Estimating Technical Prices with Machine Learning Algorithms

Joakim Bilyk, Sebastian Cramer and Teis Blem *University of Copenhagen*

This document provides a practical example of applications of machine learning algorithms to car insurance data using the `mlr3` package. A popular model used in pricing of non-life insurance policies is the frequency-severity model, where the price is decomposed into the product of the probability of a claim arrises and the expected claim size given a claim occurs. This model is fitted using machine learning algorithms such as penalized linear regression and random forests. This paper argues that the ranger model gives a particular well fit to both the frequency and severity model.

Keywords: mlr3, machine learning, regression, non-life insurance, estimating technical price, XGBoost, Ranger, Bart, Elastic net regression, Generalized Additive Models

Contents

Getting familiar with the data

The data we use in this project is on the form of a table where out main objective is model the claim size Y_i given the explanatory variables X_i in the table. In particular, we want to construct an estimator that predicts an expecected claim size given the information available i.e. the quantity $\mathbb{E}[Y \mid X]$.

Missing values. As with any statistical modelling we start by doing some exploratory analysis of the data `freMPL1`. As it was seen in the previous section, the number of variables (columns) missing datapoints were only one. The one in question is `RecordEnd` which has 14143 out of 30595 missing values. This does not bather us since we have the variable `Exposure` giving the time difference between `RecordBeg` and `RecordEnd` in calendar years. Furthermore we have that `RecordBeg` ranges from 2004-01-01 to 2004-12-31 and `RecordBeg + 365.25*Exposure` being at most 2004-12-31 meaning that all contracts span within the year 2004. Assuming no seasonality trends this would incentivice us to remove the two variables `RecordBeg` and `RecordEnd`.

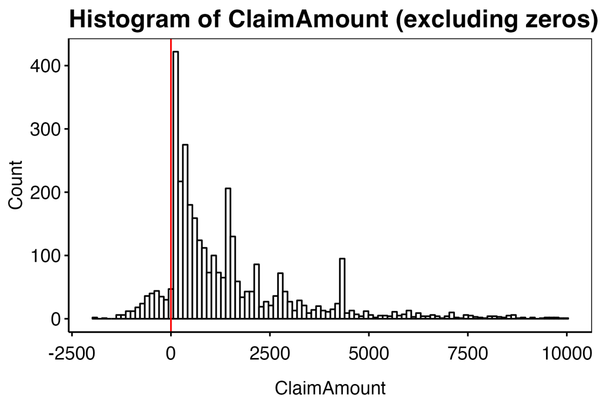
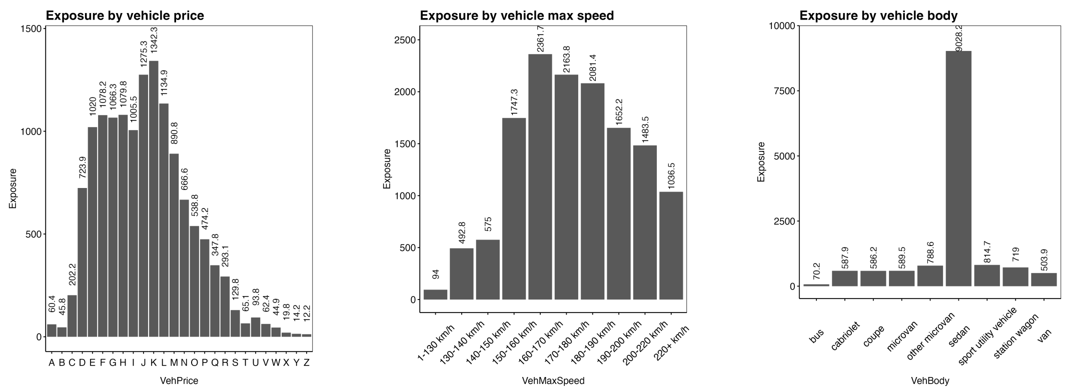


Figure 1: Histogram of the variable 'ClaimAmount'.

ClaimAmount and ClaimInd. The variable of interest, `ClaimAmount`, excibits a strange behaviour as it contains 285 strictly negative values. This is seen in figure 1. As this does not intuitively makes sense we will set these values as zero and ensure that the `ClaimInd` reflects this change.

VehEngine and VehEnergy. *(Vehicle specific, 1)* Regarding the categorical variables we notice that some levels is has sparse data. The variables `VehEngine` and `VehEnergy` has both the levels `GPL` and `electric`. We do however not have any substantial datapoints as in total these levels contain 8 observations. As the total dataset has 30595 observations in total we choose to remove these observations.



(a) (b) (c)

Figure 2: Exposure by respectively vehicle price (a), max speed (b) and body (c).

VehPrice, VehMaxSpeed and VehBody. *(Vehicle specific, 2)* In figure 2 we see that `VehPrice` is well represented in most price categories. We do however har a shorter supply of data in the tails. We will therefore combine the lowest three categories A through C, the levels R through T and lastly U through Z. Regarding to the vehicle max speed we see that a very few number of observations are in the lowest category `1-130 kmh`. We therefore combine the lowest level with the level `130-140 kmh`. The only vehicle body with very few observations is `bus` with only 159 observations. We do however see that `bus` act much like the category `sedan` with respect to frequency and severity and so these are combined under `sedan`.

VehAge, VehUsage and VehClass. *(Vehicle specific, 3)* The remaining three vehicle specific variable is well represented throughout all levels. The only scarce observation is `VehUsage` being `Professional run`. We therefore combine `Proffesional` and `Professional run` under `Professional`. We leave the remaining levels as is.

SocioCateg and Gender. *(Socails)* When constructing a statistical model, one does not simply have to consider which variables have the most explanatory value but one also have to take into consideration the lawfullness of discriminating customors based on covariates as gender, race and so forth. It is common knowledge that insurance companies cannot discriminate based on gender and so we will not use the variable `gender` as explanatory variable even though it might improve the model predictions. The variable `SocioCateg` representing the socioeconomic status of the insured ranges between category 1 and 99. It is not at the moment clear whether this is a covariate that may be used in pricing, so we will prefer not using it. However, if it does indeed improve the fit without overfitting, we may get some additional information from this variable. The data is indicate that the custumors in general are in the category 50 with a few other levels having significant more observations than others. For this reason we combine the catagories from 1 to 49 into `A` and the catagories 51 to 99 into `C` and keep the category `B` as is.

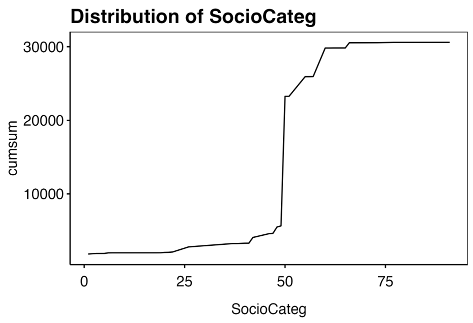


Figure 3: Cummulative distribution of the variable 'SocioCateg'.

LicAge, DrivAge and MariStat. *(Customer specific)* In general in France, one can acquire a drivers license at the age of 18 and so we have the obvious restriction `LicAge <= DrivAge - 18` and we will in general have that the license age will be approximately 18 years less than the drivers license. It is therefore reasonable to discuss whether to include both variables. We would however assume that for an older person the license age would be more important than for youngsters. Furthermore, we would assume that the `ClaimInd` and the `LicAge` are negative correlated. We will therefore include both. Both `Alone` and `Other` is well represented in `MariStat`.

HasKmLimit, RiskVar, Garage and BonusMalus *(Policy related and others)* The variables remaining HasKmLimit, RiskVar, Garage and BonusMalus are well represented throughout all levels and so no action is taken here.

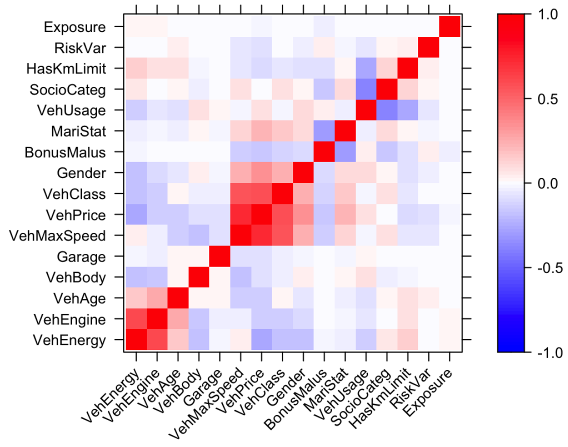


Figure 4: Spearman correlation matrix.

