Binary classification of insurance claims

150022854 - Applied Stats

April 14, 2016

Abstract

The following paper describes the modelling process with respect to the BNP Paribas Kaggle competition. The objective was to design a classifier for a binary response variable using the provided set of anonymised covariates. After extensive testing, the final model chosen was an ensemble of boosted trees, fitted through gradient boosting. Such a classifier provided a generalisation log-loss of 0.47010.

1 Introduction

The dataset of interest was related to insurance claims. The objective was to correctly classify those claims in two classes: claims whose approval could be accelerated (coded as 1s) and claims which could not (coded as 0s). Variables in this dataset (113 Mb), of dimensions 114321x133, were completely anonymised, so no clear meaning can be attributed to them. No ordinal variable was present. However, it presented a significant amount of missing values: NAs were 33% of all dataset. With respect to the hardware and software machinery used, the R language was chosen and run on a an *Core i7* CPU and 8 GB of RAM.

The modelling process was carried out in group, with every member designing its own classifiers and sharing code. This procedure was repeated through multiple rounds. Therefore, this report will present the author's individual choices, even though the influences of the whole group were significant.

2 Data cleaning and preparation

The first problem to tackle was the big number of NAs. Hence, to get rid of these without loosing a considerable amount of the data, multiple imputation was chosen. In first instance however, only numerical covariates were selected, on the idea that this would have simplified the estimation process. The *mice* algorithm in the *Mice* package, which executes MI through Fully Conditional Specification (Van Buuren, 2007). Such method is based on the idea of constructing an imputation model for every Y_j (*i.e.* every variable) by specifying a conditional density: $P(Y_j|X, Y_{-j}, R, \theta_j)$ where:

- X is a complete set of covariates. Obliviously, to initialize the algorithm, the missing X are firstly simply guessed.
- Y_{-j} is the set of variables in Y expect Y_j.
- R the set of binary response indicators for each variable k: $R = (R_1, ..., R_k)$. If $R_j = 1$, Y_j is missing.
- θ_j is a set of parameters for Y_j.

FCS is done by iterating over all conditionally specified imputation models, which can differ for every Y_j. In such an algorithm, one iteration consist of a passage through all Y. This process it then repeated for m times (usually $m \in [3, 10]$) generating m datasets, upon which the user can execute its own analysis of interest. The m set of estimates are then pooled in a single set, using certain variance adjustments for the increased uncertainty due to missing data (Barnard & Rubin, 1999). MI through FCS seemed to provide satisfactory results which are generally unbiased and with appropriate coverage (Van Buuren, 2007).

However, due to the relatively big size of the dataset, such imputation method proved to require long computing time (to impute just one meagre iteration, the author left his machine to work for about 36 hours). Therefore, just one imputed dataset was produced, which introduced significant source of bias in the early model. In the third and final model, a simple median/mode imputation method was used, which seemed to allow for lower generalisation error than the single dataset constructed with FCS. In this case, all covariates (both quantitative and categorical) were included. Nevertheless, if more computing time had been available, MI would have been the most reasonable choice.

3 Models

The preliminary model was designed with simplicity as a goal: its only role was to provide a benchmark upon which to base the generalisation error improvements. To satisfy such aim, a GLM was chosen. Specifically a model of the form:

$$\ln\left(\frac{p_i}{1-p_i}\right) = \eta_i = \beta X + \epsilon \tag{1}$$

$$\epsilon \sim Bin(n, p) \tag{2}$$

Hence, a logistic regression model. A single imputed dataset with FCS was used. Such attempt produced a generalisation log-loss (estimated using a validation set) of 0.559.

The second model that the author tried, was an ensemble of trees, fitted using the gradient boosting algorithm (see (Hastie, Tibshirani, & Friedman, 2001) pp. 353-361 for a detailed explanation). The dataset used was again based on the FCS imputation method.

In its most simple essence, boosting creates a sequence $m = 1, ..., M$ of trees that are grown sequentially, that is each tree is grown by using information based on previous trees. For this competition, gradient boosting was used: at each iteration m we calculate the negative gradient of the loss function evaluated at the previous iteration $m-1$:

$$r_{im} = -\left[\frac{\delta L\left(y_i, f(x-i)\right)}{\delta f(x_i)}\right]_{f=f_{m-1}} \tag{3}$$

Subsequently, a regression tree is fitted to those targets r_{im}, obtaining terminal regions R_{jm}, $j = 1, ..., J_M$. For every region, we then find the corresponding constants γ_{jm}

$$\gamma_{jm} = \underset{\gamma}{argmin} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma) \tag{4}$$

with which we will update our $f_m(x)$, *i.e.* our classifier at iteration m:

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm}) \tag{5}$$

The easiest way to avoid over-fitting is to control the number of iterations M, even though simulations showed that generalisation error for boosted models does not always increase with over-fitting. Nevertheless, as a rule of thumb, each individual tree should be no bigger than 8 leaves.

However, before applying such a model, a more efficient computational environment was needed. For this reason, the *H2O* package was used, which allowed for parallelization over all the CPU cores. Thank to this improvement in performance, 5-fold CV was used to tune the learning rate and number of trees for the gboost algorithm. The results are summarized on Table 1. The final model was based on a learning rate of 0.01 and 2000 trees: on the Kaggle validation set it returned a log-loss of 0.4968.

