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 : Statistique, Analyse et Modélisation Multidisciplinaire,
Centre PMF Université Paris 1, 90, rue de Tolbiac, 75013 Paris, 20ème étage
[2]Allianz France, 20 Pl. de Seine, 92400 Courbevoie

Abstract

The Poisson model is a commonly used method for modeling count data with exogenous variables, but it can be limiting when dealing with data that has a high proportion of zeros. To address this issue, we propose the use of the Hurdle model as an alternative approach. We discuss the properties of the Hurdle model and demonstrate how it can be implemented using both parametric and nonparametric estimates, including the XGBoost method. To evaluate the effectiveness of our proposed XGBoost Hurdle model, we apply it to a car insurance dataset from a French insurance company. This dataset includes a significant number of drivers with zero accidents per year. Our results show that the XGBoost Hurdle model outperforms several other models when applied to this data type.

Keywords— Insurance pricing; XGBoost algorithm; GLM, Hurdle Model
E-mails— Dafnis.Krasniqi@etu.univ-paris1.fr

1 Introduction

The Poisson distribution is commonly used to model count data in relation to exogenous variables, but it becomes inappropriate when the response variable y has mostly zero values and exhibits overdispersion (variance exceeding the mean). In such cases, an alternative regression model is necessary. Some options for handling overdispersion in data with mostly zero-valued response variables include the negative Binomial distribution (as described by Cameron and Trivedi (1990), Cameron and Trivedi (2005), Hilbe (2011)), the Zero-Inflated model (Lambert (1992)), and the Hurdle model (Mullahy (1986), Heilbron (1994)). The negative Binomial distribution is a generalization of the Poisson distribution that allows for overdispersion, often used in count data models with a high number of zeros and a long tail of positive values. It can be written as an extension of the Poisson model with an additional parameter for overdispersion. The Zero-Inflated model is used for data with an excess of zero values and consists of two parts: a binomial model for the probability of a zero value occurring and a Poisson model for the count data, given that the value is not zero. The Hurdle model is similar, with a binary model for the probability of a non-zero value occurring and a count model for the count data given that the value is non-zero. The Zero-Inflated and Hurdle models can be useful for data with a high proportion of zero values and overdispersion.

This paper proposes a novel approach to modeling zero excess count data using a Hurdle model with statistical learning models, specifically the XGBoost method (Chen and Guestrin (2016)). Traditional Hurdle models often rely on parametric or semi-parametric estimates, which can have limitations such as assumptions of equal error variances, a default distribution for the response variables, and a linear relationship between the dependent and independent variables. These assumptions may not always hold in real data, and parametric approaches may not be able to adequately model complex, sophisticated, non-linear relationships, and high-degree interactions. Using nonparametric models, such as boosting models or the XGBoost method, we can model complex relationships and interactions without requiring stringent assumptions.

Hurdle count data models were first introduced by Mullahy (1986). Since then, they have garnered significant attention in various fields, with many studies exploring their applications and properties. Gurmu (1998) proposed a generalization of hurdle models for analyzing overdispersed or underdispersed count data and applied their research to the analysis of Medicaid utilization. In a separate study, Greene (2007) compared Hurdle and zero-inflated models, providing insight into these approaches' relative strengths and limitations. Rudra and Biswas (2019) used the Hurdle model to analyze the use of manufactured cigarettes. In their study, many smokers did not purchase manufactured cigarettes, resulting in many zero values in the target variable.
Hurdle models constructed with statistical learning models have also been discussed in the literature. Povak et al. (2013) proposed a Hurdle model for detecting pollution in rivers. They used a random forests model based on the large number of zero values in the target variable for the non-polluted rivers. Kong et al. (2020) proposed a Hurdle model for detecting species in nature using a dataset with unbalanced classification. Their model was constructed from two neural networks.
Hurdle models have been applied in the insurance industry to analyze the number of claims made by insured drivers. Boucher, Denuit, and Guillén (2008) demonstrated that the Hurdle model is a useful alternative to classical Poisson or negative binomial models. Zhang, Pitt, and Wu (2022) also developed a multivariate Hurdle model using the expectation-maximization (EM) algorithm. Previous research has primarily relied on parametric models for model construction. To our knowledge, no studies have applied nonparametric estimation in Hurdle models for insurance data.

