Machine Learning Methods to Perform Pricing Optimization. A Comparison with Standard GLMs

Giorgio Alfredo Spedicato, Christophe Dutang, Leonardo Petrini

Machine Learning Methods to Perform Pricing Optimization. A Comparison with Standard GLMs

by Giorgio Alfredo Spedicato, Christophe Dutang, and Leonardo Petrini

1 Abstract

As the level of competition increases, pricing optimization is gaining a central role in most mature insurance markets, forcing insurers to optimise their rating and consider customer behaviour; the modeling scene for the latter is one currently dominated by frameworks based on Generalised Linear Models (GLMs). In this paper, we explore the applicability of novel machine learning techniques such as tree boosted models to optimise the proposed premium on prospective policyholders. Given their predictive gain over GLMs, we carefully analyse both the advantages and disadvatanges induced by their use.

Keywords: Pricing Optimization, Conversion, Machine Learning, Customer Behaviour, Boosted Trees.

2 Introduction

Policyholder retention and conversion has received increasing attention within the actuarial practice in the last two decades. In particular, the widespread diffusion of web aggregators has eased the comparison of different insurers' quotes for customers. Therefore, it is now popular for an insurance company to model not only the cost of the coverage offered but also the insurance demand. Indeed, the likelihood of a prospective customer accepting a given quotation and the probability of retaining a current customer are key drivers of maintaining and enhancing the profitability of an insurer's portfolio. Such probabilities depend not only on the classical ratemaking variables used to determine expected loss costs, but also on competitive market variables (e.g. distance between insurer's quote and best/average market price), customer behavior and demographics. Actuarial ratemaking, current policyholder retention modeling, and prospective policyholder conversion probabilities modeling lead to the so-called Pricing Optimization (PO). Specifically, this paper aims to investigate how machine learning methodologies can improve policyholder retention and conversion estimation over that of classical GLMs.

Few academic papers have used or compared predictive models different from logistic regression, as far as the authors know. On the other hand, Telecommunication firm customer retention has been a classical topic of business analytics for at least a decade; see for example (Hung, Yen, and Wang 2006). More precisely (Milhaud, Loisel, and Maume-Deschamps 2011) focused on the life-insurance context by using Random Forests, and (Fu and Wang 2014) applied survival analysis techniques to model policyholders' time to cancellation in a P&C portfolio. Further, (Guelman, Guillen, and Perez-Marin 2012) used Random Forests to model lapse probabilities, whilst (Yeo et al. 2001) used Neural Networks to model retention considering premium variations; (Guillen and Guelman 2014) proposed a causal approach to model price elasticity . Finally, a survey of classical regression models applied to policyholder behaviour can be found in (Dutang 2012).

From a machine learning perspective, the estimation of retention and conversion represents a supervised classification problem, traditionally solved in the actuarial practice with a logistic regression. A key advantage offered by logistic regression is the easy interpretability of fitted parameters combined with a reasonable computational speed. Nevertheless, machine learning techniques such as Regression and Classification Trees, Random Forests, Gradient Boosted Machines, and Deep Learners (Kuhn and Johnson 2013) have recently acquired increasing popularity in many business applications.

The interest of actuarial practitioners in machine learning models has grown in recent years, e.g. (Frees, Derrig, and Meyers 2014; Frees, Meyers, and Derrig 2016). (Pozzolo 2011) also used various machine learning

algorithms to predict claim frequency in the Kaggle Allstate competition. In addition, (Guelman 2012) showed the benefits of applying Gradient Boosting Methodologies instead of classical Poisson GLMs for predicting claim frequency. Whilst machine learning techniques appear to outperform the application of classical logistic regression in many applications, two issues hamper their widespread adoption in actuarial science. First, their parameters are often relatively more difficult to interpret (the "black box" issue). Second, the computational time required can be overwhelming compared to the time required to fit a GLM. To the authors' knowledge, a systematic review of machine learning techniques comparing predictive performance gain on logistic regression, interpretability, and computational time to model policyholders' retention and conversion is still lacking in actuarial literature, and shall hence be presented here.

The rest of the paper is organised as follows: Section 3 provides a brief overview of business considerations. In Section 4, we review predictive models and methodologies for binary classification problems. In Section 5, the presentation of a dataset is followed by the estimation and the comparison of models previously presented, along with an example of price optimization. Finally, Section 6 concludes the paper.

In order to achieve these tasks, a real dataset coming from a direct insurer will be used in our study to model conversion. More precisely, the database used is from two recent months of personal motor liability cover quotations. Distinct sets of data will be used for the model fitting, performance assessment and pricing optimization steps mentioned above. We underline that the methodologies used here to model conversions can be transposed to retention modeling without any difficulty. To allow easy replicability of the analysis, open source software has been used, such as the R environment (R Core Team 2017), and the H2O data mining software (H2O.ai team 2017).

3 Business context overview

The Casualty Actuarial Society (CAS) defines PO as "the supplementation of traditional actuarial loss cost models to include quantitative customer demand models for use in determining customer prices. The end result is a set of proposed adjustments to the cost models by customer segment for actuarial risk classes", see (Ratemaking 2014).

The PO approach includes considerations of both customer behavior and the market environment, making it depart slightly from traditional loss cost-based rate making. Although the methodology is innovative, concerns are being raised by Consumers Advocates, and there is some initial scrutiny from Regulators. For instance (National Association Insurance Commissioners 2015; Baribeau 2015) question to what extent the explicit inclusion of price elasticity in the process of setting rates makes insurance prices unfair. PO has been extensively treated by actuarial practitioners in numerous forms; see for example (Duncan and McPhail 2013; Serhat and McPhail 2013; Serhat 2013), and to a lesser extent by academics within insurance science. Furthermore, PO should be carried out by distribution channel as customers do not behave in the same way on the web or in front of a tied-agent, see e.g. (Rulliere, Loisel, and Mouminoux 2017).

PO can help increase the profitability of current and prospective business by taking into account both the policyholders' loss propension and the business environment in which the insurer operates. In fact, in almost every country policyholders can compare quotes being offered by multiple competing insurance carriers, making it crucial for the insurer to maximize the gain associated with current and potential policyholders. As a consequence, PO should not only model the prospective cost associated with the coverage provided, but also consider the likelihood of preventing a customer from accepting deals coming from the competition. More specifically, a conversion analysis should take into account factors such as individual features (including demographics, characteristics of the insured risk); the proposed premium (in particular their monetary variation); the relative rank of the premium with respect to what is currently offered on the market. A similar analysis can be performed in order to estimate the probability of keeping current customers, the retention analysis, by taking into account similar individual features; the history of premiums and claims.

In practice, performing PO requires four elements: a risk premium model, in order to obtain the expected burning cost; a competitive market analysis (CMA) to model the competitors' premiums given the characteristics of a policyholder; a customer price elasticity model to predict the volume of new business and renewals

reflecting market competition in business analysis; optimization models to integrate all the aforementioned models and predict the profit volume given a change in prices, and to identify the best price changes for a given financial objective. (Santoni and Gomez Alvado 2007) and (Manin and Bayley 2010) provide a general overview of PO from an insurance perspective.

A review of recent practictioners' presentations has drawn a few key points to attention. The personal motor business is one of the markets in which such tecniques have been applied the most, facilitated by policy portfolio sizes and the large volume of data collected. For instance, (Serhat and McPhail 2013; Serhat 2013) model retention and conversion in US markets using non-linear GLMs. Another example of PO based on customer value metrics for direct business can be found in (Bou Nader and Pierron 2014).
(Duncan and McPhail 2013) present four different approaches that can be used to perform pricing optimization:

1. *individual policy optimization*: the final price proposed to the policyholder is re-calculated at an individual level.
2. *individual policy optimization re-expressed in ratebook form*: individually fitted prices are modeled as target variables within a standard predictive model (e.g. GLM). A traditional ratebook structure is therefore obtained.
3. *direct ratebook optimization*: very similar to the above method.
4. *real time optimization*: this method stresses the importance of continuously "refreshing" the consumer behaviour and loss models with real - time updated data.

Although individual policy optimization provides the best performance by revenue maximization, it is worth noting that regulation or operational constraints could lead one to choose less refined approaches.

The current paper focuses its attention on applying predictive modeling to perform conversion modeling as an alternative to standard GLMs. To illustrate, a conversion model targets the dicotomic variable "Convert", which can take two values: Convert (Yes), Reject (No). A logistic regression within the generalized linear model family has been traditionally used to address such analysis (Anderson et al. 2007), and it is currently easily applied by taking advantage of actuarial pricing software (e.g. Emblem, Pretium, Earnix, ...).

