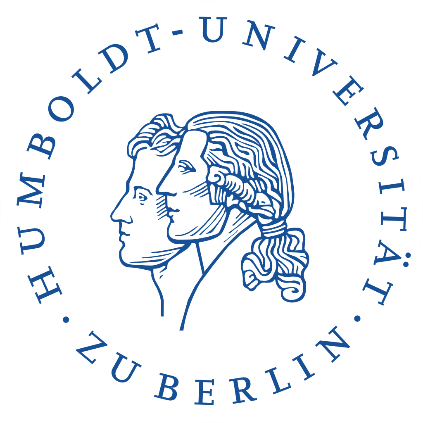
Business Analytics and Data Science

Assignment Report

****

Professor Dr. Stefan Lessmann

Johannes Haupt

*12/02/2017*

**Academic Year 2016 - 2017**

*Group 45*

*Lorenzo Montrone ID 428438*

*Patrick Gill ID 566186*

**Index**

1. Introduction
2. Analysis

2.1 Experimental Setup

2.2 Data

2.3 Performance Measure

2.4 Model Evaluation

3. Conclusions and Limitations

*Bibliography*

**1. Introduction**

Two of the main objectives of companies is to acquire new customers and to retain those customers that have already been acquired. The value of a customer goes beyond its actual purchase: long term customers usually spend more and more often, are less vulnerable to competitor offers and can spread positive feedbacks about the company.  It is, therefore, important to take into account the so called “customer lifetime value”, namely the value of a customer relationship based on the present value of the projected future cash flows from its relationship (Farris et al. 2010). It is the role of the customer relationship management (CRM) to deal with the entire lifetime of a company's customers and with this regard predictive modelling has been widely studied and applied by the data mining and machine learning communities to tackle those problems (Bahnsen et al. 2015).

Customers of e-commerce shops often order in that specific shop just once, this phenomenon can be compared with the “customer churn” problem, on which there is a wide literature both from a strict marketing perspective and from computer science applications.

In this paper, we are going to deal with a real-world data set of an e-commerce shop with the attempt to predict those clients that are not going to make another purchase in the future in order to target them with a coupon. We are, therefore, going to combine the two branches of literature on the problem of applying predictive models to the data to grasp the underlying structure and making predictions.

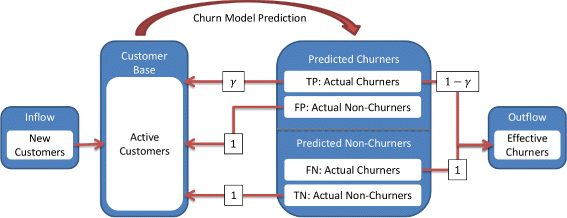
****

Figure 1: Flow Analysis of a churn campaign (Verbraken 2012 et al.)

A coupon poses a cost of foregone profit to the retailer when it is used. In cases where a customer would have made a follow-up purchase even without the coupon incentive, the coupon value is effectively wasted. For this reason, rather than sending coupons to all customers, we want to target only specific and promising customers.

In many other real-world applications, the differences between different misclassification errors can be quite large. Cost-sensitive learning has therefore received much attention in recent years to deal with such an issue (Elkan C. 2001).

In our paper, we are going to apply a set of models and evaluate them on the base of the expected net benefit of their predictions using a customized evaluation function with an empirical tested threshold (Sheng et al. 2006).

1. **Analysis**

**2.1 Experimental Setup**

As mentioned above, the underlying prediction problem is a binary classification which is additionally characterized by *data imbalance* and *cost sensitivity.* As an imbalanced classification problem, we consider a classification problem where “the examples of one class significantly outnumber the examples of another class” (López et al., 2013). Moreover, “the minority class represents the most important concept to be learned” (lbid). Although there is no fixed definition of what “significantly outnumber the examples of another class” exactly means, i.e. there is no benchmark ratio that defines datasets as imbalanced, we consider the underlying *imbalance-ratio* (ratio of number of majority cases to the number of minority cases) of 4.3 within the target variable as a justifying indicator of an imbalanced data set. This imbalance needs to be acknowledged in the analysis since “most commonly used classification algorithms do not work well for such problems because they aim to minimize the overall error rate, rather than paying special attention to the positive class” (Chen et al. 2004). This bias towards predicting the majority class is particularly problematic if there are different costs associated with wrongly classifying the cases (cost sensitivity) which is the case in the underlying problem (see above).