The paper is structured as follows: Section 2 introduces the classical Hurdle model and describes its parametric estimation. We also demonstrate that the Hurdle model can be decomposed into two independent models without loss of information. In Section 3, we explore nonparametric estimation using statistical learning techniques and develop prediction models. In Section 4, we apply the proposed models to a real-world data set, a car insurance portfolio of a French insurance company. Finally, in Section 5, we discuss the results and implications of our analysis.

2 The Poisson Hurdle model

The Poisson Hurdle model, also known as the "two-part model," is a well-known parametric model for predicting count data. It consists of two distinct processes: one that generates zeros and one that generates positive integers. The Hurdle model posits that a Bernoulli random variable, which depends on the exogenous variables, determines whether the count variable takes on a zero or positive value. The distribution of positive values, also dependent on the exogenous variables, is governed by a positive integer-valued distribution. These two processes are assumed to be independent of each other.

This section introduces the classical parametric Hurdle model and discusses methods for estimating its parameters and selecting the most appropriate model. This widely used and well-established approach serves as the foundation for our analysis.

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 are supposed to explain the response variable y. Our goal is to propose a model to predict this variable y from x.

Definition 1 *The Poisson Hurdle regression model is defined for all $i = 1...n$ as:*

$$P\left[Y_i = y_i | \, x_i, \pi_i, \lambda_i\right] = \begin{cases} \pi_i & \text{if } y_i = 0 \\ (1-\pi_i)\dfrac{\lambda_i^{y_i}}{(e^{\lambda_i}-1)y_i!} & \text{if } y_i > 0 \end{cases}$$

where π_i represents the probability of the binary part of the model and $\dfrac{\lambda_i^{y_i}}{(e^{\lambda_i}-1)y_i!}$ represents the probability of a positive count as determined by the Poisson zero-truncated distribution.

A critical issue in modeling count data is selecting appropriate link functions for the probability distribution. A common parametric choice is to use a logit link for the parameter π_i and a log link for the parameter λ_i.

Table 1: The link functions for the two models

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

In Table 1, β_1 represents the vector of regression coefficients for the covariates x_i in the Bernoulli model. In contrast, β_2 represents the vector of regression coefficients for the covariates x_i in the Poisson zero-truncated model. In both models, we can select the variables $x_i \in \mathbb{R}^p$ that we wish to include in the analysis.

The expectation of the hurdle model can be calculated as the sum of the Bernoulli component's expected value and the count component's expected value. Specifically, let Y be the random variable representing the count data, and let π_i be the probability of observing a zero value and λ_i be the expected count for non-zero values. Then, the expectation of the hurdle model can be written as follows:

$$\begin{aligned} E(Y_i|x_i, \pi_i, \lambda_i) &= \pi_i \cdot 0 + (1-\pi_i) \cdot E(Y_i \mid Y_i > 0) \\ &= (1-\pi_i) \cdot \frac{\lambda_i e^{\lambda_i}}{e^{\lambda_i} - 1} \end{aligned}$$

The expected value of the Bernoulli component is simply zero since it can only take on values of 0 or 1. The expected value of the count component, on the other hand, is equal to the expected count for non-zero values, $E(Y_i \mid Y_i > 0)$, multiplied by the probability of observing a non-zero value, $(1-\pi)$.

The variance of the hurdle model can be calculated as the sum of the Bernoulli component's variance and the count component's variance.

