Parametric and XGBoost Hurdle Model for estimating accident frequency

Dafnis Krasniqi, Jean-Marc Bardet, Joseph Rynkiewicz

Parametric and XGBoost Hurdle Model for estimating accident frequency

Dafnis Krasniqi [1,2], Jean-Marc Bardet [1] and Joseph Rynkiewicz [1]

[1]SAMM, Universit Paris 1 Panthon-Sorbonne, France
[2]Allianz France

Abstract

A well known and used method for modeling count data from exogenous variables is based on a Poisson model. Nevertheless, it has many shortcomings when faced with the problem of counting data with an excess of zeros. In this article, we focus on the Hurdle model as an alternative model. We will analyze its properties and build it as well with parametric and non-parametric estimates, notably with an XGBoost method. These new models will then be applied to a car insurance portfolio of a French insurance company, in which the number of annual accidents per driver presents a significant excess of zero accident. We show that the performance of the Hurdle model improved by XGBoost estimation is superior to that of several alternative models.

Keywords— Insurance pricing; XGBoost algorithm; GLM; Hurdle Model

1 Introduction

A standard Poisson distribution is often used to model count data as a function of exogenous variables. But when the response variable y, is mostly zero valued, the mean and variance of the response variable are not identical and generally the variance is larger than the mean, resulting of an overdispersion Berk and Macdonald (2008). Therefore, the standard Poisson model is no longer suitable for this kind of data. To solve this overdispersion problem, it is advisable to change the regression model.
Many models can handle this problem. We can quote for example the negative Binomial distribution (Gamma-Poisson distribution mixture) (Hilbe (2011), Cameron and Trivedi (2005), Cameron and Trivedi (1990)), the Zero-Inflated model (Lambert (1992), Greene (1994), Cameron and Trivedi (1999)) and the Hurdle model (Mullahy (1986), Heilbron (1994)). In our study, we will focus exclusively on the Hurdle model since we can assume that all zeros come from a "structural" source. Rougthly speaking, a Hurdle model can be described as follows: a first binary random variable (depending on the exogenous variables) will decide whether the response variable y is zero or positive, while a positive integer valued random variable (depending on the exogenous variables), following for example a truncated Poisson or negative binomial distribution, will model for positive data. In other terms, a Hurdle model is a mixture of two probability distribution, a Dirac measure at 0 and a positive integer valued probability distribution. Many papers have been published on this topic and in various disciplines (Rudra and Biswas (2019), Bohara and Krieg (1996)). In the field of insurance, we can mention, for example, Boucher, Denuit, and Guillén (2008b), Boucher, Denuit, and Guillén (2007b). They studied these models to calculate premiums on the Spanish insurance market's real data and obtained excellent results. We can also quote Zhang, Pitt, and Wu (2022).

The objective of this study is to improve the usual Hurdle models for cound data with exogenous variables. Usually, these models are built from parametric or semi-parametric estimations, *i.e.* the binomial choice as well as the positive distribution, depend on a finite dimensional parameter. Here, we will propose to use non-parametric distributions such as decision trees or boosting models built from the exogenous variables.

In general, classical or parametric statistical models require certain assumptions and have limitations, such as equal error variances, taking into account a default distribution for the response variables, a linear relationship between the dependent and independent variables, which, in real data, may not be available. In addition, most of these approaches lack the ability to model complex, sophisticated, non-linear relationships and high degree interactions. For all these reasons, we will focus on statistical learning models.

Hurdle models built from learning models already exist in the literature. We can cite two examples. Povak et al. (2013) proposed a Hurdle model for pollution detection in rivers. Their Hurdle model fits particularly well the data they deal with because many rivers are not affected by pollution and therefore many zeros appear in the target variable. Hence, a learning models based on random forests is applied. Kong et al. (2020) proposed a

Hurdle model to detect species in nature. The problem is that many species are not classified, which makes the database unbalanced. Their Hurdle model is built from two neural networks.

In the field of insurance, several authors used an Hurdle model to analyze the number of claims declared by an insured driver. We can quote Boucher, Denuit, and Guillén (2008a) or Boucher, Denuit, and Guillén (2007a). They showed that the Hurdle model is an interesting alternative to the classical Poisson or negative binomial models. We can also quote Zhang, Pitt, and Wu (2022), which developed a multivariate Hurdle model optimized by the expectation-maximization (EM) algorithm. In all these insurance papers, the models are built with parametric models. To our knowledge, there are no other papers using Hurdle models based on non-parametric estimation on insurance data.

This study is presented as follows: in Section 2, the classical Hurdle model is presented as well as the parametric estimations. We will show that this Hurdle model can be modeled into two independent models without loss of information. In Section 3, non-parametric estimation based on machine learning methods will be considered, and prediction will be developed in Section 4. In the last section, Section 5, the models are applied to a real dataset, a car insurance portfolio of a French insurance company.

2 The Hurdle model

In parametric models, the Hurdle model is well known for predicting count data (another name is "the two-part model"). The Hurdle model is a modified counting model containing two processes, one generating zeros and the other generating positive integers. The concept behind the Hurdle model is that a Bernoulli random variable with parameter depending on the exogenous variables governs the binary outcome, *i.e.* whether a count variable has a zero or positive value. A positive integer valued distribution governs, depending again on the exogenous variables, governs the conditional distribution of positive values. The two models are assumed to be independent of each other. Here there is a fundamental difference with the zero-inflated Poisson (ZIP) model because the latter is considered a mixture model a Poisson distribution and a Dirac measure at 0, implying that both the distribution are not independent.
In this section, as a first approach, a classical parametric Hurdle model is presented, as well as the estimation of its parameters, and the selection of the best model.

2.1 Statistical models

Problem Statement Let $\mathcal{D}$ be a data set defined:

$$\mathcal{D} = \{(x_i, y_i)_{1\leq i\leq n},\ \text{with } x_i \in \mathbb{R}^p, y_i \in \{0,1,2,..\} \text{ for } 1 \leq i \leq n\}.$$

The exogenous variables are represented by a $\mathbb{R}^p$-vector x and is supposed to explain the response variable y. Our goal is to propose a model to predict this variable y from x and we begin with a Poisson Hurdle regression model defined by:

$$Pr\left[Y_i = y_i\right]_{\{1\leq i\leq n\}} = \begin{cases} \pi_i & \text{if } y_i = 0 \\ (1-\pi_i)\dfrac{\frac{\lambda_i^{y_i}}{e^{\lambda_i} y_i!}}{1-e^{-\lambda}} = (1-\pi_i)\dfrac{\lambda_i^{y_i}}{(e^{\lambda_i}-1)y_i!} & \text{if } y_i > 0 \end{cases} \tag{1}$$

In the equation (1), $\frac{1-\pi_i}{1-e^{-\lambda}}$; can be interpreted as the probability of crossing the obstacle. A crucial problem in modeling count data is the appropriate choice of link functions. The main example of a parametric choice is the following: the parameter π_i is built from a *logit* link and the parameter λ_i from a log link.

Bernoulli Model	Poisson zero truncated Model
$\text{Logit}(\pi_i) = x_i^T\beta_1$	$\log(\lambda_i) = x_i^T\beta_2$
$\pi_i = \frac{e^{x_i^T\beta_1}}{1-e^{x_i^T\beta_1}}$	$\lambda_i = e^{x_i^T\beta_2}$

where β_1 and β_2 are the regression coefficients for the covariates x_i, respectively.

