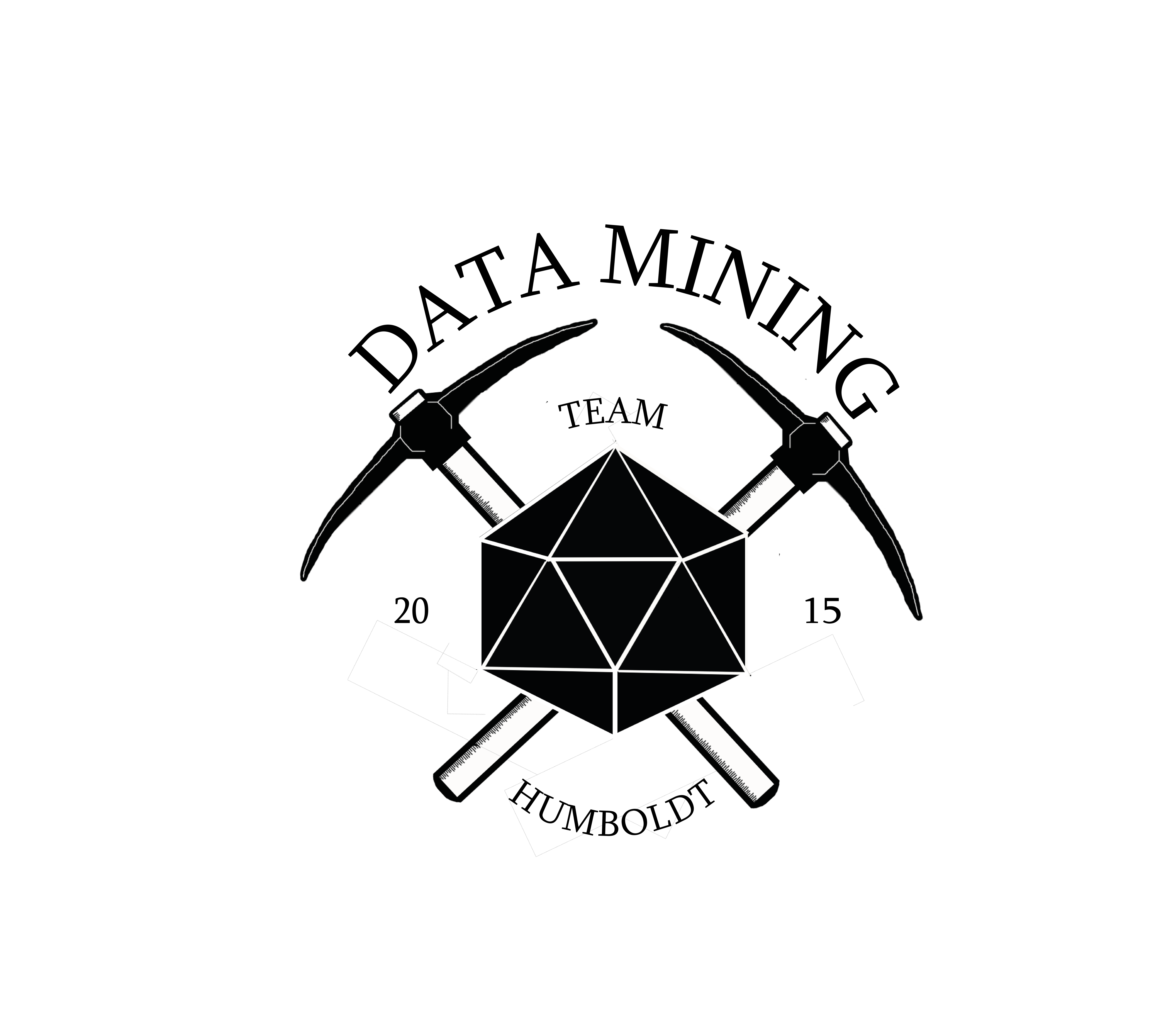
|  |
| --- |
|  |
| Business Analytics and Data Science  Special Working Task WS 2015/2016 |
|  |

|  |
| --- |
|  |



Maximilian Andres, 574900

Andra-Selina Pietsch, 572551

Frederik Pahde, 576260

Oleksiy Ostapenko, 5762570

Inhaltsverzeichnis

[1 Introduction 3](#_Toc440991543)

