Term paper

"Business Analytics and Data Science"

Prof. Stefan Lessmann & Johannes Haupt

WS 2017/18

Batuhan İpekçi
527060, MEMS, HU-Berlin

Gairik Aluni
396888, ICT Innovation, TU-Berlin

Jan Esmaeili Fathabadi
591488, Economics, HU-Berlin

Shijian Xiao
586283, Economics, HU-Berlin

1. Introduction

In 2003, the total sum of data that existed, had a volume of 5 exabytes, which is equal to 1 Mio. terabyte (Sagiroglu and Sinanc 2013). One of the main challenges for businesses was taking decisions under high uncertainty, resulting from "incomplete, insufficient, and time-lapsed data" (Ngai et al. 2017).

By 2012, Google alone processed approximately 24.000 terabytes of data per day (Davenport, 2012) and in 2015 5 exabytes of new data were generated every 2 days (Sagiroglu and Sinanc 2013). In 2011, a survey was published by Bloomberg Businessweek, coming to the conclusion that 97% of all companies with revenue exceeding $100 mio. have implemented some form of business analytics. And in 2012 Evans et. al stated that each dollar invested in analytics would bring approximately 10,66$ in return (Evans and Lindner 2012).

But what exactly is described by the term Business Analytics?
Generally, it can be understood as the purification of data (Gronau et. al 2016). The New York Stern School found a more precise description, formulating it as „...the study of data through statistical and operations analysis, the formation of predictive models, application of optimization techniques and the communication of these results to customers, business partners and colleague executives." (Felden 2016).
Different techniques like data mining, predictive analytics, applied analytics and statistics are combined to not only receive insights on what happened in the past, but to also go one step further and derive assumptions for the future about what will happen next.

The possibilities of appliance of such analytics are obviously widely spread. For example,
Netflix manages to give precise recommendations to their customers about movies, by analyzing feedback, reviews and similar Click-stream data (Davenport and Harris 2007). Amazon claimed to generate 30% of their total sales by making similar recommendations about what else could be interesting, each time a purchase is made (Manyika et al. 2011) which are based on the transaction data from their customers buying behavior.

1.1. Problem Statement

Similarly, in this assignment, we will apply such methodology of business analytics to the context of an online retailer. The market conditions do not allow for the retailer to charge customers for returning a product that they have ordered, even though it is associated high costs. Thus, given real-world transaction data from the retailer, we will train a predictive model on one dataset, knowing for each item if it was returned. This model will then be used to predict the probability of return for the items of another dataset, for which the true outcome is unknown. The final goal will then be to maximize the net revenue gain, by giving a recommendation for which items the retailer should take action to prevent the customer for submitting the respective order. We will not discuss the possibilities for such action.

2. Used Packages

The following packages were used for completing the task:

caret (Kuhn 2017): This package helped us in nearly every step of the model training process. We used it to center and scale the original numeric data in the preprocessing phase, as well as to split the data into a training and a testing set. It contains several learning algorithms such as neural networks and random forest, and can further tune a models parameters via cross validation.

caretEnsemble (Deane-Mayer and Knowles 2017): The "caretEnsemble" is based on the "caret" package. Using "caretEnsemble", we could combine different learning algorithms into one model to increase the predictions accuracy. The algorithms you want to use are defined in "caretList", parameter tuning should be defined well for each algorithm separately.

doParallel (Microsoft Corporation, Weston 2017): We used this package to accelerate the algorithm calculation time. It allows the computer to use multiple processor core to take the training task.

forcats (Wickham 2017): The "forcats" package helped us to reorder and modify the factor levels as well as to collapse and replace some very rare levels. Further, it is very helpful in reordering the factor levels based on our demand such as the first appearance.

Hmeasure (Anagnostopoulos and Hand 2015): It helped us to assess our model's performance based on criteria such as AUC, sensitivity, H-measure.

klaR (Weihs 2015): We used this package to calculate the weight of evidence (woe) of each factor level and each variable.

mlr (Bischl et al. 2016): The "mlr" package is based on the package "caret" and has similar functionality. Thus, it was used in a similar way, as well as for feature extraction such as wrapper implementation.

nnet (Venables and Ripley 2016): We used the "nnet" package to build a single-hidden-layer neural network model.

pROC (Robin et al. 2017): This package was used to plot model performance, namely the receiver operating characteristic (ROC curves) and the area under the curve(AUC).

randomForest (Liaw and Wiener 2015): This package helped us to build a random forest.

rpart (Therneau et al. 2018): This package was used to build a decision tree.

Xgboost (Chen et al. 2018): We used the "xgboost" package to run an extreme gradient boosting algorithm. In gradient boosting we ran gradient descent algorithms iteratively until certain thresholds were achieved.

woe (Greif 2013): We benefited from a custom package for WOE-based discretization of numeric variables in the dataset (https://github.com/tomasgreif/woe). Tomas Greif's package "woe" on Github has been particularly useful in giving us the option to prune the decision tree that is embedded in the discretization function. On our discretion, we found the optimal pruning.

3. Preprocessing

Data preprocessing is a crucial step in „Knowledge Discovery in Databases", a process which is defined as mapping low-level data into compact, abstract or/and useful forms (Fayyad et al. 1996). The data preprocessing pipeline consists of statistical exploration (moments, frequencies and dependencies), data preparation (standardization, outlier and missing value treatment), and variable transformation (encoding, discretization, linear and nonlinear transformations). Our interpretation of this process is a combination of these methods targeted at obtaining a valid solution to the specific business problem as explained before. Our model is primarily concerned with minimizing costs resulting from our prediction.

