# 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 then to start a loyalty initiative such as offering a discount or a new mobile phone to the customer. To gain valuable knowledge out of the data telecommunication providers have about their customers, 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 as explained above. This set will be used to build our model. Our goal is then 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 (for example for data cleaning and reduction). We will start with data cleaning such as missing value and outlier handling. Next within *transformation* we will focus on data reduction (through correlation analysis and feature selection) as well as encoding. Within the next step of *data mining* we will apply data mining 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 (J48) and Support Vector Machines. At the end of chapter 3 we will interpret the results and evaluate 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. During the whole procedure we will always keep in mind that the KDD process is not a strict sequence, but rather an iterative process.

# Pre-processing Exploratory data analysis

Having received the real world data for the task, our team has been faced to the problem of the data being huge in size and having high number of variables. It therefore appeared of to be impossible to analyse the existence of possible amoralities in the data manually (inconsistencies, incompleteness, noisiness). In the pre-processing phase of the KDD Process our team has therefore applied several techniques of the Explorative Data Analysis in order to gain more detailed insights into the given dataset, detect the outliers and amoralities, uncover the underling data structure and to understand the given variables.

## Explorative Data Analysis

We have applied the following techniques before the firs iteration:

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

### Histogram of numerical variables

We plotted and analysed all 138 numerical variables (!only numerical) of our dataset. Figure … shows an example of the 5 variables’ histograms.



Figure 1

Viewing all the 138 plots we could discover that our variables were mostly (not in all cases) distributed normally or following some kind of exponential density function. It became evident that many numerical variables had a huge amount of observations valued with “0”, which appeared to be a standard value in many cases. Thus for example the variables “age1” (blue) and “age2“ (red) – the age of the first and second adult in the household I are plotted in the Figure 2.



Figure 2

It is evident that in both variables many observations are valued with “0”, which is not a valid value for these variables. “=” appeared to be a standard value for these variables. A standard values, “0” would falsify the results in the Data Mining phase, and will have to be treated as missing values in the transformation phase of the KDD process in the first and second iterations. Figure 3 shows the plot of the variables “age1” and “age2” without the missing “0” values.

From the plot is evident that most of the first household members are between 40 and 45 years old. Furthermore, the plot does not exhibit any noticeable problems concerning the outliers or missing values.

We took a look on the histograms of several variables in order to find the possible outliers. Such, the Figure … 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).

As can be seen from the left histogram of the Figure 4 due to the he amount of observations, the outliers cannot be seen in the histogram when showing the whole data. In contrary, while only plotting the data starting from a higher value (e.g. the median or third quantile) makes it possible to visually identify the existence of the outliers. For the better assessment of the existence of the outliers, we decided to use boxplots.



Figure 3



Figure 4

### 2. Box-plotting the data

Box plots are commonly used for the visual outlier identification. For all numerical variables a boxplot was created. Figure 5 shows an example of a boxplot 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. Depending on the range value, and thus on the height of the upper antenna, different number of values can be considered as outliers. The three single values in the range 15000 – 30000 can be definitely considered as outliers.

The outlier handling will be described in more detail in the following chapter in the individual iterations. The boxplots have given a good impression about the existence of the outliers and the distribution of the values of the variables. 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

### 3. Scatterplots

In order to create an impression about the dependency structure of several interrelated variables several scatter plots have been created. Several scatter plot matrixes have been created, 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.

### 4. 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 estimating the median of a variable and evaluating the range between the maximal and the minimal value of a variable can be understood if a variable has outliers. Using the “summary” function of R we have calculated the following characteristic values of every numeric variable: mean, median, 1st and 3rd quartiles, as well as maximum and minimum values.

Thereafter, we have divided the variables among the group members and 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.

In general, the numerical summary appeared to be a good identifier for the detection of variables with outliers. Thus, for example, variables who’s mean/average deviation appeared to be big, can be in most cases successfully further investigated on subject of outlier detection.



Figure 6

### 5. Correlation

In order to visualize the correlation between the variables oft he dataset, the correlation plot was used that 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

**1st Iteration Pre-processing: Data Cleaning:**

**N/A VALUES FREDDY**

**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 show 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 the simplification we have handled the outliers detected by the named methods for every numerical variable.

## Boxplot method

The distance between the first and the third quartiles (interquartile distance -IQR) of a boxplot is measure of variation of the variable around the median. Whiskers of a boxplot are commonly calculated using the factor of 1,5 or 3 and multiplying it with the IQR. It is a common praxis to consider the values that lie outside the boarders of the whiskers as outliers.