[2 Preprocessing 4](#_Toc440991544)

[2.1 Explorative Data Analysis 4](#_Toc440991545)

[2.1.1 Histogram of numerical variables 4](#_Toc440991546)

[2.1.2 Box-plotting the data 6](#_Toc440991547)

[2.1.3 Scatterplots 6](#_Toc440991548)

[2.1.4 Descriptive numerical summaries 7](#_Toc440991549)

[2.1.5 Correlation 7](#_Toc440991550)

[3 First Iteration 9](#_Toc440991551)

[3.1 Preprocessing: Data Cleaning 9](#_Toc440991552)

[3.1.1 Missing Value Handling 9](#_Toc440991553)

[3.1.2 Outlier Detection and Handling 9](#_Toc440991554)

[3.2 Transformation 11](#_Toc440991555)

[3.2.1 Data reduction: Deletion of highly correlated data 11](#_Toc440991556)

[3.2.2 Feature Selection 11](#_Toc440991557)

[3.2.3 Scaling of continuous variables 12](#_Toc440991558)

[3.3 Data Mining 12](#_Toc440991559)

[3.4 Evaluation 15](#_Toc440991560)

[4 Second Iteration 17](#_Toc440991561)

[4.1 Preprocessing 17](#_Toc440991562)

[4.1.1 Multivariate outlier detection 17](#_Toc440991563)

[4.2 Transformation 17](#_Toc440991564)

[4.2.1 Feature Selection 17](#_Toc440991565)

[4.2.2 Principal Component Analysiss 18](#_Toc440991566)

[4.3 Data Mining 19](#_Toc440991567)

[4.4 Evaluation 21](#_Toc440991568)

[5 Sources 22](#_Toc440991569)

# Introduction

The telecommunication market is a highly competitive market. Customers are able to choose between a variety of suppliers and it is relatively easy to switch between them. This results in a very high churn rate and represents a serious problem for the suppliers as the costs for attracting new customers exceed the costs of customer retention (e.g., Lu, 2002). An important and necessary task for the telecommunication companies is to predict the probability of customers to cancel the contract in the near future. This information can be used to start a loyalty initiative such as offering a discount to the customer. To gain valuable knowledge out of the customer data, data mining techniques can be used.

This working task deals with such a customer churn problem. The provided real-world data of 100,000 customers of an anonymous telecommunication provider are described by 173 attributes, such as customer characteristic and usage behaviour. The observed customers were with the company for at least six month and they were sampled during July, September, November of 2001 and January of 2002. At the time a customer was observed the input variables were calculated based on the previous four months. If a customer churned within a period of 31-60 days after the observation time, the data provides the information of “churn=1”, otherwise “churn=0”. The data set is divided into two parts of equal size (training and test set). One set represents the training set including the information if a customer churned. This set will be used for model training. Our goal is to predict the churn probabilities for each customer in the test set. The higher the probability the higher the risk that a customer will churn within the period of 31-60 days after the observation time. An accurate prediction gives the company sufficient time to react to the information and retain the customer.

In doing so we proceed in accordance with the KDD process (knowledge discovery in databases) (e.g., Fayyad, et al., 1996). The first step, selection, is already done as our target data is selected (see above). In chapter 2 we will do some exploratory data analysis which is part of the preprocessing to get an overview of the data. The following steps of the KDD process will be divided into two chapters (chapter 3 and 4). Our approach is to run two iterations during the model building process. Within the first iteration we will focus on rather simple methods (e.g. data cleaning). We will start with data cleaning such as missing value and outlier handling. Within transformation we will focus on data reduction and encoding. Within the next step of data mining we will apply algorithms to search for patterns in our data. In our case the data mining model is classification and seven popular classification methods will be applied: Logistic Regression, Neural Network, Random Forest, Naïve Bayes, K-nearest Neighbors, Decision Tree and Support Vector Machines. At the end of chapter 3 we will evaluate the results, decide which approaches worked well and which should be improved. In the second iteration we will handle the needs for improvement identified in the first iteration and focus on rather complex methods if required. We will see for example that during preprocessing and transformation the missing value and outlier handling, feature selection and feature extraction should be improved. During data mining we will only focus then on those models in the second iteration that performed well in the first iteration. Moreover we will also apply a heterogeneous ensemble method that makes a linear combination of the individual models. In doing so we want to further increase the predictive accuracy of the individual models.