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

where π_i is the probability of observing a zero value and λ_i is the expected count for non-zero values. Note that 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

There are several ways to estimate the regression parameters (β_1 and β_2) for the hurdle model. One common approach is to use maximum likelihood estimation (MLE). Assuming that the observations $Y_1, \ldots, Y_n$ are independent, the log-likelihood function is given by:

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

with $\Omega_0 = \{i | y_i = 0\}$, $\Omega_1 = \{i | y_i \neq 0\}$, β_1 and β_2 are the vectors of regression coefficients for the Bernoulli and count components, respectively.
The log-likelihood function of the Hurdle model (1) can be decomposed into two independent log-likelihood functions, which allows us to study each component separately and potentially improve the overall performance.

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

We obtain maximum likelihood estimates by separately maximizing $\log(L_1(\beta_1, y))$ and $\log(L_2(\beta_2, y))$.

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

$$\begin{aligned}\log(L_2(\beta_2, y)) &= \sum_{i \in \Omega_1} \log\left(\frac{\left(e^{x_i^T \beta_2}\right)^{y_i}}{y_i!(e^{e^{x_i^T \beta_2}} - 1)}\right) \\ &= \sum_{i \in \Omega_1} y_i \log\left(e^{x_i^T \beta_2}\right) - \left(\sum_{i \in \Omega_1} \log\left(y_i!\right) + \sum_{i \in \Omega_1} \log\left(e^{e^{x_i^T \beta_2}} - 1\right)\right) \end{aligned} \tag{2}$$

As a result,

$$\hat{\beta}_1 = \underset{\beta_1 \in \mathbb{R}^{p+1}}{\operatorname{argmax}} \log(L_1(\beta_1, y)) \quad \hat{\beta}_2 = \underset{\beta_2 \in \mathbb{R}^{p+1}}{\operatorname{argmax}} \log(L_2(\beta_2, y))$$

However, such an estimator does not have a closed-form expression. Therefore, we need to use iterative algorithms, such as the Newton-Raphson or gradient algorithms, to estimate the value of β. To select the best exogenous variables, several techniques can be used, such as the AIC (see Akaike (1973)) or BIC (see Schwarz (1978)) criteria. These model selection methods can help us identify the combination of variables that best explains the data while avoiding overfitting.

$$AIC = -2 \cdot \log(L(\widehat{\beta}_1, \widehat{\beta}_2, y)) + 2 \cdot k$$

$$BIC = -2 \cdot \log(L(\widehat{\beta}_1, \widehat{\beta}_2, y)) + \log(n) \cdot k$$

with $\log(L(\hat{\beta}_1, \hat{\beta}_2, y))$ is the maximized value of the likelihood function, k the number of estimated parameters and n the sample size.

Therefore, increasing the value of k will generally result in a higher AIC and BIC, indicating a lower-quality model. When comparing multiple models, selecting the model with the lowest AIC or BIC is usually recommended, as this indicates the best balance between model fit and parsimony.

3 The XGBoost Hurdle model

The traditional Hurdle model consists of two components: a binary variable that determines whether the response variable is zero or positive and a positive integer-valued variable that models the positive data using a zero truncated Poisson regression. However, these parametric or semi-parametric approaches may not be able to model complex, non-linear relationships and high-degree interactions. Using XGBoost, a nonparametric statistical learning method, we can model these complex relationships without the need for stringent assumptions. We optimize a logistic loss function and a Poisson zero truncated loss function using XGBoost to construct our new Hurdle model, representing a novel contribution to the literature, particularly in actuarial science.

Incorporating nonparametric models, such as those optimized using the XGBoost algorithm, can significantly improve model performance when parametric models are insufficiently flexible to capture the complexity of the data. This is particularly useful in actuarial science, where accurate modeling of insurance risks is crucial for making informed business decisions. Our approach offers a flexible and effective solution for modeling count data in the insurance industry by using nonparametric models to optimize the logistic and Poisson zero truncated loss functions. One key advantage of our approach is the ability to directly optimize the models on the data rather than relying on predetermined distributional assumptions. This allows the models to capture the underlying patterns and relationships in the data, leading to improved predictive accuracy. Additionally, using the XGBoost algorithm allows for efficient optimization of the models, making it a practical and scalable approach for analyzing large datasets.