4 Predictive modeling for binary classification

4.1 Modeling steps

In this section, a brief overview of predictive models based on the books (Kuhn and Johnson 2013), (Breiman 2001) and (Bett 2014) is presented. Predictive modeling involves the application of various mathematical techniques to a dataset composed of a response variable and a set of predictors. This process aims to find the best model in terms of predictive performance, where the performance needs to be measured differently from the methods suggested by classical statistics. Specifically, while in classical statistics a model is defined in order to better explain a phenomenon, in predictive modelling a strong emphasis is set on how well a prediction can be done on unseen data. Moreover, the importance of assessing predictive performance on a subsample of data different from the one used to calibrate the model is always emphasized.

Regarding the model building process, (Kuhn and Johnson 2013) list the following steps:

1. Data pre-processing: this task consists of cleaning the data, possibly transforming predictors (feature engineering) and selecting those that will be used in the modeling stage (feature selection).
2. Data splitting: the dataset is divided into a training, a validation, and a test set, thereby reducing the "overfitting" that occurs when a model appears to be extremely performant on the same data used for finding the underlying structure (e.g. training set), whilst showing significantly less performance on unseen data.
3. Fitting the selected models on the training set: most families of models need one or more tuning parameters to be set in advance to uniquely define a model; these parameters cannot be derived analytically and their class is also known as "hyper-parameters". A grid search (or an optimized variant) can be employed to find the optimal combination of parameters with respect to a specific performance

metric. For binary classification, performance metrics include the Area Under the Curve (AUC), the Gini index, the logarithmic loss, and the Kappa statistic.
4. Model selection: an assessment of which model among the ones tested performs best on a test set, making the results generalizable to unused data.

4.2 Data pre-processing

Data pre-processing techniques generally refer to the addition, deletion, and transformation of the data. This part of the process is crucial for determining the success or failure of the entire analysis, since most Machine Learning techniques are sensitive to the format and scale of the predictors.

Firstly, several modeling techniques require predictors to have a common scale of measure. Center scaling is the most commonly used transformation for achieving this objective, helping improve the stability of numerical calculations at the expense of reduced interpretability. In some cases it can also be useful for removing the skewness of the predictors, attained by taking advatange of methods such as the Box and Cox transformation (Box and Cox 1964).

Secondly, a proper analysis of outliers is required in many instances. Outliers are observations that appear exceptionally far from the rest of the data, and can impact the final performance of the model by introducing a global bias. Usually, a visual inspection of a variable's distribution is the first step for dealing with this issue, and once the suspect points have been identified, their values should be questioned with care in order to ensure that they indeed belong to the data generating process. With the exception of some predictive models that are naturally insensitive to outliers (e.g. tree based models, and Support Vector Machines), in all other instances outliers should be removed. In this regard, special techniques like the spatial sign transformation (Serneels, De Nolf, and Van Espen 2006) can help.

Thirdly, missing values, or observations with no value for some or all variables, should be treated appropriately. As for outliers, a careful exploration into potential structural reasons for such phenomena may be needed. The intuition is that missing data can be caused by a different process underlying data creation, and the simple removal of these data points may negatively affect overall performance. Nevertheless, whenever the proportion of missing values is too high to be ignored, methods such as imputation (e.g. the k-nearest neighbour model imputation or regression with auxiliary variables) can be used.

Increasing the number of variables is not always beneficial. Thus, an initial selection of predictors might be useful. For example, highly correlated predictors may be removed to help interpretability without loss of predictive performance. Many predictive models already contain intrinsic measures of variables' predictive importance, so they perform an implicit feature selection. Models without feature selection may be negatively affected by uninformative predictors. To avoid this, specific methodologies have been built in to perform an initial screening of predictors: "wrapper methods" and "filter methods". "Wrapper methods" conduct a search of the predictors to determine which, when entered into the model, produce the best result. "Filter methods" perform a bivariate assessment of the strength of the relationship between each predictor and the target.

Further, degenerate or "near-zero-variance" variables (predictors characterized by few distinct values whose frequencies are severely disproportionate) may create computational issues in some models. Principal Component Analysis (PCA) and Independent Component Analysis (ICA) transformations can be used to reduce the number of input variables, i.e. a smaller set of generated variables seeks to capture the majority of the information, leading to more parsimonious models. Such approaches also prevent multicollinearity, at the cost of less interpretable variables.

Finally, some predictors require recoding in order to be conveniently handled. For example, encoding nominal or categorical variables into multiple dummy variables is always a necessary step before fitting any model. Manual binning of continuous variables is a widely used approach to overcome marginal non-linearity between the outcome and any continuous variable. However, (Kuhn and Johnson 2013) identifies three drawbacks of this approach: loss of performance (since many predictive models are able to find complex non-linear

relationships between predictors and binning may reduce this feature); loss of precision; increase of the false positive rate.

4.3 Model Training, Tuning and Performance Assessment

Model training consists of fitting a model through an iterative update of variables and/or parameters. Through this, the modeler should be mindful of overfitting which can appear when a model is excessively complex. This is due to a modeling strategy that over-emphasizes patterns unique to the specific dataset on which the model has been fitted. Overfitted models have poor predictive performance. Thus, it is necessary to obtain a reliable way for estimating models' predictive performance.

Hence, subdividing the dataset between a training part, where models are fit and tuned, and a test part, used to estimate the models' performance is fundamental. As further detailed in (Kuhn and Johnson 2013), the use of resampling techniques can help obtain a less biased estimate of model performance. For instance, one approach commonly used is the k-fold cross validation, e.g. the training dataset is split into k roughly equally sized sub-samples during the estimation process. Once the k models are estimated, the out of fold observations are used as a validation set on which the performance metrics figures are computed. Consequently, the overall model fit is obtained by averaging the k cross validated performance fit estimates.

In addition, when estimating models within a given model family , it must be noted that the vast majority of current machine learning techniques identify models by specifying one or several hyper-parameters. As introduced in the previous section, the optimal values of hyper-parameters cannot be directly estimated from data, hence requiring a grid search to tune the final model. Comparing the performance metrics obtained on several models with different sets of hyper-parameters cannot be (generally) performed on the cartesian product of all possible combinations. As the computation time or dimensionality increases, a random search becomes more appealing. Recently, Bayesian Optimization has been gaining popularity as an alternative (Kuhn 2016). Specifically, the Bayesian approach includes a first cycle of random search to explore the space of the hyper-parameters, with a consequent second cycle of numerical optimisation based on a Gaussian process. The advantage of this approach is that every step relies neither on a random step, nor on a subjective discrete list of hyper-parameters, but upon a probabilistic model.

Since our work dedicates most of its efforts to the analysis of a binary response, a special note on how to assess the predictive performance of competing models in such environments is given. As a preliminary step, defining a threshold for the probabilities given as predictive ouputs by a model in order to determine whether an observation is to be considered as an "event" or "non-event" is needed. The default threshold is 1/2 as for a totally random classification. The resulting cross tabulation of actual and predicted events/non-events after the cut-off has been applied generates a Confusion Matrix (CM), which becomes the starting point for assessing binary classifier performance.

Table 1: Confusion Matrix notation

Predicted	Observed: Event	Observer: Non-Event
Event	True Positive (TP)	False Positive (FP)
Non-Event	False Negative (FN)	True Negative (FP)

The structure of a generic CM is given in Table 1. Let the total sample size be $N = TP + TN + FP + FN$. A first measure of classifier precision is the accuracy $O = \frac{TP+TN}{N}$. Nevertheless, alternative measures are generally more appropriate when the outcome distribution is severely unbalanced like in the Conversion dataset treated in this work. For example, the Kappa statistic, $K = \frac{O-E}{1-E}$ can be used, where E is the accuracy of the uninformative classifier (the relative frequency of the greatest class), and O is the assessed predictive model accuracy.

From the confusion matrix, two additional statistics for assessing binary classification problems can be derived: sensitivity and specificity. Sensitivity is the probability that a sample is correctly classified, given that it

is a true "event" sample: $Se = \frac{TP}{TP+FN}$. Conversely, specificity is the probability that a true non-event is correctly classified $Sp = \frac{TN}{FP+TN}$ and the specificity's complement to one is known as the false classification rate. For a given predictive precision, increasing the sensitivity (i.e. lowering the cut-off probability to identify new samples as events) lowers the specifity.

It is possible to graphically display the trade off between the two measures by the so-called ROC curve, which displays the relation between sensitivity (y-axis) and the false classification rate (x-axis). A model with no discriminatory power has a ROC curve along the 45-degree line in the unit square, whilst a better performing model exhibits a curve moving towards the top-left corner. The ROC curve allows one to obtain a synthetic measure of classification power for a given predictive model. The Area under the Curve (AUC) for the ROC curve is a measure bounded between 0.5 (uninformative of the 45-degree line) and 1 (perfect discriminatory capability of the heavyside curve). Finally, the Gini index is a commonly used linear transformation of the AUC: ($Gini = 2 * AUC - 1$).

