Customer Churn Prediction using Quotation Data

Master Thesis

Submitted on: December 6, 2021

at the University of Cologne

Name:	Abdurahman Maarouf
Adress:	Schulstrasse 31
Postcode, Area:	53332, Bornheim
Country:	Germany
Matriculation number:	736481
Supervisor:	Prof. Dr. Dominik Wied

Contents

1 Introduction

Predicting customer churn in order to retain customers has become one of the most important issues for companies. The goal is to estimate probabilities for a costumer churning in the next period of time, in order to be able to detect potential churners before they leave the company. To tackle this issue, more and more advanced Machine-Learning-Algorithms are used guaranteeing high accuracy in their out-of-sample predictions.
Fortunately for most of the companies, churn rates from one period to another are very small. However in classification models predicting a rare event can become challenging. In this so called "Imballanced Classes" issue certain arrangements to the underlying training data need be made. Without these arrangements and with highly imballanced classes, a poor algorithm will simply never predict the outcome of the minority class. In a dataset with 1000 customers containing 5 churners for example, this loss-minimizing algorithm would have an in-sample accuracy of 99.5%.
In order to avoid the high amount of "False-Negative" classifications there are many methods ranging from upsampling the minority class or downsampling the majority class to more advanced techniques. In this work we will present and compare the different methods while applying them to the underlying problem.
We also want to emphasize (or not) the importance of using quotation data for predicting customer churn. A company can track (potential) customer behavior on their distribution channels. Nowadays, in most cases the products or services are offered online on websites, which makes it easy to track website visitor data. In the context of dealing with customer churn this data can be matched to the customers already having a product or contract of this company. We believe (?) that the number of visits of a current customer in the last period (?) plays a big role in predicting the probability of that customer leaving in the next period. (Coming from high correlation between Nvisits and churn)
In order to evaluate the importance of not only the number of website visits but also the other explanatory variables there is typically a trade-off during model selection. The trade-off is between the model complexity or corresponding accuracy and the model interpretability. Deep neural networks or boosted trees belong to the complex models which are famous for their high accuracy in the fields of computer vision and natural language processing. Understanding and interpreting the model is of no big interest in these areas. However in the topic of this work and in many other areas understanding which variables lead to the resulting outcome of the model becomes desirable. The most transparent models in terms of interpretability are linear or logistic models. There the magnitude and sign of the corresponding coefficients (after being testet for significance) illustrate the changes of the outcome for a change in the specific explanatory variable. These models however lack in terms of accuracy when being compared to the comlex ones. In this work we will present the accuracy and interpretability of "Explainable Boosting Machines" developped by (?) for predicting customer churn. It aims to combine the high accuracy of complex models on the one hand and the interpretability of linear models on the other hand.

2 Data and Methodology

2.1 Understanding the Problem

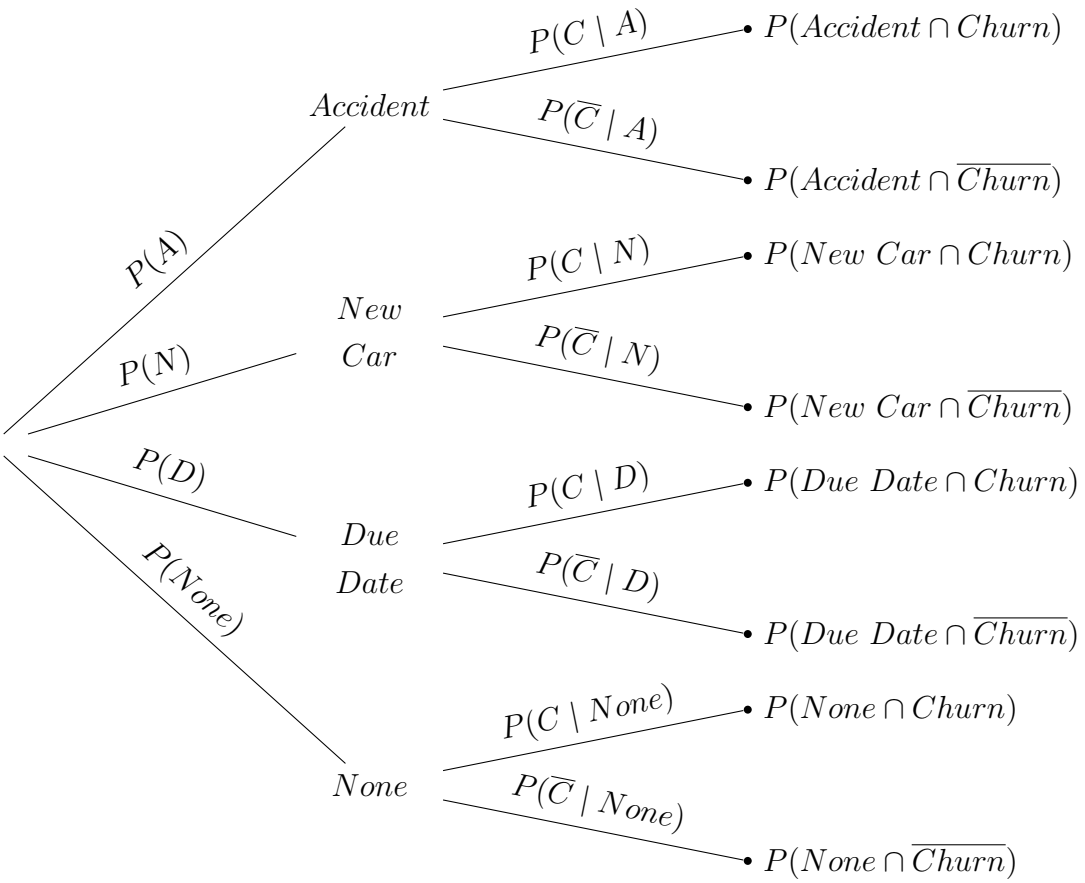
For this work we use customer data from a big insurance company in Germany. Due to data pretection the data is anonymized which does not affect the model accuracy and interpretability in any form. We focus on the product of automobile liability insurance, which is by law a mandatory service every car owner must hold in Germany.
Typically car owners close a deal with an insurance company which can be terminated by the end of each year. In rare cases both sides agree on a contract with a due date during the year. If the contract does not get terminated it is automatically extended for another year. Besides the option to terminate the contract at the due date there is also an option to terminate it earlier in a few special cases. These cases mainly involve car accidents and vehicle changes of the contractor. To sum up, here are the three cases in which a churn can (but not must) occur during a year:

Event A:	Contractor is involved in an Accident.
Event N:	Contractor buys a new Car.
Event D:	Due date during the year.

The problem of modelling customer churn needs to be seperated into the probability of a costumer leaving during the year and at the end of a year (why noch ausführen). In this work we will focus on predicting churns occuring during the year. The purpose is to build a model which can be used at any time t of the year besides on January the 1st to predict the probability of a costumer leaving the company in the next period of time $(t, t + s]$.
It can be argued that in order to provide a model with maximized utility for production one would want to keep s small. For example a company would highly benifit from a model, which can predict the churn-probability of tomorrow or the next week. However we will see that having a small s will decrease the accuracy of our models (drastically?), creating a trade-off situation between model accuracy and the benifits of a small s. With a smaller period the classes of the data become more imballanced, creating a higher challange of preprocessing the training data and feeding the algorithm enough information on potential churners. Furthermore, a small s decreases the scope of action for a company to retain potential customers leaving.

Figure 1 (richtiger Verweis) illustrates how the probability of a churn during the year can be decomposed using the Events A (Accident), N (New Car), D (Due date during the

year) and C (Churn).



By assumption we set $P(C \mid None) = 0$ as the amount of terminated contracts during the year which are not being caused by a new car, an accident or a due date is very small and can be omitted. Therefore we leave these cases out of our data (?). Also, the probability $P(D)$ can only take values 0 and 1, as either the due date of a customer lies in the next period of time $(t, t+s]$ or not. What we are interested in predicting is the overall probability of a churn, which can be rewritten as:

$$\begin{aligned} P(C) &= P(A \cap C) + P(N \cap C) + P(D \cap C) \\ &= P(A)P(C \mid A) + P(N)P(C \mid N) + P(D)P(C \mid D) \end{aligned}$$

One idea would be to model the three branches of A, N and D seperately. The logic behind this is that different models and sets of explanatory variables have the best fit for the probabilities of the three branches. Not only the the three branches but also the unconditional and conditional probabilities of a single branch may vary in their best modelling approach. For prediciting an accident a model of type (XY) is more suitable, whereas (YZ) would go better with modelling the probability of churn given an accident occured. In the course of this work we will begin with a general modelling approach trying to predict the probability $P(C)$. At a later stage we will compare the accuracy outcomes of seperately predicting the branch probabilities with the baseline approach.

2.2 Data

Most customer churn predicition models have been using only one timestamp to train a model. We want to emphasize the usage of multiple timestamps and show that it significantly improves prediction accuracy. The improvement is a result of an increased training set size and the ability of the model to especially learn more about the rare class. Furthermore, the model becomes more generalizable in time which is crucial if this model is applied in production. (see: How training on multiple time slices improves performance in churn prediction). Explain more. Use Graphics like in the paper.
Part of this work will also be to evaluate if the churn probability of customers is time invariant. Therefore we statistically test the equality of monthly and yearly (and weekly?) mean churn rates using the ANOVA (or other? seasonality tests). We will see that it is time variant (or not?) which underlines the importance of including (or not) different timestamps containing different years/months(/weeks) in the data for the model to learn the time differences (or not). (These will simply be appended to the data as additional rows creating a panel dataset.)
Therefore to build the models we use historical data of the insurance company. More specifically we pick N (one or many?) timestamp(s) t_i with $i = 1, ..., N$ in the past and collect all the active contracts to that(these) timestamp(s). One row corresponds to one active contract at t_i. So if one hypothetical contract is active in all N periods it will appear as N seperate rows in our data. To each row we merge the status of that contract in $t_i + s$. Furthermore, we join the number of requests corresponding to that contract in the period $[t_i - m, t_i]$. The period length m will be another parameter to tune.
Describe the ETL Process?