# Preprocessing

Because of high dimensionality of the dataset it is hardly possible to analyse the existence of possible abnormalities in the data manually. In the preprocessing phase we apply several techniques of Explorative Data Analysis in order to gain more detailed insights into the given dataset.

## Explorative Data Analysis

The relative distribution of the target variable will be important for the relative frequency of the classes in the folds of cross validation. The given dataset exhibits the following class distribution of the variable „churn“: good – 25219 times (50,4%), bad – 24718 times (49,6%).

We apply the following techniques:

* histogram
* box plot
* scatter plots
* descriptive numerical summaries (mean, median, max, min etc.)
* correlation tables

### Histogram of numerical variables

We plotted and analysed all numerical variables of our dataset. Figure 1 shows an example of the five variables’ histograms.



Figure 1: Histrogram of numerical variables

Viewing all plots we discover that our variables are mostly normally distributed or follow some kind of exponential density function. It became evident that many numerical variables have a huge amount of observations valued with “0”, which appeared to be a standard value in many cases. Figure 2 shows for example the variables “age1” (blue) and “age2“ (red) –age of first and second adult in the household.



Figure 2: Age first and second household member

It is evident that in both variables many observations are valued with “0”, which is not a valid value for these variables. As standard value, “0” would falsify the results in the Data Mining phase, and should be treated as missing value in the transformation phase. Figure 3 shows the plot of the variables “age1” and “age2” without the missing “0” values.



Figure 3: Age of first and second household member

We take a look at the histograms of several variables in order to find possible outliers. Figure 4 shows the distribution of the variable “adjrev” - billing adjusted total revenue over the life of the customer, for all values (left) and only for the values greater than 8000 (right). Plotting data starting from a higher value (right figure) makes it possible to better identify outliers. Nevertheless we will use boxplots for outlier detection.



Figure 4: Billing adjusted total revenue over the life of the customer

### Box-plotting the data

For all numerical variables we create a boxplot, e.g. in Figure 5 with the range value of 3 for the variable “afjrev” (billing adjusted total revenue over the life of the customer). From the box plot it is evident that this variable has outliers. The outlier handling will be described in more detail in the following chapter in the individual iterations. It is important to say, the outlier analysis should be considered after the missing value handling, since the handling of missing/default values (such as default “0”) will influence the distribution of a variable and so the outlier detection.



Figure 5: Boxplot, billing adjusted

### Scatterplots

In order to get an impression about the dependency structure of several interrelated variables we create several scatter plots. Moreover, we have created several scatter plot matrixes, similar to the one presented in the Figure 6. From the presented scatterplot matrix for example can be seen, that the total revenue (totrev) tends to grow with the growing mean total monthly recurring charge (totmrc\_Mean). Furthermore, it is evident that there are many outliers in each of the scatterplot – the multidimensional outlier handling might be useful in this dataset.



Figure 6: Scatterplot Matrix

### Descriptive numerical summaries

Numerical summaries of variables are useful in many aspects while trying to gain more detailed insights into the data. Thus, for example, by calculating the median of a variable and evaluating the range between maximal and the minimal value of a variable can be understood if a variable has outliers. Using the “summary” function of R we calculate the following characteristic values of every numeric variable: mean, median, 1st and 3rd quartiles, as well as maximum and minimum values.

Thereafter, we scanned the numerical summary for the conspicuous values and errors. Thus, we have discovered a negative minimum value of the variables “REV\_MEAN” and “TOTMRC\_MEAN” that appeared to be an invalid outlier, since this variable cannot be negative as they represent revenue and the monthly recurring charges.

### Correlation

In order to visualize the correlation between the variables of the dataset, the correlation plot was used, the correlation plot is presented in the Figure 7. The dark blue colour indicates a strong correlation between variables. From the plot it is evident that there are several variables that correlate strongly (between 0,8 and 1). Consequently it will make sense to filter out the strongly correlated variables in the transformation phase of the KDD process.