Many methods have been proposed to tackle the problem of class imbalance which can be broadly grouped into three categories:

*Data sampling:* This approach summarizes methods “in which the training instances are modified in such a way to produce a more or less balanced class distribution that allow classifiers to perform in a similar manner to standard classification” (López et al. 2013).

*Algorithmic modification:* These methods focus on adapting the base learning algorithm “to be more attuned to class imbalance issues” (lbid).

*Cost-sensitive learning:* These methods aim to acknowledge the existence of misclassification costs and implement these costs either “at the data level, at the algorithmic level, or at both levels combined […] and therefore trying to minimize higher cost errors” (lbid).

The following part explains the specific setup of our analysis to implement the above-mentioned concepts within the underlying classification problem. First, we split the data set into training set and test set with a ratio of 0.8. We then consider, through both an empirical and a theoretical perspective, the following set of base models:

A *neural network* model was chosen for its ability to approach for nonlinear classification tasks. As demonstrated by Hornik et al. (1989) “standard multilayer feedforward network architectures using arbitrary squashing functions can approximate virtually any function of interest to any desired degree of accuracy, provided sufficiently many hidden units are available.” We also choose to implement two *Homogeneous Ensemble Algorithms*, namely *Gradient Boosting* and *Random Forest*.

A *random forest* is an ensemble of (many) decision trees in which each tree is generated using a random subset of variables and usually fully grown. The final predictions are calculated (in most standard algorithms) as a majority vote of the single tree predictions. This allows it to be more robust with respect to noise (Breiman, 2001). *Stochasting Gradient Boosting* (Friedman 2002), also uses trees as base models and incorporates ideas from bagging. However, the base model decision trees are not fully grown but shallow instead. This results in an ensemble of simple predictors that consequently make mistakes due to their simplicity. However, the classification errors of a single predictor are then used to optimize the subsequent predictor by assigning a larger weight to the misclassified cases that the subsequent predictor then tries to correct. *Logistic regression* was chosen as a benchmark model for the classification problem.

Those models were preferred to others due to their superior performance in recent benchmark studies, such as Caruana et al. (2006), which performed an evaluation based on different metrics of 10 different models. Their results highlight Random Forest and Neural Networks as the models that returned the best average performance before calibration outperforming models such as KNN, logistic regression, Naive Bayes and Decision Trees. Additionally, within competitions of the machine learning community such as *Kaggle Competitions* or *Data Mining Cup*, they often belong to the models that give the best results together with *heterogeneous ensemble* methods.

We tune each of these models by looping through a specified parameter grid to find the optimal parameter combination that yields the best results. In order to find out “the best results” we apply the cost metric explained in part 2.3 since it correctly accounts for the problem specific misclassification costs. We use 5-fold cross-validation on the training set and thus get 5 performance measure estimates for each parameter combination. We calculate the mean of each estimate and extract the best parameter combination based on that criteria. Finally, we predict the best model on the test set and measure again performance based on the two metrics.

After building this set of unprepared base models as a benchmark we consider the class imbalance and use the resampling technique *SMOTE* which has been first proposed by Chawla et al. (2002). This technique consists in over-sampling the minority class creating “synthetic” examples rather than by over-sampling with replacement. The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining the *k* minority class nearest neighbors (Chawla et al. 2002).

After resampling the training set to a class-ratio of 1:1, we apply the same steps as above to get a comparison between the algorithm performance with and without resampled data. In the third part, we put the base models together in a *heterogenous ensemble* to evaluate if this approach further improves performance. Thereby, we combine the predictions of each of the *best tuned* single models and train a stochastic gradient boosting algorithm on their base. This algorithm is preferred since the base model predictions are likely highly correlated and it is more robust with respect to multicollinearity.