Although a standardized methodology is usually followed in data preprocessing, this process includes an important element of task-specific subjectivity. Fayyad et al. suggests coining a term "interestingness threshold" passing which being a prerequisite for us to acknowledge a "pattern" as a "knowledge" which is always user oriented and domain specific (Fayyad et. al. 1996). From a wider perspective, Bre classifies our threshold for interestingness as "actionability" which purports extracting pattern with an aim to quantify its "use" for its particular purpose (Bre 2013). During this project, we encountered with a trade-off between descriptive and predictive preprocessing. Given the specificity of our task, we favored the preparations and transformations which might yield a higher predictive accuracy over those which increases the interpretability of data. At each data preprocessing step, we tried to validate our results through running comparative logistic regressions. Improvements in the predictive performance of this basic regression model informed us whether we should apply a transformation in data for more advanced models.

Both of our initial known and unknown datasets have been infeasible in many ways for many prediction models. First of all, all of the variables, with an exception of item_price, were categorical. Some models are incapable in dealing with this variable type. This issue is even more complicated when we account for the high cardinality of some of these variables together with the new categories arising in the unknown set. Secondly, we observed that some missing values and outliers in some variables are not distributed randomly and are in a clear connection to our target variable, return. Thirdly, we lacked domain knowledge which is necessary for an intuition-based transformation. Moreover, we did not know anything about the data preparation and cleaning process.

Most of the time, the interactions among and clusters across the variables we extracted did not yield a nontrivial increase in predictive performance. Furthermore, we were unable to find meaningful casual relations between any of the independent variables and our target variable, let alone being able to apply statistical tests to these relations. The strategy we pursued to overcome these problems was, hence, to preprocess the data rather in a useful manner for the prediction model than by making it suitable for inferential analysis.

We imputed missing values and outliers while taking their relation to the target variable into account. Similarly, we applied Weight of Evidence (WOE) transformation to categorical variables and WOE-based binning to numeric variables. In order to attain more depth, we decided to use both categorical and numeric transformations of the same variables in our dataset. By doing this, we attained some improvement during application of our advanced models. However, because WOE transformations pose a risk of overfitting, we calculated the WOE scores in a different sample of our training set and we excluded the use of this sample while we train our model. By doing this, we succeeded to obtain higher predictive performance than that before the preprocessing steps.

3.1. Missing Values, Outlier Treatment, and Standardization

Variable	Frequency in “known”	Frequency in “class”	Mean return
item_color	34	21	0.05
item_size	5475	2768	0.28
delivery_date	8292	4053	0.00
user_dob	10023	5119	0.47

Table 2: Missing Values

As mentioned previously, some missing values in our sample were highly correlated with our target variable. As seen in Table 2, except user_dob, application of mean/median imputation or supervised imputation to these variables might cause a significant information loss. In order to retain the hidden information behind these missing values, we opted for keeping them as another category in item_size and item_color, and for more intensive treatment in user_dob and delivery_date. Moreover, we decided to impute missing values in dates before we transformed them into new numeric variables. For missing values in user_dob, it was safe to transform its class to a date and use median imputation, given the fact that their mean of return is very close to the mean of return of our whole known set. Because our known dataset is in general well-balanced with a mean of return of 0.48, we expected that the missing values “?” in user_dob would not disturb the general pattern.

Apart from missing values, there were outlier dates in user_dob and delivery_date which were also unrealistic. For instance, there were more than 100 years old customers or orders whose delivery_date is earlier than order_date. With regards to these values in user_dob, we applied min-max imputation by selecting threshold dates. The situation in delivery_date was more delicate. The missing value "?" in delivery_date was considered as late delivery and was imputed by the latest delivery date whose corresponding mean return rate is also 0. The delivery date which are earlier than order date was also considered a missing value whose mean return rate was 0.32 We imputed this type of the missing value by the most frequent value after we imputed the missing value "?", which is also the latest delivery date.

There were occasional writing mistakes and inconsistencies in the levels of item_size and item_color. For instance, in item_size the levels "xl" and "XL" were different categories although in reality they consist of the same category, whereas in item_color there were similar colors that we could bundle under fewer categories. However, we kept them as they are. An alternative method would be semantic grouping, where we collect all levels under their respective size or a similar color in real life. The reason why we have not chosen this approach is that we do not know anything about the data generation and selection process. These two different categories might have been derived from different databases with distinct qualities and these different databases might be related to different return rates. Our best strategy to retain this information was to avoid from changing these suspicious levels. Further, intuitively we considered the relation between return and item_color or item_size less substantial than the relation between return and the way the whole data is collected.

The only numeric variable in the original sets was item_price. Both in our unknown and known sets, its distribution is positively skewed and minimum, quartile, and median values are the same. Exceptionally, the kurtosis of item_price in the unknown set is much higher than that in the known set. It is primarily because of the maximum value in the unknown set which is much higher than that of the known set. Before standardization, we fixed the maximum value in the unknown set to that in the known set and applied a supervised discretization in order to capture price categories whose orders are likely to be returned (Supervised discretization will be discussed in depth later on). The reason to apply discretization before standardization was the aim of minimizing the risk of losing information during standardization (Gan et al. 2007).