Figure 7: Correlation Matrix

# First Iteration

## Pre-processing: Data Cleaning

### Missing Value Handling

In the explorative data analysis we detected a high amount of missing values. Thus, it is important to handle these values in an appropriate way in order to be able to train efficient predictive models. The first step is to define what a missing value actually is. Many variables already use the standard coding for missing values which is N/A. But there are still some variables that use a different encoding, e.g. 0 or ‘U’. We detected these differing encodings manually by looking at the data. We decided to replace these missing values by the standard representation for missing values (N/A).

Having a standard coding for missing values we calculate the missing value rate for each variables. Variables having a missing value rate larger than 50% get ignored since it makes no sense to work with this data.

The remaining missing values are handled doing an imputation. We use the *impute*-function of the *Hmisc*-package that implements a univariate imputation. Categorical missing values are replaced by the mode and numerical variables by the median which is more robust against outliers than the average. Another option would have been to keep the information that a value is missing but we decided against it.

### Outlier Detection and Handling

An outlier is a value that is significantly outside the range of the other values. While the explorative data analysis has shown that there are outliers in most of the variables of our dataset, in the data transformation phase these outliers should be detected and handled.

In the first iteration our team has decided to only handle the outliers detected by the means of the simple one-dimensional outlier detection method. In this iteration we have started with the simple boxplot outlier detection method experimenting with the heights of the antennas of the boxplot, finally used the robust z-score method as a final method for the outlier detection in this iteration. For the reasons of simplification we have handled the outliers detected by the named methods for every numerical variable.

#### Boxplot method

In order to find the extreme outliers for the numerical variables we have calculated the first and the second quartiles of the variable and the IQR. We have then calculated the whiskers as 3\*IQR. The values that were outside the limits set by the whiskers were considered as outliers. We handled the outliers by setting their value to the value of its closest whisker, which allowed us to get rid of the observation that are significantly out of the range of the values of the variable without influencing the values of the median and the quartiles of the variable.

The effect of the method is shown in the Figure 8 on example of the variable “rev\_Mean” (mean monthly revenue). On its left-hand side the figure shows a boxplot of the variable “rev\_Mean” before the outlier handling, and a boxplot of the same variable after the outlier handling on its right-hand side (cf. Buttler, n.d., pp. 7f).



Figure 8: Boxplots of the variable „rev\_Mean“

#### Z-score method

Being similar to the boxplot outlier detection method described above, z-score method appeared more robust to us because it calculates a score for each values based on the standard deviation.

We used the z-score formula ( ) to calculate the distance of the observation from the population’s mean measured in standard deviation. Every observation that appeared to have the z-score higher than 3 (or lower than -3) was considered to be an outlier. Every outlier was then set to the value the population’s mean ( added the three standard deviations (3\* (cf. Lessmann, 2015, Weinberg/Abramowitz, 2002 pp. 105f).

The result of the z-score outlier handling is presented exemplary on the variable “rev\_Mean” in the Figure 9.



Figure 9: Boxplots of the variable „rev\_Mean“ with range of 3 before(left) and after(rright) the outlier handling with z-score method

## Transformation

### Data reduction: Deletion of highly correlated data

In the first iteration, for the reasons of simplicity, our team has decided to forgo the sophisticated data reduction techniques like principle component analysis. Instead, we decided to simply remove highly correlated variables in order to reduce the number of variables. By using the caret package we removed the variables that are correlated to any other variables by the correlation factor of higher then 0.9.

### Feature Selection

One part of vertical data reduction is the variable selection where we want to find a subset of relevant variables. The underlying assumption is that not all of the available variables are relevant for the prediction. Out of a variety of feature selection approaches we decide within the first iteration to use a simple filter approach. In our case we use the rfe function out of the caret package (http://topepo.github.io/caret/filters.html). Through an underlying ANOVA model it is tested for each variable if the mean is statistically different between the two classes “churn=1” and “churn=0”. If so, the variable is regarded as relevant and should be considered in the prediction models. There are several disadvantages of a filter approach compared to other approaches, e.g. that redundancy and interactions of variables are not taken into account. Nevertheless considering that we have high-dimensional data we use the simple approach in the first iteration because of its much lower computational costs. As a result of the filter we get 68 variables that can be viewed as relevant and thus will be used as input variables for the data mining process.