Lastly, we create another ensemble that is modified in such a way that it induces a better degree of diversity in base model predictions by manipulating some of these models to be over-optimistic (tendency to predict the positive case with a higher probability) – compared to the rather over-pessimistic, unmanipulated base models.

Finally, we compare all approaches based on the cost measure and decide for the best approach that we will then use for predicting the *class data.*

**2.2   Data**

The raw form of the provided data set does not yet allow for further analysis and must therefore be processed accordingly to get the data into a feasible format. The following lines present a documentation of the data processing that has been done.

As a first step, we identify those variables that are by their nature categorical and transform them into the according format. In R, a useful approach to handle categorical variables is the usage of the “factor” format which transforms variables into categorical ones and enables automatic creation of dummy variables within algorithms. Hence, we transform the following variables into factors: *title, giftwrapping, coupon, newsletter, delivery, referrer, cost shipping, form of address, model, goods value, advertising code* and *return customer.* Furthermore, we transform the date variables (*order date, account creation date, delivery date actual* and *delivery date estimated*) into the “year-month-day” format. The *weight* variable is discretized by binning it into four intervals and transformed into a factor after.

From the data table, we remove the *ID* column since it is not needed as a separate variable in the data frame and the *points redeemed* variable, because it is 0 throughout the entire data frame and thus does not add any information. The *advertising\_code* variable is a factor with 50 levels, we further created a binary variable from it to consider a possible reduced effect of advertising code (e.g. not a specific code that matters but the sole existence of any code).

From the provided date variables, we calculate three new variables: *delivery\_time\_actual* (the difference between order date and actual delivery date)*, delivery\_time\_estimated* (difference between order date and estimated delivery date)*, difference\_registration\_order* (difference between registration date and order date). Further, we take *order\_date* and *account\_creation\_date,* extract month and year of these dates and put them into new variables, in order to abstract from the exact dates and perhaps detect some time dependent trend or seasonality. Moreover, *account\_creation\_date* includes 3412 *NA*’s.  In general, when it comes to handling missing values in a data set, there are several assumptions that can be made about the occurrence of these values which in turn affect the methods that are used to deal with them. Missing values can be considered as “structurally missing”, if there is an inherent logic or reason for those values to be missing. They are considered as “randomly missing”, if there is no apparent reason. If values are structurally missing they can further have “informative missingness”, e.g. the fact that they are missing is directly related to the outcome. Therefore, it is crucial to understand *why* missing values occur (Kuhn and Johnson, 2013). Based on information about the “logic” of missing values and the nature of the underlying problem to solve one can decide for proper techniques to handle those values.

In the case of missing data for *account\_creation\_date\_new* we see that 98% of those values occur when *order\_date\_new* is 2013-04 which in turn is the minimum date of the dataset. We therefore assume that it occurred due to a technical problem at collection of the underlying data set and impute these dates by the value of *order\_date\_new* since these values are identical in 98% of cases in the dataset.  Missing values in *form\_of\_address* are recoded as “Missing”.

*Deliverytime\_estimated* contains 55 values that are in the range of -1455 to -1459 because *delivery\_date\_estimated* is in the year 2010 and thus approximately three years before *order\_date.* Therefore, we consider them as technical errors and set them to -1 to mark them. We do the same for those 12 values that are in the range of 998174 – 998177. In this case, we mark them by assigning the value of 372 (the last “regular” value is 371). Another structural break in that variable must be taken into account as there is a sudden jump in the value range from 187 to 367-371. Since 2910 cases lie in that area, we do not apply outlier transformation and instead mark them by assigning the value 365. Missing values for *deliverytime\_actual* occur because these transactions contain either no physical goods, canceled items or no items at all. Hence, there has not been a physical delivery and therefore no date is provided. Consequently, we replace these NA’s by assigning a 0. After those transformations, we recalculate the *diff\_delivery* variable.