Correlations. As discussed previously the age of the driver and the license age i very correlated and so one may consider whether or not both variables are needed. In practice we do however see that age and experience are used when modelling the premium. Secondly, we see that the covariates vehicle engine and energy are very correlated and also the three variables vehicle max speed, price and class are well correlated. Thirdly, we see that vehicle usage and socio category are correlated. This is likely because wealthier people more often have a car for professional use.

Training a regression model

The technical premium is based upon a frequency/severity model. We have the base model assumptions as: Let $Y := Y_{t+\Delta t}$ be the claim risen by a policy during the interval $[t, t+\Delta t)$ with Δt representing the exposure. Notice that in princible Δt is a stopping time defined as

$$\Delta t := \min\big(1, \inf\{s \geq t : Y_s > 0\}\big),$$

meaning the policy is terminated at the time $t + \Delta t$ with $t + \Delta t = t + 1$ if $Y = 0$. Notice that the exposure is censored if $\Delta t_i < 1$ and $Y_i = 0$. We furthermore have the covariates $X \in \mathcal{X}$ with $\mathcal{X}$ being a p-dimensional space. We are interested in the object $\mathbb{E}[Y \mid X]$ being the expected claim risen given the covariates X. Our main assumption is that this expectation is decomposed into

$$\mathbb{E}[Y \mid X] = \mathbb{E}[Y 1_{Y>0} \mid X] = \mathbb{E}[Y \mid X, 1_{Y>0}] \cdot \mathbb{E}[1_{Y>0} \mid X] = \mu_X \cdot p_X,$$

where μ_X is the expected claim in the event, that a claim arises i.e. the severity and p_X is the probability that a claim arises i.e. the frequency. However since the data is censored with exposure $\tilde{\Delta t} \leq \Delta t$ we need to consider how we may translate a predictive model into a price function for new policies. In practice, we include exposure as a covariate.

We will search for the most efficient estimators for both the severity and frequency. We will denote these estimators by $m_\mu(X)$ and $m_p(X)$ where we will denote the model by a superscript $m_j^{(*)}$ with $*$ denoting the model for $j = \mu, p$.

Modelling severity

We model the severity by modelling `ClaimAmount ~ X_1 + X_2 + ...` for `X_i` being either a numerical variable or a 0/1 variable. As the response is numerical we choose the L^2 loss given by the loss function

$$L(y_1, y_2) = (y_1 - y_2)^2.$$

We will by default not use any other feature selection as described in the previous sections. We will start by testing which of the following five models does best under the L^2 loss

1. The generalized additive model,
2. Elastic net linear model,
3. eXtreme Gradient Boosting or short XGBoost,
4. Bayesian Additive Regression Tree or shot BART,
5. Random forests implemented through `ranger`.

Generalized Additive Model

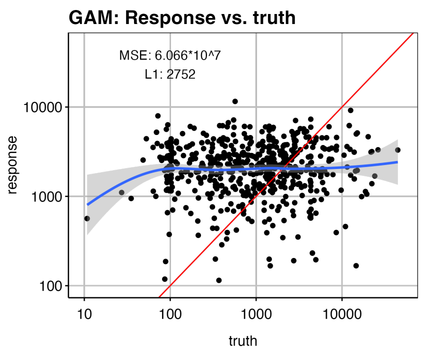
The generalized additive model we do not necessary assume that $Y \mid X$ is normal distributed and may be modeled as a linear combination of features. Instead we assume that the conditional mean satisfies the following

$$g\left(\mathbb{E}[Y \mid X]\right) = \beta_0 + \sum_{i=1}^{p} f_i(X_i) := \eta \tag{1}$$

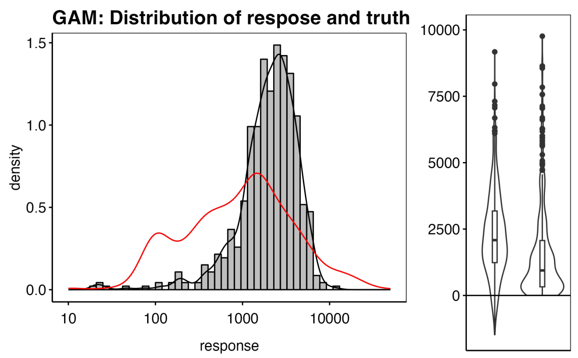
where g is the link function relating the parameter η with the mean. The functions f_i for $i = 1, ..., p$ are smoothing functions. Furthermore we in general assume that $Y \mid X \sim \mathcal{E}$ for some exponential distribution. However in this case we choose the link function $g(x) = x$ (identity) and the response distribution as gaussian, that is we assume the model

$$Y \mid X = \beta_0 + \sum_{i=1}^{p} f_i(X_i) + \varepsilon,$$

with $\varepsilon \sim \mathcal{N}(0, \sigma^2)$. For the smoothing functions we choose splines and apply them to the continuous variables `LicAge`, `BonusMalus` and `DrivAge`. The rest of the variables are indicators and so no smoother is applied. Fitting this model to the training data and predicting onto the test data yields the below pairs `(truth,response)`.



(a)



(b)

Figure 5: (a): Predictions for the GAM with x being the actual value and y being the estimate. The red line represent the mapping $y = x$. (b, left): Distribution of the response and actual values. The red density function represents the empirical density of the test data set. (b, right): A combined boxplot and violin plot of the distribution of the response (left) and the actual value (right).

One sees that the model predicts negative claims. This is obviously problematic since this would imply that the technical premium would be negative as well. Other than this, the model does a poor job in capturing the heavy tail of the true distribution. This is not a problem with the model but rather a lack of extreme value considerations.

We can check the model predictions on a couple of specific policies, namely the row numbers 6257, 25018, 30503 and 30563. Let us briefly take a look at the four policies.

Table 1: Policy covariates for policies of interest (6257, 24131, 25018, 25196, 30503 and 30563).

	Exposure	LicAge	VehAge	MariStat	SocioCateg	VehUsage	DrivAge	HasKmLimit	BonusMalus	VehBody	VehPrice	VehEngine	VehEnergy	VehMaxSpeed	VehClass	ClaimAmount	RiskVar	Garage	ClaimInd	id
6254	0.083	274	3	Alone	B	Private+trip to office	42	0	50	sedan	F	injection	regular	160-170 km/h	B	0.000	17	None	0	6257
24123	0.200	0	3	Alone	A	Private+trip to office	18	0	100	coupe	F	injection overpowered	regular	1-140 kmh	A	3067.356	16	None	1	24131
25010	0.103	1	3	Alone	A	Private+trip to office	18	0	100	sedan	G	direct injection overpowered	diesel	160-170 km/h	B	0.000	8	None	0	25018
25188	0.083	276	1	Alone	B	Private+trip to office	42	0	50	sedan	H	direct injection overpowered	diesel	160-170 km/h	B	0.000	3	None	0	25196
30495	0.052	703	8-9	Other	C	Private	89	0	50	cabriolet	M	injection	regular	180-190 km/h	M1	0.000	1	Collective garage	0	30503
30555	0.408	855	10+	Other	C	Private	89	0	50	sedan	G	injection	regular	160-170 km/h	B	0.000	12	None	0	30563

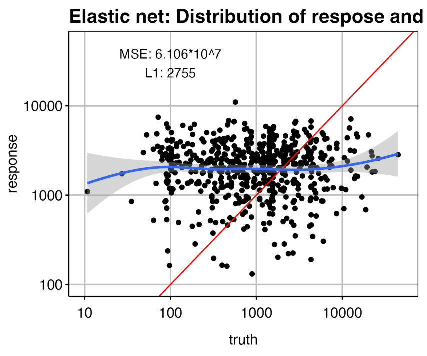
Then applying the estimator we obtain the estimates.