Note that the expectation of this model is obtained as follows:

$$\begin{aligned} E(Y_i) &= (1-\pi_i)E(Y_i \mid Y_i > 0) \\ &= (1-\pi_i)\frac{\lambda_i}{1-e^{-\lambda_i}} \end{aligned} \tag{2}$$

and its variance is:

$$\begin{aligned} Var(Y_i) &= (1-\pi_i)Var(Y_i \mid Y_i > 0) + \pi_i(1-\pi_i)\left[E(Y_i \mid Y_i > 0)\right]^2 \\ &= (1-\pi_i)\left[\frac{\lambda_i}{1-e^{-\lambda_i}} + \left(\pi_i - e^{-\lambda_i}\right)\left(\frac{\lambda_i}{1-e^{-\lambda_i}}\right)^2\right] \end{aligned}$$

The Hurdle models therefore take into account the overdispersion since the variance is greater than the expectation ($\mathbb{E}[Y] \leq Var[Y]$).

2.2 Estimation of the parameters

To estimate the parameters β_1 and β_2 of the model (1), a maximum likelihood estimation can be used. With the hypothesis that $Y_1, \ldots, Y_n$ are independent, the log-likelihood can be written:

$$\log(L(\beta_1, \beta_2, y)) = \sum_{i \in \Omega_0} \log\left(\frac{e^{x_i^T \beta_1}}{1 - e^{x_i^T \beta_1}}\right) + \sum_{i \in \Omega_1} \log\left(1 - \frac{e^{x_i^T \beta_1}}{1 - e^{x_i^T \beta_1}}\right) + \sum_{i \in \Omega_1} \log\left(\frac{\left(e^{x_i^T \beta_2}\right)^{y_i}}{y_i!(e^{e^{x_i^T \beta_2}} - 1)}\right) \tag{3}$$

with $\Omega_0 = \{i | y_i = 0\}$, $\Omega_1 = \{i | y_i \neq 0\}$.

The log-likelihood of this model is separated into two parts

$$\log(L(\beta_1, \beta_2, y)) = \log(L_1(\beta_1, y)) + \log(L_2(\beta_2, y)).$$

We obtain maximum likelihood estimates by separately maximizing $L1$ and $L2$.

$$\log(L_1(\beta_1, y)) = \sum_{i \in \Omega_0} \log\left(\frac{e^{x_i^T \beta_1}}{1 - e^{x_i^T \beta_1}}\right) + \sum_{i \in \Omega_1} \log\left(1 - \frac{e^{x_i^T \beta_1}}{1 - e^{x_i^T \beta_1}}\right)$$

$$\begin{aligned} \log(L_2(\beta_2, y)) &= \sum_{i \in \Omega_1} \log\left(\frac{\left(e^{x_i^T \beta_2}\right)^{y_i}}{y_i!(e^{e^{x_i^T \beta_2}} - 1)}\right) \\ &= \sum_{i \in \Omega_1} y_i \log\left(e^{x_i^T \beta_2}\right) - \left(\sum_{i \in \Omega_1} \log(y_i!) + \sum_{i \in \Omega_1} \log\left(e^{e^{x_i^T \beta_2}} - 1\right)\right) \end{aligned} \tag{4}$$

As a result,

$$\hat{\beta_1} = \underset{\beta_1 \in \mathbb{R}^{p+1}}{\operatorname{argmax}} \log(L_1(\beta_1, y)) \qquad \hat{\beta_2} = \underset{\beta_2 \in \mathbb{R}^{p+1}}{\operatorname{argmax}} \log(L_2(\beta_2, y))$$

However, such an estimator does not have an analytical form. Thus, to get a point estimate of β, iterative algorithms are used, in particular the Newton-Raphson or the gradient algorithms.

2.3 Model selection

There are several techniques to choose the best combination of numerical and categorical exogenous variables. The AIC (see Akaike (1973)) or BIC (see Schwarz (1978)) criteria can be used as model selection with

$$AIC = -2 * \log(L(\hat{\beta_1}, \hat{\beta_2}, y)) + 2 * k$$

$$BIC = -2 * \log(L(\hat{\beta_1}, \hat{\beta_2}, y)) + \log(n) * k$$

with k the number of estimated parameters and n the sample size.

3 XGBoost Hurdle model

As we have seen, two link functions (logistic and logarithmic) involving the exogenous variables and the parameters β_1 and β_2 were used for the parametric Hurdle model. However, such parametric model is often not enough powerful to take account of the complexity of the data. A possible improvement is to replace these parametric link functions by nonparametric functions, which can be independently optimized on the data (see equation (3)). And we chose to use the stochastic algorithm XGBoost to independently optimize a logistic loss function and a Poisson zero truncated loss function.

3.1 XGBoost

XGBoost method was introduced in Chen and Guestrin (2016) and Sommer, Sarigiannis, and Parnell (2019). It is based on successive construction of **m** binary trees. Each tree learns and improves the previous one. These so-called weak models (simple trees) are boosted to obtain a more powerful model. These tree optimization methods with gradient descent were introduced by Breiman in the '90s and perfected later by Friedman (2000). The first difference between XGBoost and other tree-based methods is that it is computationally optimized because the data are processed in compressed blocks, which allows them to be ordered much faster and processed in parallel. The second difference is that XGBoost method uses the second-order Taylor expansion of the objective function to find the minimum. As an example, consider the objective function of the XGBoost model at the t-th iteration:

$$L^{(t)} = \sum_{i=1}^{n} \ell(y_i, \hat{y}^{(t-1)} + f(x, \alpha_t)) + \Omega(f(x, \alpha_t))$$

where ℓ is the loss function, $f(x, \alpha_t)$ the output of the t-th tree, α_t the parameters concerning the t tree, and Ω is the regularization. This regularization can take the form of the L1 or L2 regularization or a linear combination of both.

One of the (many) critical steps for a fast computation is the approximation:

$$L^{(t)} \approx \sum_{i=1}^{n} \ell(y_i, \hat{y}^{(t-1)}) + g_i f(x_i, \alpha_t) + \frac{1}{2} h_i f^2(x_i, \alpha_t) + \Omega(f(x_i, \alpha_t))$$

with $g_i = \left[\frac{d}{d\hat{y}} \ell(y_i, \hat{y})\right]$ and $h_i = \left[\frac{d^2}{d\hat{y}^2} \ell(y_i, \hat{y})\right]$.

The second-order Taylor approximation is easy to compute because most of the terms are the same as in a given iteration. For a given iteration, the expression can be computed once and reused as a constant for all splits:

$$L^{(t)} \approx \sum_{i=1}^{n} \underbrace{\ell(y_i, \hat{y}^{(t-1)})}_{constant} + \underbrace{g_i}_{constant} f(x_i, \alpha_t) + \frac{1}{2} \underbrace{h_i}_{constant} f^2(x_i, \alpha_t) + \Omega(f(x_i, \alpha_t))$$

So, the only thing left to calculate is $f(x_i, \alpha_t)$ and $\Omega(f(x_i, \alpha_t))$.

