Data Mining in Online Marketing - a case-based analysis

by

Ferrari, Damiano #591538

Haeusler, Konstantin # 551078

Wehrmann, Franziska #592500

Statistical Programming Languages

Humboldt University Berlin

Berlin, March 10, 2018

1 Introduction

Recently, the market share of online retailers increased drastically. The economic importance of this sector is reflected in the growth of revenues generated by online retailers: from 24.6 billion Euro in 2012 to 52.8 billion Euro in 2015 (which represents 1.5% of the German GDP at that time)[1]. This development is accompanied by huge logistic efforts. To avoid undesired shipping costs, one challenge of corporate data mining is to reduce these costs to a minimum. Several papers have outlined the importance of data mining in supporting managerial decision making Manyika et al. (2011). There is a broad literature about each step of the data mining process, from data preprocessing Crone et al. (2006) to the design of data mining algorithms Lessmann et al. (2015). The research in this area is undergoing massive transformations due to technological and methodological innovations, e.g. modern storage capacities allow to process huge amounts of data and new statistical methods evolve in order to analyze this data adequately.

In this seminar paper, a case study is presented that comprises the essential tools of corporate data mining in the context of online marketing. A raw data set of an online retailer with 150.000 observations and 13 variables is given; for two thirds of the observations exists a binary 'return'-variable $y \in \{0, 1\}$ that indicates if a product has been returned by the customer to the retailer (hereafter referred as *known*). The task is to train a model that predicts with high accuracy whether a product of the last third (hereafter referred as *unknown*) of the data set is likely to be returned or not. In order to support the managerial decision making, these predictions are then used to minimize a loss function that consists of type one error $\mathbb{P}(\hat{y} = 1|y = 0)$ and a type two error $\mathbb{P}(\hat{y} = 0|y = 1)$.

		True Value	
		item kept (0)	item returned (1)
Prediction	item kept (0)	0	$0.5 * -(3 + 0.1 * \textit{item price})$
	item returned (1)	$-0.5 * \textit{item price}$	0

Table 1: cost matrix, consisting of type one and type two errors and depending only on item price

[1]Revenues of Online Retailers in Germany 2015, source: https://de.statista.com/statistik/daten/studie/29201/umfrage/umsatz-im-online-handel-in-deutschland-seit-2008/

This seminar paper is structured as follows: *Section* 2 introduces the data. *Section* 3 explains the data cleaning process; *Section* 4 provides some examples of feature engineering; *Section* 5 illustrates novel data set preparation techniques for training and cross validation; *Section* 6 presents the algorithms, their parameter tuning and concludes with their performance scores.

2 Data

The data set has been provided by a german online retailer, more information about the retailer is not known. In total, it contains 150.000 observations, each observation reflects an order placed by a customer at the online store. For the whole data set, 13 (not very informative) variables are given, shown in Table 2. For two thirds of the data set (100.000 observations, *known*), a binary *return* variable $y \in \{0, 1\}$ is given that indicates whether the customer returned the product to the online retailer. The goal of this case study is to train various algorithms in order to predict with high accuracy if the orders of the remaining (*unknown*) third of the data set will be returned by the customer or not.

variable name	type	head	description
order_item_id	int	1, 2, 3, ...	ID of each order placed
order_date	chr	2012-09-04, 2012-11-03, ...	date when order has been placed
delivery_date	chr	2012-09-06, 2012-11-07, ...	delivery date
item_id	int	1507, 1745, 2588, ...	ID of each item
item_size	chr	'unsized', 10, XXL, ...	size of item
item_colour	chr	green, blue, red, ...	colour of item
brand_id	int	102, 64, 42, ...	brand ID
item_price	int	24.9, 75, 79.9, ...	price of item
user_id	int	46943, 60979, 72232, ...	customer ID
user_title	chr	Mrs, Mrs, Mrs, ...	title of customer
user_dob	chr	1964-11-14, 1973-08-29, ...	customer's date of birth
user_state	chr	Rhineland-Palatinate, Brandenburg	shipping destination
user_reg_date	chr	2011-02-16, 2011-05-21, ...	registration date of customers
return	int	0, 1, 1, ...	return variable

Table 2: first look on raw data set, containing 13 variables plus the *return* variable for the *known* data set. Abbreviations: int - integer, chr -character.

3 Feature Engineering

3.1 definition and explanation of variables

In addition to the usual data cleaning process (treatment of missing values, outliers etc.), it is necessary to extract more information from the raw data set by creating new features. Some of the main newly created features are presented in Table 3.

