider not to be overfitted.

3.3 Coupon assignment and revenue maximization

Our final LightGBM model is used to make predictions for the various discounts offered by the retailer. Discounts are to be granted in the following range: 15%, 20%, 25%, 30%. First we need to create our data prediction set for week 90. 26 features can be reproduced for this last week. The target variable 'product_bought' is deleted. Furthermore, we needed to recalculate two variables: the lag variables for week 90 separately and as we do not know which products are bought in week 90, we added for each shopper its whole preference list. The next step is to create four different test datasets for each level of discount, including the columns from the week 90 data set. Then we make predictions for each discount and finally assign our final LightGBM model and then shopper and product features are merged from week90 dataset to the individual data records for the discounts. This "pre-processing" is a necessary step for getting to our final dataset of our logical layer. The logical layer aims at giving the top-five discounts to the 2000 shoppers by maximizing also the expected revenue of the retailer. The input in our coupon_assignment function are the final prediction datasets for the discounts (15, 20, 25, 30) and as output we get the coupon recommendation dataset yielding the top 5 coupons with maximized revenue we recommend to the retailer to distribute in week 90. We calculate the expected revenue with the values in the column 'price' and multiply it with the purchase probabilities. After grouping by shopper and product, we receive the highest ranked top 5 discounts.

The comparison between our models show: The LightGBM solution matches 6.3%

and the baseline heuristic solution matches 3.2% of the ground truth coupons. Hence, the LightGBM is 93% better than the heuristic model in terms of redeemed coupons.

4 Discussion

We trained different LightGBM models to get the model with the best parameters as our final prediction model for individual shopper purchase probabilities. The proposed LightGBM model provides a forecasting accuracy of 0.79, an AUC of 0.87 and a binary log loss of 0.41. Looking at the performance metrics, we can assume that our model does not overfit. Features of the LightGBM model that contributes to the good performance are: 'avg_no_weeks_between_two_purchases', 'lag_weeks_of_product_per_customer' and 'avg_no_weeks_between_two_purchases', showing the the trained model learns very well the time aspect in our dataset. This means that our model has recognized the temporal structure of the dataset. Our pre-processing approach, with our two-fold train-test splitting and the negative sampling part, also helps the model to perform better. Our first train-test splitting takes place before feature engineering and therefore ensures that our model generalizes well. The second train-test-split for prediction uses for the train set of weeks [0-87]. This means, that the LGBM model predicts final predictions for week 90 that are three weeks apart from the training data. The model therefore did not see these weeks during training. However, the final model predicts very well, our work can be expanded to include a second training round with weeks [2-89], to overcome the discrepancy between training and predicted dataset.

The negative samples make our model more robust in terms of performance metrics, since taking them out, leads to a highly overfitted AUC. Furthermore, we tackled target leakage, by creating a correlation plot to see which variables highly correlate with our target variable. We also checked features that were derived of those highly correlated ones and removed them from our final dataset. The newly calculated lag feature for week90 adds just minimal noise (target leakage) to the whole dataset.

For the sake of this study, using more sophisticated models like a LSTM model, GRU (as it also provides a forgetting gate) or a recurrent neural network with different

input layers, could have been an option since all models have been widely used with time-series patterns. Our methodological approach, with the training of a LGBM model to get best purchase predictions and our added logical layer which assigns top 5 coupons that yield the highest revenue uplift, are inline with the theory presented in our introduction. Our approach solves the mathematical equation put forward.

5 Conclusions and further research

The main objective of the paper is to extract meaningful information from a large set of historical transaction data from a grocery retailer to improve its promotional decision making in the future. In this study, the multi-dimensional features concerning shopper purchase histories form the ground base for optimizing our proposed machine learning model, which in turn helps us to assign to each shopper 5 coupons that maximizes the revenue of the retailer. The results show that a well-optimized LightGBM machine learning (ML) model performs better than any other model used in the study. Although ML models have the ability to map temporal features, it is very important to make the most of the feature engineering and pre-processing, which is then following by fine-tuning the model. Store data and shopping baskets are valuable information that retailers can gather and build their future marketing strategies on. Although they are not easy to perform, such optimizations can increase revenue and enhance awareness when introducing new products, foster customer loyalty and improve the overall retailer’s competitive position. Product choice prediction and target marketing are topics that can be explored in depth as avenues for further research. Rendering the model involving even more feature engineering and including different time spans of training data for modeling, may boost the performance. Further improvements in the advancement of ML models and neural network models can also sharpen our model. Understanding and tracking infrequent purchases that were not involved in our analysis and purchase predictions could help us further improve the performance of our models in the future.