In practice, AUC and Gini are certainly the most used metrics to solve binary classification problems. However, while they stress the discriminating power of a predictive model, they are not able to measure how correct the predicted probabilities are. Since we want to assess the prediction power, it is also worth exploring different metrics. The log-loss metric, $logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(p_i) + (1 - y_i)\log(1 - p_i))$ which targets the discrepancy between actual and estimated probabilities, and will be thus jointly used with AUC/Gini in our analysis.

In most cases, a graphical representation can also help interpret the results, both within and between models. In order to do so, calibration charts and lift charts are used. Calibration charts help in assessing the predictive performance of different models. Once observations are scored, estimated event probabilities are assigned to buckets of estimated probability ranges, i.e. by dividing the probability domain (0-1) into numerous buckets. Calibration charts display the midpoint of the probability bucket in the x-axis and the observed empirical event rate on the y-axis. Given a well performing model, the resulting points would lie on the midline. Lift charts are created by sorting estimated event probabilities in decreasing order. Then, on the x-axis the cumulative percentage of samples is plotted, while on the y-axis the cumulative percentage of true "events" is shown. Lift charts are specifically tailored to compare the discriminating performance of binary classifiers, and allow one to answer questions like: "what is the expected percentage of total events considering the top % observation scored by probability?". Being close to the 45-degree line indicates absence of discriminating advantage, while a perfect classifier would be represented by the hinge function.

Finally, one can decide to apply resampling techniques to mitigate instances in which the frequency of binary classes is not even. For example, "downsampling" reduces the frequency of the most represented class, whilst "upsampling" randomly draws additional samples from the minority class. Both downsampling and upsampling (and other derived methods) aim to even the frequency of classes as much as possible.

4.4 Common predictive models for binary classification

(Kuhn and Johnson 2013) organizes classification algorithms into three main categories. *Linear Classification Models* are based on a scoring function that can be expressed as a linear combination of predictors. In addition to the classical GLMs introduced by (McCullagh and Nelder 1989), the following algorithms should be mentioned: the Penalized Logistic Regression (Elastic Net) and the Linear Discriminant Analysis (LDA). The Elastic Net introduces two forms of penalties into the GLM formula, namely the Ridge and Lasso ones. They permit better handling of feature selection, overfitting and multi-collinearity, see e.g. (Zou and Hastie 2005). The LDA finds a linear combination of features characterizing or separating two or more classes of objects or events, see e.g. (Fisher 1940).

Non Linear Classification Models are composed of a heterogeneous group of techniques. Below we list some relevant elements. Neural Networks (and, in particular, Deep Learning (DL)): a DL model consists of multiple strata of "neurons" that collect inputs, transform a linear combination of such inputs into a non-linear transformation through the so-called activation functions, and return the output to the subsequent stratum. DLs have been successfully applied in a myriad of applications including image recognition and natural

language processing. Flexible Discriminant Analysis (FDA): FDA combines ideas from LDA and regression splines. The classification function of FDA is based on a scoring function that combines linear hinge functions, see e.g. (Hastie, Buja, and Tibshirani 1995). K-Nearest Neighbors (KNN): a KNN model classifies each new observation according to the most frequent class of its k-nearest neighbors according to a specified distance metric. KNN is one of the oldest and most important classifiers found in statistical literature, see e.g. (Fix and Jr 1951). Naive Bayes Classifier (NB): NB is based on the Bayes rule of Probability Calculus assuming independence among predictors, that is $Pr\left(C_k|x_1,|x_2,\ldots\right) \propto pr\left(C_k\right)\prod_{i=1}^{n} Pr\left(x_i|C_k\right)$. Despite the strong assumption, predictive performance is often high and computational resources are relatively low, see e.g. (Rish 2001). Support Vector Machines (SVM): a SVM performs classification tasks by creating hyperplanes defined by linear or non-linear functions, see e.g. (Cortes and Vapnik 1995).

Tree-Based Approaches consist of tree-like nested if-then statements for the predictors that partition the data. This approach generates a structure of "nodes" and terminal "leaves", within which a model is used to predict the outcome. The following tree based models will be explored. The C5.0 algorithm is one of the most significant representatives of a classical tree based approach for performing classification, see e.g. (Quinlan 2004). Random Forest blends tree-based logic with the bootstrap aggregation approach ("bagging") by creating a so-called "forest" of trees, rather than a single tree. Each of these trees is a weak learner built on a subset of rows and columns. The classification from each tree can be seen as a vote and the most votes determine the classification, see e.g. (Liaw and Wiener 2002). This endeavour successfully reduces variance in the final set of predictions. Gradient Boosted Machines applies the boosting concept on either regression or classification tree models. Similarly to bagging, the boosting approach combines the results of multiple models. The key difference is that each subsequent model is recursively applied on the results of the previous one. In particular, as the previous model misclassifies the sample more frequently, more weight starts being given to the subsequent model, see e.g. (Friedman 2001). A notable extension of classical GBM is the eXtreme Gradient Boosting (XGBoost), see e.g. (Chen and Guestrin 2016) which has been chosen as the preferred algorithm by winners of many Kaggle competitions.

Finally, ensembling models of different families often provides higher predictive accuracy than can be obtained by any of the original models. Such a technique can be based on a SuperLearner, a machine learning algorithm that finds the optimal combination of predictions generated by the original constituents, see e.g. (LeDell, Sapp, and van der Laan 2014). In our numerical experiments, we fitted all the models previously listed. In this paper, we only keep the most common and best predictive models, for which we give a brief outline below.

4.5 GLMs and their Elastic Net extension

The following section is based on (Hussami et al. 2015) to which the interested reader is directed for details. GLMs extend the standard linear model by relaxing the normality and the constant variance assumptions. The components of a GLM are a random component (from the exponential family and in our case the Bernoulli distribution), a systematic component (that is a linear combination of explanatory variables and the regression coefficients β) called the linear predictors and a link function between the mean of the response variable and the linear predictors. GLM are fit by maximizing log-likelihood, as (Anderson et al. 2007) shows, using iterative numerical methods. The variable selection task is performed by a chi - square based test, as in classical GLMs.

Nevertheless the extension of GLMs, the elastic net penalty approach, has achieved widespread use in machine learning for variable selection and regularization. Here the function to be optimized is $\max\left(LogLik - Pen\right)$, the penalty[1] being $\lambda * \left(\alpha * \|\beta\|_1 + (1-\alpha) * \frac{1}{2}\|\beta\|_2\right)$. In particular, $\lambda > 0$ controls the penalty strength while α represents the relative weight of the ridge and lasso components (Nykodym et al. 2016) within the elastic net penalty. The elastic net regularization reduces the variance in the predictions and makes the model more interpretable. In fact, imposing a penalty on coefficient size leads to a sparse solution (throwing off non - significant variables) and shrinks coefficients.

[1] $\|\beta\|_1 = \sum_{k=1}^{p} |\beta_k|$ and $\|\beta\|_2 = \sqrt{\sum_{k=1}^{p} \beta_k^2}$.

The α parameter controls the penalty weight between the l_1 (the lasso, least absolute shrinkage and selection operator) and l_2 (the ridge regression) penalties. If the λ tuning parameter is sufficiently large, it brings coefficient values toward zero, starting from the less relevant ones. As a consequence, lasso has proven to be a good selection tool in many empirical applications. The l_2 term is the ridge penalty component , which controls the coefficients' sum of squares. Is is easier and faster to compute than lasso, but instead of leading to null coefficients, it yields shrinked values. Its advantage is that it increases numerical stability and that it has a grouping effect on correlated predictors. The λ value expresses the overall amount of regularization in the model and its optimal value is usually estimated by grid search.

Finally, regarding model interpretation, the linear relationship of the explanatory variables underlying GLMs allow one to quickly assess the importance of any terms within the model. The standardized coefficient (the raw coefficient estimate divided by the standard error of estimate) represents a raw measure of the relative importance associated to that variable. The standardized coefficient also serves to determine the p-values of test statistics.

4.6 Random Forest

Tree models consist in a series of (possibly nested) if-then statements that partition the data into subsets. The "if-then" statements define splits that eventually define terminal nodes depending on predictors' values, also known as children or leaves. Given a tree, any new observation has a unique route from the root to a specific terminal node. Trees can either be used for classification or regression problems, and are hence also known as Classification And Regression Tree (CART), see e.g. (Breiman 2001). Regression trees are used to predict continuous responses whilst classification ones are used to predict class probabilities. Furthermore, it is possible to convert binary trees into "rules" that are independent sets of if-then statements; this practice can often be advantageous, as pointed out by (Kuhn and Johnson 2013). Both tree and rule based approaches belong to the family of general partitioning-based classification algorithms.