We extracted three new numeric variables from date variables, user.age, delivery.time and user.reg.time. Their quartile, median, minimum, and maximum values together with kurtosis and skewness does not differ across our known and unknown sets. We observed that none of our numeric variables showed characteristics of different sample distributions across known and unknown datasets. Further, similar to item_price, we applied standardization to these new variables. And relying on similar distributions across our both known and unknown datasets, we avoided from further outlier removal. We suspected that the outlier values could possess some information that we might could not afford to lose.

Later on, we tested some Box-Cox Transformations (logarithmic, quadratic, and inverse), following textbook suggestions (Garcia et al. 2015). However, we did not observe any significant improvement in model performance following these transformations.

3.2. Factor cardinality, Weight of Evidence, and Dummy Encoding

The most challenging issue for us during data preprocessing was to include variables with high cardinality in the analysis. Aside from the computational costs of models such as logistic regression on categorical variables, we encountered with new levels in the unknown set which rendered tree-based models also useless. Furthermore, our observation was that exclusion of these high-cardinality variables affected the prediction accuracy adversely. A remedy to this problem has mainly been done through encoding and clustering techniques. In Section 3.1, it was pointed out that although the levels of item_size and item_color were proper for semantic grouping, we avoided applying this technique mainly because it worsened our model performance.

Weight of Evidence (WOE) was the main transformation technique from which we benefited during this study. WOE is an approach based on calculating the logarithm of the Bayes factor, an alternative test hypothesis in statistics, and it has been widely used in many different contexts from banking to geography where a risk assessment of individual values is required (Good 1985). Moeyersoms and Martens proposed WOE as a valid technique in data preprocessing, by means of which not only relational or behavioral instances behind categories can be processed throughout a measure of distance, but also the inclusion of high-cardinality attributes is done without an expansion of dimensionality (Moeyersoms and Martens 2015). The former quality can be thought as an advantage of using WOE against standard number encoding, while the latter as against dummy encoding. In addition to these advantages, WOE is also a useful tool in supervised binning within numeric variables, where numbers are grouped according to their average relative risk (Zdravevski et al. 2011). Since WOE is useful in transformation both from categorical to numeric and from numeric to categorical, it enables multiple transformations of a single variable (Moeyersoms and Martens 2015). Following this line of thought, we have extracted two new variables for each of our categorical variables. For instance, for user_id, we have created both a numeric woe.user_id which contains the WOE scores and a categorical user_id.cat which consisted of bins of the WOE scores. Later on, we applied dummy encoding to the new categorical variables in order to prepare them for a larger set of predictive models. Namely, the variable names beginning with "cat." are dummy encoded.

Nonetheless, we should mention certain limitations of this approach. At first, the data preprocessing through WOE transformation causes models to overfit the dataset more likely. Second, there are sparsely populated and highly informative levels which correspond suspiciously always only to 1 or 0, in those cases standard WOE calculation outputs infinity values. Third, there are levels which occurred only once and therefore it might be incorrect to assume any value for them at all. Fourth, there are new levels in unknown set, and previous WOE scores do not inform us for new levels. Fifth, WOE is a relative measure of risk and does not differentiate across proportions of data points (Zdravevski et al. 2011). Sixth, WOE transformation is a univariate statistical approach, hence it cannot be used to analyze correlations and interactions across multiple variables.

Aiming to overcome the first limitation, we calculated WOE scores of categorical variables in a separate sample of the dataset (Moeyersoms and Martens 2015). We have not used this part of the dataset during model training. We used this hold-out dataset only to calculate the scores, then disregarded it altogether at further steps. For the second limitation, we have followed Zdravevski et al.'s advice and added insignificant number of artificial data points by using the klaR package (Zdravevski et al. 2011). Note that third limitation is also similar to the second, hence it might similarly be dealt with. However, we desired a further differentiation between levels which only occurred once and levels which occurred more than once but correspond 100% to 1 or 0. Therefore, we regarded the third and fourth limitation as being the same, i.e. we rejected any information that comes from an individual level that only occurs once and treated them in the same category as the new levels in the unknown set. We calculated frequencies of each variable in the union of known and unknown datasets and then set their WOE score to 0. By calculating and using the frequencies in another column, we also obtain necessary differentiation to overcome the fifth limitation. We will deal with the sixth limitation in the next section.

3.3. Interaction among Variables, Clustering, and Feature Extraction

	woe.brand_id	woe.item_id	user_title.Mr	woe.user_state	delivery_time	user.age
woe.brand_id	[redacted]	0.000	0.033	0.052	0.000	0.080
woe.item_id	0.000	[redacted]	0.025	0.0524	0.000	0.009
user_title.Mr	0.033	0.025	[redacted]	0.000	0.000	0.055
woe.user_state	0.054	0.0524	0.000	[redacted]	0.435	0.202
delivery_time	0.000	0.000	0.000	0.435	[redacted]	0.000
user.age	0.080	0.009	0.055	0.202	0.000	[redacted]

Table 3: Interaction (p-values of testing interaction coefficients)

In order to explore the sixth limitation mentioned in Section 2.2, we listed some possible interactions among our numeric variables. The results in Table 3 are p-values which represent test results on coefficients of multiplication of columns. They are derived from the output of the logistic regression which was run one by one for each couple of variables. The interactions between item_price and other variables were relevant but redundant since we know that specific item_size and item_color corresponds to specific prices. Similarly, the interactions between delivery.time and brand_id or item_id are redundant, since different items or brands might have different structures of delivery speed. Because we are analyzing from the perspective of woe scores, these similarities might have arisen by the relationship between independent variables within themselves rather than their interaction with the target variable. The most remarkable interactions were those between delivery.time and user.age; user_title.Mr and user.state; and woe.brand_id and woe.item_id. Unfortunately, including these interactions into our model did not result in higher performance.