3.2 Part 1: Binary XGBoost

The first model is a statistical learning model with a loss function inspired by the log-likelihood of logistic regression. To build such a binary model, it is necessary to slightly modify the initial data. In particular, it is required to define a new target variable, which is defined as follows:

$$y_i^* \mapsto \begin{cases} 1 & \text{if} y_i \in \Omega_1, \\ 0 & \text{if} y_i \in \Omega_0. \end{cases}$$

where $\Omega_0 = \{i | y_i = 0\}$ and $\Omega_1 = \{i | y_i \neq 0\}$

To learn on binary data, let's define the following model; the loss function ℓ_1 associated with this problem can be written as follows:

$$\ell_1(y_i^*; f(x_i, \alpha_t)) = -\frac{1}{n} \left[\sum_{i}^{n} y_i^* \times \log(f(x_i, \alpha_t)) + (1 - y_i^*) \times \log(1 - f(x_i, \alpha_t)) \right]$$

Where ℓ_1 is the loss function and $f(x_i, \alpha_t)$ is the output of the t-th tree. To compute a probability, the model will sum all the trees and put them into a sigmoid function:

$$\pi^{ML}(x_i) = \frac{e^{\sum_{m=1}^{M} \beta_m f(x_i, \alpha_m)}}{1 + e^{\sum_{m=1}^{M} \beta_m f(x_i, \alpha_m)}}. \tag{5}$$

With the result of (5), we obtain values between 0 and 1 ($\pi_i^{ML} \in [0, 1]$) that can be interpreted as a probability: π_i^{ML} and $1 - \pi_i^{ML}$ are respectively is the probability to belong to class 0 and 1.

3.3 Part 2: XGBoost of zero-truncated Poisson distribution

The second part of XGBoost Hurdle model is a zero-truncated Poisson model, which is devoted to predict the positive number of accidents. Here again we consider a change of the response variable y. The original target variable y will be truncated by 0, only the positive values will be kept. The new target variable for this model is defined as follows:

$$\tilde{y}_i = y_i, \text{ for } i \in \Omega_1$$

with $\Omega_1 = \{i | y_i \neq 0\}$.

Concerning the database X, we will also remove some rows. For example, $x_{/0}$ is the database without the rows that have a value for the target variable equal to 0. To learn about this data, the second loss function will be defined.

$$\ell_2(\tilde{y}_i; f(x_i, \alpha_t)) = -\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i \times \log(f(x_i, \alpha_t)) - f(x_i, \alpha_t)) \tag{6}$$

Where ℓ_2 is the loss function, $f(x_i, \alpha_t)$ is the output of the t-th tree.

This loss function is available by default in the XGBoost model when a target variable y following a Poisson distribution. Unfortunately, this loss function (6) does not work in our case because a loss function that corresponds to the log-likelihood of the zero-truncated Poisson distribution is needed. To this end, a new loss function will be coded in XGBoost.

3.3.1 New loss function

To construct the new loss function, we start from the maximum likelihood of the zero-truncated Poisson distribution (4). In XGBoost we have to minimize functions, so the negative of the function (4) will be used. In addition, the parametric part will be replaced ($\lambda_i = e^{x_i\beta_2}$) by a non-parametric part ($\lambda_i = f(x_i, \alpha_m)$) and create a loss function dedicated to the boosting model.

$$\ell_2(\tilde{y}_i; f(x_i, \alpha_t)) = -\frac{1}{|\Omega_1|}\sum_{i\in\Omega_1}(\tilde{y}_i \times \log(f(x_i, \alpha_t)) - \log(e^{f(x_i,\alpha_t)} - 1)) \tag{7}$$

where ℓ_2 is the loss function, $f(x_i, \alpha_t)$ is the output of the t-th tree and $\Omega_1 = \{i|y_i \neq 0\}$.

To implement a new loss function in XGBoost, 3 ingredients are needed: the first derivative of the equation (7), the second derivative of the equation (7) and a function to evaluate. For the first derivative of (7) is equal to :

$$\frac{\partial \ell_2(\tilde{y}_i; f(x_i, \alpha_t))}{\partial f(x_i, \alpha_t)} = -\frac{\tilde{y}_i}{f(x_i, \alpha_t)} + \frac{e^{f(x_i,\alpha_t)}}{e^{f(x_i,\alpha_t)} - 1}$$

and the second derivative of (7) is :

$$\frac{\partial^2 \ell_2(\tilde{y}_i; f(x_i, \alpha_t))}{\partial f(x_i, \alpha_t)^2} = \frac{\tilde{y}_i}{f(x_i, \alpha_t)^2} - \frac{e^{f(x_i,\alpha_t)}}{(e^{f(x_i,\alpha_t)} - 1)^2}$$

For the evaluation function, the equation (7) will be directly used. The implementation of these functions in Python is available in the appendix. As for the output result, it is the expectation of the zero-truncated Poisson distribution: $E[Y_i] = \frac{\lambda_i e^{\lambda_i}}{e^{\lambda_i}-1}$. The output value of our boosting model will be as follows:

$$\lambda^{ML}(x_i) = \frac{\sum_{m=1}^{M}\beta_m f(x_i, \alpha_m) \times e^{\sum_{m=1}^{M}\beta_m f(x_i,\alpha_m)}}{e^{\sum_{m=1}^{M}\beta_m f(x_i,\alpha_m)} - 1} \tag{8}$$

with the result of (8), values greater than 1 are obtained.

3.4 Advantages and disadvantages of statistical learning models

Unlike parametric models, statistical learning models are generally not built using a finite number of parameters; no assumption on the function and the distribution of random variables are made. In the parametric model, selecting the appropriate variables for the models is necessary. In statistical learning such as the XGBoost method, all variables will be put in both models, and they will select the best variables for each tree built by minimizing the objective function.

A first drawback of statistical learning methods, particularly XGBoost methods, is they require a lot of parameters that need to be perfectly tuned. To help us in this task, a Python library called *Optuna* will be used (Akiba et al. (2019)). This library aims to find the best parameters randomly for each model. The only thing the user has to do is give a range of parameters for the model to be tested, and the library will do the necessary by performing random tests. But the library also learns during the search phase and will gradually try parameters that decrease the objective function.

A second drawback of machine leaning method such as XGBoost is the overlearning the data. Indeed, the goal of a statistical learning method is to provide a model that generalizes well over all the data and not just the training data. To contain this problem, an early stop will be implemented if the score on the training set continues to decrease during the construction phase while the score on the validation set increases.

4 The prediction strategy

After defining the two sub-models, the prediction strategy of this new XGBoost Hurdle model can be specified. The simplest strategy to follow is to multiply the two predictions of our two models as it is already done for

parametric Hurdle model (inspired by the expectation (2)). The prediction $\hat{y}_{n+1}$ obtained from a new vector x_{n+1} is defined as follows:

$$\hat{y}_{n+1} = \lfloor \hat{\mu}_{n+1} \rceil \quad \text{with} \quad \hat{\mu}_{n+1} = (1 - \pi^{ML}(x_{n+1})) \times \lambda^{ML}(x_{n+1})$$

where $\pi^{ML}(x_{n+1})$ is the predicted probability of the XGBoost binary model (see (5)), $\lambda^{ML}(x_{n+1})$ is the predicted number of the XGBoost zero-truncated Poisson (see (8)) and $\lfloor x \rceil$ represents the nearest integer at x. The $\hat{y}_{n+1}$ and the $\hat{\mu}_{n+1}$ will be used in different metrics.