### Scaling of continuous variables

To scale the continuous variables, we considered two methods: Min/Max and z-Transformation. The Min/Max method is more sensitive for outliers because the result is bounded. Thus we decide to use the z-Transformation (standardization).

The first step to implement the z-Transformation was to detect the continuous variables. To detect the continuous variables we considered two ways of coding. The first way was to implement an algorithm which detects the continuous variables by comparing the number of different numeric attributes of a variable with a specific number. If the specific number is lower than the number of different numeric attributes of a variable, the variable is continuous. The second way was to implement an algorithm which detects the continuous variables of the data by searching for the continuous variable name. We got the continuous variable name out of the description table.

The second algorithm is used to detect the continuous variables. The first algorithm leads to a misinterpretation of the variable if the prescribed number of different numeric attributes is lower than the number of categories represented by numeric attributes of a variable.

The second step to implement standardization is a function which scales a vector by z-Transformation. Instead of using the scale() function we coded the function z.scale() to standardize a vector.

The final step is to code a function which scales just the continuous variables of the data by standardization.

## Data Mining

After the data got cleaned and transformed in the preprocessing step the calculation of the predictive models can be done. We decided to train seven popular classification models in the first iteration: Logistic Regression, Neural Network, Random Forest, Naïve Bayes, K-nearest Neighbours, Decision Tree (J48) and Support Vector Machines.

Some of them are robust against outlier (Random Forest, Decision Trees, Naïve Bayes and K-nearest Neighbours) and others are not (Logistic Regression, Neural Network, Support Vector Machines). Thus, it makes sense to work with two different trainingsets. One of them contains the original input that includes outliers and in the other one the outliers are handled as stated in section 3.1.2.

We decided to use the Split-Simple approach in the first iteration: The trainingset (70%) for model training and a testset (30%) for the assessment of the trained models. This is an easy approach to avoid that the model is assessed using the same data used for training (Resubstitution estimate). Since the occurrence of both classes (churn/no churn) is approximately the same (see section 2.1) the data split can be done randomly.

For the model calculation we used the package caret. This package contains implementations for all classification models used to solve the predictive modelling task. There exist some meta-parameter for most of the models that can be set manually by the developer. For each of these models we defined a grid with possible parameters (see Table 3‑2). Caret is able to choose the best parameter combination for each model out of this grid. Moreover we decided to use a 10-Fold Cross-Validation for the model selection process.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Parameter 1 | Parameter 2 | Amount Combinations |
| Logistic Regression |  |  | 1 |
| Neural Network | size = [2,5,9] | decay = [0.1,1,10] | 9 |
| Random Forest | mtry = [3,5,7,9] |  | 4 |
| Naive Bayes | laplace = [0,1] | useKernel = [FALSE, TRUE] | 4 |
| K-nearest Neighbours | K = [3, 5, 7, 9, 11, 13, 15, 17] |  | 8 |
| J48 | C = [0.1, 0.2, 0.3, 0.4, 0.5] |  | 5 |
| Support Vector Machine | cost = [0.01, 0.1, 0.2, 0.3, 0.4, 0.5] | gamma = [2, 3, 4 ] | 18 |

Table 1: Parameter Combinations for Model Selection

In order to get more reliable data we decided to train every model 10 times, each time with a new randomly chosen subsample as trainingset. This allows us to investigate the sensitivity of each model to the random split.

Since every model is calculated multiple times (10 iterations x Amount of combinations [see Table 3‑2] x 10-Fold Cross Validation) a lot of computational power is needed for the model calculation. We used a vServer where the code could run multiple hours or days. Moreover the model training was parallelized using the package *doMC (*<http://topepo.github.io/caret/parallel.html>). This approach allowed us to compare many different models and parameter combinations.

The results of the first iteration are shown in Figure 10 using boxplots. It turns out that 3 out of the 7 models performed better than the others (Logistic Regression, Random Forest and Support Vector Machines). It makes sense to have a closer look on those models in the second iteration. Therefore the results of the model selection process of those models are shown in Figure 11 and Figure 12 (Logistic Regression has no meta-parameters).