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 1 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 1: BoxPlots oft he variable „rev\_Mean“ with range of 3 before(left) and after(right) the outlier handling

## Z-score method

After having discussed the z-score in the “Business Analytics and Data Science” lecture we have decided to implement this method for the outlier detection in the first iteration. 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 a standard deviation.

We used the following 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 that 3 (of lower than -3) was considered to be an outlier. Every outlier was than set to the value the population’s mean ( added the three standard deviations (3\* (cf. Lessmann, 2015, Weinberg/Abramowitz, 2002 pp. 105f).

In its core idea and implementation, the z-score method is very similar to the simple boxplot method described above – the outliers are set to the value of the whiskers, whose length is equal to three times standard deviation of the population, neither the value of the mean nor (in most cases) the values of the quartiles are influenced. The result of the z-score outlier handling is presented exemplary on the variable “rev\_Mean” in the Figure 2.



Figure 2: Boxplots oft he 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 in order to reduce the number of variables.

Using the “caret” package we have first calculated the correlation matrix (function “cor”). The correlation matrix is a square matrix with the correlation factors for every combination of two variables. The visualization of the correlation matrix (corplot) is presented in the Figure … of the section exploratory data analysis.

The „findCorrelation(correlationMatrix, cutoff=0.8)“ function of the package “caret” searches through the correlation matrix and returns the vector of column indexes that have to be removed to reduce the pair-wise correlation. The “cutoff” parameter specifies the maximal correlation coefficient, above which the variables should be deleted.

Applying the “findCorrelations” function with “cutoff “ parameter of 0.99, ten variables could be identifies as almost perfectly correlated: "attempt\_Mean", "complete\_Mean", "complete\_Range", "attempt\_Range", "totmou", "ovrrev\_Mean", "totcalls" ,"ovrrev\_Range", "totrev", "Customer\_ID". The variable "Customer\_ID" can not be deleted. According to Rud (2001) the correlation limits of .7 and above are frequently mentioned by the benchmarks. In the first iteration we have removed from our dataset the variables that correlated to any other variables by the correlation factor of higher then .75.

# Feature Selection first iteration

The data set consists of 172 input variables and probably not all of them are relevant for the prediction. One part of vertical data reduction is the feature or variable selection where we want to find a subset of relevant variables. Out of a variety of feature selection approaches we decide within the first iteration to use a simple Filter Approach. The approach is to pre-screen the variables prior to model building and only use those variables as predictors that pass a certain statistical criterion. In our case we use the rfe function out of the caret package in R (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 therefore should be considered in the prediction model. There are several other approaches for feature selection that are more effective, for instance the Wrapper Approach or using models with built-in feature selection. The latter have the advantage that feature selection is included in the objective function that is optimized which is not given when feature selection is separated from all the other steps (like in the Filter Approach). Furthermore a disadvantage of the Filter Approach is that it is used in our case in a univariate manner, so that redundancy and interactions of variables are not taken into account. Nevertheless considering that we have high-dimensional data we decide in the first iteration for this approach 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 in the following.

Additionally it makes sense to check if the selected variables had many missing values in the original data set before imputation. It would not be reasonable to include variables that have a very high missing value rate because the information content would be low. Within our selected variable subset only two variables have more than 3% missing values, namely the variables “lor” (length of residence; 30.38% missing values) and “adults” (number of adults in household; 23.84% missing values). These two variables do not seem to be very relevant for the churn problem. But as we do not want to lose information and the variables still have values for more than two third of the observations, we keep them as predictors for now.

**Data Encoding (Scaling of continuous variables, coding of categorical variables)**

**Scaling of continuous variables**

The data encoding during the first iteration concerns just about the scaling of our continuous data. For scaling of our continuous data we considered two methods. On the one hand the Min/Max method and on the other hand the z-Transformation. Z-Transformation (standardization) is used to scale our data. Due to the Min/Max method is more sensitive for outliers. The method is more sensitive because the result is bounded.

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. Due to 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 was to code 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 was to code a function which scales just the continuous variables of the data by standardization. We used therefore a for-loop which applies the algorithm out of step one as well as the algorithm out of step two. The first algorithm detects the i-th continuous variable of the table. The algorithm out of step two standardizes the data of the i-th continuous variable. After it finished the loop the function returns the standardized data table.