-Statt "STORNO" im data-load sql query zu definieren, definieren wir es einfach in dem Feature-Engineering Teil?
-Test if customer churn probability is time invariant or time variant!!! Is it enough to argue with the Aggregate Churn rate? Show statistically (Voll gute Idee)
-How do I exclude 1.1. churners in my data? Exclude the entire contract or simply say y=0 as he didnt churn during the year?
-Safe noch als Variable einbauen, was sein Preis bei einer jetzigen Berechnung wäre (und dann Differenz zwischen Preis, den er zahlt und Preis den er angezeigt bekommt!)

3 Modelling Approach

3.1 Preprocessing

3.2 Handling Class Imbalance

Studying the rarity of an event in the context of machine learning has become an important challange in the recent two decades. Rare events, such as a customer churning in the next period, are much harder to identify and learn for most of the models. HaiYing Wang et al (Logistic Regression for Massive Data with Rare Events) study the convergence rate and distributaional properties of a Maximum-Likelihood estimator for the parameters of a logistic regression while prediciting rare events. Their finding is that the convergence rate of the MLE is equal to the inverse of the number of examples in the minority class rather then the overall size of the training data set. So the accuracy of the estimates for the parameters is limited to the available information on the minority class, even if the size of the dataset is massive.
Therefore some methods have been developped to decrease the problematic of imballanced classes. In this chapter we will present (three?) different methods which can be applied to the training set, before feeding it to the model. To handle and evaluate the outcomes of prediciting rare events also the appropriate models and model evaluation metrics must be chosen. This will be discussed in the next two chapters.

(i) Downsampling

The first basic sampling method is named downsampling. It randomly eliminates examples from the majority class in order to artifficially decrease the imballance between the two classes. The downside of this approach is that it possibly eliminates useful examples for the model to maintain a high accuracy in predicting the majority class (Mining with Rarity: A Unifying Framework). HaiYing Wang et al (Logistic Regression for Massive Data with Rare Events) also study the convergence rate and distributaional properties when applying downsampling. According to their findings the asymptotic distribution of the resulting parameters may be identical to the MLE's using the full data set. Under this condition there is no loss in terms of efficiency (minimum possible variance of an unbiased estimator devided by its actual variance).

(ii) Upsampling

The second basic sampling method is the upsampling approach. This method simply duplicates examples from the minority class until the classes are more balanced. While duplicating examples though, the chances of overfitting to these duplicates becomes a more probable threat. Also, no new data is being generated in order to let the model

learn more valuable characteristics about the minority class (Mining with Rarity: A Unifying Framework). Additionally, the computational performance of this approach can get rather poor, espacially with large datasets and highly imballanced classes. While evaluating the asymptotics of the MLE's with upsampling, HaiYing Wang et al find out that it also decreases the efficiency. A probable higher asymptotic variance of the estimators is the reason for that.

(iii) SMOTE

The more advanced SMOTE-approach (Synthetic Minority Oversampling Technique) (SMOTE: Synthetic Minority Over-sampling Technique) also creates more artifficial examples of the minority class. Instead of simply duplicating some rows SMOTE creates new nearest neighbors in terms of feature values for the minority class examples. While constructing the feature values of the new example $(n + 1)$ (where n is the size of the unsampled dataset) as a new nearest neighbor for example i of the minority class one has to differentiate between continuous and nominal features. The k-nearest neighbors for the minority class are typically constructed with the Euclidean Distance for continuous features and the Value Distance Metric for nominal features (maybe include other distance measures here? Mahalanobis Distance?).
A. Continuous features:
A.1) Construct difference between corresponding feature value of example i and one of its k nearest neighbors.
A.2) Multiply this difference with a random value drawn from a uniform distribution between 0 and 1.
A.3) Construct the feature value of the new example by adding the multiplied difference to the feature value of example i.
B. Nominal features:
B.1) Choose the feature value which is the majority vote between the feature value i and its k nearest neighbors.
B.2) Assign this value to the corresponding feature of the new example.
With this approach it is ensured that the model learns more about the neighborhood regions of the minority class. It decreases the probability, that the model overfits to the duplicates created in upsampling.

(iv) Cost-Sensitive Classifiers

One drawback of the presented sampling methods is that the sample distributions are being changed. A different starting point is to to change the objective cost function which is to be minimized during the model estimation. Thereby the sample size and the distribution stay the same. In the modified cost function we want to penalize false-negative

classifications with a higher weight. Reason behind this is to ensure that the event of interest, represented by the minority class, is predicted correctly.
(How does our cost-function look like then? Gattermann use balanced class weights which are inversely proportional to the class frequency and thereby assign more weight to samples from the minority class.)
The challange of this approach is to find the appropriate penalty weight. It is hard to measure the cost of misclassifying an insurance-customer regarding churn-probabilities. The fact that these costs can come from multiple sources which are not easily definable is one part of the reasoning.

3.3 Machine Learning Models

3.4 Model Evaluation Metrics

Bibliography