Figure 10: Error Rates of the 1. Iteration for Neural Network (NNET), Logistic Regression (LR), Naive Bayes (NB), Random Forest (RF), K-Nearest Neighbors (KNN), Support Vector Machine (SVM) and Decision Tree (J48)



Figure 11: Model Selection Random Forest (on small subset)



Figure 12: Model Selection SVM (on small subset)

## Evaluation

An evaluation of the first iteration requires that we look at preprocessing, transformation as well as the choice of algorithm under the point of view of a first attempt to predict the churn. The first attempt is applied in form of an univariate outlier detection, the approach of replace missing values with the median, without a treatment of categorical variables, a feature selection using univariate filter approach, a data set without outliers for unrobust models and a data set with outliers for robust models. We consider taking a smaller subset (5.000 samples) to compare several models as well as the meta parameters. We decided to take a smaller subset in order to save computational cost. We considered the trade-off between saving computational cost and accuracy. The best three models of the first iteration have classification accuracy between 60 and 62 %.

The classification accuracy of our first iteration depends on a simple attempt to predict the churn of 5.000 samples. To improve the prediction we take the following factors in account: A multivariate outlier detection, a more effective wrapper approach for selecting the relevant features, tune the meta parameters and to take the 50.000 samples for the final model.



Figure 13: Accuracy depending on Random Forest meta parameters



Figure 14: Accuracy depending on Support vector machines meta parameters

Based on comparing figure 13 and 14 with figure 11 and 12, it can be concluded that further meta parameter tuning lead to a higher accuracy.

The more complex code would be the second iteration of our KDD Process. We could also compare the results of the second iteration with the first iteration to answer the question: “Is there an improvement?”

# Second Iteration

## Preprocessing

### Multivariate outlier detection

We consider two algorithm for multivariate outlier detection: Feature-bagging based outlier detection with local outlier factor and angle based outlier detection.

The feature-bagging based outlier detection can be considered as an ensemble method. The ensemble method compares results of several outlier detection algorithms. Every outlier detection algorithm uses a small subset of random variables to detect outliers. Every detected outlier ascribes a probability of being an outlier. The probabilities of being an outlier are compared to find outliers with the highest probability to be an outlier.

We considered applying the HighDimOut package. The HighDimOut package provides a function called Func:FBOD and Func.ABOD.

The Func.FBOD uses bagging based outlier detection method with the help of local outlier factor (LOF). The local outlier factor describes how remote a sample is. The degree of isolation depends on the local distance to neighbors. The result of Func.FBOD is a vector which contains the score of feature-bagging based outlier detection based on local outlier factor.

The Func.ABOD uses angle based outlier detection algorithm. The used angle based outlier detection algorithm calculates the angle variance of an object to the neighbors. The angle variance is used to decide if an object is an outlier or not. An outlier has a low angle variance.

We applied Func:FBOD and Func.ABOD on a subset with 100 samples. The Func.ABOD calculated still after two hours. The results of the Func:FBOD seemed to be an improvement. Nevertheless, compared to our univariate outlier detection the computational time consumption increased for both algorithms. We have to consider the computational time consumption as a factor of using this multivariate outlier detection due to our limited computational resources. Our limited computational resources lead to the decision of using univariate instead of multivariate outlier detection for the second iteration again.

## Transformation

### Feature Selection

During the first iteration we noticed that the filter approach only selected numerical variables although we assume that also non-numerical variables are relevant for our predictions. To improve the feature selection process we now apply a wrapper approach. Using a prediction model the variables are here treated as inputs and model performance as the output that shall be optimized. We use a Random Forest Variable Importance (RFVI) method which is included in the caret package. We expect it to have higher computational costs because a random forest has to be trained prior to importance ranking, but probably it will perform better than the filter approach.

Unfortunately computational costs with all variables included are extremely high and after one and a half days we stopped the execution. When working with high-dimensional data it is common to use a hybrid strategy, so reducing the number of variables by firstly using a filter and afterwards a wrapper approach. As our filter only selected numerical variables, we continue our wrapper approach with using these selected variables and additionally all categorical variables. After implementing this adjustment we execute the random forest training and importance measurement again.