We did not test other interactions which seemed too unintuitive to us. Therefore, we stopped exploring those interactions further.

Earlier in Section 2.2, we extracted frequency columns for each categorical variable. We decided to include these frequencies in our analysis, provided that they are not in a high correlation with the WOE scores of their respective columns. We observed that the Pearson correlation between woe.delivery_date and woe.user_reg_date with the respective frequencies f.delivery_date and f.user_reg_date are high, respectively 0.98 and 0.76. We removed these two frequency columns and kept all other frequency columns for the sake of more depth in our analysis.

In addition, we have tested whether k-means clustering all of the final numeric variables together could induce any effect on the prediction performance. We found the optimal number of clusters as 10 by elbow method and categorized our data in 10 clusters. However, including this clusters in a new column did not affect our performance.

4. Split Sample Setup

After having trained a model on the dataset which includes the target variable “return”, its accuracy needs to be reviewed, in order to compare the model to others.
The simplest way would be to train the model on the known data set (including target variable) and then calculate the relative amount of correct predictions by comparing the predicted target variables with the actual. However, training a model on the same data is then afterwards should predict will most likely cause overfitting. This means the model is too precisely adjusted to it’s training data and will not be as precise when predicting data that is unknown to it. It, therefore, is important to know how well the model performs on unseen data. This can be achieved by using a method called cross validation. The idea is to train the model only on one part of the known dataset and let it predict the target variable for the other part.

The simplest version of cross validation is the so called “holdout method”. It works by dividing the known dataset into 2 splits, not necessarily of same size: train and test. Training the model is then performed by using the train set and predictions are calculated for the test part. Even though it can easily be performed, one disadvantage of this method is that the model is trained on a smaller portion of the available data.

This problem is solved when using the so called “K-fold cross validation”. Similarly to the holdout method, the dataset is split into k equally sized folds before training the model. The model is then trained k-times, so each fold & each data point is once used a test set and k-1 times as a training set. In the end, the average performance across all trials is averaged. As a result, the estimates variance is reduced and a more realistic impression of the models predictive quality can be achieved. But as each fold is once used a test set, the training algorithm has to be run k times.

As mentioned previously, we have calculated the WOE scores in a separate hold-out set and thereby aimed to avoid overfitting. The decision on the ratio of splitting the “known” dataset into three parts as woe.set, known_tr, and known_test has had a direct influence on the AUC scores we obtained. To illustrate this, in Table 4 we created 4 different partitions of the known set. Note that as training set becomes smaller and WOE hold-out set becomes larger, we obtained always higher AUC scores on Gradient Boosting. Increasing the size of the WOE set beyond 60% did not bring about a considerable improvement. When training set is equal or less than the test set, we might not train our model enough to capture the underlying distributions. On the other hand, we were encouraged by the empirical results of Moeyersoms and Martens 2015 which states that with higher sample sizes the AUC scores keep increasing. Our final decision was to split the known set into 40% hold-out, 40% training and 20% test for model assessment. During this decision process, we kept the test set at 20% because with a too small test set we might fail to attain generalization. This setup helped us to choose the best model to predict the unknown dataset "class". However, during predicting "class", we have altered this ratio to 60% WOE hold-out and 40% training sets, i.e. relied more on the best model as to generalize the knowledge throughout cross-validation. We choose the more risky option for the competition.

WOE Hold-out	Training	Test	AUC / xgboost
20%	60%	20%	0.732
40%	40%	20%	0.738
60%	20%	20%	0.744
70%	10%	20%	0.745

Table 4: Split Sample

5. Variable selection

For optimizing the performance of a model one has to consider which variables should be included. By only including variables of high significance/excluding variables of low significance, the accuracy of the model can be increased. There are multiple alternatives of reviewing the variables importance the for predicting the target variable. For our assignment, we firstly calculated the information value of our categorized variables (Package ‚InformationValue'). This method gives an impression of a categorical variable to predict a discrete target variable. A variable with an IV of < 0.02 is considered not to be significant for the prediction, an IV of $>= 0.3$ is labeled as having a strong relationship with the target.

For calculating the importance of continuous variables on a discrete target, the Fisher score is an applicable method. Calculating it for our continuous variables yielded the following values, where generally a higher score indicates a stronger relationship with the target (Gu et al, 2012).

Variable	Information Value
woe.delivery_date.cat	0.935578267
woe.item_id.cat	0.197096325
woe.user_id.cat	0.145859717
item_price.cat	0.129790872
woe.brand_id.cat	0.126157846
woe.item_size.cat	0.051852724
woe.user_dob.cat	0.043646351
woe.item_color.cat	0.023801619
woe.order_date.cat	0.010482324

Table 5: Information Value

Variable	Fisher Score
woe.delivery_date	0.44691978
f.delivery_date	0.43879323
woe.item_id	0.31398143
woe.user_id	0.26192095
woe.brand_id	0.25416408
item_price	0.18737985
woe.item_size	0.15875938
woe.user_dob	0.15041679
woe.item_color	0.10778329

Table 6: Fisher Score

Lastly, we applied the Boruta algorithm on those variables which appeared to be significant (above 0.9 for both filters), according to the previous test methods. Boruta is a wrapper feature, that based on random forest.