A number of reasons explain their popularity: (1) they are very interpretable and communicable to a non-technical audience, (2) they can handle both numeric and categorical predictors without any pre-processing, (3) they perform feature selection and can handle missing values explicitly, and any missing value is used as another level/numeric value. However, there are known drawbacks worth mentioning, such as model instability leading to possibly big changes in the tree structure given a small change in the data, and the suboptimal performance due to their naturally defined rectangular regions. Further, standard trees are prone to overfitting, since they may find splits in the data that are peculiar to the specific sample being analyzed. Tree pruning techniques have been developed to overcome this specific drawback.

In order to overcome the remaining deficiencies and increase model performance, the endevaour of combining many trees into one model (model ensembling) has become the best practice. Specifically, two main algorithms can be used: "bootstrap aggregation" (bagging), and "boosting". Broadly speaking, the bagging approach refers to fitting different trees on bootstrapped data samples, while boosting consists of sequentially fitting trees and giving more weight to the observations misclassified at the previous step. In this paper, both standard and advanced tree based models will be employed. C5.0 model (Kuhn et al. 2015) is probably the most prominent example of a standard CART model but another relevant method known in the literature is the "Conditional Inference Tree" (Zeileis, Hothorn, and Hornik 2008).

The Random Forest model (Breiman 2001) takes advantage of the bagging methodology and can be used for both classification and regression problems. Further, it is computationally attractive since a high number of independent decision trees on different bootstrapped data samples can be built at the same time during the training phase and the final predictions are obtained by averaging the individual scores of each tree. The trees are ideal for Bagging, since they can capture the complex structures of interaction in the data and if developed sufficiently in depth, they have a relatively low distortion and, as the trees are notoriously noisy, benefit greatly from averaging.

Our analysis made use of the H2O implementation of Random Forest, which introduces some methodological additions to the original algorithm, as well as a computational optimization achieved by parallelized calculation. Specifically, the main tuning parameters used in this paper for the Random Forest algorithm are the following:

1. Predictors' random sample ("mtries"): each tree uses predictors of m-sized random samples of all available predictors in order to achieve independence among trees; this parameter controls for overfitting. Suggested values for this parameter are the square root of predictors for classification problems and one third of predictors for regression problems.
2. Number of independent trees to fit ("ntrees"): by increasing this value more trees are built, making the set of predictions more accurate, but yielding diminishing returns and higher training time. A suggested starting value for this parameter is 1000.
3. Minimum rows ("min_rows"): this represents the minimum number of rows to assign to terminal nodes, and can help against overfitting. The default value for this parameter is 10.
4. Maximum depth of a tree ("max_depth"): this specifies the complexity of interactions for each tree. Increasing the value of this parameter will make the model pick up higher order interactions and hence can lead to overfitting. A reasonable range for this parameter is [4,14], and the default value is 6.
5. Histogram type ("histogram_type"): this parameter determines which type of histogram should be used for finding the optimal split of points in the trees.
6. Row subsample rate ("sample_rate"): the ratio of rows that should be randomly collected by the algorithm at every step. A lower value makes the algorithm faster and less prone to overfitting. A reasonable range for this parameter is (0,1], and the default value is 0.632.

Although ensemble methods return an better performing algorithm overall, they are often considered less interpretable than that of a standard CART. In order to deal with this issue, it is possible to estimate the relative importance of each variable in the regression/classification process.

4.7 The boosting approach: GBM and XGBoost

A Gradient Boosting Machine (GBM) is an ensemble (combination) of regression or classification trees. Unlike Random Forest models ,in which all trees are built independently from one another, in a GBM the setting of a sequential learning procedure to improve accuracy is employed, in which every new tree tries to correct the errors of previously built trees.

Formally, this endevaour is known as "Boosting", in which weak learners (usually CARTs) are sequentially combined into a strong learner using an additive approach, $\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$, where each $f_t(x_i)$ is a tree based prediction. The peculiarity of the boosting approach is that at every step the objective function to be optimized aims to reduce the discrepancy between the outcome and the prediction at the previous step, $\text{obj}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + constant$, being Ω a regularization function. The loss function can be any generic loss function, and does not have to only be the classical Mean Square Error. The above formula gives more weight to samples badly predicted at previous steps. A gradient-based numerical optimization is used to obtain the model loss. Furthermore, GBM algorithms can be parallelised to deliver computationally attractive methods.

Currently, two main variants of classical tree boosting have been proposed, namely Friedman's GBM (Friedman 2001) and the eXtreme Gradient Boosting (XGBoost) (Chen and Guestrin 2016) approach. Recently, XGBoost has gained popularity among data scientists for its faster and better performing boosting algorithm. In particular, the function to be optimized allows for regularization, algorithms can be naturally parallelized, and cross validation can be performed at each step.

A number of hyperparameters that need to be tuned are required for the model to be fully specified. Specifically, XGBoost has four possible types of boosters (in our case trees), and each one comes with dedicated parameters. Since in our study we are taking advantage of the Tree booster, the connected employed hyperparameters are briefly outlined:

1. The number of trees ("nround"): the maximum number of trees to be built during the iterative boosting process.
2. Learning rate ("eta"): controls how much each tree influences the improvement of prediction. It is a form of regularization for the process. For a lower learning rate more trees are required to reach the

same overall error rate (nround and learning rate are inversely related). A reasonable range for this parameter is [0.1,0.01], and the default value is 0.03.
3. Maximum depth of a tree ("max_depth"): same as random forest.
4. Minimum child weight ("min_child_weight"): the minimum sum of instance weight (hessian) needed to create a final leaf (child). A larger value corresponds to a more conservative model, and it hence helps against overfitting. A reasonable range for this parameter is [1,20], and the default value is 1.
5. Row/column subsample ratio ("subsample"/"colsample_bytree"): the ratio of rows/columns that should be randomly collected/selected by the algorithm at every step. A lower value makes the algorithm faster and less prone to overfitting. A reasonable range for this parameter is (0,1], and the default value is 0.5.
6. Gamma ("gamma"): this represents the minimum loss reduction for creating a further partition in a given tree. This is a very important parameter that can ease problems related to overfitting. A reasonable range is [0,1], and the default value is 0.
7. Alpha ("alpha"): it is the L1 regularization term for weights, and can be used to make a model less aggressive, similarly to Gamma. A reasonable range is [0,1], and the default value is 0.
8. Maximum delta step ("max_delta_step"): in a binary class setting with unbalanced classes (as in our study), it is important to include a constraint in the weight estimation, in order to control every update in a more conservative way. In such instances, a reasonable range is [1,10], and the default value is 0.

For simplicity, since the XGBoost parameters outlined above are very similar to those included in the H2O GBM routine (see for detail XGBOOST Parameters), we are not going to re-list them in detail for the GBM implementation. Nevertheless, one can refer to the H2O booklet for more details (Nykodym et al. 2016). Finally, a grid search approach is necessary to obtain the best configuration of the parameters GBM HyperParameters Optimization.

4.8 Deep Learning

Neural networks belong to machine learning algorithms used both for classification and regression problems. Lately, they have been successfully attracting attention in image recognition and natural language processing for their competitive results, see e.g. (Wiley 2016). Neural networks are generally used for detecting recurring patterns and regularities. Those that are characterized by more than one hidden layer are known as "Deep Neural Networks" (DNN). We will focus our attention on feed forward deep neural networks, in which signals go from the input layer to the output layer flowing through the hidden ones without any feedback loop.

Such a model has a logical structure based on interconnected processing units (neurons) structured in one input layer, one or more hidden layers, and an ouput layer. Outputs (signals) from one layer's neurons to the subsequent layer's neurons are linearly weighted, and then passed to an activation function that can take several forms. Specifically, $\alpha = \sum_{i=1}^{n} w_i \times x_i + b$ is the weighted combination of input signals in a generic neuron that is passed to an activation function $f(\alpha)$, where w_i is the weight for the x_i observation, while b represents the bias node, which behaves in a way similar to the intercept within a linear regression setting.

Given a network structure, the model is fitted by finding the optimal combination of weights w and bias b that minimizes a specified loss function, and the resulting performance is extremely sensitive to the hyper-parameter configuration. Specifically, the main parameters to be tuned are described in more detail by (Arora et al. 2015) in the framework are:

1. Network architecture ("hidden"): the number and size of the hidden layers.
2. Activation function ("activation"): common choices include the Rectifier, Tahn, and Maxout functions.
3. Regularization Parameters: it is possible to apply regularization techniques that resemble the lasso and ridge approaches.
4. Loss function ("stopping_metric"): the chosen loss function to be optimised by the model.
5. Adaptive learning parameters ("Adaptive Rate"): when set to True, the following parameters control the process of weight updating, and are useful for avoiding local minima. ρ, controls how the memory of prior weights updates, and is usually set between 0.9 and 0.999; ε takes into account the learning rate annealing and momentum allowing forward progress, and is usually set between 10^{-10} and 10^{-4}.
6. Number of iterations ("Epochs"): the number of passes over the complete training dataset to be iterated.

