# 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 Neighbors, 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 Neighbors) 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 Neighbors | 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 iteration. 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 Neighbors (KNN), Support Vector Machine (SVM) and Decision Tree (J48)



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



Figure : Model Selection SVM (on small subset)