As a next step, we discretize all “count variables” by putting them into one of the following categories: *No item, One Item, Multiple Items* or *Extreme.* To be assigned to the latter category, an item must be greater or equal than 2 AND greater than the 0.99 quantile of that count variable.

Moreover, the following new features are generated:

* *Post\_code\_area*: Using the postcode map of Germany we reduced the 99 different levels of the post code variable in 10 levels which represent different geographic areas of Germany
* *Day week*: Using the order date we extract the day of the week in which the order was placed
* *Xmas buy*: From the order date, we create this binary variable which is 1 when the order is placed 20 days or less before Christmas;
* *Wrong present*: combining *gift wrapping* and *remitted item*s;
* *Advertising and coupon*: when the customer had both an advertising code and a coupon;
* *Week-end order*: binary variable which is 1 when the order was placed Friday, Saturday or Sunday
* Lastly, we assume an interaction between referrer (customer came to the shop via a partner link or not) and coupon (has a coupon or not). Hence, we create a new factor variable *referrer\_coupon* with four levels (“Referrer and Coupon”, “Referrer and no Coupon”, “Coupon and no Referrer” and “No Coupon and no Referrer”)

The remaining numeric variables in the set are now standardized.

To enable both remaining date variables (*order\_date\_new, account\_creation\_date\_new*) to be included in the algorithms that we later use for analysis, we transform them to factors (some algorithms cannot handle variables in date format). This results in two factor variables with each 12 levels. As mentioned above, R automatically generates the necessary dummy variables from factor-type variables resulting k-1 dummies for a k-level factor variable. Thus, a dataset containing many factor variables (that each may have many levels) can quickly become infeasible in terms of dimensionality. We therefore decide to replace *email\_domain, weight, advertising\_code, order\_date\_new, postcode\_invoice* and *account\_creation\_date\_new* by their *weight of evidence* (WOE). We estimate the WOE, following Zdravevski et al. (2011), on a training set that contains 50% of the data and replace the mentioned factor variables by it. Due to a very low *information value* for *email\_domain* and *postcode\_invoice,* we remove these variables. Further, we remove *delivery* from the data set because it is “Yes” if and only if *payment* is “Cash”. Thus, it is linearly dependent and all information is already incorporated in *payment.*

**2.3   Performance Measure**

Most classifiers aim to minimize the overall error rate, which is a valid strategy only if the costs of different errors are equal. In our specific classification problem, we are dealing with an asymmetric cost matrix, in fact predicting customers to not return and therefore sending him a coupon, when he would have returned anyway has a greater cost than not sending a coupon to a customer that would have not returned. The table below shows the cost-benefit matrix of the underlying classification problem.

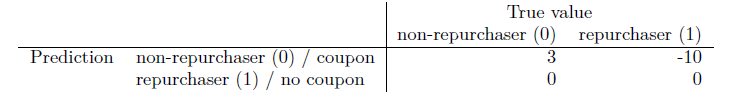


Figure 2: Cost-Benefit matrix

A crucial step in our analysis is to define the right evaluation criteria for our models in order to take into account those different costs. We, therefore, decided to implement an empirical thresholding method as it is proposed for example in Sheng et al. (2006). Since our base models predictions are probabilities, we iterate through every possible cut-off from 0.01 to 1 in 0.001 steps to decide for a case to return or not. For every cut-off, the costs are calculated according to the cost matrix.

For finding the optimal threshold we calculate an upper benchmark which is equivalent to the possible gain achievable, if every case is correctly predicted. Consequently, our performance measure calculates the percentage of the gain of every threshold compared to the upper benchmark and returns the threshold with the best value. We also take into account a lower benchmark, that is a minimum level of gain that our model should achieve in order to add any value. Due to the class distribution within the dataset and the cost matrix it can easily be computed as follows:

If the class distribution is considered as an estimate of a customer to return or not, the probability of a customer returning is 9773/51884 = 0.19. Hence, the expected profit per customer is calculated by 0.81\*3 - 0.19\*10 = 0.53 > 0. Since the expected profit per customer (without any knowledge of the likelihood of a return!) is positive, it is only reasonable to send a coupon to every customer. Consequently, our model must beat the lower benchmark, that is, the gain that is achieved without any knowledge and thus by sending a coupon to everyone.