3.1 XGBoost

XGBoost is a tree-based machine learning method introduced by Chen and Guestrin (2016). It builds multiple trees, each learning from and improving upon the previous one. These weak models are then combined to create a more robust model. Compared to other tree-based methods, XGBoost has two main advantages. First, it is computationally efficient because it processes data in compressed blocks, allowing it to be quickly ordered and processed in parallel. Second, it uses second-order Taylor expansion to find the minimum of the objective function. This means that it takes into account the curvature of the objective function, allowing it to converge to a better minimum than other methods.

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 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 we consider is a statistical learning model with a loss function based on the log-likelihood of logistic regression. This loss function is provided in the Xgboost library called the binary loss function. Modifying the initial data slightly is necessary to build such a binary model. 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 about 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:

$$\pi^{ML}(x_i) = \frac{e^{\sum_{m=1}^{M} \eta f(x_i, \alpha_m)}}{1 + e^{\sum_{m=1}^{M} \eta f(x_i, \alpha_m)}}. \tag{3}$$

with $\pi^{ML}(x_i)$ the final output, M the total number of trees in the XGBoost model, and η the learning rate. The parameters M and η are obtained by cross-validation.
The result of (3), 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 of belonging to class 0 and 1.

3.3 Part 2: XGBoost of zero-truncated Poisson distribution

The second part of the XGBoost Hurdle model is a zero-truncated Poisson model devoted to predicting 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, and 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, $x_{\not 0}$ is the database without the rows with a value for the target variable equal to 0. To learn this data, a second loss function will be defined.

$$\ell_2(\tilde{y}_i; f(x_i, \alpha_t)) = -\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i \times \log(f(x_i, \alpha_t)) - f(x_i, \alpha_t)) \tag{4}$$

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 follows a Poisson distribution. Unfortunately, this loss function (4) does not work in our case because a loss function corresponding to the log-likelihood of the zero-truncated Poisson distribution is needed. To this end, a new loss function will be coded in XGBoost. This loss function is one of the contributions of this paper.

By coding a new loss function specifically tailored to our research needs, we have made a significant contribution to the field of modeling count data. Our novel loss function is based on the Poisson zero truncated distribution, which is commonly used for modeling count data with excess zeros. It allows for the modeling of overdispersion and zero-inflation, standard features of count data in many applications. Implementing this loss function in XGBoost enables the efficient optimization of nonparametric models for count data using the XGBoost algorithm.

3.3.1 New loss function

To construct the new loss function, we start from the maximum likelihood of the zero-truncated Poisson distribution (2). In the XGBoost, we have to minimize functions, so the negative of the function (2) will be used. In addition, the parametric part will be replaced ($\lambda_i = e^{x_i\beta_2}$) by a nonparametric part ($\lambda_i = f(x_i, \alpha_m)$) and create a loss function dedicated to the boosting model.

$$\ell_2(\tilde{y}_i; f(x_i, \alpha_t)) = -\frac{1}{|\Omega_1|}\sum_{i\in\Omega_1}(\tilde{y}_i \times \log(f(x_i, \alpha_t)) - \log(e^{f(x_i,\alpha_t)} - 1)) \tag{5}$$

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

Three ingredients are needed to implement a new loss function in XGBoost: the first derivative of the equation (5), the second derivative of the equation (5) and a function to evaluate. For the first derivative of (5) 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 (5) is :

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

The equation (5) will be directly used for the evaluation function. 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:

$$\lambda^{ML}(x_i) = \frac{\sum_{m=1}^{M}\beta_m f(x_i, \alpha_m) \times e^{\sum_{m=1}^{M}\beta_m f(x_i,\alpha_m)}}{e^{\sum_{m=1}^{M}\beta_m f(x_i,\alpha_m)} - 1} \tag{6}$$

with the result of (6), values greater than 1 are obtained.