As a result, woe.order_date.cat (tentative) and woe.user_reg_date.cat (rejected) are failed to be confirmed. However, including only the survivor variables after the filters and wrapper to the model led to a decrease in AUC from 73,1% to 72,5%. We, therefore, decided not to remove any variables from the training set except the ones that are highly correlated (higher than 0.7).

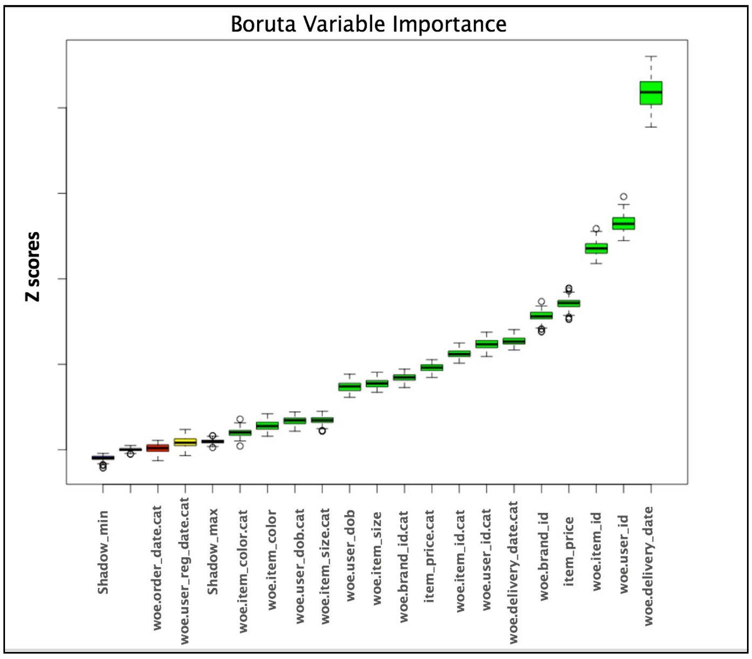

Table 7: Boruta Variable Importance

6. Models & Assessment

During model assessment we have used both the categorical and numeric transformations that we extracted from the initial variables. By doing this, we take the risk of overfitting in the expectancy of attaining more depth in model training.

Throughout the project, we favored the AUC over accuracy as a single-number performance measure. The accuracy measure tacitly assumes that the data distribution is balanced and has equal error costs. That means the False-Positive error is equivalent to the False-Negative error. Nevertheless, in real life situations, when classifiers are used to find a relatively small number of unusual occurrences, or when class distributions are skewed, accuracy is not the desired measure (Provost and Fawcett 1997).
AUC stands for the Area Under the ROC curve, whereas ROC stands for Receiver Operating Characteristics, a classic methodology from signal detection theory. In ROC space, each classifier is mapped to the point corresponding to its False-Positive and True-Positive pair. The resulting ROC curve visualizes error trade-offs given a model. It can be argued that for binary classification problems, ROC curve is independent of class distributions, error costs, and, usually, of models. ROC curve is particularly applicable "in situations where strong general conclusions cannot be made" (Provost et. al. 1998, 450). AUC is more suitable than accuracy to the ranking of the values, and interestingly, if the models are optimized for a better AUC score, we would obtain a higher accuracy, as well. Lastly, AUC has another advantage in business operations, such that it is associated with net profit in direct marketing (Huang and Ling 2003). The last advantage can be attributed to the adaptability of AUC to the probability estimations in the outputs of many supervised classification models. Therefore, the choice of the performance measure is in harmony with our task of ranking and minimizing business costs arising from the prediction.

Logistic Regression: Logistic regression is a valid choice in our case because the dependent variable (i.e., return) in this model is a binary factor. The function we used for logistic regression is "glm" under R "stats" package, where we trained the prediction model over a set of independent variables to give a probability between 0 and 1. The result we were expecting has to be either 0 or 1, i.e., binomial in nature, so the method used for training model is "logit (binomial)". The AUC score we obtained without regularization is 0.7314. Further, we applied penalties on coefficients through lasso regularization, where we found the best shrinkage parameter with cv.glmnet function which improves the result only slightly, 0.7315.

Random Forest: Decision tree has low bias and very high variance. Random forest, on the other hand, reduces the high variance by averaging the samples of decision trees. Internally it uses bootstrapping and bagging to normalize the high variance of decision trees. In bootstrap aggregation, various training sets are bagged in random order along with their responses. Bagging reduces variance but there is slight increase in bias (Breiman 1996). We used the package 'randomForest'(CRAN 2015).

Neural Networks: We use "nnet" package to apply Neural Network. The package feed-forwards neural networks with a single hidden layer (CRAN 2016). The group took the same approach for neural networks and trained the model firstly without using the CARET package and then by using the CARET package.We used cross validation within the training dataset. Initially we used 90 features in our training set, but we did not get any result since the data set was much much bigger than what R could handle. We even used “parallelize” to use all the cores but there wasn't any improvement. Later, we reduced our features count to 50, but the result we obtained was worse than what we expected and even lesser than regression model. So we proceeded to calculate the AUC without cross validation and achieved an AUC of 0.731 on a separate test set. In comparison to the other models and training the model without cross it gave a fairly better AUC, which is higher than Random forest and Regression.