**2.4 Model Evaluation**

The following table summarizes the results of the first part of the analysis:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Random Forest** | **Neural Network** | **Gradient Boosting** | **Logistic Regression** | **Stacked Ensemble** |
| **Original Data** | 0.307805 | 0.3284256 | 0.3331354 | 0.3351144 | 0.3366975 |
| **SMOTE (1:1)** | 0.302066 | 0.2675532 | 0.3287422 | 0.2881343 | --- |

Table 1: Comparison of the performance of the best tuned versions of each considered base model on the test set using the percentage of maximum gain achieved (see 2.3). Additionally, the performance of the stacked ensemble model is given.

The base models perform on a comparable level, i.e. no model significantly outperforms the others. Notably, throughout many simulations on different data, the logistic regression which is the least “sophisticated” model performed if ever slightly less well than the gradient boosting (which was most often slightly better than the others) and often enough achieved almost equal or even little better gain. When comparing the confusion matrices of the single models at the level of the optimal calculated cut-off level it becomes evident that all models tend to underestimate the likelihood of a return and thus have a strong tendency to predict the negative cases. It is therefore surprising that the models perform significantly worse when trained on SMOTE data that contain an equal amount of cases. As expected, they had a good performance on cross validation sets (since they were generated from the SMOTE training set), however, the structure these models learned on the SMOTE data did apparently not describe the structure of the real data well. Further simulations with SMOTE data that is less balanced (we used 1:1) could potentially improve overall performance.

Due to the similarity of the base model predictions, we expected the ensemble not to be significantly superior to the single models and indeed, it could not achieve an important amount of additional gain. Hence, we simulated another approach in which we try to manipulate the balance of the ensemble by combining over-optimistic predictions with the rather pessimistic predictions from the standard base models. We achieve that by inducing class weights into the random forest algorithm, thereby pushing it towards the positive class. We simulate several runs where we either manipulate the ensemble balance by the number of overconfident random forests we add, by the degree of their overconfidence or both. Results show that a higher number of less strictly manipulated random forests within the ensemble does not significantly improve the results compared to less manipulated trees that, however, are manipulated by their class weight more strictly. Consequently, we decide for a final ensemble that contains two additional random forests where the class weight that is put to the positive class is four times bigger than the class weight of the negative class.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Random Forest** | **Neural**  **Network** | **Gradient Boosting** | **Logistic Regression** | **Stacked Ensemble** | **Stacked Ensemble**  **(modified)** |
| **Original Data** | 0.307805 | 0.3284256 | 0.3331354 | 0.3351144 | 0.3366975 | 0.345035 |
| **SMOTE (1:1)** | 0.302066 | 0.2675532 | 0.3287422 | 0.2881343 | --- | --- |

Table 2: Comparison of the modified ensemble with the former models

By looking at the correlation between the predictions of the new ensemble one can see a slight improvement, although they are (as expected) still very correlated. What improves, however, is the diversity within the predictions since the two manipulated random forests predict the positive class with significant higher probability. As table 2 shows, this diversity in predictions gets accounted for in the ensemble as the performance increases by 0.9% to 34.5% of the maximum gain compared to the standard ensemble model.

Consequently, we decide for the manipulated ensemble model to predict the customer return in the final data set. Compared to the lower benchmark given the test set size of 10376 observations this prediction model generates an 58.5% improvement over sending a coupon to everyone which would be achieve a gain of 10376\*0.53 = 5499, whereas our model generates a gain of 8717.