5 Application on insurance data

This final section will first briefly describe the data and explain why they are highly relevant to our situation. Next, a description of the different measures used to judge the quality of our results will be given. Finally, the algorithms designed by us will be applied to specific insurance data and their results will be analyzed.

5.1 The data

To make a numerical illustration of our different models, the FreMTPL2freq database will be used. It contains 678,013 automobile liability contracts (see Christophe Dutang ("R-Package CASDatasets")). This dataset is very frequently used in automobile insurance and has been the subject of many studies (Noll, Salzmann, and Wuthrich (2018), Denuit, Charpentier, and Trufin (2021)). Here is the list of variables and their descriptions in the table below.

Table 1: The features in *FreMTPL2freq*

Features	Description	Type
ClaimNb	the number of accidents (between 0 and 5)	continuous
Exposure	Exposure (in fraction of years)	continuous
BonusMalus	Bonus/malus, between 50 and 350	continuous
Power	the power of the car (from 4 to 48)	continuous
CarAge	the age of the vehicle (in years)	continuous
DriverAge	the age of the driver (in years)	continuous
Brand	the brand of the car	categorical
Gas	Gas of the car: "Diesel" or "Regular"	categorical
Region	the regions (in France)	categorical
Density	number of inhabitants per km2	continuous
Area	The density value of the community	categorical

Table 2: Distribution of the variables *ClaimNb* and *Exposure*

ClaimNb	0	1	2	3	4	5	6
Nombre	643,953	32,178	1,784	82	7	2	1
Sum *Exposure*	336,616.1	20,670.8	1,153.4	52.8	3.1	1.1	0.3

The variable y_i corresponds to the number of accidents ($ClaimNB$) suffered by the driver i during the year. This variable is a discrete variable $\{0, 1, 2, ...\}$. The *Exposure* variable e_i corresponds to the period of the insured's underwriting and his exposure to the insurance risk. This variable is essential in the models as it plays the role of a weighting coefficient. It is clear that having 2 accidents in 6 months ($e_i = 0.5$) and 2 accidents in 1 year ($e_i = 1$) is not the same thing. This exposure variable will therefore aim to harmonize contracts with unequal subscription periods.

5.2 Experiment parameters and optimization

Several restatements on the variables are done to facilitate the learning of our models. This step aims to obtain a clean database without errors and extreme values. Two tasks are performed:

1. The claims will be capped at three in the target variable *ClaimNb* because, beyond that number, accidents become extremely rare and unpredictable. Drivers with more than 3 accidents will be reduced to 3;
2. The different algorithms do not accept categorical variables so we will transform them into continuous variables. Our database only has nominal qualitative variables, so the numerical encoding has no hierarchy to respect but only reproduces the distinction between the modalities. With this method, a matrix with

the modality as column name for each qualitative variable will be obtained. Before carrying out this transformation and to avoid having matrices that are too empty (a lot of 0's), it is advisable to do some grouping of modalities. For quantitative variables, no transformation will be performed. We can introduce them directly into the different algorithms.

Five algorithmes are applied to the freMTPL2freq data: GLM Poisson, Hurdle, Decision Tree, Poisson Boosting Machine (PBM) with deep 3 and 5 and ML-Hurdle. The decision tree and the Poisson Boosting Machine are deduced from the article by

Noll, Salzmann, and Wuthrich (2018)
[1].

So, there are two parametric models and three non-parametric models. Concerning the database, it is divided into two. we will have a training base (80% of the total base) and a testing base (20% of the total base). When the nonparametric methods are applied, the training database is divided into two to obtain a validation database to adjust the hyperparameters.

For each method, a Monte Carlo Cross-Validation with K = 30 is performed. This method allows us to provide robust results. For more details, the different types of cross-validation are presented in great detail in the thesis of Cornec (2009).

To optimize the parametric Poisson GLM model, the stepwise approach is used to minimize the BIC with the library *glm* on R. The parameters of the regression are given in the appendix (15). Then, the Hurdle is optimized with the *pscl* library on R with the same approach. To optimize the Decision Tree and the Poisson Boosting Machine (PBM) Noll, Salzmann, and Wuthrich (2018) use the library *rpart* on R. For the ML-Hurdle model, XGBoost and Optuna library will be used to perform our modeling. The final results of the optimization of this model are available in appendix (18, 19, 20 and 21).

5.2.1 Comparisons of the methods

To compare our models, we will use three different measures:

Poisson deviance:

$$d(y_i;\hat{\mu}_i) = \sum_i^n 2y_i \left[\log\left(\frac{y_i}{\hat{\mu}_i}\right) - 1 + \frac{\hat{\mu}_i}{y_i}\right] \times 100$$

The comparison between the predictions of our model and the observed events:

$$d(y_i;\hat{\mu}_i) = \left[\frac{\sum_i^n \hat{\mu}_i - \sum_i^n y_i}{\sum_i^n y_i}\right] \times 100$$

The confusion matrix:

To perform the construction of this matrix, the $\hat{y}_i$ will be used.

Table 3: C-M

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	...	$\hat{y}_i = n$
$y_i = 0$	..	..	..	..
$y_i = 1$	..	..	..	..
...	..	..	..	..
$y_i = n$	..	..	..	..

5.3 Results

In this section, the results of the different algorithms will be analyzed. The results of each model will be compared on a test and training basis.

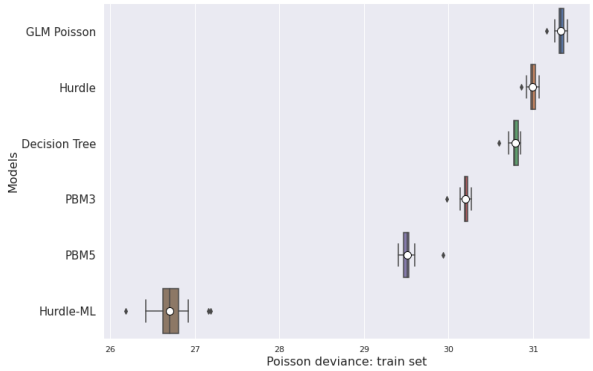
In table (4), the results for deviance and the difference between predictions and observed in percentage are presented. In this table, three models of Noll, Salzmann, and Wuthrich (2018) are also included. The authors of this paper seek only to optimize the Poisson deviance but neither the confusion matrices nor the difference between what they predict and what is observed. We used their open-source code to compute the two missing criteria.