Estimator	$\hat{R}_{L^2}$	$\hat{R}_{L^1}$	6257	24131	25018	25196	30503	30563
$m_\mu^{(GAM)}$	$6.066 \cdot 10^7$	2752.16	2463.49	4684.1	6047.67	2263.16	2009.47	1694.34

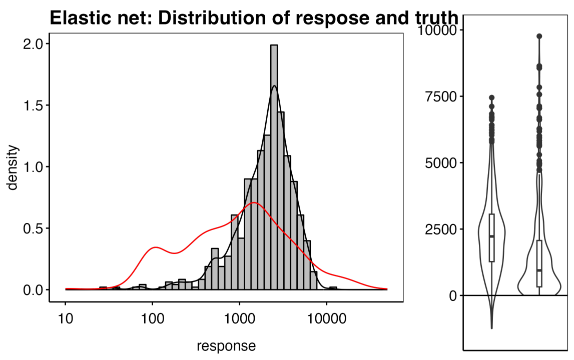
Elastic net linear model

The elastic net is a penalized linear model where the estimate is based on the algorithm

$$m_\mu^{(EN)}(X) = \left(\beta_\lambda^{EN}\right)^\top X$$



(a)



(b)

Figure 6: (a): Predictions for the elastic net regression model with x being the actual value and y being the estimate. The red line represent the mapping $y = x$. (b, left): Distribution of the response and actual values. The red density function represents the empirical density of the test data set. (b, right): A combined boxplot and violin plot of the distribution of the response (left) and the actual value (right).

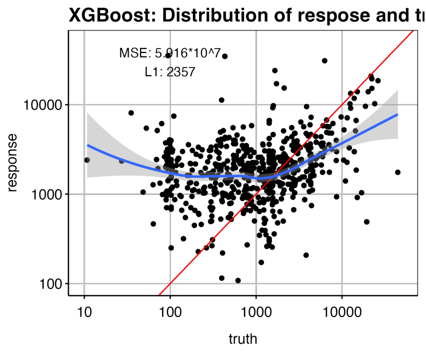
where β-parameter is determined through the minimization problem

$$\hat{\beta}^{\text{glmnet}}_{\lambda} = \arg\min_{\beta\in\mathbb{R}^p} \left\{ \hat{R}_n(\beta) + J_\lambda(\beta) \right\}, \qquad J_\lambda(\beta) = \lambda \left[\alpha \sum_{j=1}^{p} |\beta_j| + \frac{1-\alpha}{2} \sum_{j=1}^{p} \beta_j^2 \right].$$

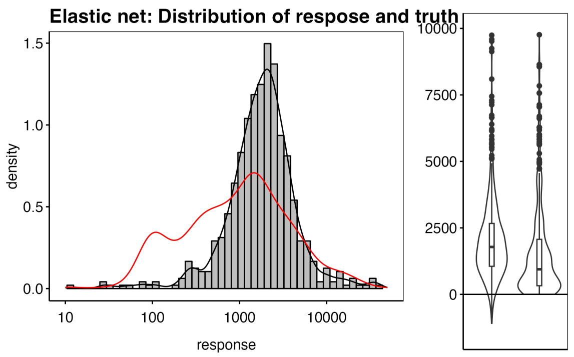
Through 5-fold cross validation we choose the parameters $\lambda = 0.205564$ and $\alpha = 0.000963$. We may consider the estimate for the policies from above.

Estimator	$\hat{R}_{L^2}$	$\hat{R}_{L^1}$	6257	24131	25018	25196	30503	30563
$m_\mu^{(EN)}$	$6.106 \cdot 10^7$	2754.79	2338.48	1843.77	3206.72	2142.26	1143.72	1322.23

XGBoost



(a)

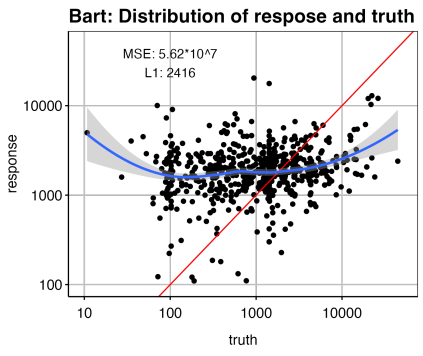


(b)

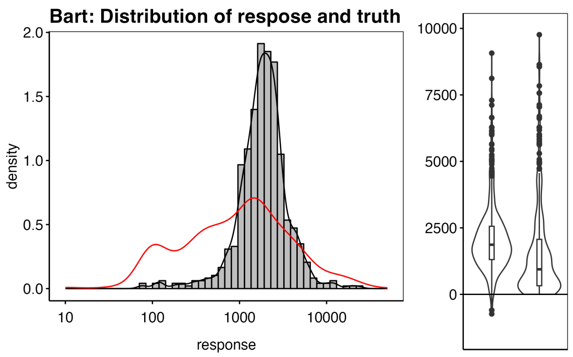
Figure 7: (a): Predictions for the XGBoost regression model with x being the actual value and y being the estimate. The red line represent the mapping $y = x$. (b, left): Distribution of the response and actual values. The red density function represents the empirical density of the test data set. (b, right): A combined boxplot and violin plot of the distribution of the response (left) and the actual value (right).

Estimator	$\hat{R}_{L^2}$	$\hat{R}_{L^1}$	6257	24131	25018	25196	30503	30563
$m_\mu^{(XGB)}$	$5.916 \cdot 10^7$	2357.03	4416.08	3226.21	6291.82	2634.91	3856.08	3239.16

BART



(a)

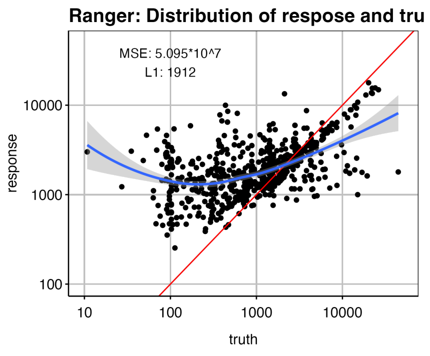


(b)

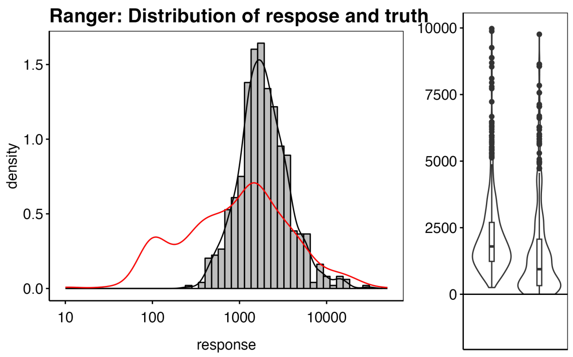
Figure 8: (a): Predictions for the BART regression model with x being the actual value and y being the estimate. The red line represent the mapping $y = x$. (b, left): Distribution of the response and actual values. The red density function represents the empirical density of the test data set. (b, right): A combined boxplot and violin plot of the distribution of the response (left) and the actual value (right).

Estimator	$\hat{R}_{L^2}$	$\hat{R}_{L^1}$	6257	24131	25018	25196	30503	30563
$m_{\mu}^{(Bart)}$	$5.62 \cdot 10^7$	2415.75	9366.62	3189.97	3488.9	2064.3	6641.56	1847.67

Ranger



(a)



(b)

Figure 9: (a): Predictions for the ranger regression model with x being the actual value and y being the estimate. The red line represent the mapping $y = x$. (b, left): Distribution of the response and actual values. The red density function represents the empirical density of the test data set. (b, right): A combined boxplot and violin plot of the distribution of the response (left) and the actual value (right).

Estimator	$\hat{R}_{L^2}$	$\hat{R}_{L^1}$	6257	24131	25018	25196	30503	30563
$m_{\mu}^{(Ranger)}$	$5.095 \cdot 10^7$	1912.16	10258.91	3956.79	5320.21	1683.94	4550.7	3248.58

Combining results

Estimator	$\hat{R}_{L^2}$	$\hat{R}_{L^1}$	6257	24131	25018	25196	30503	30563
$m_\mu^{(GAM)}$	$6.066 \cdot 10^7$	2752.16	2463.49	4684.1	6047.67	2263.16	2009.47	1694.34
$m_\mu^{(EN)}$	$6.106 \cdot 10^7$	2754.79	2338.48	1843.77	3206.72	2142.26	1143.72	1322.23
$m_\mu^{(XGB)}$	$5.916 \cdot 10^7$	2357.03	4416.08	3226.21	6291.82	2634.91	3856.08	3239.16
$m_\mu^{(Bart)}$	$5.62 \cdot 10^7$	2415.75	9366.62	3189.97	3488.9	2064.3	6641.56	1847.67
$m_\mu^{(Ranger)}$	$5.095 \cdot 10^7$	1912.16	10258.91	3956.79	5320.21	1683.94	4550.7	3248.58

We choose the Ranger model going forward because it has the lowest MLE and it seems the most precise in estimating, seen on figure 9a. This is maybe because random trees deal well with sparsity and thus the model has worked around the high correlation of the variables.