Feature	Description	Explanation
item_retrate	Defined for each item as the mean of returns over the observations (in the train set) that share the same item_id (N_{item} is the number of such objects).	Very distinct for different items, for example some products will look different in picture and in real life. So just for being a specific item it will be a little more or less likely to be returned.
delivery_time	Difference in days between the delivery date and order date.	Customers do not like to wait a long time for their orders and are more likely to be unsatisfied over a certain delivery time.
price_comp	Created by assigning the percentage deviation from the average price of the specific item to each purchase.	Lets assume a customer bought an item on sale, it is more likely that he will keep the item with respect to buying it full price.
item_double	Counts when an item is ordered at least twice by the same user.	It is safe to assume the customers' behaviours follow an inductive pattern, which means they will order items that they enjoyed over and over, making those more likely to be kept. On the other hand costumers are known to order multiple items and return the ones which do not fit them.
item_type	Categorises each item through its size.	For example an item whose size is M is unlikely to be a pair of shoes.

Table 3: Overview of the main features that are used for the predictive model.

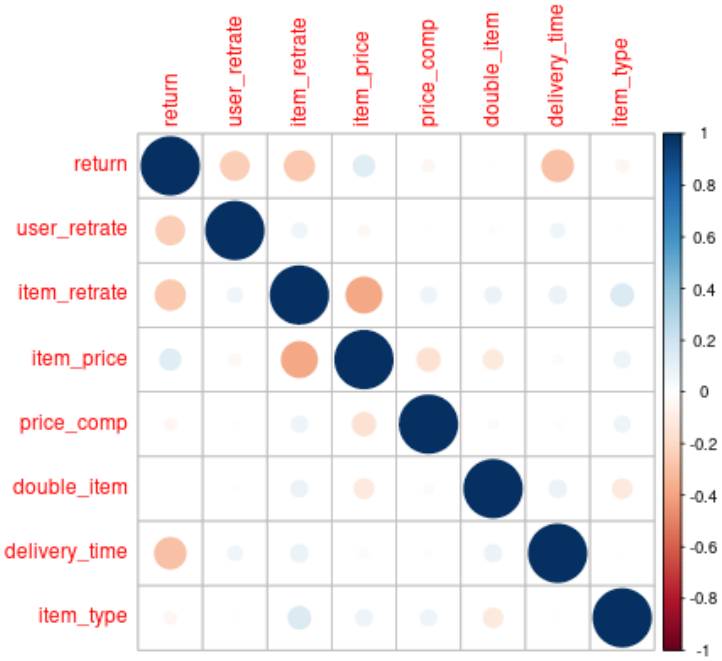


Figure 1: Correlation of features that are used for prediction. Red indicates positive correlation, blue inverse correlation. Intensity of color and size of circle depict correlation magnitude.

3.2 Feature engineering: the case of item_retrate

Lets assume to work on item_retrate, the same reasoning is also valid for user_retrate with only minor changes and can be extended to a wide class of features.

As seen in Table 3, item_retrate is a numerical value for each observation whose item_id appears in the train set at least once. For the observations in test set with an item_id that does not appear in train set, the value of item_retrate can not be calculated, therefore the value *unknown* is assigned. Since most models can not handle mixed class variables, item_retrate must be turned into a factor variable.

3.2.1 Interval boundary choice

For every observation $item\_retrate \in [0, 1]$ is assigned: it is an estimator of the probability that the order will be returned from a random user. It is therefore clear that it is modelled as a Bernoulli trial with

$$P_1 := item\_retrate$$

$$P_0 := 1 - item\_retrate$$

Two questions arise naturally:

1. Is there a reliable way to choose interval boundaries when we need to turn a numerical variable into factor?

   Of course it is preferable to keep numerical variables unchanged, but problems can come up such as novel items in the test set for which it is impossible to infer an item_retrate, and items which were ordered only one time for which the proposed estimation is clearly not appropriate.

2. In which way does N_{item}, the number of times a particular item appears in the training set, influence the reliability and precision of our estimator item_retrate?

   Let's introduce a trivial example: if an item is chosen such that $N_{item} = 3$, only four values are possible

   $$item\_retrate \in \left\{0, \frac{1}{3}, \frac{2}{3}, 1\right\}$$

   In this case it would not make much sense to create a category of item_retrate between 0.4 and 0.5.

To start with, a first interval can be defined

$$I := \big[0.48 - \alpha, 0.48 + \alpha\big]$$

for some α. Note that I represents the items whose item_retrate is within average. In absence of any a priori knowledge about the theoretical distribution of item_retrate it is advisable to select α so that

$$\#\big\{x \in train \mid item\_retrate(x) \in I\big\} = \frac{nrow(train)}{t}$$

where t is the number of quantiles and depends on N_{item}; x represents an observation.
In this setting the mean returns calculation can be modelled as one repetition of t independent Bernoulli trials, therefore following a binomial distribution.
As a first approach the theoretical $P_1 = \frac{1}{2}$ and number of Bernoulli trials 6 are chosen, a local upper bound for the percent correct estimation of P_1 through item_retrate is found.