3.4 The prediction strategy

Once the two sub-models have been defined, we can specify the prediction strategy for our new XGBoost Hurdle model.

Definition 2 *The prediction $\hat{y}_{n+1}$ for a new vector x_{n+1} is given by:*

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

where $\pi^{ML}(x_{n+1})$ is the predicted probability of the XGBoost binary model (see (3)), $\lambda^{ML}(x_{n+1})$ is the predicted number of the XGBoost zero-truncated Poisson (see (6)) and $\lfloor x \rceil$ represents the nearest integer at x.

$\hat{\mu}_{n+1}$ is the predicted mean of the XGBoost Hurdle model. It is a method used to analyze the count data and improves the parametric hurdle model because it uses statistical learning models instead of parametric models to make predictions.

4 Application on insurance data

This section will first provide a brief overview of the data and explain why it is highly relevant to our situation. Next, we will describe the various measures used to evaluate the quality of our results. Finally, we will apply our custom-designed algorithms to specific insurance data and analyze the resulting output carefully.

4.1 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 (n.d.)). 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), Pocuca et al. (2018), Miljkovic and Fernández (2018)). Here is the list of features and their descriptions in the table below.

Table 2: The features in *FreMTPL2freq*

Features	Description	Type
ClaimNb	the number of accidents (between 0 and 5)	continuous
Exposure	Exposure (in a 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 3: The distribution of the variables *ClaimNb* and *Exposure* is shown. In this dataset, there are 643,953 drivers with no claims, with exposure to the risk of 336,616. Then there are 32,178 drivers with one claim, with exposure to the risk of 20,670. And so on.

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. Two accidents in 6 months ($e_i = 0.5$) and two accidents in 1 year ($e_i = 1$) are not the same. This exposure variable will therefore aim to harmonize contracts with unequal subscription periods.

4.2 Experiment parameters and optimization

To improve the performance of our models, we apply a series of restatements to the variables in our data. This helps to clean the database and eliminate errors and extreme values. This process involves two key steps:

1. To ensure the accuracy and reliability of our predictions, we will cap the number of claims at three in the target variable. This is because accidents become extremely rare and unpredictable after the third claim. Therefore, we will treat drivers with more than three accidents as if they had only three accidents;

2. The different algorithms do not accept categorical variables, so we will transform them into continuous variables.

   Our data only contains nominal qualitative variables, so we will use numerical encoding to represent these variables without considering any inherent hierarchy. This will result in a matrix with column names representing the different modalities for each qualitative variable. Before performing this transformation, we will group some modalities to avoid creating excessively sparse matrices with many zeros. For quantitative variables, no transformation is necessary, and we can directly input them into the algorithms.

Five algorithms are applied to the freMTPL2freq data: GLM Poisson, Hurdle, Decision Tree, Poisson Boosting Machine (PBM) with deep 3 and 5, and the XGBoost Hurdle. The decision tree and the Poisson Boosting Machine are deduced from the article by Noll, Salzmann, and Wuthrich (2018) [1]. Therefore, there are two parametric models and three nonparametric models. Regarding the database, it is divided into two parts: a training set (80% of the database) and a testing set (20% of the database). When the nonparametric methods are applied, the training set is further divided into a validation set to adjust the hyperparameters. A Monte Carlo Cross-Validation with 30 different bootstrapped samples is performed for each method. This method allows us to obtain robust results. For more information, the various types of cross-validation are discussed in detail in the thesis of Cornec (2009).