The bad performance of the GAM model shows that the response might have a very non-additive character which also speaks in favor of the Ranger model since random trees deal very poorly with additive models.

Modelling frequency

We have chosen to have exposure as an explanatory variable, since we from the plots can see a somewhat linear tendency between the exposure and Claimind. Thus the idea is that the more exposure you have the more likelihood there is of you making a claim. We did not use the weight method, because we do not want an observation with more exposure to have a bigger impact, but rather that the risk is explained by the exposure.

When we evaluate the risk of the estimators we use the log loss function given as

$$L(y,p) = -\Big(y \cdot \log p + (1-y)\cdot \log(1-p)\Big).$$

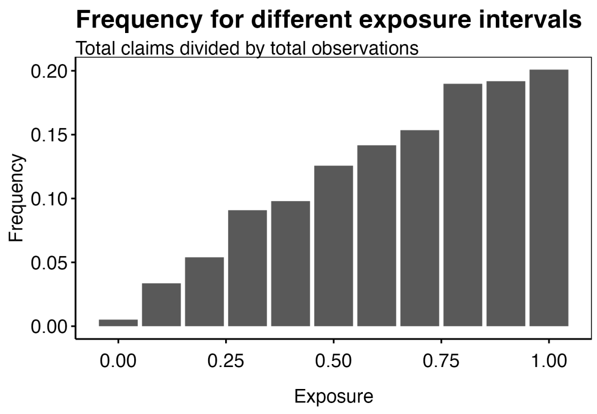


Figure 10: The figure shows that the frequency is linear in exposure.

Generalized Additive Model

We fit a Generalized additive model by fitting `ClaimInd` to the covariates. Specifically this is done by choosing the logit link function g in equation (1). The logit link function is given by

$$g(p) = \log\left(\frac{p}{1-p}\right) = \log p - \log(1-p) \in \mathbb{R}.$$

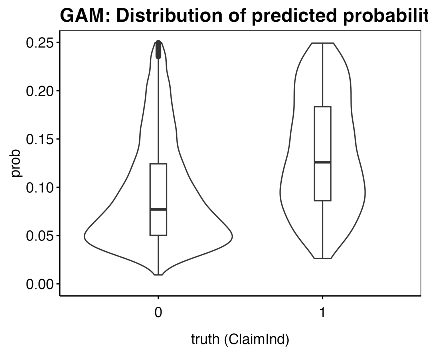


Figure 11: A combined boxplot and violin plot of the distribution of the predicted probability for respectively the non claim (left) and claim (right) policies.

We can check the model predictions on a couple of specific policies, namely the row numbers 6257, 25018, 30503 and 30563. Let us briefly take a look at the four policies.

Then applying the estimator we obtain the estimates.

Estimator	$\hat{R}_{LL}$	6257	24131	25018	25196	30503	30563
$m_p^{(GAM)}$	0.3142643	0.03748	0.0671	0.06536	0.03278	0.05206	0.08789

Elastic net linear model

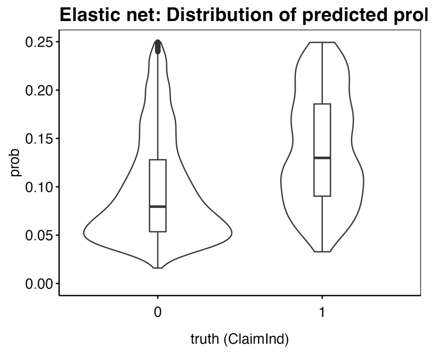


Figure 12: A combined boxplot and violin plot of the distribution of the predicted probability for respectively the non claim (left) and claim (right) policies.

We can check the model predictions on a couple of specific policies, namely the row numbers 6257, 25018, 30503 and 30563. Let us briefly take a look at the four policies.

Then applying the estimator we obtain the estimates.

Estimator	$\hat{R}_{LL}$	6257	24131	25018	25196	30503	30563
$m_p^{(EN)}$	0.3132799	0.04128	0.06164	0.06382	0.03882	0.03934	0.06055

XGBoost

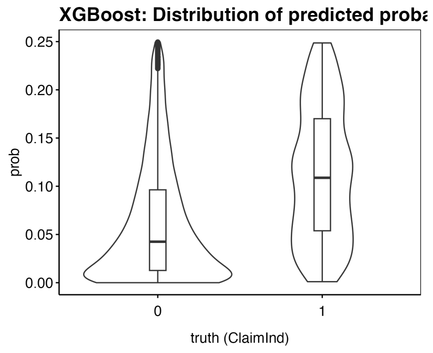


Figure 13: A combined boxplot and violin plot of the distribution of the predicted probability for respectively the non claim (left) and claim (right) policies.

We can check the model predictions on a couple of specific policies, namely the row numbers 6257, 25018, 30503 and 30563. Let us briefly take a look at the four policies.

Then applying the estimator we obtain the estimates.

Estimator	$\hat{R}_{LL}$	6257	24131	25018	25196	30503	30563
$m_p^{(XGB)}$	0.2803893	0.01441	0.7285	0.30647	0.01226	0.00022	0.00174

BART

We can check the model predictions on a couple of specific policies, namely the row numbers 6257, 25018, 30503 and 30563. Let us briefly take a look at the four policies.

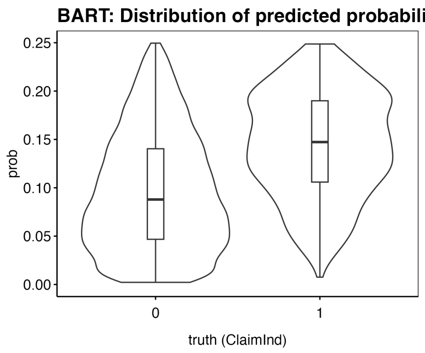


Figure 14: A combined boxplot and violin plot of the distribution of the predicted probability for respectively the non claim (left) and claim (right) policies.

Then applying the estimator we obtain the estimates.

Estimator	$\hat{R}_{LL}$	6257	24131	25018	25196	30503	30563
$m_p^{(Bart)}$	0.3093065	0.02454	0.07736	0.06091	0.02523	0.01419	0.09361

Ranger

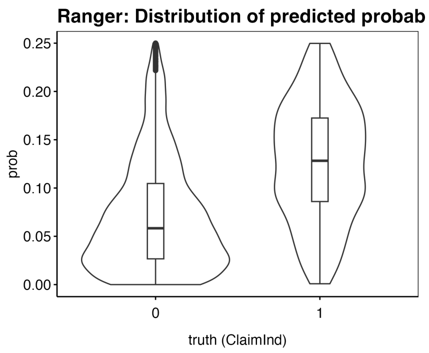


Figure 15: A combined boxplot and violin plot of the distribution of the predicted probability for respectively the non claim (left) and claim (right) policies.

We can check the model predictions on a couple of specific policies, namely the row numbers 6257, 25018, 30503 and 30563. Let us briefly take a look at the four policies.

Then applying the estimator we obtain the estimates.

Estimator	$\hat{R}_{LL}$	6257	24131	25018	25196	30503	30563
$m_p^{(Ranger)}$	0.2260974	0	0.70167	0.34536	0.00667	0.08533	0.13714

Combining results

Estimator	$\hat{R}_{LL}$	6257	24131	25018	25196	30503	30563
$m_p^{(GAM)}$	0.3142643	0.03748	0.0671	0.06536	0.03278	0.05206	0.08789
$m_p^{(EN)}$	0.3132799	0.04128	0.06164	0.06382	0.03882	0.03934	0.06055
$m_p^{(XGB)}$	0.2803893	0.01441	0.7285	0.30647	0.01226	0.00022	0.00174
$m_p^{(Bart)}$	0.3093065	0.02454	0.07736	0.06091	0.02523	0.01419	0.09361
$m_p^{(Ranger)}$	0.2260974	0	0.70167	0.34536	0.00667	0.08533	0.13714

Once again the ranger model outperforms the other models. Furthermore, it seems that the probabilities assign to the test data indicates, that the estimator correctly assigns noteworthy larger probabilities to the policies that indeed ended up with a claim. Lastly, one might want to investigate if the model predicts probabilities very close to zero. If so, we would maybe for business reasons rather penalize these probabilities while transferring the increased income as savings to the policies with a lower competitive price.