**3. Conclusion and Limitations**

We applied several machine learning algorithms that showed to perform comparably well within both literature and empirical applications, trained and optimized them on the data set of an e-commerce shop in order to predict the likelihood of a customer to return. After putting lots of effort into optimizing the quality of the data and additionally generating new features from the data, the single algorithms already performed relatively well considering that they improve the gain by more than 50% that would be otherwise generated without applying any predictive modeling and thus sending a coupon to everyone. Evaluating the performance to the upper benchmark is not easily applicable since 34.5% of maximum gain implies still quite some misclassification, however it is impossible to know *ad hoc* how much gain is practically possible, i.e. how much structure finally is in the data that can get picked up and how much randomness is within a data set that is by its nature often non-deterministic compared to data from nature sciences for example. Moreover, we combined the base models into two ensemble models where the manipulated ensemble could further achieve a gain of approximately 1%.

In order to further improve the performance of our model we would have liked to implement several more approaches but could finally not do so due to lacking human- and thus time- and computational resources. These approaches automatically define the limitations of our models. Among others, a greater variety of SMOTE data combined with a vaster set of models would be worth to test. In our case, we tried a balanced set and did not further apply our ensembles to SMOTE data due to their bad performance on the base models. It would also be worth to further expand the approach of manipulating a broader set of base models within an ensemble and in addition use a more diverse set of models like Bayesian models or support vector machines. The latter, unfortunately turned out to be too computationally demanding and could thus not be implemented in a feasible manner. Consequently, with a larger set of base models, more sophisticated ensemble strategies like greedy hill-climbing could be applied and more possible stacking models, including meta-parameter tuning. This could also reduce the variance in performance of the ensemble models and stabilize the results as it has to be noted that the final ensemble model could not outperform the standard ensemble approach in all simulations that have been run on different data sets (although it did in the majority). Lastly, it might have some gains to put special effort to variable importance and feature extraction. However, the results we got where quite contradicting when using different approaches and performance of accordingly reduced data sets did not add any improvement. We therefore did not focus on any more sophisticated methods in that area and decided to keep all information in the data set.

*Bibliography*

Bahnsen, Alejandro Correa, Djamila Aouada, and Björn Ottersten. "A novel cost-sensitive framework for customer churn predictive modeling." Decision Analytics 2.1 (2015): 5.

Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

Caruana, Rich, and Alexandru Niculescu-Mizil. "An empirical comparison of supervised learning algorithms." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.

Chawla, Nitesh V., et al. "SMOTE: synthetic minority over-sampling technique." Journal of artificial intelligence research 16 (2002): 321-357.

Chen, Chao; Liaw, Andy; Breiman, Leo. “Using Random Forest to Learn Imbalanced Data”. Report ID: 666 (2004). Department of statistics, UC Berkley.

De Jonge, Edwin, and Mark van der Loo. "An introduction to data cleaning with R." Statistics Netherlands, The Hauge (2013).

Elkan, Charles. "The foundations of cost-sensitive learning." International joint conference on artificial intelligence. Vol. 17. No. 1. Lawrence Erlbaum Associates Ltd, 2001.

Fayyad, Usama M. Advances in Knowledge Discovery and Data Mining. Menlo Park, CA: AAAI, 1996. Print.

Farris, Paul W., et al. Marketing metrics: The definitive guide to measuring marketing performance. Pearson Education, 2010.

Friedman, Jerome H. "Stochastic gradient boosting." Computational Statistics & Data Analysis 38.4 (2002): 367-378.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2.5 (1989): 359-366.

López, V., et al. “An Insight into Classification with Imbalanced Data: Empirical Results and Current Trends on Using Data Intrinsic Characteristics”. Information Sciences 250 (2013): 113-141.

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. New York: Springer.

Shearer, Colin. "The CRISP-DM model: the new blueprint for data mining." Journal of data warehousing 5.4 (2000): 13-22.

Sheng, Victor S., and Charles X. Ling. "Thresholding for making classifiers cost-sensitive." AAAI. 2006.

Verbraken, Thomas, Stefan Lessmann, and Bart Baesens. "Toward profit-driven churn modeling with predictive marketing analytics." Cloud Computing and Analytics: Innovations in E-business Services. Workshop on E-Business (WEB2012). 2012.

Zdravevski, Eftim, Petre Lameski, and Andrea Kulakov. "Weight of evidence as a tool for attribute transformation in the preprocessing stage of supervised learning algorithms." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.