# 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.

These models differ in the way they behave when having outliers within the input data. Some of them are robust against outlier (Random Forest, Decision Trees, Naïve Bayes and K-nearest Neighbours) and the 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 REF.

We decided to use the Split-Simple approach in the first iteration. That means the available labeled data is split into two subsamples: a trainingset (70%) for the 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 REF) 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 REF). 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 |

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 [s. Table REF] 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” (s. <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 REF 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 iteragtion. Therefore the results of the model selection process of those models are shown in figures REF and REF (Logistic Regression has no meta-parameters).



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



Figure 2: Model Selection Random Forest (on small subset)



Figure 3: Model Selection SVM (on small subset)

**Evaluation**

An evaluation of the first iteration requires that we look at the 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 (Nadaraya-Watson estimator) and a data set without outlier 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 the accuracy. The best three models of the first iteration have classification accuracy between 60 and 62 per cent.

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, an advance imputation procedures for missing values, tune the meta parameters and to take the 50.000 samples for the final model.

The following figures **eight** and **nine** showing the accuracy depending on meta parameters. 

Figure



Figure nine:

Based on comparing figure nine and eight with the figure **FREDDY CAPITEL ,** 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?”

**2nd Iterartion**

# Feature Selection second iteration

During the first iteration we noticed that the filter approach only selected numerical variables although we assume that also some categorical 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 is the output that shall be optimized. For our problem we decide to use a Random Forest Variable Importance (RFVI) method which is included in the caret package. We expect it to have higher computational costs as a random forest has to be trained prior to importance ranking, but probably it will perform better than the filter approach. The RFVI consists of three steps:

* Step 1: For each tree in the random forest the classes are predicted for the out-of-the-bag cases and the number of correct class votes is counted.
* Step 2: Each variable is randomly manipulated (continuous as well as categorical variables).
* Step 3: For each tree and each manipulated variable step 1 is repeated.

The assumption is that manipulating an important variable should change the accuracy whereas an unimportant variable should not have any effect. The figure from step 3 is now subtracted from the figure of step 1 and then averaged over all trees to get the raw importance score (RIS). The RIS gives the average loss in accuracy due to manipulating the variables’ values and is therefore the indicator for the variable importance ranking. Figure XX shows the result from our RFVI and we can see for instance that XXX is regarded as the most important variable. Furthermore we notice that many categorical variables are regarded as highly important. This confirms our decision to not rely on the filter approach. The categorical variables are now recoded into binary ones which we can see for example by looking at variable XXX. This is the binary variable for the original categorical variable XXX with level XXX. This leads to a large increase in dimensionality as we have XXX variables now in total.

The variable importance ranking is only the first step of feature selection. Now we have to decide how many variables should be selected. As there is no general rule of how many variables should be included (such as a threshold of the RIS), we firstly compare the importance ranking with our results from the wrapper approach.

…

Moreover we decide to include a categorical value if only one level of the variable is part of our selected variables (EXAMPLE).

Having done that we get XX selected features, out of which XX% are continuous and XX% are categorical variables. Again we checked the missing value rate of the selected features and …

## Principal Component Analysiss

In the exploratory data analysis we have seen that there are several highly correlated variables. In the second iteration in order to reduce the dimensionality of the given dataset we are using the technic of principal component analysis (PCA). PCA’s central idea is to reduce the number of variables of a dataset while retaining as much variation (and by that also the information content) as possible. The PCA is applied as the last data pre-processing 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 of a dataset. In the PCA a correlation matrix of the previously normalized (and standardized) dataset is created. The eigenvectors of the (symmetric) correlation matrix are the uncorrelated (orthogonal) principal components of the dataset. To find the principle components of the numeric variables of the dataset the r function “princomp(x, ...)“ has been used. The sum of the eigenvalues represent the total variance I the dataset, the corresponding eigenvalues of the eigenvectors represent the proportion of variance explained by each eigenvector of the correlation matrix (principle component). The plot in the Figure 1 pictures the amount of variance explained by the first 10 components.



Figure 1: Variance explained by the first 10 ecomponents

For the further analysis only the components which have the squared standard deviation (eigenvalues) higher then one will be retained, meaning that this components explain at least the same amount of variance as the original variables. By applying the PCA, the 87 numeric variables (“Customer\_ID” excluded) could be reduced down to 16.

Data Mining

Evaluation

**Conclusion**

Sources

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

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].

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

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

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.

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