The stepwise approach is used to optimize the parametric Poisson GLM model to minimize the BIC with the library glm on R (see Marschner, 2011). Then, the Hurdle is optimized with the same approach with the pscl library on R (see Jackman, 2020). To optimize the Decision Tree and the Poisson Boosting Machine (PBM) Noll, Salzmann, and Wuthrich (2018), use the library *rpart* on R.
For the XGBoost Hurdle model, XGBoost and Optuna library (see Akiba et al., 2019) will be used to perform our modeling. XGBoost methods require a lot of parameters that need to be perfectly tuned. To help us in this task, the library *Optuna* will be used. This library aims to find the best parameters randomly for each model. The user only has to give a range of parameters for the model to be tested, and the library will do the necessary by performing random tests. An early stop will be implemented to provide a model that generalizes well over all the data and not just the training data. If the score on the training set continues to decrease during the construction phase while the score on the validation set increases.

4.2.1 Comparisons of the methods

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

Poisson deviance:
The Poisson deviance is closely related to the negative log-likelihood function, which is often used as a loss function in Poisson regression. The log-likelihood function for the Poisson distribution is given by:

$$L(y, \hat{\mu}) = \sum_{i=1}^{n} (y_i \log \hat{\mu}_i - \hat{\mu}_i - \log y_i!)$$

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

where y_i is the observed count for the ith observation, and $\hat{\mu}_i$ is the predicted count for the ith observation. The Poisson deviance can be obtained by subtracting the log-likelihood from its saturated value:

$$d(y_i; \hat{\mu}_i) = 2L_{\text{saturated}} - 2L(y, \hat{\mu})$$
$$d(y_i; \hat{\mu}_i) = 2\sum_{i=1}^{n} \left(y_i \log\left(\frac{y_i}{\hat{\mu}_i}\right) - (y_i - \hat{\mu}_i) \right)$$

where $L_{\text{saturated}}$ is the negative log-likelihood function for a saturated model. A saturated model is a model that perfectly fits the data. The Poisson deviance can therefore be interpreted as the deviation of the model from a saturated model, with a smaller deviance indicating a better fit of the model to the data.

Comparing the model predictions with the observed value:
In this analysis, we compare the sum of the predicted values to the sum of the observed values to measure the difference between the two. A smaller difference indicates a better fit of the model to the data.

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

The confusion matrix (C-M):
We constructed a confusion matrix because the first two metrics were insufficient for understanding the model's performance. These metrics only provided a broad overview of the model's accuracy without giving insight into its ability to predict rare events. The confusion matrix allowed us to examine the model's predictions in more detail and evaluate its performance in extreme cases. To build this matrix, we used the predicted values $\hat{y}_i$.

Table 4: C-M

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

4.3 Results

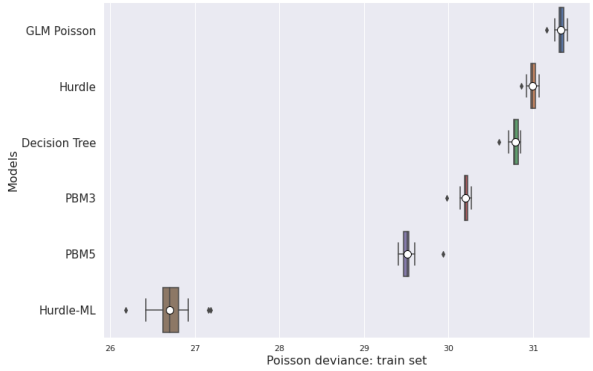
In this section, the results of the different algorithms will be analyzed and compared on a test and training basis. Three models of Noll, Salzmann, and Wuthrich (2018) are also included. The authors of this paper seek only to optimize the Poisson deviance but not the confusion matrices. We used their open-source code to compute the missing criteria.

Table 5: Results: The different models are in the first column. The results of the Poisson deviance for the training and test sets are described in the second and third columns. In the fourth and fifth columns, the comparison score of the sum of predictions and the sum of observations are described for the training and test sets.