7. Input dropout rate ("input_dropout_ratio"): this determines what percentage of the features for each training row is to be omitted from training in order to improve generalization.
8. Hidden layers' dropout rate ("hidden_dropout_ratios"): the fraction (default set to 0.5) of the inputs for each hidden layer to be omitted from training in order to improve generalization.
9. Maximum sum of the squared weights into neurons ("max_w2"): this parameter is helpful whenever the chosen activation function is not bounded, which can occur with Maxout and Rectifier.

5 Numerical evidence

In this section, we present the dataset used for the numerical illustration. Then, we fit a standard GLM to model policy conversion, which will serve as a benchmark against the competing machine learning algorithms. Secondly, we apply non-linear and tree based machine learning techniques to predict policy conversion, and we compare all the methods by accuracy and discriminating power. Finally, we perform a price optimization in order to assess the benefits of machine learning techniques compared to the standard approach.

5.1 Brief description of the dataset

In our study of conversion, we use a database of 1.2 million quotes for fitting and model selection. 80 percent of the database is used for fitting models (the training dataset) whereas the remaining 20 percent of the database is used for testing the performance (the test dataset). Furthermore, an additional 100 thousand records (optimization dataset) are used for the PO exercise one. More precisely, models have been fit on the train data set and compared in terms of predictive performance on the test set. The relevant models have been applied on the PO data set to perform the optimization exercise. Table 2 shows models' performance on the training dataset, while Table 3 gives PO performance on the optimization dataset. Tables 4-7 display descriptive statistics of the training dataset.

The available variables used for predictors are listed below:

- **Id & conversion status**: *quoteId* (db key) and *converted* (binary variable).
- **Premium & competitive position related variables**: *premiumCompany*, *premiumMarket* and *burningCost* represent the company's premium, the market best price (average of the three lowest premiums) and the pure premium (loss cost) respectively. *ratioCompanyMkt* and *deltaCompanyMkt* have been calculated as the ratio and the difference between the company premium and market premium respectively. The currency is euro.
- **Vehicle characteristics**: *vehicleMake*, *vehicleModel*, *vehiclePower*, *vehicleFuelType*, *vehicleAge*, *vehiclePurchaseAge*, *vehicleKm*, *vehicleUsage* are brand, model, engine characteristics, age and usage style variables. In addition, *vehicleMakeandModel* groups the most frequent combinations of vehicles makes and models.
- **Demographics**: *policyholderTerritory* and *territoryBigCity* indicate the policyholder living region and whether the policyholder's town is a high density city. Policyholder's age, gender, marital status and occupation are recorded into the *policyholderAge*, *policyholderGender*, *policyholderMaritalStatus*, and *policyholderOccupation* variables respectively.
- **Insurance and claim history variables**: *bonus malus* and *policyholderPreviousClaims* indicate attained bonus malus level and whether any previous claims has been filled within five years before. *quoteTimeToPolicy* indicate the difference (in years) between the quote and the effective date. Finally, the *previousCompany* variable indicates whether the previous company was a direct company or a traditional one.

Our analysis focuses on the key variables (conversion rate and premium variables), as well as other variables. A classical bivariate analysis representing the relationship between conversion rate and selected predictor variables is reported in the Appendix.

5.2 Model Comparison and Selection

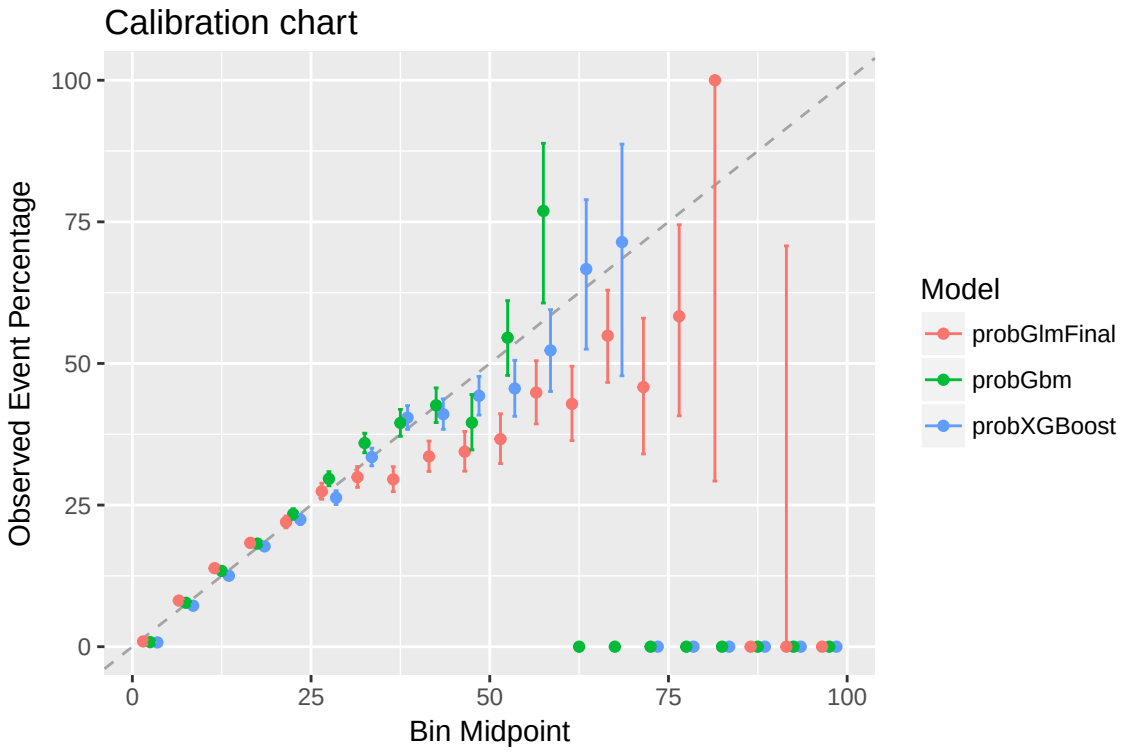
The previously fitted models have been used to predict the conversion probability on the test dataset. They have been ranked according to performance metrics and the better performing one is going to be selected to predict prospects' conversion on the test dataset and on the PO one. Finally, the obtained results will be compared against a GLM model.

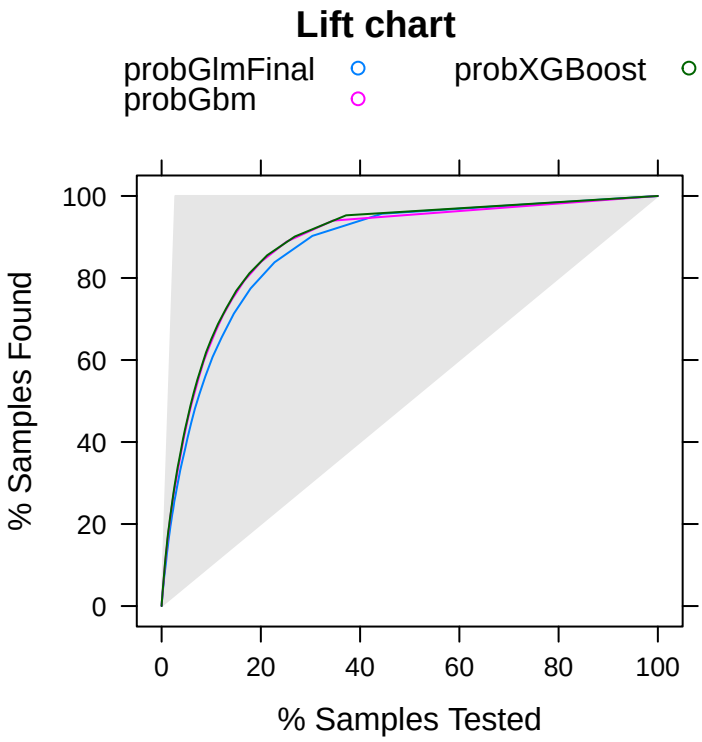
Table 2 shows the total predicted converted policies on the test data set, the log-loss and the AUC measures for each model, while the actual total number of converted policies is shown on top. The total estimated converted policies is calculated as the sum of quote - level estimated conversion probabilities. Clearly, the best model is the one that is able to predict the number of converted policies with the highest degree of accuracy, such that the the log-loss is minimized and the AUC is maximized.