References

- COHEN, M. C. AND G. PERAKIS (2020): “Optimizing promotions for multiple items in supermarkets,” in *Channel Strategies and Marketing Mix in a Connected World*, Springer, 71–97.
- GABEL, S. AND A. TIMOSHENKO (2020): “Product Choice with Large Assortments: A Scalable Deep-Learning Model,” *Available at SSRN 3402471*.
- LGBM (2021): “Parameters Tuning,” <https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html#tune-parameters-for-the-leaf-wise-best-first-tree>, accessed 21.03.2021.
- QADEER, K., W. U. REHMAN, A. M. SHERI, I. PARK, H. K. KIM, AND M. JEON (2020): “A Long Short-Term Memory (LSTM) Network for Hourly Estimation of PM_{2.5} Concentration in Two Cities of South Korea,” *Applied Sciences*, 10, 3984.
- STATISTA (2021): “U.S. Coupon Market Trends - Statistics and Facts,” <https://www.statista.com/topics/1156/coupon-market-trends-in-the-united-states/>, accessed 21.03.2021.
- VALASSIS (2019): “Valassis 2K19 Coupon Intelligence Report,” www.valassis.com.
- ZAMMITO, F. (2019): “What’s considered a good Log Loss in Machine Learning ?” <https://medium.com/@fzammito/whats-considered-a-good-log-loss-in-machine-learning-a529d400632d>, accessed 21.03.2021.