The technical price

The technical price is given by the product

$$\pi = \mu_X \cdot p_X.$$

Notice that the length of the contract i.e. the exposure is used as a covariate in the frequency/severity model, and so the effect of longer contracts is already priced in. By default new contracts are sold with a yearly length, however in the case of estimating on the dataset we use the given `Exposure`. Since the risk is the less with the ranger algorithm we estimate π by

$$\hat{\pi} = \hat{m}_{\mu}^{(Ranger)}(X) \cdot \hat{m}_{p}^{(Ranger)}(X) \cdot e.$$

We can apply the technical price onto the dataset `freMPL1` by running the following code:

```r
# Estimating mu
m_mu_ranger <- sev_ranger_tuned_regr$predict_newdata(df)$response
m_mu_ranger_2 <- df %>%
  mutate(Exposure = 1) %>%
  sev_ranger_tuned_regr$predict_newdata(.)
m_mu_ranger_2 <- m_mu_ranger_2$response
# Estimating p
m_p_ranger <- freq_ranger_tuned$predict_newdata(df_freq)$prob[,2]
m_p_ranger_2 <- df_freq %>%
  mutate(Exposure = 1) %>%
  freq_ranger_tuned$predict_newdata(.)
m_p_ranger_2 <- m_p_ranger_2$prob[,2]
# Inserting estimates
freMPL1[,"pi_hat"] <- m_mu_ranger * m_p_ranger
freMPL1[,"mu_hat"] <- m_mu_ranger
freMPL1[,"p_hat"] <- m_p_ranger
freMPL1[,"pi_hat_new"] <- m_mu_ranger_2 * m_p_ranger_2
```

We can summarize the models predictions on the policies below.

Policy	Exposure	$\hat{\mu}_X$	$\hat{p}_X$	$\hat{\pi}$
6257	0.08	10258.91	0	0
24131	0.2	3956.79	0.70167	2776.35
25018	0.1	5320.21	0.34536	1837.37
25196	0.08	1683.94	0.00667	11.23
30503	0.05	4550.7	0.08533	388.33
30563	0.41	3248.58	0.13714	445.51

If we set the exposure to one we get new estimates and the technical price the insurance firm will give to new contracts.

Policy	Exposure	$\hat{\mu}_X$	$\hat{p}_X$	$\hat{\pi}$
6257	1	10258.91	0.19805	2031.75
24131	1	3956.79	0.61167	2420.24
25018	1	5320.21	0.36131	1922.24
25196	1	1683.94	0.13167	221.72
30503	1	4550.7	0.24122	1097.73

Policy	Exposure	$\hat{\mu}_X$	$\hat{p}_X$	$\hat{\pi}$
30563	1	3248.58	0.23381	759.53

In the table above we have altered the dataset with exposure set to 1 for all rows, to predict the price of a new contract i.e. with exposure of one year. One sees that the prices for the new contracts are larger than the technical price for the historical as the exposure is larger. Furthermore as expected we see that the estimate $\hat{\mu}_X$ does not depend on the exposure.

Repreducing statement

Packages

```
library(CASdatasets)
library(lattice)
library(evmix)
library(ggplot2)
library(mlr3)
library(mlr3learners)
library(mlr3extralearners)
library(dbarts)
library(mlr3mbo)
library(mlr3measures)
library(mlr3tuning)
library(ranger)
library(mlr3viz)
library(fastDummies)
library(dplyr)
```

Statistical data

The data we will be working with is the `freMPL1` (*French Motor Personal Line datasets*) dataset from the package `CASdatasets` (*Computational Actuarial Science datasets*). The author write the following regarding the dataset

> This collection of ten datasets comes from a private motor French insurer. Each dataset includes risk features, claim amount and claim history of around 30,000 policies for year 2004.

A detailed description of the variables may be read in the below table.

Variable	Description
Exposure	The exposure, in years.
LicAge	The driving licence age, in months.
RecordBeg	Beginning date of record.
RecordEnd	End date of record.
VehAge	The vehicle age, in years.
Gender	The gender, either “Male” or “Female”.
MariStat	The marital status, either “Alone” or “Other”.
SocioCateg	The social category known as CSP in France, between “CSP1” and “CSP99”.
VehUsage	The vehicle usage among “Private”, “Private+trip to office” “Professional”, “Professional run”.
DrivAge	The driver age, in years (in France, people can drive a car at 18).
HasKmLimit	A numeric, 1 if there is a km limit for the policy, 0 otherwise.
BonusMalus	A numeric for the bonus/malus, between 50 and 350: <100 means bonus, >100 means malus in France.
VehBody	The vehicle body, among “bus”,“cabriolet”,“coupe”,“microvan”,“othermicrovan”,
VehPrice	The category of the vehicle price from “A” (cheapest) to “Z” (most expensive).

Variable	Description
VehEngine	The vehicle engine, among “carburation”,“directinjectionoverpowered”,“electric”, “GPL”, “injection”, “injection overpowered”.
VehEnergy	The vehicle energy, among “diesel”, “eletric”, “GPL”, “regular”.
VehMaxSpeed	The VehMaxSpeed, among “1-130km/h”,“130-140km/h”,“140-150km/h”,“150-160 km/h”, “160-170 km/h”, “170-180 km/h”, “180-190 km/h”, “190-200 km/h”, “200-220 km/h”, “220+ km/h”.
VehClass	The vehicle class (unknown categories), among “0”, “A”, “B”, “H”, “M1”, “M2”.
ClaimAmount	Total claim amount of the guarantee.
RiskVar	Unkonw risk variable between 1 and 20, possibly ordered.
Garage	The garage, if any, among “Collective garage”, “None”, “Private garage”.
ClaimInd	Claim indicator of the guarantee. (this is not the claim number)

Feature formatting

The following changes has been made to the data `freMPL1`:

1. The columns `RecordBeg` and `RecordEnd` are discarded.
2. The negative `ClaimAmount` entries is overwritten with zeroes and the `ClaimIndicator` is changed accordingly.
3. The observations `VehEngine` being equal to either `electric` or `GPL` is removed.
4. The levels A-C, R-T and U-Z are combined in `VehPrice`.
5. The lowest two `VehMaxSpeed` are combined.
6. The category `bus` is layed under `sedan` in the column `VehBody`.
7. The `gender` column is also discarded.
8. The column `SocioCateg` is combined into three layers: `A`, `B` and `C`.

These changes may be read in the script below.

```r
## Feature selection and formatting
data("freMPL1",package = "CASdatasets")
#Add id
freMPL1[,"id"] <- 1:dim(freMPL1)[1]

#1: RecordBeg and RecordEnd discarded
freMPL1 <- freMPL1 %>%
  select(-RecordBeg,-RecordEnd)

#2: Claim amount and indicator
freMPL1$ClaimAmount[freMPL1$ClaimAmount<0] <- 0
freMPL1$ClaimInd <- ifelse(freMPL1$ClaimAmount>0,1,0)

#3: Electric or GPL vehicals
freMPL1 <- freMPL1 %>%
  filter(!(VehEngine %in% c("electric","GPL")))

#4: Combining price categories
levels(freMPL1$VehPrice)[1:3] <- "A-C"
```

```
n <- length(levels(freMPL1$VehPrice))
levels(freMPL1$VehPrice)[(n-5):n] <- "U-Z"
n <- length(levels(freMPL1$VehPrice))
levels(freMPL1$VehPrice)[(n-3):(n-1)] <- "R-T"

#5: Combining max speed levels
levels(freMPL1$VehMaxSpeed)[1:2] <- "1-140 kmh"

#6: Bus set to sedan
levels(freMPL1$VehBody)[levels(freMPL1$VehBody) == "bus"] <- "sedan"

#7: Gender discarded
freMPL1 <- freMPL1 %>%
  select(-Gender)

#8: SocioCateg change levels
freMPL1 <- freMPL1 %>%
  #Get numerical value of SocioCateg
  mutate(helper = as.numeric(substr(SocioCateg,4,5))) %>%
  #Overwrite SocioCateg
  mutate(SocioCateg = factor(ifelse(helper > 50, "C",
                                    ifelse( helper < 50, "A",
                                            "B")),
                             levels = c("A","B","C"))) %>%
  select(-helper)
```