Table 2 shows the ten most important variables resulting from our RFVI. Here we can see for instance that “eqpdays” (number of days (age) of current equipment) is regarded as the most important variable. Furthermore we notice that many categorical, non-numerical variables are regarded as important. This confirms our decision to not rely on the filter approach. The non-numerical, categorical variables are now recoded into binary ones which we can see for example by looking at variable “dualbandN”. This is the binary variable for the original variable “dualband” with level “N”. This leads to a large increase in dimensionality with more than 2000 variables now in total.

|  |  |  |
| --- | --- | --- |
| No. | Name of variable | RIS |
| 1 | eqpdays | 7.446 |
| 2 | hnd\_price | 6.311 |
| 3 | mou\_Mean | 5.901 |
| 4 | mou\_opkv\_Mean | 5.578 |
| 5 | iwylis\_vce\_Mean | 5.467 |
| 6 | comp\_vce\_Mean | 5.387 |
| 7 | Models | 5.262 |
| 8 | mouowylisv\_Mean | 4.904 |
| 9 | dualbandN | 4.861 |
| 10 | complete\_Mean | 4.817 |

Table 2: Ten most important variables by Random Forest Variable Importance

We decide to include all variables having a raw importance score (RIS) larger than zero. Moreover we include a categorical variable if at least one level of the variable is part of our selected variables (e.g., the variable “ethnic” consists of 17 levels, but due to our importance ranking only eight levels (coded each as a binary variable) are regarded as important; nevertheless we include the whole variable “ethnic”). Having done that we get 93 selected input variables.

### Principal Component Analysiss

In the exploratory data analysis we have seen that there are several highly correlated variables. In order to reduce the dimensionality of the given dataset we use a principal component analysis (PCA). PCA is applied as the last data preprocessing step, because its outcome, completely new set of uncorrelated variables with high variation, is not easily interpretable from the business point of view (cf. Jolliffe, 2002).

The variance of a dataset is a measure of the information content. In PCA a correlation matrix of the previously normalized dataset is created. The eigenvectors of the correlation matrix are the uncorrelated principal components of the dataset. To find the principle components of the numerical variables the R function “princomp(x, ...)“ has been used. The sum of the eigenvalues represent the total variance in the dataset, the corresponding eigenvalues of the eigenvectors represent the proportion of variance explained by each eigenvector (principle component). Figure 15 pictures the amount of variance explained by the first ten components.



Figure 15: Variance explained by the first ten components

For further analysis only components with an eigenvalue larger than one will be retained, meaning that these components explain at least the same amount of variance as the original variables. By applying PCA, 87 numeric variables (“Customer\_ID” excluded) could be reduced down to 16.

## Data Mining

In the second iteration we decided to have a closer look on those models which performed good in the first iteration: Logistic Regression, Random Forest, Support Vector Machines. Additionally we had a look on Stochastic Gradient Boosting which turned out to be very efficient as well.

In order to get more accurate results we use a heterogeneous ensemble method. The models that have shown strong results (Logistic Regression, Random Forest, Support Vector Machines and Stochastic Gradient Boosting) are included in the ensemble model. We considered different combinations of base models for the ensemble. The result of the ensemble is the average over all individual class probabilities. The other option would have been to use the *caretEnsemble*-package. But there are two problems using this package. The first one is that only one trainingset can be used for all individual models using this package. Thus, it would not be possible to distinguish between models that are robust against outliers and models that are not, as done in iteration 1. The other problem is that the results of this ensemble method have a very high variance.

Moreover we use the Lift Measure as evaluation metric since the final model will be evaluated using this measure. We split our trainingset (18,000) and a testset (2,000) for the model evaluation. The results of the evaluation process are shown in Figure 16.

