*Article*

# Predicting Motor Insurance Claims Using Telematics Data—XGBoost versus Logistic Regression

**Jessica Pesantez-Narvaez, Montserrat Guillen * and Manuela Alcañiz**

Department of Econometrics, Riskcenter-IREA, Universitat de Barcelona, 08034 Barcelona, Spain; jessica.pesantez@ub.edu (J.P.-N.); malcaniz@ub.edu (M.A.)
* Correspondence: mguillen@ub.edu; Tel.: +34-934-037-039

Received: 9 May 2019; Accepted: 12 June 2019; Published: 20 June 2019

**Abstract:** XGBoost is recognized as an algorithm with exceptional predictive capacity. Models for a binary response indicating the existence of accident claims versus no claims can be used to identify the determinants of traffic accidents. This study compared the relative performances of logistic regression and XGBoost approaches for predicting the existence of accident claims using telematics data. The dataset contained information from an insurance company about the individuals' driving patterns—including total annual distance driven and percentage of total distance driven in urban areas. Our findings showed that logistic regression is a suitable model given its interpretability and good predictive capacity. XGBoost requires numerous model-tuning procedures to match the predictive performance of the logistic regression model and greater effort as regards to interpretation.

**Keywords:** dichotomous response; predictive model; tree boosting; GLM; machine learning

## 1. Introduction

Predicting the occurrence of accident claims in motor insurance lies at the heart of premium calculation, but with the development of new artificial intelligence methods, the question of choosing a suitable model has yet to be completely solved. In this article, the recently proposed methods of XGBoost (Chen and Guestrin 2016) and logistic regression are considered and compared regarding their predictive performance in a sample of insured drivers, along with their telematic information.

The advantages and disadvantages of XGBoost compared to logistic regression are discussed and this study showed that a slightly improved predictive power is only obtained with the XGBoost method, but this has complicated the interpretation of the impact of covariates on the expected response. In the case of automobile insurance, where the premium calculation is regulated and has to be fully specified, the weight of each risk factor in the final price needs to be disclosed and the connection between the observed covariate value and the estimated probability of a claim needs to be shown. If these conditions are not met, the regulating authority may deny the insurance company the right to commercialize that product. This study discussed, nevertheless, why the use of an XGBoost algorithm remains interesting for actuaries and how methods both old and new might be combined for optimum results. This study does not examine any other boosting methods. However, excellent descriptions can be found in Lee and Lin (2018), while extensions to high dimensional datasets are presented in Lee and Antonio (2015), both of which presented cases studies of insurance applications. Many of those alternatives placed their emphasis on algorithm speed, but in terms of their essential setups they do not differ greatly from XGBoost.

To compare the two competing methods, a real dataset comprising of motor insurance policy holders and their telematics measurements were used, that is, real-time driving information collected and stored via telecommunication devices. More specifically, GPS-based technology captures an insured's driving behavior patterns, including distance travelled, driving schedules, and driving speed, among many others. Here, pay-as-you-drive (PAYD) insurance schemes represent an alternative method for pricing premiums based on personal mileage travelled and driving behaviors. Guillen et al. (2019), Verbelen et al. (2018), and Pérez-Marín and Guillén (2019) showed the potential benefits of analyzing telematics information when calculating motor insurance premiums. Gao and Wüthrich (2019) analyzed high-frequency GPS location data (second per second) of individual car drivers and trips. Gao and Wüthrich (2018) and Gao et al. (2019) investigated the predictive power of covariates extracted from telematics car driving data using the speed-acceleration heatmaps proposed by Wüthrich (2017). Further, Hultkrantz et al. (2012) highlighted the importance of PAYD insurance plans insofar as they allow insurance companies to personalize premium calculation and, so, charge fairer rates.

The rest of this paper is organized as follows. First, the notation is introduced and the logistic regression and XGBoost methods are outlined. Second, our dataset is described and some descriptive statistics are provided. Third, the results of our comparisons in both a training and a testing sample are reported. Finally, following the conclusion, some practical suggestions are offered about the feasibility of applying new machine learning methods to the field of insurance.

## 2. Methodology Description

In a data set of $n$ individuals and $P$ covariates, there is a binary response variable $Y_i$, $i = 1, \ldots,$ $n$ taking values 0, 1; and a set of covariates denoted as $X_{ip}$, $p = 1, \ldots, P$. The conditional probability density function of $Y_i = t$ ($t = 0, 1$) given $X_i$ $(X_{i1}, \ldots, X_{iP})$, is denoted as $h_t(X_i)$. Equivalently, it can be said that $\mathrm{Prob}(Y_i = t) = h_t(X_i)$, and that $\mathrm{E}(Y_i) = \mathrm{Prob}(Y_i = 1) = h_1(X_i)$.

### 2.1. Logistic Regression

Logistic regression, a widely recognized regression method for predicting the expected outcome of a binary dependent variable, is specified by a given set of predictor variables. McCullagh and Nelder (1989) presented the logistic regression model as part of a wider class of generalized linear models. A logistic regression is distinguished from a classical linear regression model primarily because the response variable is binary rather than continuous in nature.

The logistic regression uses the logit function as a canonical link function, in other words, the log ratio of the probability functions $h_t(X_i)$ is a linear function of $X$; that is:

$$\ln \frac{h_1(X_i)}{h_0(X_i)} = \ln \frac{\mathrm{Prob}(Y_i = 1)}{\mathrm{Prob}(Y_i = 0)} = \beta_0 + \sum\nolimits_{p=1}^{P} X_{ip}\beta_p, \tag{1}$$

where $\beta_0, \beta_1, \ldots, \beta_P$ are the model coefficients[1], and $\mathrm{Prob}(Y_i = 1)$ is the probability of observing the event in the response (response equal to 1), and $\mathrm{Prob}(Y_i = 0)$ is the probability of not observing the event in the response (response equal to 0).