Frequency	Estimator value
1.5 %	0
9.5 %	$\frac{1}{6} = 0.17$
23.5 %	$\frac{1}{3} = 0.33$
31.0 %	$\frac{1}{2} = 0.50$
23.5 %	$\frac{1}{3} = 0.67$
9.5 %	$\frac{1}{6} = 0.83$
1.5 %	1

Frequency	Estimator value
0.1 %	0
1.0 %	$\frac{1}{10} = 0.1$
4.4 %	$\frac{1}{5} = 0.2$
11.7 %	$\frac{3}{10} = 0.3$
20.5 %	$\frac{2}{5} = 0.4$
24.6 %	$\frac{1}{2} = 0.5$
20.5 %	$\frac{3}{5} = 0.6$
11.7 %	$\frac{7}{10} = 0.7$
4.4 %	$\frac{4}{5} = 0.8$
1.0 %	$\frac{9}{10} = 0.9$
0.1 %	1

Table 4: Classification outcomes for $N_{item} = 6$ (left) and $N_{item} = 10$ (right).

For a binomial distributed random variable X holds

$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Since the function

$$(p^n)(1-p)^n$$

has maximum for $p = \frac{1}{2}$ for every $n \in \mathbb{N}$, we have that 31% (c.f. Table 2, left) is a local maximum in a neighbourhood of $\frac{1}{2}$. This means that there is a local upper bound for percentage successful estimation of P_1 of only around 31% for an interval $[0.4, 0.6]$. Rising N_{item} to 10 can improve this result (see right side of Table 2).

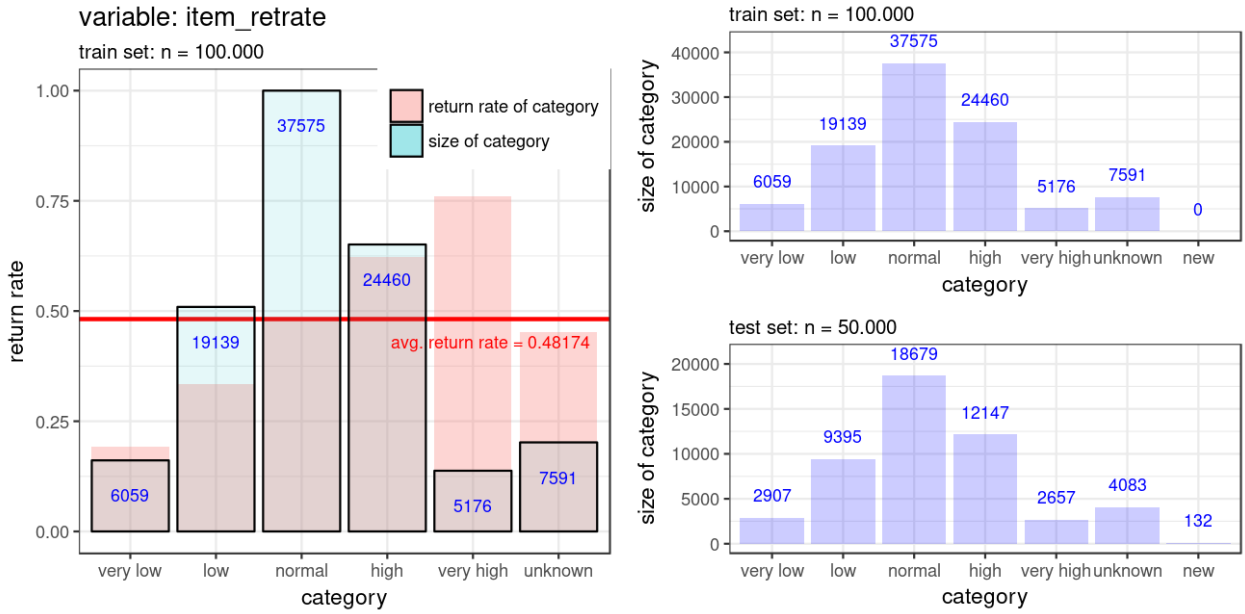
This means that the local maximum success estimation of P_1 already raised to 65.6% $(= 20.5\% + 24.6\% + 20.5\%)$ of the times for the same interval $[0.4, 0.6]$.

In conclusion it is better to raise N_{item} whenever feasible, in this case until too many item_retrate are placed in the *unknown* category. The final choices are $N_{item} \geq 15$ and $t = 7$ for item_retrate; $N_{item} \geq 6$ and $t = 5$ for user_retrate.