Table 2: Models's performance metrics comparison

Model	Quote Nb	Log-loss	AUC
observed	6800	NA	NA
XGBoost	6826	0.0890	0.9064
GBM	6314	0.0896	0.9050
Random Forest	6817	0.0923	0.8955
Deep Learning	7438	0.0936	0.8925
GLM	6831	0.0940	0.8896

One can observe that all model predictions are quite close the observed one in terms of quote number, with the exception of deep learning. Conversion modeling requires precise assessment of underlying conversion probabilities, which is measured by log-loss. By ranking the models according to increasing log-loss, it is clear that boosted models (GBM and XGBoost) show the highest performance in terms of predictive accuracy.





Interestingly, GBM and XGBoost show higher precision in terms of estimated probability when compared to a GLM model, even though the underlying conversion probability of most quotes is very low. Also, they keep estimating the conversion rate unbiasely at levels of expected outcome higher than the GLM model. The lift chart shows that there is not much difference between the three predictive models (GLM, GBM and Ensemble-GBM) even if the last two are slightly superior in terms of lift.

It is indeed difficult to compare the models in term of computational requirements. Using the H2O package infrastructure, fitting a GLM (even using an elasticnet approach) takes a fraction of the time needed to select a model within the cited ML approaches. The subjectivity of hyper - parameter search (grid depth, use of Bayesian Optimization, ...) in addition to the time required to fit a hyperparameter-definite ML model explains the difficulty of such comparisons.

5.3 Application to Pricing Optimization

The pricing optimization process considers knowledge of consumer behaviour vital to maximizing expected profit. In our application, the company's knowledge of consumer behaviour is represented by a risk premium model that estimates the expected cost of the insurance cover that will be provided, as well as by a conversion model, which estimates the probability of a prospect entering the portfolio. Also, information on the competitive environment such as the distance between market price and the company's premium should be taken into account.

From a mathematical point of view, we assume that the insurer sets the best premium π for each prospect that maximizes the expected underwriting result weigthed by the conversion probability. That is $uw(\pi) = p(\pi) \times (\pi - L)$, where $p(\pi)$ is the conversion probability given the π proposed premium and L the expected cost. In the subsequent part of the analysis, we will assume that the insurer can modify the proposed premium without any restriction.

The following hyphotheses were used to perform the pricing optimization exercise:

- the insurer follows an individual optimization approach;

- the individual quoted premium can vary between -10% and + 10% (ε) around the baseline;
- the company is able to estimate the expected cost of providing coverage at policy level, L_i, thanks to a risk premium model.
- the company calculates the expected underwriting result $uw(\pi_i) = \pi_i - L_i$.

We use the notation $p_i(\pi_i)$, π_i, L_i to refer to the i-th quote's conversion probability, premium and burning cost respectively. Therefore, the expected number of conversions is simply $E(Q) = \sum_i p_i(\pi_i)$, the expected gross premium volume is $E(PR) = \sum_i p_i(\pi_i)\pi_i$, and the expected underwriting margin is $E(UW) = \sum_i uw_i(\pi_i)$. The L_i expected cost per quote is considered known and given by the risk premium model.

The individual PO is carried out as follows:

- the insurer calculates an initial technical premium π_i^0 for the ith prospect, without competitive marketing considerations, e.g., adding a fixed loading to the estimated cost L_i.
- on a discretized grid ranging in $[\pi_i^0(1-\varepsilon), \pi_i^0(1+\varepsilon)]$
  - it computes the conversion probability $p_i(\pi_i)$, and the expected underwriting result $uw_i(\pi_i)$. This also re - evaluates the competitive position for each scenario.
  - it chooses the individual customer's premium variation to maximize the underwriting result from within the grid.

This approach is known as individual optimization, since it assumes that the insurer is able to target any single prospect with an individual quote. The presence of any marketing and regulatory restrictions would hence lead the insurer to less refined strategies as previously discussed. In addition , this deterministic analysis on one period does not take into account any solvency considerations, as well as the renewals of prospects beyond the first period. Also, it assumes no structural changes in the market (e.g. no reactions from competitors) during the process of its implementation.

Thus, it is possible to compare the actual quote number and observed margin (computed on converted quotes) with the amount predicted by each predictive model, as shown in the table below for the selected elastic-net GLM, GBM, and XGBoost models. The final "PO Gain" column, calculated as the difference of the optimized margin and the baseline one, shows the gain of using an individual PO approach. As previously anticipated, the PO exercise has been carried out on a data set (the optimization dataset) that contains quotes used neither to train nor to test the ML models.

Table 3: Pricing optimization exercise, summary of results

Model	Conversion Nb	Baseline margin	Optimized margin	PO gain
observed	2522	-88203	NA	NA
GLM	2409	-73000	749.5	73749
GBM	2492	-76341	2384.8	78726
XGB	2654	-86815	2298.3	89114

In Table 3, we observe that all boosted models are closer than GLM in terms of estimated conversions. In particular, the GBM figure is the closest one. On the other hand, the estimated margin shows that the XGBoost estimate is very close to the actual figure. It is worth pointing out that the margin figure should be compared to the total gross premium of around 600K. After the individual optimization exercise has been performed, the "PO gain" column shows that the difference between the baseline and the optimized underwriting margin varies between 74K (GLM figure) and 89K (XGBoost one).

6 Conclusion

Our work has applied recent predictive modeling methodologies with the dual goals of taking into account customers' behavior and of optimizing underwriting margins. The size of the considered dataset is relatively

large compared to the market size, leading to reasonably generalizable results other personal lines in non-life insurance.

We observed that machine learning models may offer higher accuracy and discriminating power when compared to classical GLM models. Our exercise also confirmed the excellent performance of boosted tree based models. It is still an open question whether the predictive performance gain of machine learning methods is enough to suggest their widespread adoption. Interestingly, we found that both GLM and XGBoost approaches produced very similar results in terms of optimized premium volume. Nevertheless, competitive results of the logistic regression can be explained by the fact that the marginal conversion rate observed for variables that have been found to be extremely important, such as premiums distance and time to policy, seem monotonic and can be approximately linear (e.g. in the log scale).

Furthermore, we noted that the performance difference between ML approaches and classical GLM is relatively higher on the AUC scale than on the log-loss scale. This can also be visually seen by observing the lift curve. Therefore, it is suggesed that machine learning models can offer a competitive advantage when used for actively marketing to prospective customers, rather than when optimizing the premium amount.

Regarding the computational resources, it is clear that fitting a GLM takes a very small fraction of the computational time required by most machine learning models. This is due to the fact that a GLM approach does not require any hyperparameter tuning, unlike the vast majority of machine learning algorithms. Precisely, since it is not generally possible to find a priori the optimal configuration of the hyperparameters space, a grid search approach is necessary. The subjectivity of defining the grid space and depth adds another level of complexity when comparing the tuning process and the timing requirements across different families of models.

7 Acknowledgements

This work has been sponsored by the Casualty Actuarial Society (CAS), the Actuarial Foundation's research committee, and the Commitee on Knowledge Extension Research (CKER) of the Society of Actuaries (SOA). The authors wish to give a special thanks to David Core, Director of Professional Education and Research (CAS), and Erika Schulty, Research Associate (SOA) for their support.

Finally, the opinions expressed in this paper are solely those of the presenters. Their employers neither guarantee the accuracy nor reliability of the contents provided herein nor take position on them.

8 Appendix

8.1 Infrastructure

The R software (R Core Team 2017) has been used for this analysis, taking advantage of packages `tidyverse` and `data.table`, as well as the H2O (H2O.ai team 2017), and caret (Jed Wing et al. 2016) machine learning infrastructures.

Parallelization of computing tasks is available for both infrastructures, either on multicore processors or on clusters of computers. Furthermore, the H2O infrastructure permits an easy interface to the Apache Spark (Zaharia et al. 2016) framework specifically devoted to perform parallel processing on big data. Finally, all the software used in the project is open source, making our analysis easily replicable.

8.2 Tables and graphics for the descriptive section

The table and the plot indicate that the portfolios' loss ratio is slightly above 80%, while the histograms show a clearly skewed distribution of premium and losses, as expected. In addition, the overall conversion

rate (actual conversions divided by number of quotes) is around 2.5%. Bivariate tables display the conversion rate by each level of key predictors.