Appendix

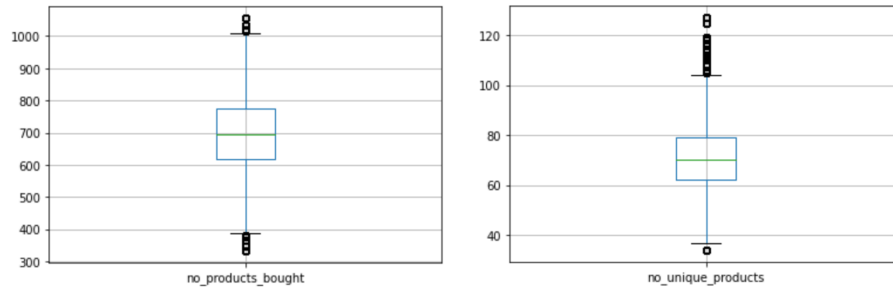


Figure 1: EDA of no_unique_products and no_products_bought

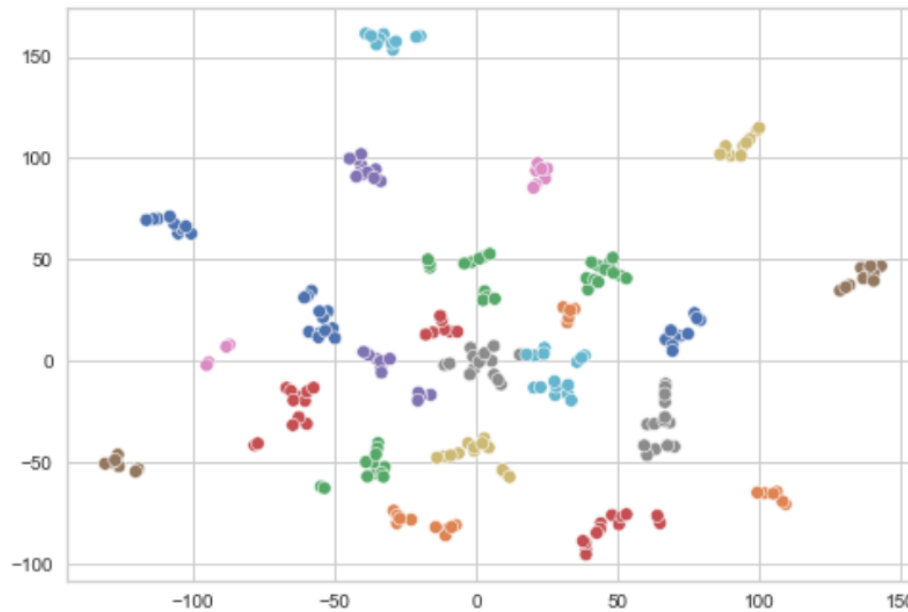


Figure 2: Product Categories with k=25 clusters

Table 1: Features Engineering

Feature	Decription	Inlcuded
week	week t	x
shopper	shopper i	
product	product j	
price	product price	x
discount	offered discount in %	x
product_bought	binary indicator - product j was purchased (1/0)	
avg_no_weeks_between_two_purchases	average number of weeks between two purchases of product j	x
lag_weeks_of_product_per_customer	no weeks since last purchase of product	x
purchase_temporal_distribution	indicator for temporal distribution of prchases per shopper per product	x
category_label	category label	x
discount_offered	binary indicator - product j was discounted (1/0)	
purchase_w/o_dis	binary indicator - product j was purchased without a discount (1/0)	
no_purchase_w_dis	binary indicator - product j was not purchased with a discount (1/0)	
discount_effect	binary indicator - product j was purchased with a discount (1/0)	
max_price	maximum price of product	x
min_price	minimum price of product	x
no_products_bought	number products bought by a customer i	x
spend_per_customer	customer lifetime value (sum € spend by a customer i)	x
no_unique_products	number unique products bought by customer i	x
discount_purchase	number products bought at discount by a customer i	x
product_sells	number of times the product was sold	x
product_dis_sells	number of times a product was bought with a discount	x
product_dis_sells_share	share of product with a discount	x
no_products_bought_per_product	no product j purchases for customer i	x
customer_prod_dis_purchases	number purchases of a product j at discount by customer i	x
customer_prod_bought_dis_share	share a product j is bought at a discount by customer i	x
customer_prod_dis_offers	number discount offers of a product j for customer i	x
customer_product_dis_offered_share	share of deemed coupons per customer i	x
customer_product_share	share of product j was bought by customer i in comparison to all other products that were bought by customer i	x
customer_mean_product_price	average price of an item bought by a customer i	x
customer_discount_buy_share	the percentage of products bought at discount by customer i	x
week_basket_size	number products bought by a customer i in week t	
week_basket_value	Total value products in € by a customer i in week t	
mean_basket_value	the average basket value in € of customer i	x
mean_basket_size	the average basket size of customer i	x

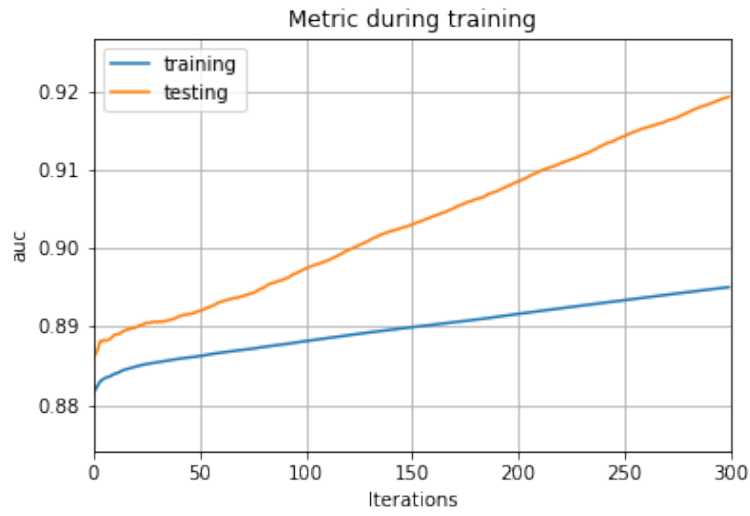


Figure 3: The AUC of the training and testing set of the final LightGBM after 300 iterations

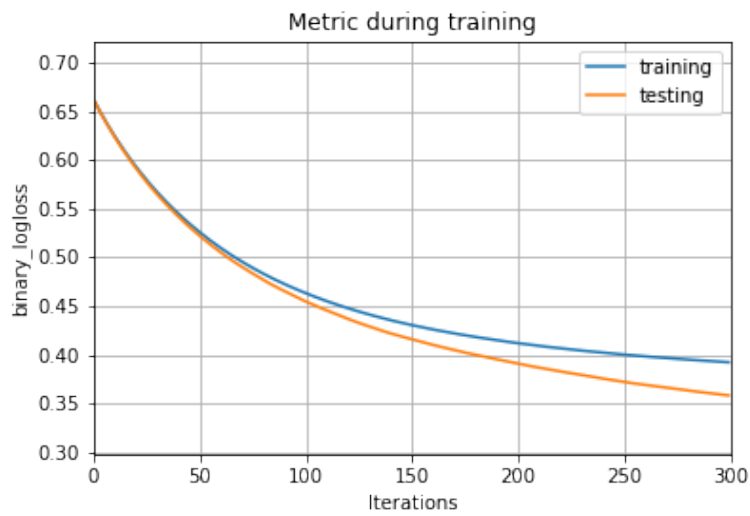


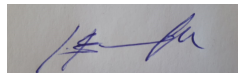
Figure 4: The binary log loss of the training and testing set of the final LightGBM after 300 iterations

Declaration of Authorship

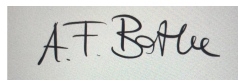
We hereby confirm that we have authored this paper independently and without use of others than the official team members and than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, March 21, 2021

Sophie Bonczyk

A handwritten signature in blue ink, appearing to be 'S. Bonczyk', on a light-colored background.

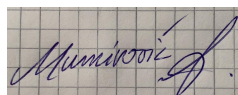
Anna Franziska Bothe

A handwritten signature in black ink, reading 'A.F. Bothe', on a light-colored background.

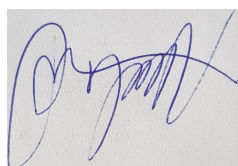
Christopher Gerling

A handwritten signature in blue ink, reading 'Christopher Gerling', on a light-colored background.

Asmir Muminovic

A handwritten signature in black ink, reading 'Muminovic', on a grid-lined background.

Arash Moussavi Tasouj

A handwritten signature in blue ink, appearing to be 'Arash Tasouj', on a light-colored background.