The categories of item_retrate are presented in Table 5. Their distribution in Figure 2.

category	very low	low	average	high	very high	unknown	new
mean return	[0,0.26)	[0.26,0.4)	[0.4,0.56)	[0.56,0.71)	[0.71,1]	$N_{item} < 15$	item_id

Table 5: Definition of tau categories. Every order is sorted to an tau category, depending on the price of the item that got ordered.



(a) (b)

Figure 2: (a) Display of the artificially created feature item_retrate. red shading: average return rate of observations of this category with comparison to average return rate of all items in training set (red line, avg.returnrate = 0.48174). blue histogram: number of observations in each category. (b) Distribution of variable item_retrate in train and test set. In both data sets the newly created variable follows a similar distribution.

3.2.2 Weight of Evidence

A method to measure the meaningfulness of a feature is the Weight of Evidence (WOE).

$$WOE(category) := ln\left(\frac{\mathbb{P}(return = 1 \mid item\_retrate = category)}{\mathbb{P}(return = 0 \mid item\_retrate = category)}\right)$$

Since the feature item_retrate was developed by considering the rate of return in the past, the WOE follows the anticipated logarithmic conduct for the categories "very low" to "very high". An WOE = 0 denotes an average chance of return, which is especially the case for the "normal" category. Also "unknown" and "new" have an WOE around zero, since it is not possible to deduce their previous return rate. The WOE of the variable item_retrate is shown in figure 3.

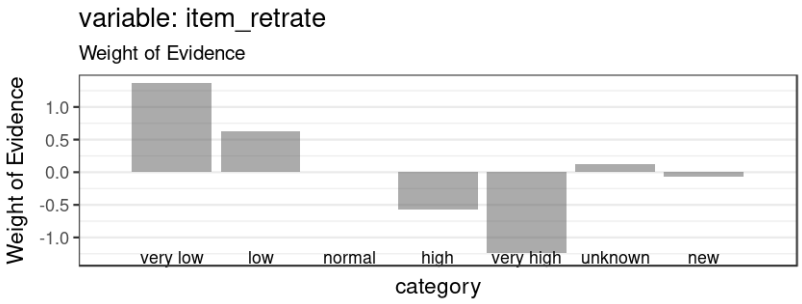


Figure 3: Weight of Evidence (WOE) of the feature "item_retrate". Categories "very low" to "very high" were created by grouping the items according to their return rate in the training set. This created a strong feature in terms of WOE.

3.3 item_type

High cardinality attributes are impractical for most predictive algorithms. The original data set contained the variable item_size, which overall consists of more than 100 categories. Even after merging duplicate naming such as (... xs, s, m, l, ...) and (... XS, S, M, L, ...), the variable is still of high cardinality. One method of reducing the categories of a variable is semantic grouping (c.f. Moeyersoms and Martens (2015)). Categories were generated with respect to common sizing systems for shoes and clothes. Additionally the category "rest" stores items that could not be allocated to shoes or clothes and "unsized" stores items without a specific size. By using semantic grouping it was not only possible to reduce over 100 categories to only four, but also a difference in the return rate of the categories can be observed. Figure 4 shows that "unsized" has a considerably lower return rate than average (Also the return rate of "rest" is lower than average, but because of the small size of the category it will not add much information to the model.)

4 Preparation of Data Set for Models

4.1 Decomposition due to uncertain categories

When creating the variables item_retrate and user_retrate (section 3.2), it is only justified to infer from items/user that occur more than 15/6 times (explanation see 3.2.1). All remaining items/user were stored in the category "unknown". However, this category can not be treated like the categories "very low" to "very high" that are directly connected to the previous return rate of the item/user. To ensure that the predictive algorithm does not treat the category incorrectly, the data set gets decomposed into certain and uncertain subsets, as shown in Table 6. The decomposition ensures that in each subset the remaining columns

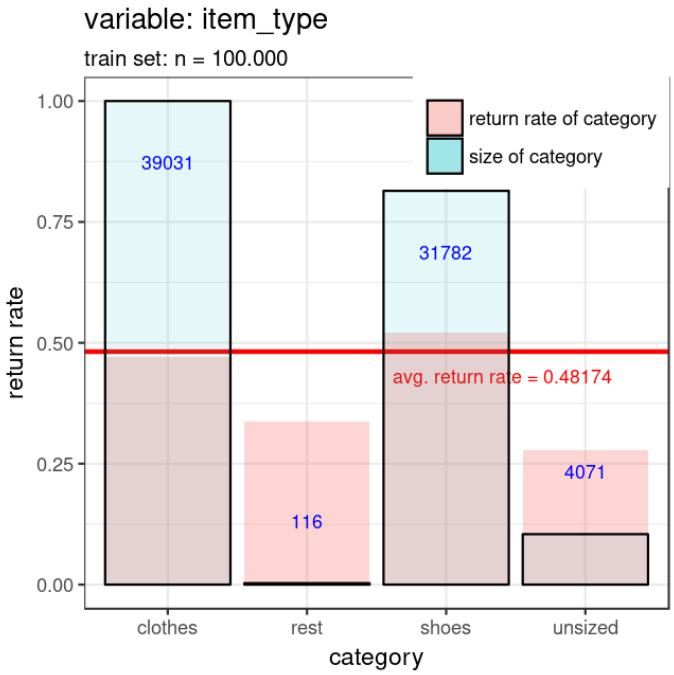


Figure 4: Display of the artificially created feature item_type. red shading: average return rate of orders in this categories in training set. blue histogram: number of orders of each category.