Table 4: Conversion Rate Summary

converted	Num	Freq
N	103027	0.9761
Y	2522	0.0239

Table 5: Conversion by competitiveness (ratio)

ratioCompanyMktBin	num	conversions	ratio
[0.476,0.900)	104876	13106	0.1250
[0.900,0.969)	94077	5813	0.0618
[0.969,1.027)	99795	3624	0.0363
[1.027,1.084)	101219	2146	0.0212
[1.084,1.142)	101721	1249	0.0123
[1.142,1.209)	102093	734	0.0072
[1.209,1.292)	102538	396	0.0039
[1.292,1.405)	101433	211	0.0021
[1.405,1.603)	99236	97	0.0010
[1.603,8.502]	74727	24	0.0003

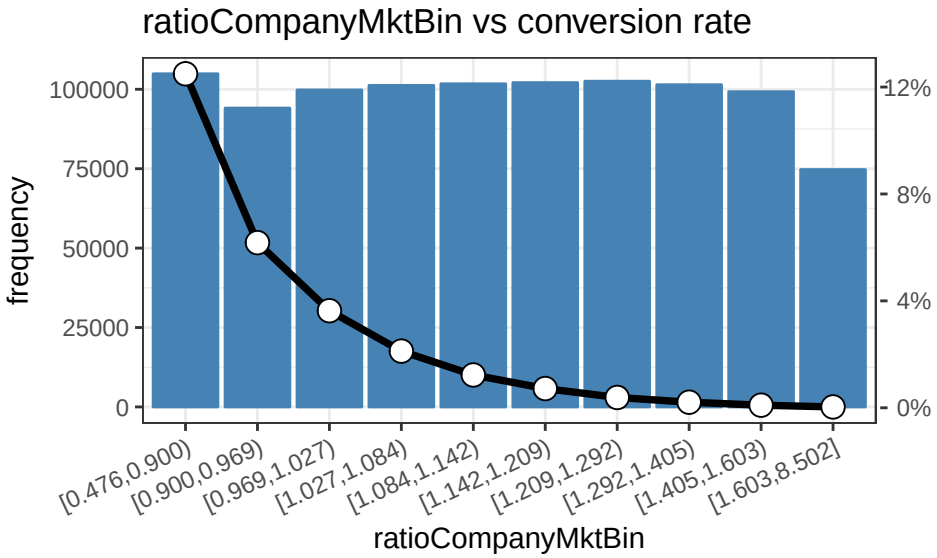


Table 6: Conversion by policy effective date delay

quoteTimeToPolicyBin	num	conversions	ratio
[0.00000,0.00822)	119750	4253	0.0355
[0.00822,0.02466)	122738	5826	0.0475
[0.02466,0.04658)	153243	5510	0.0360
[0.04658,0.06575)	136130	3310	0.0243
[0.06575,0.08767)	153873	3322	0.0216
[0.08767,0.10959)	176746	3175	0.0180
[0.10959,0.37260]	119235	2004	0.0168

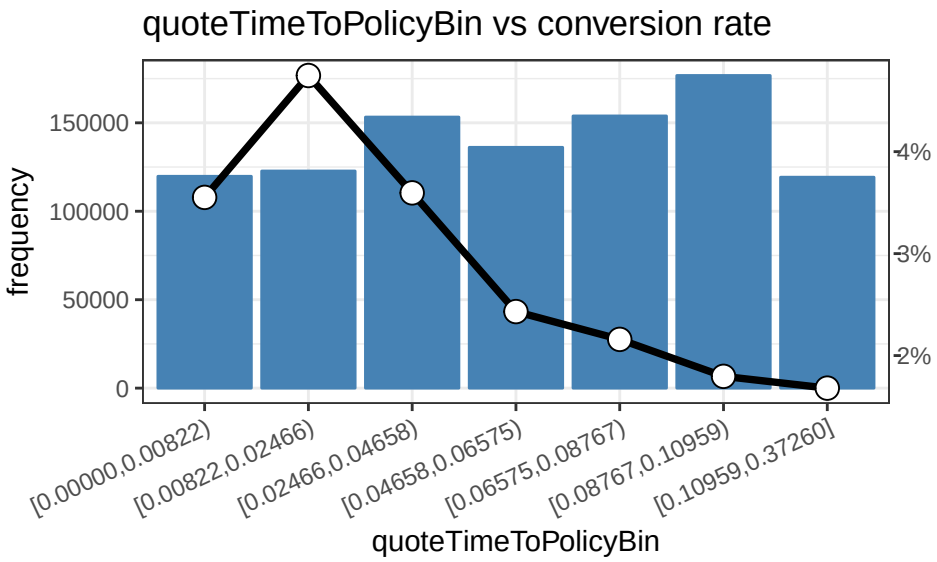
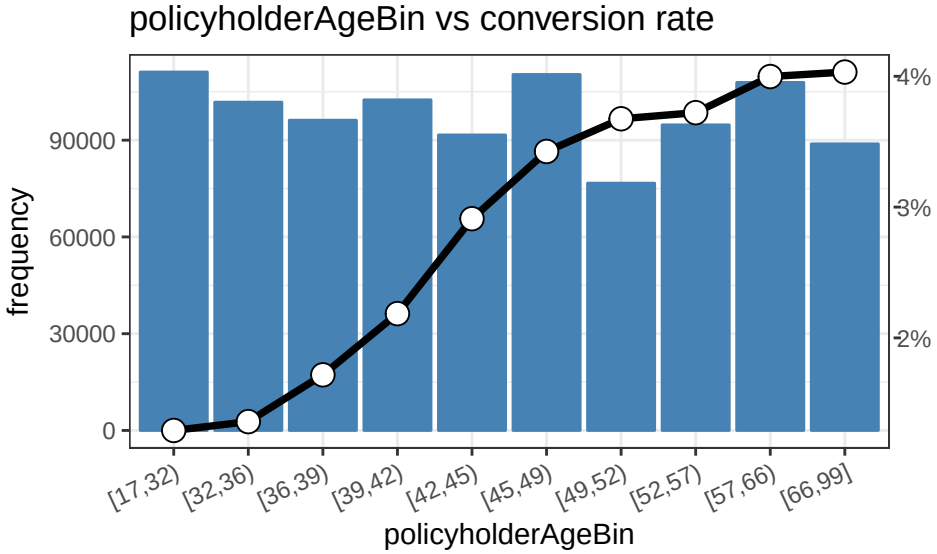


Table 7: Conversion by policyholder's age

policyholderAgeBin	num	conversions	ratio
[17,32)	111121	1436	0.0129
[32,36)	101780	1384	0.0136
[36,39)	96184	1652	0.0172
[39,42)	102495	2240	0.0219
[42,45)	91612	2667	0.0291
[45,49)	110358	3780	0.0343
[49,52)	76675	2818	0.0368
[52,57)	94735	3526	0.0372
[57,66)	107900	4314	0.0400
[66,99]	88855	3583	0.0403



8.3 Models' specific implementation notes

8.3.1 GLM

The elastic net approach has been used in this analysis, since it allows one to perform variable selection and prevent overfitting and collinearity among predictors, with no dramatic increase of computational time.

Two models have been tested:

1. A model with an unbinned coefficient (thus assuming marginal linearity of categorical predictors).
2. A model with a binned one. Bins were constructed on continuous covariates based on deciles.

The log - loss criterion identified the unbinned model to be the most performing one.

8.3.2 Random Forest

H2O Distributed Random Forest (DRF) can be initialized using the code snippet shown in the corresponding appendix section. The model has been tuned using a random discrete grid search approach and log-loss as criteria. Two sets of grids have been used: the first one runs over a relatively wide range set of each parameter, while the second one on a narrower one. The second grid ranges have been defined on the basis of the best models found at the previous step.

Precisely, each grid has been set to run for both a maximum number of models (200 models), and a maximum amount of time (4 hours). In addition, we have judiciously selected the ranges of the tuning parameters. In the second grid we have: max depth (8-12), min rows (10-20), sample rate (60% - 80%) and ntrees (100 - 300).

8.3.3 Boosting (GBM and XGBOOST)

Regarding GBM, the H2O infrastructure has been adopted to perform the required pre-processing, as well as training and prediction. In terms of hyper-parameter tuning, a grid search has been carried out, by adopting the random discrete strategy included in H2O, by training up to 100 models for a maximum of 4 hours during the first cycle. Further, a second and finer grid search on the same parameters has been performed. On other other hand, the dedicated library has been used to implement XGBoost models: as a preliminary step, categorical variables have been one-hot encoded, and simultaneously a sparse matrix has been created to help computational efficiency. Next, an initial cartesian grid search has been performed on parameters of main importance. Then, a more refined tuning grid search has been performed on some parameters, and "max_delta_step" has been included to take into account the presence of unbalanced classes. Finally, the learning rate and the number of rounds were optimised jointly.