Preparing the dataset

We lastly format the data by creating a data frame called `data` which we will henceforth be referring to. The data frame is created with splitting the catagorical variables into indicators given whether or not a covariates is equal to a given level. For instance the `VehEnergy` column with the levels `diesel`, `eletric`, `GPL` and `regular` are split into the columns `VehEnergy_diesel`, `VehEnergy_eletric`, `VehEnergy_GPL` and `VehEnergy_regular`, where per row only one is equal to 1 and the rest equal to 0.

```
df <- freMPL1 %>%
  dummy_cols(.,select_columns = c("VehAge","MariStat","VehUsage",
                                  "VehBody","VehPrice","VehEngine",
                                  "VehEnergy","VehMaxSpeed","VehClass",
                                  "Garage","SocioCateg","RiskVar"),
             remove_selected_columns = TRUE)
colnames(df) <- make.names(colnames(df))
```

Severity models

Starting a task

We start by preparing test and training data and start a task.

```
#Get the severity data
df_sev <- df %>%
  filter(ClaimInd == 1) %>%
  select(-Exposure,-id)

#Start a task
task_sev <- df_sev %>%
```

```r
  #Make sure not to include ClaimInd
  select(-ClaimInd) %>%
  #Start tast with target ClaimAmound
  as_task_regr(.,
               target = "ClaimAmount",
               id= "Severity")

#Split data into training and test
set.seed(20230313) #We choose a seed
splits <- partition(task_sev,
                    ratio = 0.8)
df_sev_train <- df_sev[splits$train,]
df_sev_test <- df_sev[splits$test,]

#Start task on training data
task_sev_train <- as_task_regr(df_sev_train, target = "ClaimAmount")
```

Defining learners

Now we define a learner for each of the five estimators some will be tuned in the following section.

```r
### Generalized Additive Model
sev_gam_tuned_regr <- lrn("regr.gam")
indicators <- colnames(df)[!(colnames(df) %in% c("LicAge","BonusMalus",
                                                 "DrivAge","ClaimAmount",
                                                 "Exposure","ClaimInd","id"))]
sev_gam_tuned_regr$param_set$values$formula <- reformulate(c("s(LicAge)",
                                                         "s(BonusMalus)",
                                                         "s(DrivAge)",
                                                         indicators),
                                                       "ClaimAmount")
### Elastic net
sev_glmnet_learner <- lrn("regr.glmnet",
                        s= to_tune(0, 1),
                        alpha=to_tune(0, 1))

### XGBoost
sev_xgb_learner <- lrn("regr.xgboost",
                    eta = to_tune(0, 0.5),
                    nrounds = to_tune(10, 5000),
                    max_depth = to_tune(1, 3))

### BART
sev_bart_learner <- lrn("regr.bart",
                        ntree = to_tune(73,400))

### RANGER
sev_ranger_learner <- lrn("regr.ranger",
                        mtry.ratio = to_tune(0.1,1),
                        min.node.size = to_tune(1, 50),
                        num.trees = 50)
```

Estimating hyperparameters

We then use 5-fold cross validation to tune the parameters in the four models.

```r
#### Elastic net
set.seed(20230313) #We choose a seed
# 1: Estimating hyperparameters with 5-fold cross validation
sev_glmnet_learner_instance <- tune(
  #method = tnr("random_search"), ### tuning method
  method = mlr3tuning::tnr("mbo"), ### tuning method
  task = task_sev_train,
  learner = sev_glmnet_learner,
  resampling = rsmp("cv", folds = 5), #### resampling method: 5-fold cross validation
  measures = msr("regr.mse"), #### root mean squared error
  terminator = trm("evals", n_evals = 200) #### terminator
)
#Save the instance as it require a lot of
#computational time
saveRDS(sev_glmnet_learner_instance, file = "rds/sev_glmnet_learner_instance.rds")
# 2: Define new tuned learner
sev_glmnet_tuned_regr <- lrn("regr.glmnet")
sev_glmnet_tuned_regr$param_set$values <- sev_glmnet_learner_instance$result_learner_param_vals

### XGBoost
set.seed(20230313) #We choose a seed
# 1: Estimating hyperparameters with 5-fold cross validation
sev_xgb_learner_instance = tune(
  #method = tnr("random_search"), ### tuning method
  method = mlr3tuning::tnr("mbo"), ### tuning method
  task = task_sev_train,
  learner = sev_xgb_learner,
  resampling = rsmp("cv", folds = 5), #### resampling method: 5-fold cross validation
  measures = msr("regr.mse"), #### root mean squared error
  terminator = trm("evals", n_evals = 200) #### terminator
)
#Save the instance
saveRDS(sev_xgb_learner_instance, file = "rds/sev_xgb_learner_instance.rds")
# 2: Define new tuned learner
sev_xgb_tuned_regr <- lrn("regr.xgboost")
sev_xgb_tuned_regr$param_set$values <- sev_xgb_learner_instance$result_learner_param_vals

### BART
set.seed(20230313) #We choose a seed
# 1: Estimating hyperparameters with 5-fold cross validation
sev_bart_learner_instance = tune(
  #method = tnr("random_search"), ### tuning method
  method = mlr3tuning::tnr("mbo"), ### tuning method
  task = task_sev_train,
  learner = sev_bart_learner,
  resampling = rsmp("cv", folds = 5), #### resampling method: 5-fold cross validation
  measures = msr("regr.mse"), #### root mean squared error
  terminator = trm("evals", n_evals = 200)
)
#Save the instance
saveRDS(sev_bart_learner_instance, file = "rds/sev_bart_learner_instance.rds")
```

```r
# 2: Define new tuned learner
sev_bart_tuned_regr <- lrn("regr.bart")
sev_bart_tuned_regr$param_set$values <- sev_bart_learner_instance$result_learner_param_vals

### RANGER
#Estimate hyperparameters
set.seed(20230313) #We choose a seed
# 1: Estimating hyperparameters with 5-fold cross validation
sev_ranger_learner_instance = tune(
  #method = tnr("random_search"), ### tuning method
  method = mlr3tuning::tnr("mbo"), ### tuning method
  task = task_sev_train,
  learner = sev_ranger_learner,
  resampling = rsmp("cv", folds = 5), #### resampling method: 5-fold cross validation
  measures = msr("regr.mse"), #### root mean squared error
  terminator = trm("evals", n_evals = 200) #### terminator
)
#Save the instance
saveRDS(sev_ranger_learner_instance, file = "rds/sev_ranger_learner_instance.rds")
# 2: Define new tuned learner
sev_ranger_tuned_regr <- lrn("regr.ranger")
sev_ranger_tuned_regr$param_set$values <- sev_ranger_learner_instance$result_learner_param_vals
```

We may list the estimates in the following table.

Hyperparamater	**Search space**	**Estimate**
Elastic net		
`s`	$[0, 1]$	0.205564
`alpha`	$[0, 1]$	0.000963
XGBoost		
`eta`	$[0, 0.5]$	0.246407
`nrounds`	$[10, 5000]$	379
`max_depth`	$[1, 3]$	3
BART		
`ntree`	$[73, 400]$	267
Ranger		
`mtry.ratio`	$[0.1, 1]$	0.56226
`min.node.size`	$[0.1, 1]$	1

Fitting the models

```r
### Generalized Additive Model
sev_gam_tuned_regr$train(task_sev_train)

#### Elastic net
sev_glmnet_tuned_regr$train(task_sev_train)

### XGBoost
sev_xgb_tuned_regr$train(task_sev_train)
```

```r
### BART
sev_bart_tuned_regr$train(task_sev_train)

### RANGER
sev_ranger_tuned_regr$train(task_sev_train)
```

Predicting

```r
### Generalized Additive Model
sev_gam_predictions <- sev_gam_tuned_regr$predict_newdata(df_sev_test)
sev_gam_mse <- mean((sev_gam_predictions$response-sev_gam_predictions$truth)^2)
sev_gam_L1 <- mean(abs(sev_gam_predictions$response-sev_gam_predictions$truth))

#### Elastic net
sev_glmnet_predictions <- sev_glmnet_tuned_regr$predict_newdata(df_sev_test)
sev_glmnet_mse <- mean((sev_glmnet_predictions$response-sev_glmnet_predictions$truth)^2)
sev_glmnet_L1 <- mean(abs(sev_glmnet_predictions$response-sev_glmnet_predictions$truth))

### XGBoost
sev_xgb_predictions <- sev_xgb_tuned_regr$predict_newdata(df_sev_test)
sev_xgb_mse <- mean((sev_xgb_predictions$response-sev_xgb_predictions$truth)^2)
sev_xgb_L1 <- mean(abs(sev_xgb_predictions$response-sev_xgb_predictions$truth))

### BART
sev_bart_predictions <- sev_bart_tuned_regr$predict_newdata(df_sev_test)
sev_bart_mse <- mean((sev_bart_predictions$response-sev_bart_predictions$truth)^2)
sev_bart_L1 <- mean(abs(sev_bart_predictions$response-sev_bart_predictions$truth))

### RANGER
sev_ranger_predictions <- sev_ranger_tuned_regr$predict_newdata(df_sev_test)
sev_ranger_mse <- mean((sev_ranger_predictions$response-sev_ranger_predictions$truth)^2)
sev_ranger_L1 <- mean(abs(sev_ranger_predictions$response-sev_ranger_predictions$truth))
```