Focusing on Poisson deviance, the $Hurdle - ML$ model gets the best average score with 26.70 on the training set and 29.88 on the test set. The Hurdle model with parametric models (average scores of 30.99 on the training

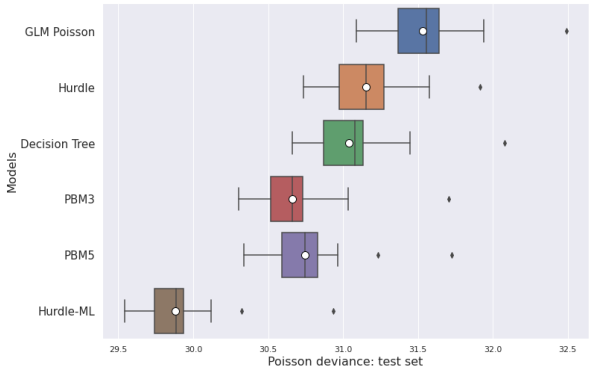
[1] the codes of these two models are available in the following GitHub see here

Table 4: Rsultats

	P. deviance train	P. deviance test	$\neq$ accidents train	$\neq$ accidents test
GLM Poisson	31.33	31.53	7.34e-08	-0.71
Hurdle	30.99	31.15	-1.18e+00	-1.85
Decision Tree	30.79	31.04	-1.68e-02	-0.68
PBM3	30.20	30.66	1.22e-04	-0.61
PBM5	29.51	30.74	2.67e-05	-0.40
Hurdle-ML	26.71	29.88	3.92e-01	-0.38



(a) Poisson deviance for the different models on the training set



(b) Poisson deviance for the different models on the test set

Figure 1: Poisson deviance on the training and test set in both boxplots.

set and 31.15 on the test set) or the GLM Poisson (average scores of 31.33 on the training set and 31.51 on the test set) are far from these results. They fail to be as accurate as of the statistical learning models. They are not able to distinguish between good and bad drivers. Concerning the methods developed in Noll, Salzmann, and Wuthrich (2018), they are more accurated than the parametric models, but less than the Hurdle-ML model. This is also clearly visible in the two graphs 1a and 1b.

Table 5: M-C GLM Poisson; train

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	547056	311	1	0
$y_i = 1$	27240	86	1	0
$y_i = 2$	1493	21	0	0
$y_i = 3$	83	0	0	0

Table 6: M-C GLM Poisson; test

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	96509	53	0	0
$y_i = 1$	4836	16	0	0
$y_i = 2$	267	3	0	0
$y_i = 3$	15	0	0	0

Table 7: M-C Hurdle; train

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	547368	0	0	0
$y_i = 1$	27326	0	0	0
$y_i = 2$	1514	0	0	0
$y_i = 3$	83	0	0	0

Table 8: M-C Hurdle; test

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	96562	0	0	0
$y_i = 1$	4852	0	0	0
$y_i = 2$	270	0	0	0
$y_i = 3$	15	0	0	0

Looking at the confusion matrices for the test data, we should not forget that we are faced with a complicated problem: working on a counting model with extremely unbalanced data (see table 2). First, the two parametric models are found to perform poorly. They fail to detect potentially dangerous drivers accurately. The GLM can only detect 16 dangerous drivers on average. The worst performing model is the Hurdle model with the parametric models. The latter classifies all drivers as $\hat{y} = 0$. Thus, all predictions of the Hurdle model will be between 0 and 0.5. This model fails to detect drivers with very high risks. Next, the two models of Noll, Salzmann, and Wuthrich (2018) (PMB3 trees),are much better than the parametric models. They manage to detect more potentially dangerous drivers. Among these two models, the best is the Poisson Boosting Machine (see (12)).

Table 9: M-C Decision trees; train

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	546554	814	0	0
$y_i = 1$	27146	180	0	0
$y_i = 2$	1505	9	0	0
$y_i = 3$	83	0	0	0

Table 10: M-C Decision trees; test

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	96422	140	0	0
$y_i = 1$	4820	32	0	0
$y_i = 2$	268	2	0	0
$y_i = 3$	15	0	0	0

Table 11: M-C PBM3; train

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	546402	963	3	0
$y_i = 1$	26934	392	0	0
$y_i = 2$	1489	25	0	0
$y_i = 3$	81	2	0	0

Table 12: M-C PBM3; test

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	96382	180	1	0
$y_i = 1$	4784	68	1	0
$y_i = 2$	266	4	0	0
$y_i = 3$	15	0	0	0

Table 13: M-C ML-Hurdle; train

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	546810	563	0	0
$y_i = 1$	26186	1147	0	0
$y_i = 2$	1426	78	0	0
$y_i = 3$	79	2	0	0

Table 14: M-C ML-Hurdle; test

	$\hat{y}_i = 0$	$\hat{y}_i = 1$	$\hat{y}_i = 2$	$\hat{y}_i = 3$
$y_i = 0$	96389	168	0	0
$y_i = 1$	4728	117	0	0
$y_i = 2$	275	5	0	0
$y_i = 3$	17	0	0	0

Finally, the Hurdle ML had the best performance on the test set. It manages to detect 117 potentially dangerous drivers (see (14)). But these confusion matrices (10, 12, or 14) also show that false positives are predicted and some "secure" drivers are assigned with a positive number of accidents. The algorithm classifies them this way for two reasons. These drivers have all the characteristics of drivers who have accidents but have had nothing. These drivers have indeed had accidents during the year, but they did not report them to their insurance company because they prefer to pay them in cash.

6 Conclusion

Throughout this study, we aimed to show the accuracy of a XGBoost Hurdle model for zero excess count data. This method is built using the XGBoost library by coding a new loss function. Four other methods (two parametric and two non-parametric models) are also studies. The XGBoost Hurdle method performs better on all criteria than the other methods.

One of the shortcomings of the XGBoost Hurdle method (and of the other methods tested) is that it cannot find claims greater than 1, i.e., very dangerous drivers. There are several ways to improve such methods. For example, we can put double Hurdles or even consider a cascade structure to solve the data imbalance problem.

Acknowledgements

I would like to thank all the staff of Allianz France for their assistance and for the financing.

References

Akaike, Hirotogu (1973). "Information Theory and an Extension of the Maximum Likelihood Principle". In: *Selected Papers of Hirotugu Akaike*. New York, NY: Springer New York, pp. 199–213.

Akiba, Takuya et al. (2019). "Optuna: A Next-generation Hyperparameter Optimization Framework". In: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Berk, Richard A. and John Macdonald (2008). "Overdispersion and Poisson Regression". In: *Journal of Quantitative Criminology* 24, pp. 269–284.

Bohara, Alok K. and Randall G. Krieg (1996). "A Poisson Hurdle Model of Migration Frequency". In: *The Journal of Regional Analysis and Policy* 26, pp. 37–45.

Boucher, Jean P., Michel Denuit, and Montserrat Guillén (2007a). "Risk Classification for Claim Counts". In: *North American Actuarial Journal* 11.4, pp. 110–131. DOI: 10.1080/10920277.2007.10597487. eprint: https://doi.org/10.1080/10920277.2007.10597487. URL: https://doi.org/10.1080/10920277.2007.10597487.

— (2008a). "Modelling of Insurance Claim Count with Hurdle Distribution for Panel Data". In:

Boucher, Jean-Philippe, Michel Denuit, and Montserrat Guillén (2007b). "Risk Classification for Claim Counts". In: *North American Actuarial Journal* 11.4, pp. 110–131. DOI: 10.1080/10920277.2007.10597487. eprint: https://doi.org/10.1080/10920277.2007.10597487. URL: https://doi.org/10.1080/10920277.2007.10597487.