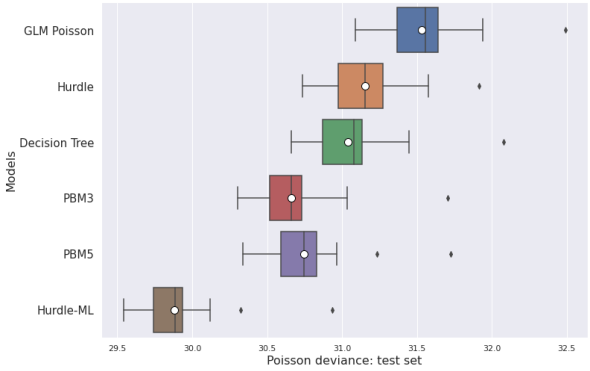
	P. deviance train	P. deviance test	$\neq$ accidents train	$\neq$ accidents test
GLM Poisson	31.33	31.53	0.00%	-0.71%
Hurdle	30.99	31.15	-1.18%	-1.85%
Decision Tree	30.79	31.04	-1.68e-02%	-0.68%
PBM3	30.20	30.66	1.22e-04%	-0.61%
PBM5	29.51	30.74	2.67e-05%	-0.40%
XGBoost Hurdle	26.71	29.88	0.39%	-0.38%

In terms of Poisson deviance, both the Hurdle model with parametric models (average scores of 30.99 on the training set and 31.15 on the test set) and the GLM Poisson (average scores of 31.33 on the training

set and 31.51 on the test set) perform significantly worse compared to the statistical learning models. These models are unable to distinguish between good and bad drivers. The methods developed in Noll, Salzmann, and Wuthrich (2018) perform better than the parametric models, with the Decision Tree achieving average scores of 30.79 on the training set and 31.04 on the test set and the PBM3 achieving average scores of 30.20 on the training set and 31.66 on the test set. However, the best performance is achieved by the XGBoost Hurdle model, which has average scores of 26.70 on the training set and 29.88 on the test set.
Upon examining the scores for the difference between the sum of the predicted and observed values, it can be seen that all models perform well and are close to 0. This is reassuring as it demonstrates that the XGBoost Hurdle model has excellent scores on the Poisson deviance without compromising its performance on this metric.



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



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

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

According to the two graphs 1a and 1b, it is clear that the XGBoost Hurdle model performs the best. These graphs contain box plots for each method, with 30 values plotted. The box plot for the XGBoost Hurdle model is significantly farther from the others for the training set, indicating that it may have slightly overfitted the data. However, it also performs well on the test set.

Table 6: C-M 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 7: C-M 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 8: C-M 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 9: C-M 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

Table 10: C-M 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 11: C-M 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 12: C-M 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 13: C-M 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 14: C-M XGBoost 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 15: C-M XGBoost 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

Looking at the confusion matrices for the test data, we should remember that we are faced with a complicated problem: working on a counting model with extremely unbalanced data.

- 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 $\hat{y} = 0$. This model fails to detect drivers with very high risks.
- The two models of Noll, Salzmann, and Wuthrich (2018) (PMB3 and decision 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 (13)). On the test set, it detects 68 potential drivers with one claim.
- The XGBoost Hurdle model showed the best performance on the test set, accurately identifying 117 potentially risky drivers with a single claim (see (15)). While there were some misclassifications for this model, the results were still favorable compared to the other models. For example, the PMB3 model identified 68 good potential drivers with one claim but also had 180 misclassifications ($180/68 = 2.65$), while the XGBoost Hurdle model had a lower ratio of $168/117 = 1.44$ misclassifications per good prediction. Overall, the XGBoost Hurdle model demonstrated a solid ability to distinguish between good and risky drivers. Some speculation for the misclassifications: the algorithm classifies these drivers in this way for two reasons. These drivers have all the characteristics of drivers who are likely to have accidents but have not had any. They may have been involved in accidents during the year but did not report them to their insurance company because they preferred to pay them in cash (see Lemaire (1985)).