are "pure" (do not contain category "unknown" or "new"). For example subset .u (second row) is the subset, where the column "user_retrate" got removed and "item_retrate" contains only certain categories. Therefore the model is trained on this subset without the feature item_retrate and accordingly predicts the test.u subset. Actually for training the model not only the train.u subset can be used, but also the train.f subset, since it is only important that the remaining columns are pure.

user_retrate	item_retrate	rest	model	size (train/test)
1	1	1	.f - full model	25266 / 10082
u/n	1	1	.u - without user_retrate	67143 / 35703
1	u/n	1	.i - without item_retrate	1857 / 813
u/n	u/n	1	.iu - without item_retrate and user_retrate	5734 / 3402

Table 6: Decomposition of data set due to uncertain categories. u/n stands for "unknown" and "new" categories (= uncertain categories), 1 for certain categories. Categories "unknown" and "new" in item_retrate and user_retrate are uncertain and should not be included in models. Decomposition in pure subsets ensures that model only is fed with certain categories.

In total four models get trained and used for prediction (full model, without item_retrate, without user_retrate, without item_retrate and user_retrate). This decomposition increased the performance from $loss_{full} = -244229.0$ and $AUC_{full} = 65.50\%$ to $loss_{dec} = -238879.0$ and $AUC_{dec} = 66.75\%$ (with a tuned neural network).

4.2 On multiple τ thresholds

By observing the given loss table for this specific task, it is evident that depending on item price, type 1 errors can be more penalising than type 2 errors and vice versa. In order to optimise the results, an analysis of its explicit dependence from item_price is due.
The essential boundaries of when type 1 and type 2 errors are equally relevant are to be assessed, resulting in a meaningful first split of the data. In most cases $\tau < 0.5$ and $\tau > 0.5$ are found to be optimal when type 2 error loss is bigger and smaller than type 1 error loss respectively.
For this particular case study only one boundary is to be found because it is the root of a grade 1 polinomial

$$\Delta := -0.5 \cdot 5(3 + 0.1 \cdot X) + 0.5 \cdot X \tag{1}$$

To decrease the loss even further, a split of the train and test sets depending on item_price is a natural extension of this first remarks. The difference of type 2 and type 1 error magnitudes Δ is a linear function of item_price so a split in a quantilelike way is recommended, but with one of the boundaries being 30 euros (see equation 1).

tau category	1	2	3	4	5	6
item_price	[0,30)	[30,59)	[59,79)	[79,90)	[90,120)	[120, max(item_price)]

Table 7: Definition of tau categories. Every order is sorted to an tau category, depending on the price of the item that got ordered.

Implementing this dependence in the cross validation code is essential not only to be able to estimate accuracy through AUC (and other measures) and tune parameters , but also to obtain an estimate of the specific total loss of a prediction.

Running the cross validation for each of

$$T_k := \{observation \in Split.Train \mid \tau = k\}$$

produces significantly different optimal values for τ (see Table 8), which is a confirmation of what was expected from the above reasoning.

5 Neural Network

5.1 Remarks on Neural Network

A neural network tries to linearise the boundaries of the dataset by applying different layers of weighting and non linear transformations, such as sigmoids (c.f. LeCun et al. (2015)).
nnet in particular tends to perform worse than e.g. a random forest out of the box but often outperforms it once parameter tuning has been carried out.
Parameters that can be tuned are *size* and *decay*: size is the number of units in the only hidden layer of nnet, decay prevents from overfitting.

5.2 Additional Data Preparation

The performance of neural networks increases, when the data is numerical (instead of categorical). Different techniques can be used to convert a categorical variable to a numerical, but not every technique is suitable for every case.
Since item_retrate and user_retrate are based on their connection to the target variable, assigning the Weight of Evidence (WOE) to each category does not aggravate overfitting even more. The WOE is also used for the categories of delivery_time and price_comp, but to prevent overfitting, the WOE was calculated on a small subset of the train set, that is not going to be used for training anymore (c.f.Moeyersoms and Martens (2015)):
If the variable only has few categories, it is beneficial to introduce dummy variables. Instead of having one variable item_type with four categories, there will be four binary dummy variables that indicate whether the item is of this category.