— (2008b). "Modelling of Insurance Claim Count with Hurdle Distribution for Panel Data". In: *Advances in Mathematical and Statistical Modeling*. Boston: Birkhäuser Boston, pp. 45–59. ISBN: 978-0-8176-4626-4. DOI: 10.1007/978-0-8176-4626-4_4. URL: https://doi.org/10.1007/978-0-8176-4626-4_4.

Cameron, A. and Pravin Trivedi (Sept. 1999). "Regression analysis of count data. 2nd ed". In: vol. 41. DOI: 10.1017/CBO9780511814365.

Cameron, A. Colin and Pravin K. Trivedi (2005). *Microeconometrics: Methods and Applications*. Cambridge University Press. DOI: 10.1017/CBO9780511811241.

Cameron, A.Colin and Pravin K. Trivedi (1990). "Regression-based tests for overdispersion in the Poisson model". In: *Journal of Econometrics* 46.3, pp. 347–364. ISSN: 0304-4076. DOI: https://doi.org/10.1016/0304-4076(90)90014-K. URL: https://www.sciencedirect.com/science/article/pii/030440769090014K.

Chen, Tianqi and Carlos Guestrin (2016). *XGBoost: A Scalable Tree Boosting System*. cite arxiv:1603.02754Comment: KDD'16 changed all figures to type1. DOI: 10.1145/2939672.2939785. URL: http://arxiv.org/abs/1603.02754.

Christophe Dutang, Arthur Charpentier. "R-Package CASDatasets". In: (). URL: https://www.openml.org/search?type=data&sort=runs&id=41214&status=active.

Cornec, Matthieu (June 2009). "Probability bounds for the cross-validation estimate in the context of the statistical learning theory and statistical models applied to economics and finance". Theses. Université de Nanterre - Paris X. URL: https://pastel.archives-ouvertes.fr/tel-00530876.

Denuit, Michel, Arthur Charpentier, and Julien Trufin (2021). *Autocalibration and Tweedie-dominance for Insurance Pricing with Machine Learning*. arXiv: 2103.03635 [stat.ML].

Friedman, Jerome H. (2000). "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* 29, pp. 1189–1232.

Greene, William (1994). *Accounting for Excess Zeros and Sample Selection in Poisson and Negative Binomial Regression Models*. Working Papers. New York University, Leonard N. Stern School of Business, Department of Economics. URL: https://EconPapers.repec.org/RePEc:ste:nystbu:94-10.

Heilbron, David C (1994). "Zero-altered and other regression models for count data with added zeros". In: *Biometrical Journal* 36.5, pp. 531–547.

Hilbe, Joseph M. (2011). *Negative Binomial Regression*. 2nd ed. Cambridge University Press. DOI: 10.1017/CBO9780511973420.

Kong, Shufeng et al. (2020). *Deep Hurdle Networks for Zero-Inflated Multi-Target Regression: Application to Multiple Species Abundance Estimation*. arXiv: 2010.16040 [cs.LG].

Lambert, Diana (1992). "Zero-inflated poisson regression, with an application to defects in manufacturing". In: *Technometrics* 24.1, pp. 1–14.

Mullahy, John (1986). "Specification and testing of some modified count data models". In: *Journal of Econometrics* 33.3, pp. 341–365. URL: https://EconPapers.repec.org/RePEc:eee:econom:v:33:y:1986:i:3:p:341-365.

Noll, Alexander, Robert Salzmann, and Mario V. Wuthrich (2018). "Case Study: French Motor Third-Party Liability Claims". In: *Innovation Practice eJournal*.

Povak, Nicholas A et al. (2013). "Machine learning and hurdle models for improving regional predictions of stream water acid neutralizing capacity". In: *Water Resources Research* 49.6, pp. 3531–3546.

Rudra, Sujan and Soma Chowdhury Biswas (2019). "MODELS FOR ANALYZING OVER-DISPERSED HURDLE NEGATIVE BINOMIAL REGRESSION MODEL: AN APPLICATION TO MANUFACTURED CIGARETTE USE". In: *Journal of Reliability and Statistical Studies*.

Schwarz, Gideon (Mar. 1978). "Estimating the Dimension of a Model". In: *The Annals of Statistics* 6.2, pp. 461–464. DOI: 10.1214/aos/1176344136. URL: https://doi.org/10.1214%2Faos%2F1176344136.

Sommer, Johanna, Dimitrios Sarigiannis, and Thomas P. Parnell (2019). "Learning to Tune XGBoost with XGBoost." In: *CoRR* abs/1909.07218. URL: http://dblp.uni-trier.de/db/journals/corr/corr1909.html#abs-1909-07218.

Zhang, Pengcheng, David Pitt, and Xueyuan Wu (2022). "A NEW MULTIVARIATE ZERO-INFLATED HURDLE MODEL WITH APPLICATIONS IN AUTOMOBILE INSURANCE". In: *ASTIN Bulletin*, 124. DOI: 10.1017/asb.2021.39.

Appendices

Zero-truncated poisson loss function

```
def first_grad(predt, dtrain):
    '''Compute the first derivative.'''
    y = dtrain.get_label() if isinstance(dtrain, xgb.DMatrix) else dtrain
    return (-y/predt) +(np.exp(predt)/(np.exp(predt)-1))

def second_grad(predt, dtrain):
    '''Compute the second derivative.'''
    y = dtrain.get_label() if isinstance(dtrain, xgb.DMatrix) else dtrain
    return (y/predt**2 - (np.exp(predt)/(np.exp(predt)-1)**2))

def PoissonZeroTruncatedNegLogLik(predt, dtrain):
    ''''Poisson Zero truncated error function.'''
    predt[predt < 0] = 0 + 1e-6
    grad = first_grad(predt, dtrain)
    hess = second_grad(predt, dtrain)
    return grad, hess

def EvalPoissonZeroTruncatedNegLogLik(preds: np.ndarray, dtrain: xgb.DMatrix)
                                        -> Tuple[str, float]:
    ''''Poisson Zero Truncated Evaluation function.'''
    actuals = dtrain.get_label()
    preds[preds < 0] = 1e-6
    resultats = -actuals * np.log(preds) + np.log(np.exp(preds)-1)
    return "PoissonZeroTrunc-eval", float(np.sum(resultats)/len(preds))

booster =xgb.train(params,
                   dtrain=dmat_train,
                   num_boost_round=1000,
                   obj=PoissonZeroTruncatedNegLogLik,
                   feval=EvalPoissonZeroTruncatedNegLogLik,
                   evals=[(dmat_train, 'dtrain'), (dmat_valid, 'dtest')],
                   early_stopping_rounds=10,
                   evals_result=results)
```