Gradient boosting: The gradient boosting model was the model which gave us the a very good prediction. Gradient boosting helps in regression and classification by creating a trained prediction model by ensembling prediction models (Friedman 1999). Gradient boosting model comes under xgboost package. The convergence time of xgboost is very less even with high number of features. We can define the trainControl() with method as cross validation and expand.grid parameters for better tuning and improved prediction.

Ensemble model - Stacking: This is the model that gave the best result: AUC 0.7387. It is slightly higher than gradient boosting. Stacking (sometimes called stacked generalization) involves training a learning algorithm to combine the predictions of several other learning algorithms (Wolpert 1992). First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. In our case the different algorithms used are combination of Random Forest and Gradient boosting. Since our training set contains 90 features, it was not possible for the machine to include more models. As the combiner algorithm we have used linear regression. Since we had dataset of 90 features, we could not combine more models. By using a combination of Random Forest and Gradient boosting we got the best result. We choose this model on the basis of experiment and by reading literature for the commonly used ensembling models.

Models	AUC
Logistic Regression	0.7314
Random Forest	0.7276
Neural Networks	0.7316
Gradient boosting	0.7382
Ensemble model - Stacking (Random forest + gradient boosting)	0.7387

Table 8: Model AUC

7. Cost Sensitivity

In our case, there are two types of prediction errors: The first was predicting customers would keep the item and, thus, do not send a message whilst customers actually returned the item. The cost of the first type error includes a small amount of fixed cost and loss related to item value. The second type error was to predict that customers will return the good and to send a warning whilst customers did not return the article. The respective costs only relate on the item value. All costs can be seen in the cost matrix (Table 8). The cost of error two increases faster than that of error one when the item value increases.

Prediction \ True Value	Item kept (0)	Item returned (1)
Item kept (0)	0	0.5*5*-(3+0.1*itemvalue)
Item returned (1)	-0.5*itemvalue	0

Table 9: Cost Matrix

Optimal Bayes (OB)

The Optimal Bayes method is to find out a probability which can minimize total expected cost of both error types. Elkan (2001) suggests to minimize the total expected cost by equalizing the expected costs of these two errors. We defined T as a probability threshold and used our model to calculate the probability of the good returned. It is then optimal to predict that the good will be returned, only if the probability of the good being returned is larger than T. In our case, T should be $\frac{itemvalue}{15+1.5*itemvalue}$. To some extent, optimal Bayes neglects the true data structure. When the classifier is misspecified, optimal Bayes can't perform well. Furthermore, it is also not suitable for the unbalanced dataset.

Empirical Threshold (ET): Sheng and Ling (2006) suggest to find out the best threshold for minimizing the cost by its empirical performance. Empirical threshold requires no accurate estimate of the probability. A sequence of thresholds is listed and tested respectively in our model to find out which one performs best. Two kinds of thresholds are tested in our report. The first (ET1) is one threshold for all items. The second (ET2) is one individual threshold for each item value. Sheng and Ling (2006) showed that the empirical threshold performs as well as other cost sensitivity methods. However, there may exist overfitting problems when the sample is not large enough.

Optimal Bayes and empirical threshold (OBET): We tried to use the advantages of both cost sensitivity methods. Because the cost matrix is dependent on the item values, it is not suitable to set only one certain threshold for items with different values. We prefer to assign an empirical threshold to each price level. But this is likely to generate overfitting. To avoid this, we set optimal Bayes as the threshold when the frequency of the item price level is lower than a certain frequency-threshold. We then set 3 frequency thresholds: 5, 10 & 20.

Our testing set for testing the cost performance of different cost sensitivity method included 20.000 items. Because the neural network is more demanding for the data structure, we

decreased the variables in neural network so that it can run the parameter tuning part in a shorter time. In Table 9, both results of the OB, ET1 and ET2 are compared. We can see that ET2 always performed best, while OB and ET1 had similar performances. In Table 10, we can see that the OBET performance decreased when increasing the frequency threshold. After solving the overfitting problem, we saw that OBET performed much better than OB and ET1 for all the models. The stacking ensemble model outweighs the gradient boosting in most of the case in a little. But they are still on the same level. Logistic regression has similar performance with neural network, ahead of random forest but behind gradient boosting and stacking ensemble. The decision tree always has the worst grade.
In Table 10, the results of OBET with different frequency threshold are compared.

Total Cost for Testing Set		Cost Sensitivity Method		
		OB	ET1	ET2
Model List	Logistic Regression	-210488.9	-213543.3	-198319.2
	Decision Trees	-251919.6	-245856.4	-237772.9
	Neural Networks	-209116.3	-210450.6	-197852.8
	Gradient boosting	-207037.5	-206369.4	-194979.6
	Random Forest	-220319.2	-216682.5	-203519
	Stacking Ensemble	-206485.8	-207206.9	-194624.4

Table 10: Cost Comparison of OB, ET1, ET2

Total Cost for Testing Set		Frequency Threshold		
		5	10	20
Model List	Logistic Regression	-199625.2	-201604.8	-202627.8
	Decision Trees	-239249.2	-240992.7	-241677.4
	Neural Networks	-199086.9	-200993.7	-201773.9
	Gradient boosting	-196279.9	-198332.6	-199262.8
	Random Forest	-204775.4	-207474.6	-208432.0
	Stacking Ensemble	-195552.5	-197344.4	-198006.8