Frequency models

Starting a task

We start by preparing test and training data and start a task.

```r
#Get the frequency data
df_freq <- df %>%
  mutate(ClaimInd = factor(ClaimInd, levels = c(0,1))) %>%
  #Make sure not to include ClaimAmount
  select(-ClaimAmount,-id)

#Start a task
task_freq <- df_freq %>%
  #Start tast with target ClaimInd
  as_task_classif(.,
                 target = "ClaimInd",
                 id= "Frequency")

#Split data into training and test
set.seed(20230313) #We choose a seed
```

```r
splits <- partition(task_freq,
                    ratio = 0.8)
df_freq_train <- df_freq[splits$train,]
df_freq_test <- df_freq[splits$test,]

#Start task on training data
task_freq_train <- as_task_classif(df_freq_train,
                                   target = "ClaimInd",
                                   id= "Frequency_train")
```

Defining learners

Now we define a learner for each of the five estimators some will be tuned in the following section.

```r
### Generalized Additive Model
freq_gam_tuned <- lrn("classif.gam",
                      predict_type = "prob")
indicators <- colnames(df)[!(colnames(df) %in% c("LicAge","BonusMalus","DrivAge",
                                                 "ClaimAmount","ClaimInd","id"))]
freq_gam_tuned$param_set$values$formula <- reformulate(c("s(LicAge)",
                                                         "s(BonusMalus)",
                                                         "s(DrivAge)",
                                                         indicators),
                                                       "ClaimInd")
### Elastic net
freq_glmnet_learner <- lrn("classif.glmnet",
                          s = to_tune(0, 1),
                          alpha = to_tune(0, 1),
                          predict_type = "prob")

### XGBoost
freq_xgb_learner <- lrn("classif.xgboost",
                        eta = to_tune(0, 0.5),
                        nrounds = to_tune(10, 5000),
                        max_depth = to_tune(1, 3),
                        predict_type = "prob")

### BART
freq_bart_learner <- lrn("classif.bart",
                        ntree = to_tune(73,400),
                        predict_type = "prob")

### RANGER
freq_ranger_learner <- lrn("classif.ranger",
                      mtry.ratio = to_tune(0.1,1),
                      min.node.size = to_tune(1, 50),
                      num.trees = 50,
                      predict_type = "prob")
```

Estimating hyperparameters

We then use 5-fold cross validation to tune the parameters in the four models.

```r
#### Elastic net
set.seed(20230313) #We choose a seed
# 1: Estimating hyperparameters with 5-fold cross validation
freq_glmnet_learner_instance <- tune(
  #method = tnr("random_search"), ### tuning method
  method = mlr3tuning::tnr("mbo"), ### tuning method
  task = task_freq_train,
  learner = freq_glmnet_learner,
  resampling = rsmp("cv", folds = 5), #### resampling method: 5-fold cross validation
  measures = msr("classif.logloss"), #### root mean squared error
  terminator = trm("evals", n_evals = 200) #### terminator
)
#Save the instance as it require a lot of
#computational time
saveRDS(freq_glmnet_learner_instance, file = "rds/freq_glmnet_learner_instance.rds")
# 2: Define new tuned learner
freq_glmnet_tuned <- lrn("classif.glmnet",
                         predict_type = "prob")
freq_glmnet_tuned$param_set$values <- freq_glmnet_learner_instance$result_learner_param_vals

### XGBoost
set.seed(20230313) #We choose a seed
# 1: Estimating hyperparameters with 5-fold cross validation
freq_xgb_learner_instance = tune(
  #method = tnr("random_search"), ### tuning method
  method = mlr3tuning::tnr("mbo"), ### tuning method
  task = task_freq_train,
  learner = freq_xgb_learner,
  resampling = rsmp("cv", folds = 5), #### resampling method: 5-fold cross validation
  measures = msr("classif.logloss"), #### root mean squared error
  terminator = trm("evals", n_evals = 100) #### terminator
)
#Save the instance
saveRDS(freq_xgb_learner_instance, file = "rds/freq_xgb_learner_instance.rds")
# 2: Define new tuned learner
freq_xgb_tuned <- lrn("classif.xgboost",
                      predict_type = "prob")
freq_xgb_tuned$param_set$values <- freq_xgb_learner_instance$result_learner_param_vals

### BART
set.seed(20230313) #We choose a seed
# 1: Estimating hyperparameters with 5-fold cross validation
freq_bart_learner_instance = tune(
  #method = tnr("random_search"), ### tuning method
  method = mlr3tuning::tnr("mbo"), ### tuning method
  task = task_freq_train,
  learner = freq_bart_learner,
  resampling = rsmp("cv", folds = 5), #### resampling method: 5-fold cross validation
  measures = msr("classif.logloss"), #### root mean squared error
  terminator = trm("evals", n_evals = 75)
)
#Save the instance
saveRDS(freq_bart_learner_instance, file = "rds/freq_bart_learner_instance.rds")
```

```
# 2: Define new tuned learner
freq_bart_tuned <- lrn("classif.bart",
                                predict_type = "prob")
freq_bart_tuned$param_set$values <- freq_bart_learner_instance$result_learner_param_vals

### RANGER
#Estimate hyperparameters
set.seed(20230313) #We choose a seed
# 1: Estimating hyperparameters with 5-fold cross validation
freq_ranger_learner_instance = tune(
  #method = tnr("random_search"), ### tuning method
  method = mlr3tuning::tnr("mbo"), ### tuning method
  task = task_freq_train,
  learner = freq_ranger_learner,
  resampling = rsmp("cv", folds = 5), #### resampling method: 5-fold cross validation
  measures = msr("classif.logloss"), #### root mean squared error
  terminator = trm("evals", n_evals = 200) #### terminator
)
#Save the instance
saveRDS(freq_ranger_learner_instance, file = "rds/freq_ranger_learner_instance.rds")
# 2: Define new tuned learner
freq_ranger_tuned <- lrn("classif.ranger",
                          predict_type = "prob")
freq_ranger_tuned$param_set$values <- freq_ranger_learner_instance$result_learner_param_vals
```

We may list the estimates in the following table.

Hyperparamater	**Search space**	**Estimate**
Elastic net		
s	$[0, 1]$	0.001227
alpha	$[0, 1]$	0.923369
XGBoost		
eta	$[0, 0.5]$	0.196547
nrounds	$[10, 5000]$	1747
max_depth	$[1, 3]$	3
BART		
ntree	$[73, 400]$	380
Ranger		
mtry.ratio	$[0.1, 1]$	0.100018
min.node.size	$[0.1, 1]$	3

Fitting the models

```
### Generalized Additive Model
freq_gam_tuned$train(task_freq_train)

#### Elastic net
freq_glmnet_tuned$train(task_freq_train)
```

```r
### XGBoost
freq_xgb_tuned$train(task_freq_train)

### BART
freq_bart_tuned$train(task_freq_train)

### RANGER
freq_ranger_tuned$train(task_freq_train)
```

Predicting

```r
### Generalized Additive Model
freq_gam_predictions <- freq_gam_tuned$predict_newdata(df_freq_test)
ll_loss <- function(y,p) {
  p <- ifelse(p == 1, 1-0.0000001,p)
  p <- ifelse(p==0, 0.0000001, p)
  return(-( y * log(p) + (1 - y)*log(1-p)))
}
freq_gam_ll <- mean(
  ll_loss(
    y = as.numeric(as.character(predictions$truth)),
    p = predictions$prob[,2]
  )
)

#### Elastic net
freq_glmnet_predictions <- freq_glmnet_tuned$predict_newdata(df_freq_test)
freq_glmnet_ll <- mean(
  ll_loss(
    y = as.numeric(as.character(freq_glmnet_predictions$truth)),
    p = freq_glmnet_predictions$prob[,2]
  )
)

### XGBoost
freq_XGB_predictions <- freq_xgb_tuned$predict_newdata(df_freq_test)
freq_XGB_ll <- mean(
  ll_loss(
    y = as.numeric(as.character(freq_XGB_predictions$truth)),
    p = freq_XGB_predictions$prob[,2]
  )
)

### BART
freq_bart_predictions <- freq_bart_tuned$predict_newdata(df_freq_test)
freq_bart_ll <- mean(
  ll_loss(
    y = as.numeric(as.character(freq_bart_predictions$truth)),
    p = freq_bart_predictions$prob[,2]
  )
)

### RANGER
```