Table 15: Glm Poisson

	Dependent variable:
	ClaimNb
VehPowerGLM5	0.192*** (0.020)
VehPowerGLM6	0.215*** (0.020)
VehPowerGLM7	0.136*** (0.019)
VehPowerGLM8	−0.073** (0.029)
VehPowerGLM9	0.214*** (0.022)
VehAgeGLM1	1.160*** (0.018)
VehAgeGLM3	−0.201*** (0.015)
DrivAgeGLM1	0.036 (0.047)
DrivAgeGLM2	−0.316*** (0.029)
DrivAgeGLM3	−0.420*** (0.024)
DrivAgeGLM4	−0.292*** (0.017)
DrivAgeGLM6	−0.068*** (0.016)
DrivAgeGLM7	0.013 (0.024)
BonusMalusGLM	0.023*** (0.0004)
VehBrandB10	0.032 (0.040)
VehBrandB11	0.109** (0.043)
VehBrandB12	0.135*** (0.019)
VehBrandB13	0.038 (0.045)
VehBrandB14	−0.106 (0.085)
VehBrandB2	0.003 (0.017)
VehBrandB3	0.020 (0.024)
VehBrandB4	0.015 (0.032)
VehBrandB5	0.078*** (0.027)
VehBrandB6	0.038 (0.031)
VehGasRegular	0.061*** (0.012)
DensityGLM	0.042*** (0.004)
RegionR11	−0.130*** (0.025)
RegionR21	0.017 (0.088)
RegionR22	−0.002 (0.053)
RegionR23	−0.122* (0.063)
RegionR25	−0.023 (0.044)
RegionR26	−0.092* (0.050)
RegionR31	−0.174*** (0.035)
RegionR41	−0.316*** (0.046)
RegionR42	−0.103 (0.094)
RegionR43	−0.108 (0.139)
RegionR52	−0.082*** (0.027)
RegionR53	0.024 (0.024)
RegionR54	−0.106*** (0.037)
RegionR72	−0.158*** (0.032)
RegionR73	−0.207*** (0.044)
RegionR74	0.127* (0.068)
RegionR82	0.010 (0.020)
RegionR83	−0.354*** (0.080)
RegionR91	−0.122*** (0.031)
RegionR93	−0.077*** (0.022)
RegionR94	0.037 (0.071)
Constant	−4.019*** (0.036)
Observations	576,292
Log Likelihood	−119,484.100
Akaike Inf. Crit.	239,064.200
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 16: Count model coefficients (truncated poisson with log link)

	Dependent variable:
	ClaimNb
VehPowerGLM5	−0.068 (0.081)
VehPowerGLM6	−0.106 (0.081)
VehPowerGLM7	−0.077 (0.080)
VehPowerGLM8	−0.070 (0.119)
VehPowerGLM9	−0.086 (0.089)
VehAgeGLM1	0.189** (0.081)
VehAgeGLM3	−0.048 (0.063)
DrivAgeGLM1	0.329* (0.174)
DrivAgeGLM2	0.262** (0.108)
DrivAgeGLM3	0.068 (0.097)
DrivAgeGLM4	−0.079 (0.076)
DrivAgeGLM6	0.080 (0.067)
DrivAgeGLM7	0.184* (0.100)
BonusMalusGLM	0.012*** (0.001)
VehBrandB10	−0.071 (0.175)
VehBrandB11	0.042 (0.180)
VehBrandB12	0.479*** (0.080)
VehBrandB13	0.113 (0.183)
VehBrandB14	−0.265 (0.409)
VehBrandB2	−0.088 (0.073)
VehBrandB3	−0.068 (0.102)
VehBrandB4	−0.185 (0.147)
VehBrandB5	−0.034 (0.117)
VehBrandB6	−0.022 (0.132)
VehGasRegular	−0.105** (0.052)
DensityGLM	0.042*** (0.016)
RegionR11	0.354*** (0.104)
RegionR21	0.328 (0.379)
RegionR22	0.087 (0.248)
RegionR23	0.627*** (0.239)
RegionR25	−0.011 (0.206)
RegionR26	0.052 (0.234)
RegionR31	0.513*** (0.133)
RegionR41	−0.045 (0.225)
RegionR42	0.354 (0.355)
RegionR43	0.409 (0.572)
RegionR52	0.059 (0.124)
RegionR53	0.058 (0.110)
RegionR54	0.128 (0.164)
RegionR72	0.450*** (0.128)
RegionR73	0.348* (0.184)
RegionR74	0.587** (0.261)
RegionR82	0.120 (0.092)
RegionR83	0.110 (0.356)
RegionR91	0.670*** (0.118)
RegionR93	0.590*** (0.090)
RegionR94	0.927*** (0.230)
Constant	−2.995*** (0.150)
Observations	576,292
Log Likelihood	−118,669.800
Note:	*p<0.1; **p<0.05; ***p<0.01

Table 17: Zero Hurdle model coefficients (binomial with logit link)

	Dependent variable:
	ClaimNb
VehPowerGLM5	0.222*** (0.021)
VehPowerGLM6	0.240*** (0.021)
VehPowerGLM7	0.137*** (0.021)
VehPowerGLM8	−0.113*** (0.031)
VehPowerGLM9	0.186*** (0.023)
VehAgeGLM1	0.732*** (0.019)
VehAgeGLM3	−0.221*** (0.015)
DrivAgeGLM1	−0.198*** (0.051)
DrivAgeGLM2	−0.454*** (0.031)
DrivAgeGLM3	−0.552*** (0.025)
DrivAgeGLM4	−0.359*** (0.018)
DrivAgeGLM6	−0.009 (0.016)
DrivAgeGLM7	0.188*** (0.026)
BonusMalusGLM	0.019*** (0.0004)
VehBrandB10	−0.005 (0.042)
VehBrandB11	0.035 (0.045)
VehBrandB12	−0.097*** (0.020)
VehBrandB13	0.006 (0.048)
VehBrandB14	−0.109 (0.089)
VehBrandB2	0.019 (0.018)
VehBrandB3	−0.00003 (0.025)
VehBrandB4	0.004 (0.034)
VehBrandB5	0.088*** (0.029)
VehBrandB6	0.014 (0.033)
VehGasRegular	0.129*** (0.013)
DensityGLM	0.028*** (0.004)
RegionR11	−0.297*** (0.027)
RegionR21	−0.208** (0.094)
RegionR22	−0.159*** (0.056)
RegionR23	−0.624*** (0.067)
RegionR25	−0.011 (0.047)
RegionR26	−0.270*** (0.053)
RegionR31	−0.452*** (0.037)
RegionR41	−0.309*** (0.049)
RegionR42	−0.135 (0.101)
RegionR43	−0.345** (0.147)
RegionR52	−0.156*** (0.029)
RegionR53	0.083*** (0.025)
RegionR54	−0.157*** (0.039)
RegionR72	−0.415*** (0.034)
RegionR73	−0.482*** (0.046)
RegionR74	−0.001 (0.073)
RegionR82	−0.076*** (0.022)
RegionR83	−0.554*** (0.084)
RegionR91	−0.454*** (0.033)
RegionR93	−0.322*** (0.024)
RegionR94	−0.249*** (0.077)
Constant	−4.177*** (0.039)
Observations	576,292
Log Likelihood	−118,669.800
Note:	*p<0.1; **p<0.05; ***p<0.01

Hyperparameters	Value
learning rate	0.011285
booster'	'gbtree'
lambda'	0.202855
alpha'	0.00030
subsample'	0.65794
colsample bytree'	0.78025
max depth'	7
min child weight'	5
eta'	3.360411e-05
gamma'	1.73966e-07
grow policy'	'depthwise'
objective'	'binary:logistic'
n estimators'	2000
eval metric	"logloss"
early stopping rounds	15