The XGBoost Hurdle model outperformed the other methods in analyzing automobile insurance actuarial data. It successfully identified many potentially risky drivers with only one claim and had a lower misclassification rate than the other models. This demonstrates the model's ability to accurately predict driver behavior and distinguish between good and risky individuals. Its strong performance on the test set further confirms its effectiveness in analyzing actuarial data.

5 Conclusion

In this study, we examined the effectiveness of a Hurdle model for modeling zero excess count data in the insurance industry. We found that the Hurdle model can be decomposed into a binomial regression and a zero-truncated Poisson regression. To improve upon traditional parametric Hurdle models, we introduced a novel approach using the XGBoost library and a custom loss function for zero-truncated Poisson regression, which we refer to as the XGBoost Hurdle model. This contribution is valuable to the existing literature on count data modeling in insurance, offering a flexible and effective alternative to traditional methods.

To compare the performance of our novel model, we also evaluated four other methods (two parametric and two nonparametric models). Our results showed that the XGBoost Hurdle model outperformed all other methods across all criteria. However, a limitation of the XGBoost Hurdle model (as well as the other methods evaluated) is that it cannot accurately identify drivers with multiple claims. To address this issue, we suggest further improvements, such as the use of double Hurdles or a cascade structure to address data imbalance. Overall, our results demonstrate the efficacy of the XGBoost Hurdle model in predicting driver behavior and distinguishing between good and risky individuals in the insurance industry.

Acknowledgements

The authors would like to thank the entire staff of Allianz France for their help and funding.

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*.

Boucher, Jean P., Michel Denuit, and Montserrat Guillén (2008). "Modelling of Insurance Claim Count with Hurdle Distribution for Panel Data". In.

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 (n.d.). "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].

Greene, William (Feb. 2007). "Functional Form and Heterogeneity in Models for Count Data". In: *Foundations and Trends® in Econometrics* 1. DOI: 10.2139/ssrn.986620.

Gurmu, Shiferaw (1998). "Generalized hurdle count data regression models". In: *Economics Letters* 58.3, pp. 263–268. ISSN: 0165-1765. DOI: https://doi.org/10.1016/S0165-1765(97)00295-4. URL: https://www.sciencedirect.com/science/article/pii/S0165176597002954.

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.

Jackman, Simon (2020). *pscl: Classes and Methods for R Developed in the Political Science Computational Laboratory*. R package version 1.5.5. United States Studies Centre, University of Sydney. Sydney, New South Wales, Australia. URL: https://github.com/atahk/pscl/.

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.

Lemaire, J. (1985). *Automobile Insurance: Actuarial Models*. Witherby. ISBN: 9780785543183. URL: https://books.google.fr/books?id=2AlOvgAACAAJ.

Marschner, Ian C. (2011). "glm2: Fitting generalized linear models with convergence problems". In: *The R Journal* 3, pp. 12–15.

Miljkovic, Tatjana and Daniel Fernández (2018). "On Two Mixture-Based Clustering Approaches Used in Modeling an Insurance Portfolio". In: *Risks* 6.2, pp. 1–18. URL: https://ideas.repec.org/a/gam/jrisks/v6y2018i2p57-d147107.html.

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*.

Pocuca, Nikola et al. (2018). *Modeling Frequency and Severity of Claims with the Zero-Inflated Generalized Cluster-Weighted Models*. DOI: 10.48550/ARXIV.1812.11829. URL: https://arxiv.org/abs/1812.11829.

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.

Zhang, Pengcheng, David Pitt, and Xueyuan Wu (2022). "A new multivariate zero-inflated hurdle model with applications in automobile insurance". In: *ASTIN Bulletin*, 1–24. DOI: 10.1017/asb.2021.39.

Appendices

Zero-truncated poisson loss function with python

```
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)
```