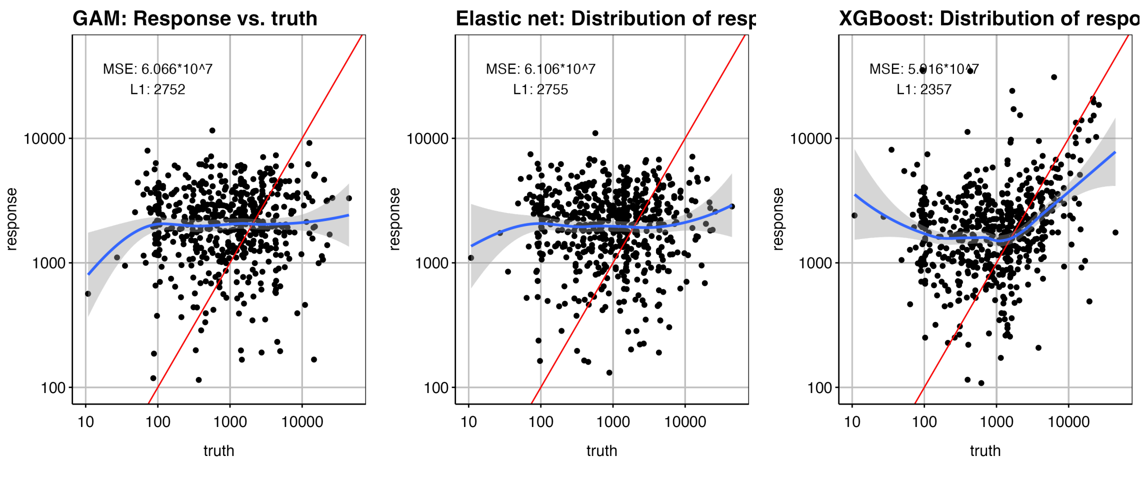
```r
freq_ranger_predictions <- freq_ranger_tuned$predict_newdata(df_freq_test)
freq_ranger_ll <- mean(
  ll_loss(
    y = as.numeric(as.character(freq_ranger_predictions$truth)),
    p = freq_ranger_predictions$prob[,2]
  )
)
```

Estimating the technical price

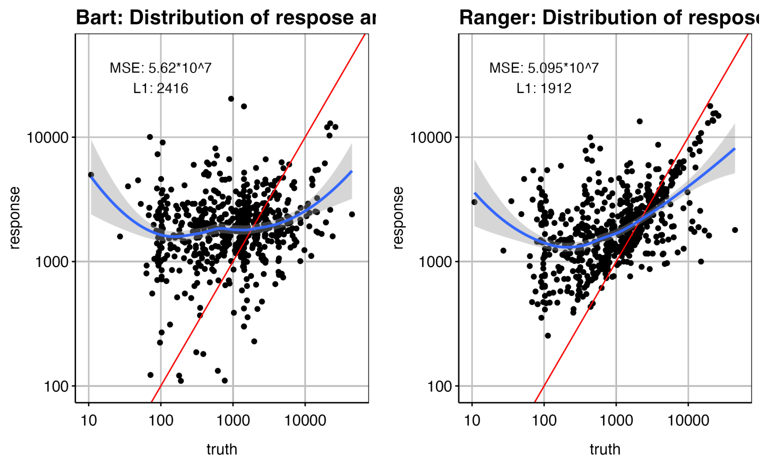
```r
# Estimating mu
m_mu_ranger <- sev_ranger_tuned_regr$predict_newdata(df)$response
m_mu_ranger_2 <- df %>%
  mutate(Exposure = 1) %>%
  sev_ranger_tuned_regr$predict_newdata(.)
m_mu_ranger_2 <- m_mu_ranger_2$response
# Estimating p
m_p_ranger <- freq_ranger_tuned$predict_newdata(df_freq)$prob[,2]
m_p_ranger_2 <- df_freq %>%
  mutate(Exposure = 1) %>%
  freq_ranger_tuned$predict_newdata(.)
m_p_ranger_2 <- m_p_ranger_2$prob[,2]
# Inserting estimates
freMPL1[,"pi_hat"] <- m_mu_ranger * m_p_ranger
freMPL1[,"mu_hat"] <- m_mu_ranger
freMPL1[,"p_hat"] <- m_p_ranger
freMPL1[,"pi_hat_new"] <- m_mu_ranger_2 * m_p_ranger_2
```

Figures

Comparing the severity models



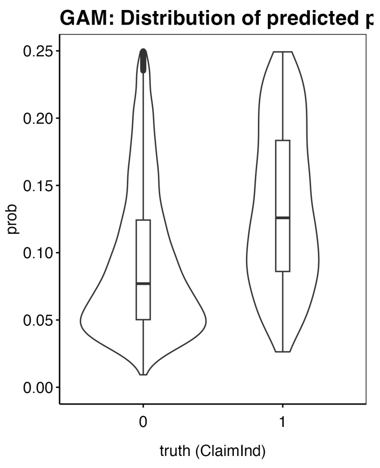
(a) (b) (c)



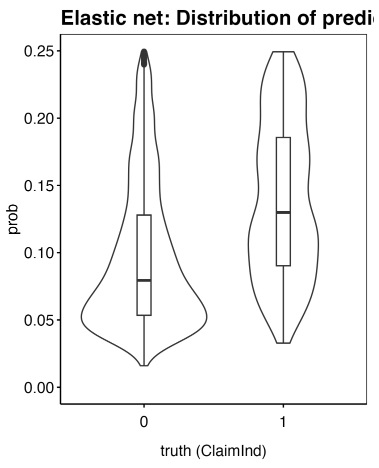
(d) (e)

Figure 16: Predictions for the five algorithms with x being the actual value and y being the estimate. The red line represent the mapping $y = x$.

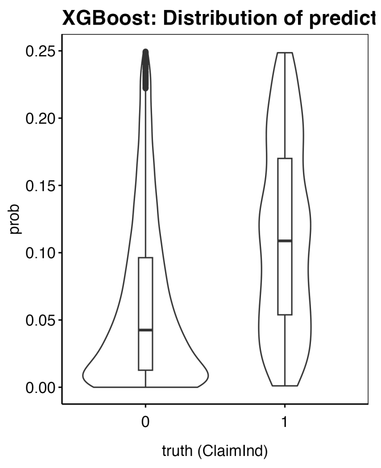
Comparing the frequency models



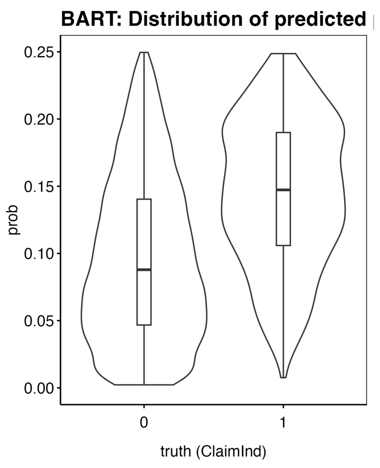
(a)



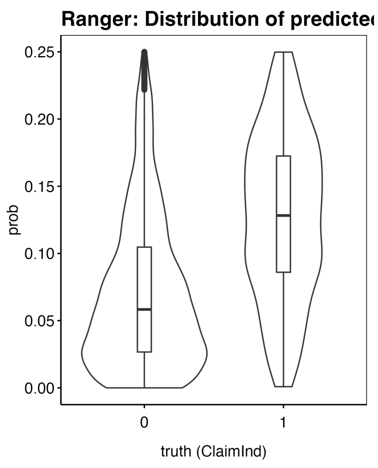
(b)



(c)



(d)



(e)

Figure 17: Combined violin and boxplot of the predicted probabilities for non-claim and claim policies.