The advantage of dummy variables is, that they are independent of the target variable but since every category requires a new dummy variable, they should only be used for lower dimensional variables (c.f. Moeyersoms and Martens (2015)).

5.3 Parameter Tuning

5.3.1 Parameters of Neural Network

To increase the performance of a model it is vital to use the optimal parameter depending on the input data. Since the data set is decomposed into four subsets, that differ in the composition of predictive variables (see section 4.1), the tuning needs to be performed on every subset separately. In the following the subset test.u (model without the variable user_retrate) is evaluated exemplarily.

The results of the parameter tuning are shown in Figure 5. The data set was divided into six subsets according to their tau-categories (see section 4.2). The tuning parameters are as follows: size of the hidden layer: [3, 5, 7, 9, 11, 13, 15] and decay: [0.01, 0.1, 0.5, 0.8, 1.0].

index	1 - 7	8 - 14	15 - 21	22 - 28	29 - 35
decay	0.01	0.1	0.5	0.8	1.0
size	3, 5, 7, 9, 11, 13, 15	3, 5, 7, 9, 11, 13, 15	3, 5, 7, 9, 11, 13, 15	3, 5, 7, 9, 11, 13, 15	3, 5, 7, 9, 11, 13, 15

Figure 5 shows that for every tau-category the loss is more stable for larger decay values. For higher decay values, the size of the hidden layer has only marginal influence on the loss. Hence it is not necessary to find the exact optimal parameter for every tau-category and train a separate model on this category (within a certain range of parameters). Further it is even beneficial to refrain from splitting into tau-categories, and instead training only one model with almost optimal parameters on the whole dataset. Then the training data carries sixfold as many observations which enables a better training of the neural network.

The sum of the six normed tuning graphs of Figure 5 is shown in Figure 6. The almost optimal parameters are extracted from the maximum of the loss. (index 25 $\rightarrow$ size of hidden layer: 9, decay: 0.8).

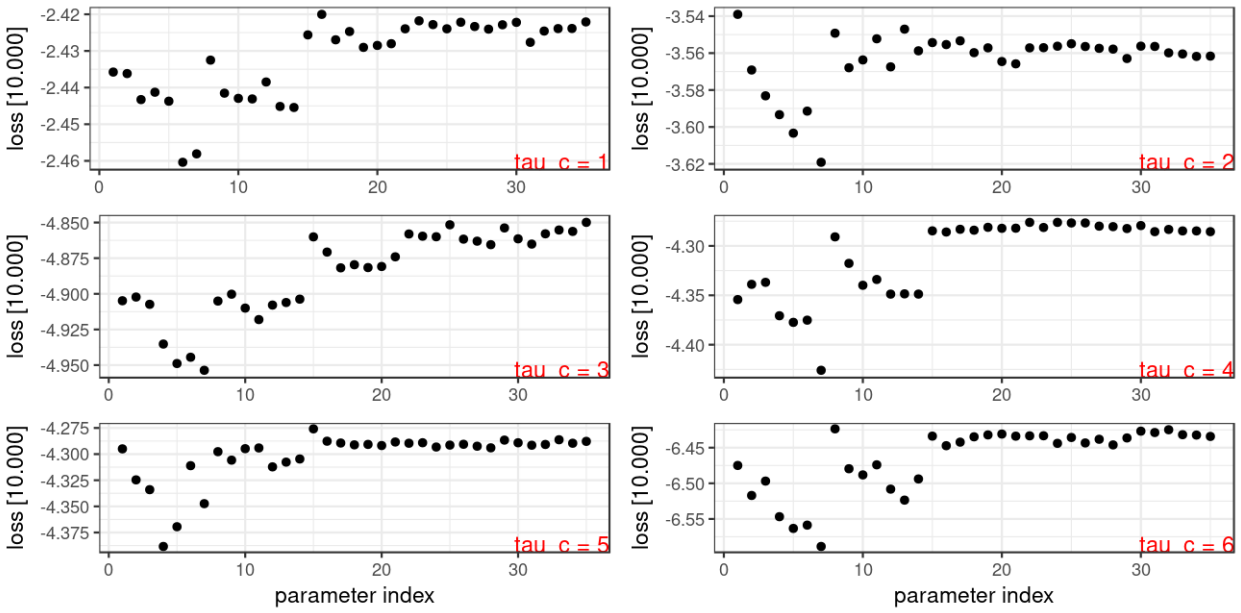


Figure 5: Tuning graph for the six different subsets (according to tau categories). For every subset 35 different models were trained and the loss of each prediction calculated. The settings change as follows: size = 3, 5, ... , 15 then dacay = 0.01, 0.1, 0.5, 0.8, 1.0. Stable loss values for bigger decays, size does not have much influence for high decay values.