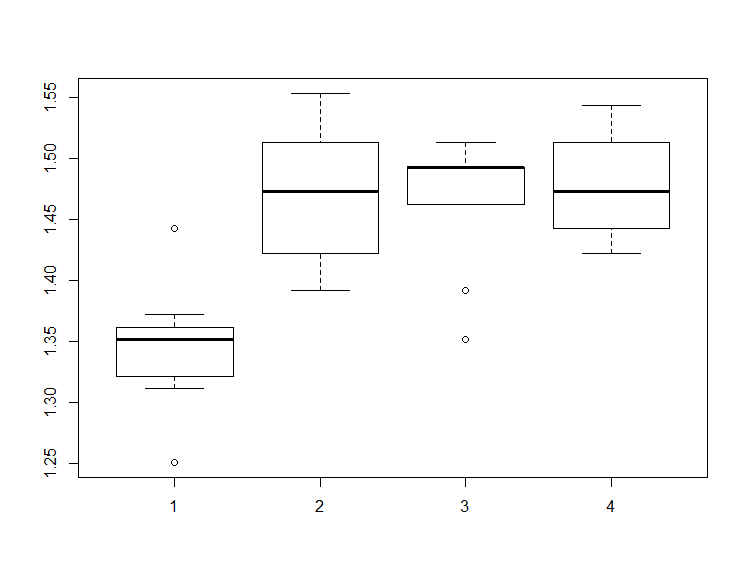


Figure 16: Results of Logistic Regression (1), Random Forest (2), Stochastic Gradient Boosting (3), Ensemble (4)

Obviously, Logistic Regression is weaker than the other two base models (Random Forest, Gradient Boosting) and influences the ensemble negatively.

Thus, it makes sense to build the ensemble without Logistic Regression. The results are shown in Figure 17.

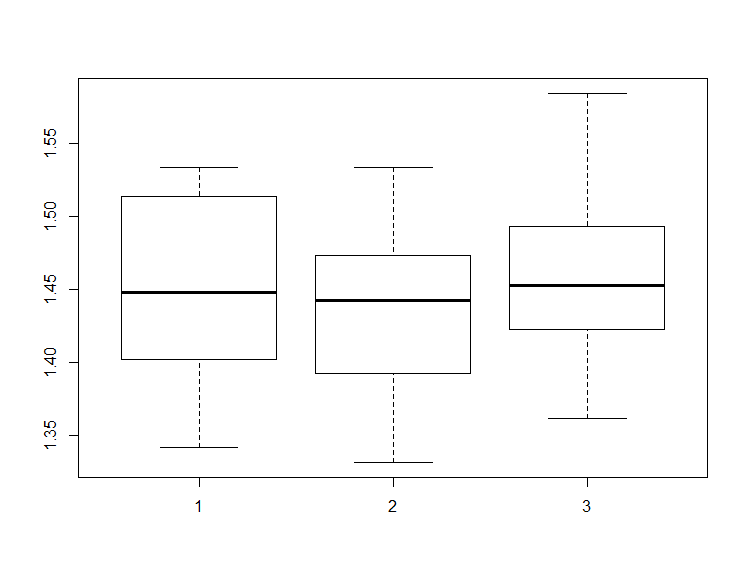


Figure 17: Results of Random Forest (1), Stochastic Gradient Boosting (2), Ensemble (3)

## Evaluation

Now the ensemble performs better and more stable than the base models. The results on the subset are satisfactory using the lift measure as metric.

Thus, we chose a heteregenous ensemble method as final predictive model. The base models are Random Forest and Stochastic Gradient Boosting Machine.

As trainingset for the final model prediction we use the whole dataset (50,000 rows).

# Sources

Buttler, G. (n.d.). Ein einfaches Verfahren zur Identifikation von Ausreißern bei multivariaten Daten. [online] Available at: http://www.statistik.wiso.uni-erlangen.de/forschung/d0009.pdf [Accessed 28 Dec. 2015].

Jolliffe, I. (2002). *Principal component analysis*. New York: Springer.

Lessmann, S. (2015). *Business Analytics and Data Science: Chapter 9 Data Preparation*.

Lu, J. (2002). Predicting customer churn in the telecommunications industry––An application of survival analysis modeling using SAS. *SAS User Group International (SUGI27) Online Proceedings*, 114-27.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. Communications of the ACM, 39(11), 27-34.

Rud, O. (2001). *Data mining cookbook*. New York: Wiley.

Weinberg, S. and Abramowitz, S. (2002). *Data analysis for the behavioral sciences using SPSS*. Cambridge, UK: Cambridge University Press.