N Trees	Learning Rate	Log-loss
1000	0.05	0.5448
1000	0.01	0.5110
2000	0.05	0.5073
2000	0.01	0.4997

Table 1: CV log-loss for gradient boosted trees at different depths and learning rates

Subsequently, another gradient trees boosting model was chosen. This time however, the missing values were imputed by median/mode and the model was fitted using a different R package: *xgboost*. The CV log-loss at different Learning Rates are displayed in Table 2

Learning Rate	Log-loss
0.1	0.5677
0.6	0.4755
0.7	0.4742
0.8	0.4766

Table 2: CV log-loss for gradient boosted trees at different learning rates. Fitted using the *xgboost* library

Finally, a simple models average ensemble was tried *i.e.* the three best group-submitted models were selected and their prediction averaged to get a new set of probabilities. Such a process should reduce the potential over-fit to the training data. However, it did not provide better predictions given that its log-loss score against a validation set was of 0.4834.

4 An unorthodox approach

After having concluded such models, the author tried to leave the solid ground of statistical theory to construct an experimental ensemble, which due to its structure, can be quite similar with a Neural Network. This ensemble was made up of 2 layers as showed in Figure 1.

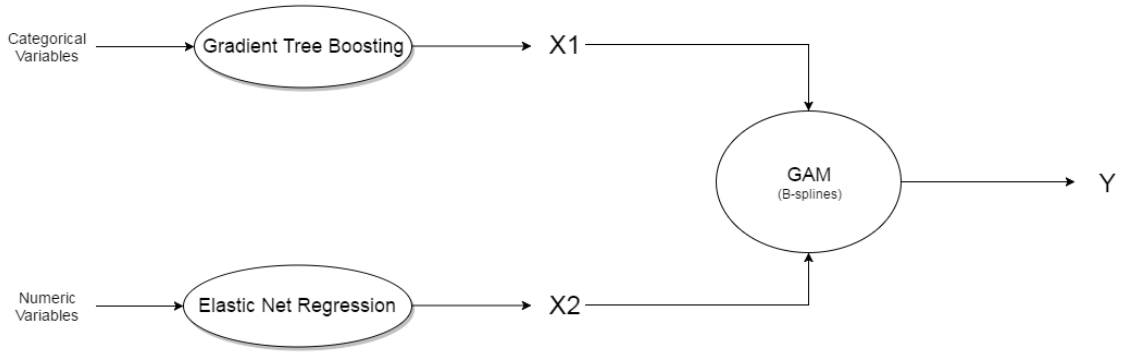

Figure 1: Flow diagram of the ensemble model attempted

The inputs of the first layer were the separate categorical and numeric variables. The former were fitted through a gradient tree boosting model, whose parameters were estimated through 5-fold CV (see Table 3). The latter through an Elastic Net (EN) regression (Zou & Hastie, 2005): also in this case the EN mixing parameter α has been chosen with 5-fold CV. The optimal shrinkage parameter λ on the contrary has been automatically estimated by the R fitting function (*h2o.glm* from the package *H2O*). See Table 4 for the relevant figures.

N Trees	Learning Rate	Log-loss
800	0.05	0.5667
1000	0.05	0.5448
1000	0.1	0.7174
2000	0.05	0.7120
2000	0.1	0.8845

Table 3: 5-fold CV log-loss for gradient boosted trees at different depths and learning rates

Log-loss	λ	α
0.5152	0.007445	1
0.5151	0.00827	0.90
0.5149	0.009307	0.8
0.5149	0.009307	0.7
0.5147	0.01241	0.6
0.5145	0.01696	0.4

Table 4: 5-fold CV log-loss for Elastic Net regression at different tuning parameters

After having fitted and predicted those $k \in [1, 2]$ models they provided a y_{ik} each. Those two sets of outputs were then fed into an outer modelling layer composed by a GAM model with *B-splines* as basis expansion. Obliviously, errors were modelled using the Binomial distribution with a logit link function. Knot selection was carried out using the *Spatially Adaptive Local Smoothing Algorithm* (SALSA) which automatically selects the best number and position of knots based on objective fit criteria and CV (Walker, Mackenzie, Donovan, & O'Sullivan, 2011). A GAM approach was selected to cope with the non-linearities on the link scale that were significant in the preliminary model.

Despite the effort, such architecture did not perform well: it produced, against the test set, a log-loss of 0.657: significantly more than the simple boosted trees on all covariates.

References

Aiello, S., Kraljevic, T., & with contributions from the H2O.ai team, P. M. (2016). h2o: R interface for h2o [Computer software manual]. Retrieved from `http://CRAN.R-project.org/package=h2o` (R package version 3.8.1.3)

Barnard, J., & Rubin, D. B. (1999). Miscellanea. small-sample degrees of freedom with multiple imputation. *Biometrika*, *86*(4), 948–955.

Chen, T., He, T., & Benesty, M. (2016). xgboost: Extreme gradient boosting [Computer software manual]. Retrieved from `http://CRAN.R-project.org/package=xgboost` (R package version 0.4-3)

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York, NY, USA: Springer New York Inc.

R Core Team. (2015). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from `https://www.R-project.org/`

Van Buuren, S. (2007). Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical methods in medical research*, *16*(3), 219–242.

Walker, C., Mackenzie, M., Donovan, C., & O'Sullivan, M. (2011). Salsa–a spatially adaptive local smoothing algorithm. *Journal of Statistical Computation and Simulation*, *81*(2), 179–191.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(2), 301–320.