In order to increase the reliability of the tuning results, they were calculated of a 3-fold cross validation, that got repeated 6 times (ensuring that the datasets are sampled differently).

5.3.2 optimal τ for loss function

With the parameters from the previous section (5.3.1), the neural network has optimal performance for our needs. As a last step the threshold value τ (which is the threshold that decides if the predicted probability is big enough to be assigned as 1 (item is returned) or stays 0 (item is not returned)) needs to be tuned to minimize the loss function. As described

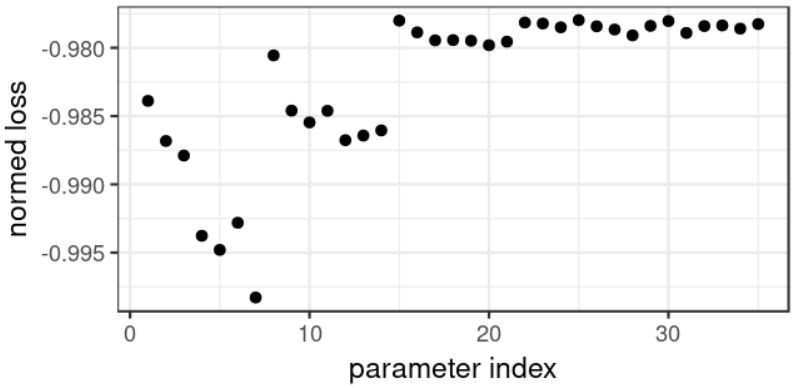


Figure 6: Sum of normed tuning graphs of the six subsets from graph 5. Loss is the highest for bigger decay values. Size only has little influence on loss values for high decay values.

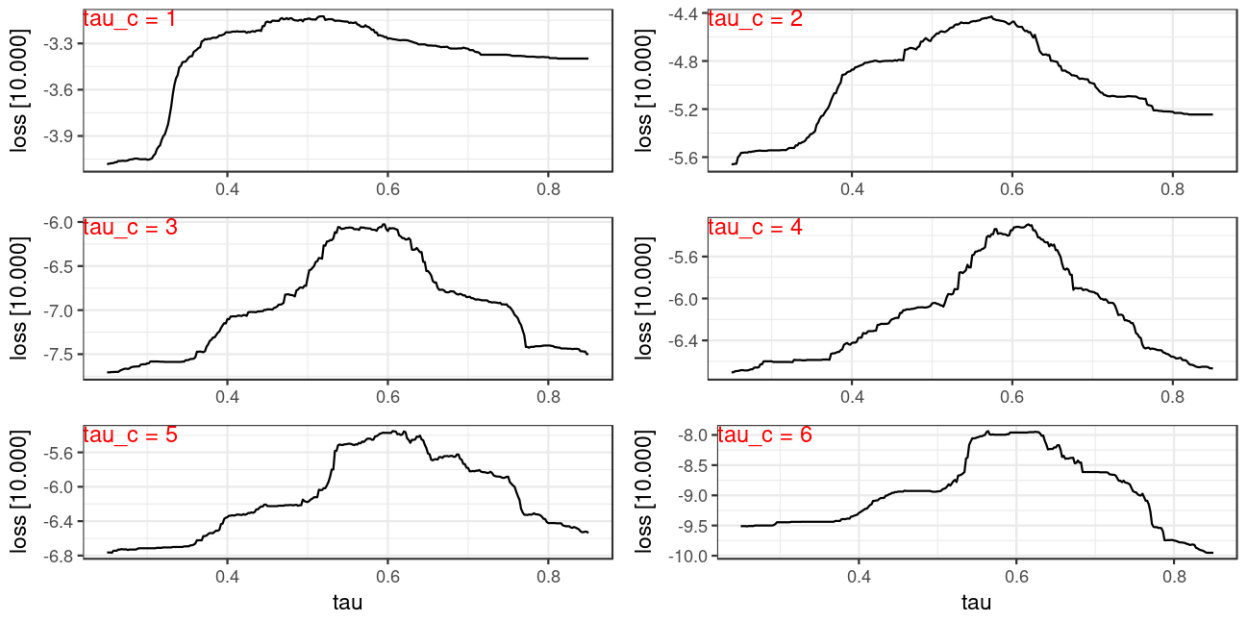


Figure 7: Loss function for the six tau-categories. In each category the loss is calculated for each tau-candidate. The negative loss shows its maximum in the range between $\tau = 0.45$ and $\tau = 0.65$.

in section 4.2, the loss depends on the item price and therefore the data set is split into six subset according to the price (subsets are called tau-categories).