Table 11: Cost comparison of OBET with Different Frequency Threshold

8. Unbalanced data

A dataset is defined as imbalanced when the classifications of the target variable are not equally distributed. It can be a problematic when the cost of incorrectly predicting the minority is much larger than misestimating the majority. Furthermore, it will be enlarged when the ratio of majority over minority increases. A popular example to explain the idea of unbalanced data is detecting cancerous patients (Breiman et al. 1984). The amount of non-cancerous patients heavily outnumbers those cancerous. However, the cost of predicting an actual cancerous patient as a normal person is much larger than misestimating the normal one as a cancerous patient.
In our case, we find that the majority (not return) and minority (return) scale is without significant difference. Their ratio is 52 against 48. It seems unnecessary to balance the dataset. But the cost of different errors is still different in our sample. Elkan (2001) mentioned that the different errors' cost is also kind of unbalanced data. In addition, we found that even the overall majority and minority scale is quite equal. The ratio of return and not return is different within each item price level. In some price interval, the returned goods largely outweigh those not returned. In other price intervals, it is reverse.

Therefore, we try to balance the sample based on each price interval. The size of each interval is one euro for each. We decide to use the oversampling algorithm in order not to lose information. Batista et al. (2004) show that when the ratio of majority and minority is not so large, the simple random oversampling method performs as good as other more complicated algorithms. In our data, the ratio of the majority and minority is not so large even in each price interval. Therefore, we used the random sampling algorithm to balance our sample based on each item price interval. After balancing it, the total dataset is about 10 percent larger.

Different models had different performances after balancing the data. The cost performances of different training methods and cost sensitivity threshold methods with balanced data can be seen in Table 11 and Table 12. Negative results, that led to an increase of the total cost, were labeled red in Table 11 and Table 12. It becomes clear, that applying the OB on the balanced sample generated higher cost, contrary to the ET. However the general performance of the training model is unchanged. The same models that resulted in lower cost when being trained on the original dataset still result in low cost when being trained on the balanced dataset. The best model, gradient boosting, still reduced the total cost by 1.000 € after balancing the data. The decision tree showed the strongest performance improvement, but overall it was still low. Because training the stacking ensemble model on the balanced data was too demanding for the computer the respective value is missing. We balanced the data expecting that positive and negative samples are equally important (Elkan 2001). But since the error costs differ, it is possible that the total cost can again be reduced when changing the positive and negative sample ratio according the respective cost.

Total cost after balanced		Cost Sensitivity Method		
		OB	ET1	ET2
Model List	Logistic Regression	-216186.7	-210564.2	-198715.8
	Decision Tree	-222454.7	-220292.2	-220012.2
	Neural Networks	-213548.9	-208231.7	-197491.4
	Gradient boosting	-213277.7	-206237.9	-193586.3
	Random Forest	-228851.5	-211399	-201517.8
	Stacking Ensemble	***	***	***

Table 12: Cost after balancing data of OB, ET1, ET2

Total cost after balanced		Frequency Threshold		
		5	10	20
Model List	Logistic Regression	-200209.0	-202407.2	-203471.2
	Decision Tree	-220765.9	-221779.4	-222152.4
	Neural Networks	-198792.3	-201326.9	-202126.2
	Gradient boosting	-194926.8	-197073.8	-197945.5
	Random Forest	-202799.1	-205749.2	-206726.4
	Stacking Ensemble	***	***	***

Table 13: Cost after balancing data of OBET with Different Frequency Threshold

9. Conclusion

There is a phrase that we often came across during this project: “Data preparation accounts for about 80% of the work of data scientists”. Attesting this phrase, most of our discussions was spent on preparation. Many ways of transformations and techniques are suggested within the group, hence we decided on testing them by the degree to which they improve the AUC score on a test partition of the data. During test phases, most of the ways turned out to be inefficient. We avoided from outlier imputation except the max-min for user_dob and delivery_date, from variable selection except very highly correlated variables (over 0.7), from unsupervised clustering and also from interaction variables. On the other hand, we observed significant improvement when applied “double” WOE encoding to each categorical variable and use both transformations for each of them. We believe that this technique has made a comparable effect as clustering does. Finally, setting the split ratio of the data amongst training, test and WOE hold-out set brought with improvement but it is always prone to the risk of overfitting. We have taken some risk for the competition as splitting 60% of “known” for hold-out and 40% for training set in order to predict the “class” whose size is 50% of the “known”, but in this report, we have used a more reasonable splitting as 40% of “known” for WOE hold-out, 40% for training, and 20% for the test. We could apply at most 5-fold cross-validation, constrained by computational resources. In this setting, the stacking ensemble through which we utilize gradient boosting and random forest with the combiner logistic regression gave the best result.

When testing different approaches of finding the threshold which minimizes the total cost resulting from our prediction, we found out that the OBET performed the best and that balancing the data can, to some extend, improved the model performance in terms of cost. According to the task description, the shops costs of a returned item is 3€ plus 10% of the item value. Calculating the total costs of all returned items from out test set generated a result of 514.595,20€. Applying our model led to total costs of approximately 200.000€. Thus, considering only our test set, our model reduced total costs by more than 300.000€.

References

Sagiroglu, S., and Sinanc, D. 2013. *"Big data: A review". Conference Paper Collaboration Technologies and Systems (CTS), 2013 International Conference:* 42 - 47.

Ngai, E. W. T. et al. 2017. "Big Data Analytics in Electronic Markets". *Electron Markets*: 27- 243.