8.3.4 Deep Learning

Generally there are no definite rules for optimal architecture design, in particular regarding the layering of the network (number and individual size of hidden layers). Some tips can be found in H2odeepLearning. We decided to perform a random discrete search across a hypherparameter space assuming:

1. No more than three hidden layers.
2. The size has been calculated as the number of continuous predictors plus the number of distinct values per each categorical predictor.
3. Scoring of the model during training: (1%, 10000) samples.
4. The activation function in each node. All available activation functions were tested (Rectified, Tanh, Maxout and the corresponding "with dropout")
5. ρ and ε as well as $l1$ - $l2$ ranges chosen according to the suggestion of the (Arora et al. 2015).

A first grid has been launched to find a more narrow combination of tuning parameters, while a second one has been defined using a more narrow parameter range defined on the basis of previous results.

References

Anderson, D., S. Feldblum, C. Modlin, D. Schirmacher, E. Schirmacher, and N. Thandic. 2007. *A Practitioner's Guide to Generalized Linear Models*. Arlington: Casualty Actuarial Society.

Arora, A., A. Candel, J. Lanford, E. LeDell, and V. Parmar. 2015. *Deep Learning with H2o*. http://h2o.ai/resources.

Baribeau, A.G. 2015. "Price Optimization and the Descending Confusion." *Actuarial Review*.

Bett, L. 2014. *Machine Learning with R*. Packt Publishing.

Bou Nader, R., and E. Pierron. 2014. "Sustainable Value: When and How to Grow?" In *Proceeding of the 30th International Congress of Actuaries*.

Box, G.E.P., and D.R. Cox. 1964. "An Analysis of Transformations." *Journal of the Royal Statistical Society. Series B (Methodological)*. JSTOR, 211–52.

Breiman, L. 2001. "Random Forests." *Machine Learning* 45 (1). Springer: 5–32.

Chen, T., and C. Guestrin. 2016. "Xgboost: A Scalable Tree Boosting System." *arXiv Preprint arXiv:1603.02754*.

Cortes, C., and V. Vapnik. 1995. "Support-Vector Networks." *Machine Learning* 20 (3). Springer: 273–97.

Duncan, A., and M. McPhail. 2013. "Price Optimization for the U.S. Market. Techniques and Implementation Strategies."

Dutang, Christophe. 2012. "The Customer, the Insurer and the Market." *Bulletin Français d'Actuariat* 12 (24).

Fisher, R.A. 1940. "The Precision of Discriminant Functions." *Annals of Eugenics* 10 (1). Wiley Online Library: 422–29.

Fix, E., and J.L. Hodges Jr. 1951. "Discriminatory Analysis-Nonparametric Discrimination: Consistency Properties." Defense Technical Information Center Document.

Frees, E.W., R.A. Derrig, and G. Meyers. 2014. *Predictive Modeling Applications in Actuarial Science*. Vol. 1. Cambridge University Press.

Frees, E.W., G. Meyers, and R.A. Derrig. 2016. *Predictive Modeling Applications in Actuarial Science: Volume 2, Case Studies in Insurance*. International Series on Actuarial Science. Cambridge University Press. https://books.google.it/books?id=5/_a7DAAAQBAJ.

Friedman, J.H. 2001. "Greedy Function Approximation: A Gradient Boosting Machine." *Annals of Statistics*. JSTOR, 1189–1232.

Fu, L., and H. Wang. 2014. "Estimating Insurance Attrition Using Survival Analysis." *Variance* 8 (1): 55–72.

Guelman, L. 2012. "Gradient Boosting Trees for Auto Insurance Loss Cost Modeling and Prediction." *Expert Systems with Applications* 39 (3). Elsevier: 3659–67.

Guelman, L., M. Guillen, and A.M. Perez-Marin. 2012. "Random Forests for Uplift Modeling: An Insurance Customer Retention Case." In *Modeling and Simulation in Engineering, Economics and Management*, edited by K.J. Engemann and A.M. Gil-Lafuente, 115:123–33. Lecture Notes in Business Information Processing. Springer Berlin Heidelberg. doi:10.1007/978-3-642-30433-0_13.

Guillen, M., and L. Guelman. 2014. "A Causal Inference Approach to Measure Price Elasticity in Automobile Insurance." *Expert Systems with Applications*.

H2O.ai team. 2017. *H2O: Scalable Machine Learning*. http://www.h2o.ai.

Hastie, T., A. Buja, and R. Tibshirani. 1995. "Penalized Discriminant Analysis." *The Annals of Statistics*.

JSTOR, 73–102.

Hung, S.-Y., D.C. Yen, and H.-Y. Wang. 2006. "Applying Data Mining to Telecom Churn Management." *Expert Systems with Applications* 31 (3). Elsevier: 515–24.

Hussami, N., T. Kraljevic, J. Lanford, T. Nykodym, A. Rao, and W. Wang. 2015. *Generalized Linear Modeling with H2o*. http://h2o.ai/resources.

Jed Wing, Max Kuhn. Contributions from, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, et al. 2016. *Caret: Classification and Regression Training*. http://CRAN.R-project.org/package=caret.

Kuhn, M. 2016. "Optimization Methods for Tuning Predictive Models." In *Cambridge R User Group*.

Kuhn, M., and K. Johnson. 2013. *Applied Predictive Modeling*. Springer.

Kuhn, M., S. Weston, N. Coulter, and M. Culp. 2015. *C50: C5.0 Decision Trees and Rule-Based Models*. http://CRAN.R-project.org/package=C50.

LeDell, E., S. Sapp, and M. van der Laan. 2014. "Subsemble: An Ensemble Method for Combining Subset-Specific Algorithm Fits." R package version 0.0.

Liaw, A., and M. Wiener. 2002. "Classification and Regression by randomForest." *R News* 2 (3): 18–22.

Manin, A., and T. Bayley. 2010. "Price Optimization: For New Business Profit and Growth." *Emphasis* 1. Towers Watson: 18–22.

McCullagh, P., and J.A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. Chapman; Hall.

Milhaud, X., S. Loisel, and V. Maume-Deschamps. 2011. "Surrender Trigger in Life Insurance: What Main Features Affect the Surrender Behavious in an Economic Context." *Bullettin Francais d'Actuariat*, no. 11.

National Association Insurance Commissioners. 2015. "Price Optimization White Paper."

Nykodym, T., T. Kraljevic, N. Hussami, A. Rao, and A. Wang. 2016. "Generalized Linear Modeling with H2o."

Pozzolo, A. dal. 2011. "Comparison of Data Mining Techniques for Insurance Claim Prediction." Master's thesis, University of Bologna.

Quinlan, R. 2004. "Data Mining Tools See5 and C5.0."

R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ratemaking, Casualty Actuarial Society Committee on. 2014. "Price Optimization Overview." *Casualty Actuarial Society*. Casualty Actuarial Society. http://www.casact.org/area/rate/price-optimization-overview.pdf.

Rish, I. 2001. "An Empirical Study of the Naive Bayes Classifier." In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, 3:41–46. 22. IBM New York.

Rulliere, J.L., S. Loisel, and C. Mouminoux. 2017. "Obfuscation and Trust: Experimental Evidence on Insurance Demand with Multiple Distribution Channels." In *8th International Conference of the French Association of Experimental Economics - Asfee*.

Santoni, A., and F. Gomez Alvado. 2007. "Sophisticated Price Optimization Methods."

Serhat, G. 2013. "II-4: Intelligent Use of Competitive Analysis."

Serhat, G., and M. McPhail. 2013. "Beyond the Cost Model: Understanding Price Elasticity and Its Applications." *CAS E-Forum* 2. Casact.

Serneels, S., E. De Nolf, and P.J. Van Espen. 2006. "Spatial Sign Preprocessing: A Simple Way to Impart Moderate Robustness to Multivariate Estimators." *Journal of Chemical Information and Modeling* 46 (3).

ACS Publications: 1402–9.

Wiley, J.F. 2016. *R Deep Learning Essentials*. Packt Publishing Ltd.

Yeo, A., K.A. Smith, R.J. Willis, and M. Brooks. 2001. "Modeling the Effect of Premium Changes on Motor Insurance Customer Retention Rates Using Neural Networks." In *Computational Science - Iccs 2001*, edited by VassilN. Alexandrov, JackJ. Dongarra, BenjoeA. Juliano, and C.J.Kenneth Renner Tan, 2074:390–99. Lecture Notes in Computer Science. Springer Berlin Heidelberg. doi:10.1007/3-540-45718-6_43.

Zaharia, M., R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, and M.J. Franklin. 2016. "Apache Spark: A Unified Engine for Big Data Processing." *Communications of the ACM* 59 (11). ACM: 56–65.

Zeileis, A., T. Hothorn, and K. Hornik. 2008. "Model-Based Recursive Partitioning." *Journal of Computational and Graphical Statistics* 17 (2): 492–514.

Zou, H., and T. Hastie. 2005. "Regularization and Variable Selection via the Elastic Net." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67 (2). Wiley Online Library: 301–20.