Figure 7 displays the loss function for the six tau-categories. In each category the loss is calculated for each possible τ value. The function reaches its maximum in the range between $\tau = 0.45$ and $\tau = 0.65$, depending on the category. The maximum of the function is the optimal τ that minimizes the loss of the subset. It is not surprising that for categories 1 to 6 (where 1 are cheapest and 6 most expensive items (see 4.2)) the optimal τ increases. (Interpretation of τ in connection to item price and return probability in section 4.2.)

5.4 Comparison with random forest

A random forest is an ensemble of trees with implemented randomisation procedures. Random forest tends to be good out of the box and apt for parallelization.

The model default parameters are (c.f. Breiman (2001))

1. m_tree: $(\#features) \cdot \frac{1}{3}$ for regression, $(\#features)^{1/2}$ for classification
2. n_tree: 500
3. nodesize: 1 (for classification), 5 (for regression)

For random forest a 10 times repeated 3 fold cross validation with implemented multiple τ optimisation was carried out.

tau category	1	2	3	4	5	6
n_tree	600	600	600	300	500	600
optimised τ	0.52	0.55	0.58	0.65	0.62	0.65

Table 8: Results of cross validation for random forest, m_tree was excluded as the best value always resulted 2

The following table shows how neural network outperformed random forest in this problem for a set of 25000 observations. Double values are expected in the test set, which contains 50000 observations.

Prediction	**Estimated total loss**
trivial 1 (vector consisting of ones only)	-415415.6
trivial 0 (vector consisting of zeros only)	-321240.2
random 01 vector with $mean = 0.48$	-365703.9
first randomForest ($\tau = 0.5$)	-321420.5
tuned randomForest (optimised multiple τ)	-260608.5
tuned nnet (optimised multiple τ)	-238879.0

Table 9: Estimated total loss of random forest and nnet in comparison to trivial predictions. Performed on 25000 observations (25% of train set.

5.5 Prediction

In order to construct values for a comparison of the performance on the test dataset, the prediction is run on the training dataset. Therefor the training dataset is randomly split into a new training and test dataset. After training and predicting with the four models (.f, .u, .i and .iu), the loss is $loss_{dec} = -238879.0$ and the average return rate of the prediction lies at 39.9%.

On the unknown dataset the predicted average return rate is $39,4\%$, which is similar to our test-run. The loss can not be computed, since the true return is unknown.

6 Conclusion

In this case-based analysis, we developed two predictive models from start to finish. Starting with the data cleaning process through training and tuning several classifiers up to the incorporation of the loss function into the cross validation. The work presented in this paper only reflects the two models (RF and NN) that we considered in the end as the most accurate ones (based on their impact on the loss function). Not reported in this paper is the development of other models that we built, e.g. extreme gradient boosting (XGBoost)[2]. However, there is definitively space for improvements and further work, e.g. other ensemble models like stacking may help to reduce bias and variance as they make use of multiple base models (c.f.Yu et al. (2006)).

One (in-)evitable error occurred during the creation of the variables user_retrate and item_retrate: the whole training data set has been used for creating these variables, because otherwise there were not enough observations in each category (since only around one third of the users have placed six or more orders). As we have used the whole training data set for creating the variables and trained the models on the same data set, it may be likely that some overfitting occurred.

Besides predicting return probabilities, there should be other actions taken into account when minimizing shipping costs. For example, online retailers can prevent return of orders by implementing a rating system on their websites such that costumers are able to assess items and thereby receive a peer-review for the products. Furthermore, conversion tables for clothing sizes that are individualized for each brand may also reduce returns. Along with these marketing actions is a more informative data collection (in close coordination with online-privacy policies) recommended. More data is not the unique solution for data mining problems, but more informative variables can help to increase predictive accuracy.

[2]The related code is available on GitHub https://github.com/Humboldt-BADS/bads-ws1718-group07

References

Breiman, L. (2001): "Random forests," *Machine learning*, 45, 5–32.

Chen, T. and C. Guestrin (2016): "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, ACM, 785–794.

Crone, S. F., S. Lessmann, and R. Stahlbock (2006): "The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing," *European Journal of Operational Research*, 173, 781–800.

LeCun, Y., Y. Bengio, and G. Hinton (2015): "Deep learning," *nature*, 521, 436.

Lessmann, S., B. Baesens, H.-V. Seow, and L. C. Thomas (2015): "Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research," *European Journal of Operational Research*, 247, 124–136.

Manyika, J., M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers (2011): "Big data: The next frontier for innovation, competition, and productivity," .

Moeyersoms, J. and D. Martens (2015): "Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector," *Decision support systems*, 72, 72–81.

Yu, L., K. K. Lai, S. Wang, and W. Huang (2006): "A bias-variance-complexity trade-off framework for complex system modeling," in *International Conference on Computational Science and Its Applications*, Springer, 518–527.