Davenport, T. H., Barth, P. and Bean, R. 2012. "How 'Big Data' Is Different". *MIT Sloan Management Review Magazine Fall 2012.*

Evans, J. R. and Lindner, C. H. 2012. "Business Analytics: The Next Frontier for Decision Sciences". *Decision Science Institute, March 2012.*

Gronau et al. 2016. "Business Analytics in der deutschen Praxis". *Controlling. 28:* 472-479.

Felden, C. 2016. "Business Analytics". *Enzyklopädi der Wirtschaftsinformatik (http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/daten-wissen/Business-Intelligence/Analytische-Informationssysteme--Methoden-der-/Business-Analytics), Accessed on 11.02.2018.*

Davenport, H. 2007. "Competing on Analytics". *Harvard Business Review Press.*

Manyika et al. 2011. "Big Data: The Next Frontier for Innovation, Competition, and Productivity." *McKinsey Global Institute.*

Bre, T. D. 2013. "Subjective Interestingness in Exploratory Data Mining" *International Symposium on Intelligent Data Analysis (IDA) 2013: Advances in Intelligent Data Analysis (XII).*

Fayyad et al. 1996. "From Data Mining to Knowledge Discovery in Databases", *AI Magazine 17 (3).*

Gan et al. 2007. "Data Clustering: Theory, Algorithms, and Applications", *ASA-SIAM Series on Statistics and Applied Probability: Philadelphia, Alexandria.*

Garcia et al. 2015. "Data Preprocessing in Data Mining"*, Springer: Switzerland.*

Good, I. J. 1985. "Weight of Evidence: A Brief Survey"*, Bayesian Statistics (2).*

Huang J., Ling, C.X. 2005. "Using AUC and Accuracy in Evaluating Learning Algorithms", *IEEE Trans. Knowl. Data Eng. 17(3).*

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

Provost F. et. al. 1998. "The Case Against Accuracy Estimation for Comparing Induction Algorithms." *Proceedings of the Fifteenth International Conference on Machine Learning.*

Provost F., and Fawcett, T. 1997. "Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distribution.", *In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining.*

Zdravevski et al. 2011. "Weight of evidence as a tool for attribute transformation in the preprocessing stage of supervised learning algorithms." (https://www.researchgate.net/publication/261351467), Accessed on 08.02.2018.

Breiman, L. 1996. “Bagging predictors”*, Machine Learning 24(2)*: 123-140.

Friedman, J. H. 1999. "Greedy Function Approximation: A Gradient Boosting Machine” (http://www-stat.stanford.edu/~jhf/ftp/trebst.pdf).

Wolpert, D. 1992. “Stacked Generalization”*, Neural Networks*, 5(2), pp. 241-259.

Packages

Breiman, L., and Cutler, A. 2015. “randomForest” Version 4.6-12 (https://cran.r-project.org/web/packages/randomForest/randomForest.pdf), Accessed on 11.02.2018.

Ripley, B., and Venables, W. 2016. “nnet” Version 7.3-12 (https://cran.r-project.org/web/packages/nnet/nnet.pdf), Accessed on 11.02.2018.

Kuhn, M. 2017 “caret,” Version 6.0-78 (https://cran.r-project.org/web/packages/caret/caret.pdf), Accessed on 11.02.2018.

Deane-Mayer, Z. A., and Knowles, J. E. 2017. “caretEnsemble,” Version 2.0.0 (https://cran.r-project.org/web/packages/caretEnsemble/caretEnsemble.pdf), Accessed on 11.02.2018.

Calaway, R., Microsoft Corporation, et. al. 2017. “doParallel,” Version 1.0.11 (https://cran.r-project.org/web/packages/doParallel/doParallel.pdf), Accessed on 11.02.2018.

Wickham, H. 2017. “forcats” Version 0.2.0 (https://cran.r-project.org/web/packages/forcats/forcats.pdf), Accessed on 11.02.2018.

Anagnostopoulos, C., H., and D. J. 2015 “hmeasure,” Version 1.0 (https://cran.r-project.org/web/packages/hmeasure/hmeasure.pdf), Accessed on 11.02.2018.

Roever, C., et. al. 2015. “klaR,“ Version 0.6-12 (https://cran.r-project.org/web/packages/klaR/klaR.pdf), Accessed on 11.02.2018.

Bischl, B. et. al. 2016 “mlr,“ Version 2.11 (https://cran.r-project.org/web/packages/mlr/mlr.pdf), Accessed on 11.02.2018.

Ripley, B., Venables, W. 2016. “nnet,“ Version 7.3-12 (https://cran.r-project.org/web/packages/nnet/nnet.pdf), Accessed on 11.02.2018.

Robin, X., et. al. 2017. “pROC,“ Version 1.10.0 (https://cran.r-project.org/web/packages/pROC/pROC.pdf), Accessed on 11.02.2018.

Therneau, T. et.al. 2018. “rpart,“ Version 4.1-12 (https://cran.r-project.org/web/packages/rpart/rpart.pdf), Accessed on 11.02.2018.

Chen, T., He, et. al. 2018. “Xgboost,“ Version 0.6.4.1 (https://cran.r-project.org/web/packages/Xgboost/Xgboost.pdf), Accessed on 11.02.2018.

Thoppay, S. M. 2015. “woe,“ Version 0.2 (https://cran.r-project.org/web/packages/woe/woe.pdf), Accessed on 11.02.2018.