Table 18: Hyperparameters of the XGBoost binary model

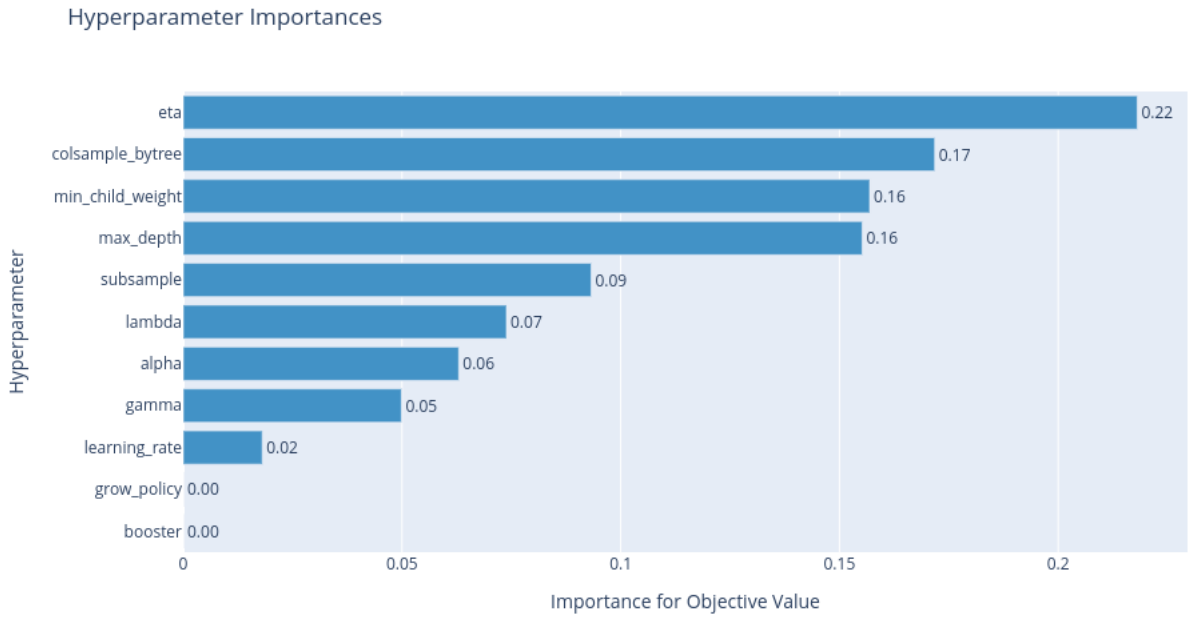


Table 19: This graph shows the most valuable parameters in optimizing the XGBoost binary model.

Hyperparameters	Value
learning rate	0.0052
booster	'gbtree'
lambda	0.0009
alpha	2.85434
subsample	0.81529
colsample bytree	0.83097
max depth	4
min child weight	2
eta	0.00539
gamma	6.91281e-06
grow policy	'lossguide'
objective	'PoissonZeroTruncatedNegLogLik'
n estimators	2000
eval metric	"EvalPoissonZeroTruncatedNegLogLik"
early stopping rounds	15

Table 20: Hyperparameters of the XGBoost Poisson zero truncated model

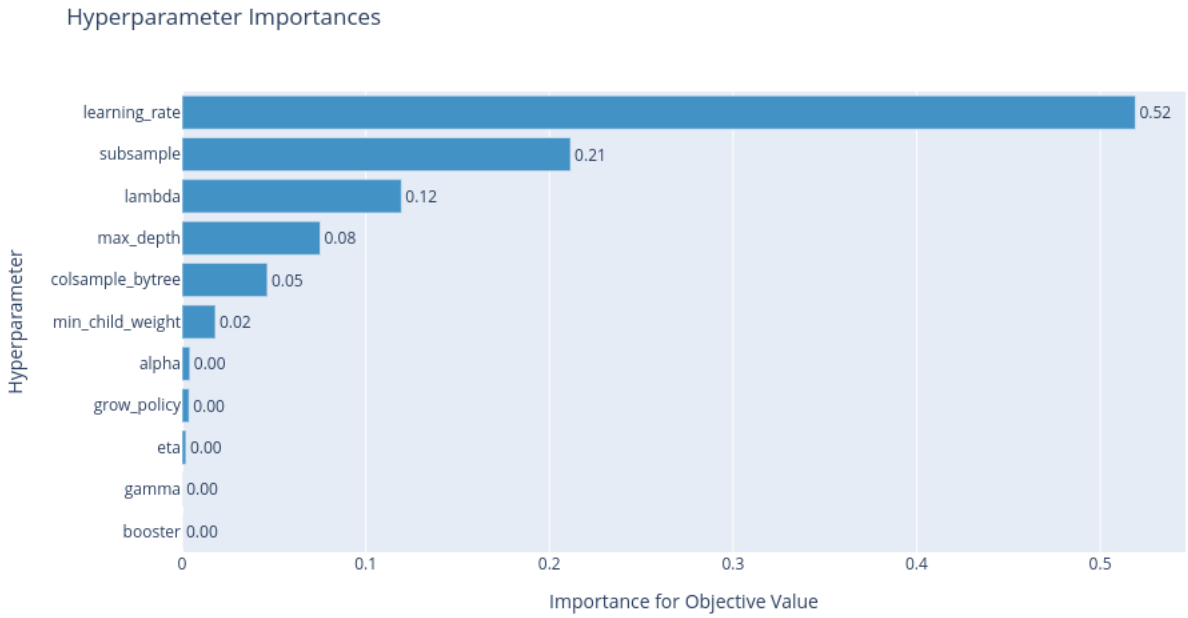
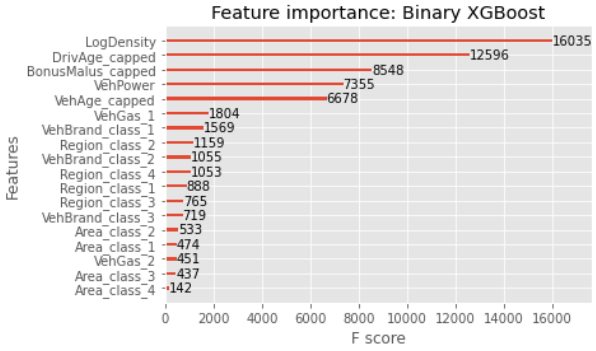
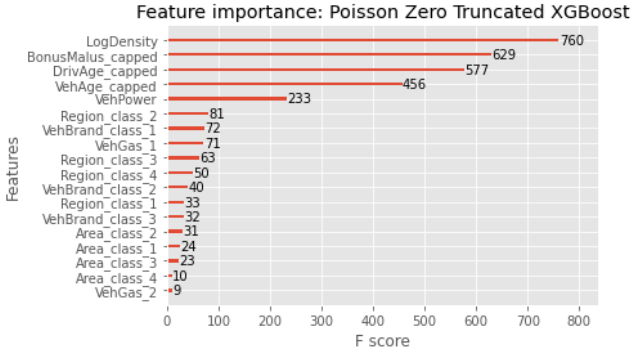


Table 21: This graph shows the most valuable parameters in optimizing the XGBoost Poisson zero truncated model.



(a) Importance of variables in the XGBoost Binary



(b) Importance of variables in the XGBoost Poisson zero truncated

Figure 2: These two graphs show the importance of the variables in the two models composing the Hurdle with statistical learning. We can see that the most used variable is Log(Density). This variable indicates the population of the city of residence.