The link function provides the relationship between the linear predictor $\eta = \beta_0 + \sum_{p=1}^{P} X_{ip}\beta_p$ and the mean of the response given certain covariates. In a logistic regression model, the expected response is:

$$E(Y_i) = \mathrm{Prob}(Y_i = 1) = \frac{e^{\beta_0 + \sum_{p=1}^{P} X_{ip}\beta_p}}{1 + e^{\beta_0 + \sum_{p=1}^{P} X_{ip}\beta_p}}. \tag{2}$$

A logistic regression can be estimated by the maximum likelihood (for further details see, for example, Greene 2002). Therefore, the idea underlying a logistic regression model is that there must be a linear combination of risk factors that is related to the probability of observing an event. The data analyst's task is to find the fitted coefficients that best estimate the linear combination in (2) and to interpret the relationship between the covariates and the expected response. In a logistic regression

[1] Note we have opted to refer here to coefficients as opposed to parameters to avoid confusion with the values defined below when describing the XGBoost method.

model, a positive estimated coefficient indicates a positive association. Thus, when the corresponding covariate increases, the probability of the event response also increases. If the estimated coefficient is negative, then the association is negative and, therefore, the probability of the event decreases when the observed value of the corresponding covariate increases. Odds-ratios can be calculated as the exponential values of the fitted coefficients and they can also be directly interpreted as the change in odds when the corresponding factor increases by one unit.

Apart from their interpretability, the popularity of logistic regression models is based on two characteristics: (i) The maximum likelihood estimates are easily found; and (ii) the analytical form of the link function in (2) always provides predictions between 0 and 1 that can be directly interpreted as the event probability estimate. For these motives, logistic regression has become one of the most popular classifiers, their results providing a straightforward method for predicting scores or propensity values which, in turn, allow new observations to be classified to one of the two classes in the response. For R users, the glm function is the most widely used procedure for obtaining coefficient estimates and their standard errors, but alternatively, a simple optimization routine can easily be implemented.

## 2.2. XGBoost

Chen and Guestrin (2016) proposed XGBoost as an alternative method for predicting a response variable given certain covariates. The main idea underpinning this algorithm is that it builds $D$ classification and regression trees (or CARTs) one by one, so that each subsequent model (tree) is trained using the residuals of the previous tree. In other words, the new model corrects the errors made by the previously trained tree and then predicts the outcome.

In the XGBoost, each ensemble model[2] uses the sum of $D$ functions to predict the output:

$$\hat{Y}_i = ₣(X_i)= \sum\nolimits_{d=1}^{D} f_d(X_i),\ f_d \in \mathrm{F},\ i = 1, \ldots, n \tag{3}$$

where F is the function space[3] of the CART models, and each $f_d$ corresponds to an independent CART structure which is denoted as $q$. In other words, $q$ is the set of rules of an independent CART that classifies each individual $i$ into one leaf. The training phase involves classifying $n$ observations so that, given the covariates $X$, each leaf has a score that corresponds to the proportion of cases which are classified into the response event for that combination of $X_i$. This score is denoted as $w_{q(X)}$.

Thus, $q$ can be written as a function $q$: $\mathbb{R}^P \rightarrow T$, where $T$ is the total number of leaves of a tree and $j$ is later used to denote a particular leaf, $j = 1, \ldots, T$. To calculate the final prediction for each individual, the score of the leaves are summed as in (3), where F = $\{f(\mathrm{X}) = w_{q(\mathrm{X})}\}$, with $q$: $\mathbb{R}^P \rightarrow T$, and $w \in \mathbb{R}^T$.

In general, boosting methods fit D models in D iterations (each iteration denoted by $d$, $d = 1, \ldots, D$) in reweighted versions. Weighting is a mechanism that penalizes the incorrect predictions of past models, in order to improve the new models. The weighting structures are generally optimal values, which are adjusted once a loss function is minimized. Then, new learners incorporate the new weighting structure in each iteration, and predict new outcomes.

In particular, the XGBoost method minimizes a regularized objective function, i.e., the loss function plus the regularization term:

$$\mathcal{L} = \sum\nolimits_{i=1}^{n} \ell(Y_i, \hat{Y}_i) + \sum\nolimits_{d=1}^{D} ή(f_d), \tag{4}$$

[2] Natekin and Knoll (2013) explain that the ensemble model can be understood as a committee formed by a group of base learners or weak learners. Thus, any weak learner can be introduced as a boosting framework. Various boosting methods have been proposed, including: (B/P-) splines (Huang and Yang 2004); linear and penalized models (Hastie et al. 2009); decision trees (James et al. 2013); radial basis functions (Gomez-Verdejo et al. 2005); and Markov random fields (Dietterich et al. 2008). Although Chen and Guestrin (2016) state $f_k$ as a CART model, the R package xgboost currently performs three boosters: linear, tree and dart.

[3] The XGBoost works in a function space rather than in a parameter space. This framework allows the objective function to be customized accordingly.

where $\ell$ is a convex loss function that measures the difference between the observed response $Y_i$ and predicted response $\hat{Y}_i$ and $ἡ = \mu T + \frac{1}{2}\lambda\|w\|_2^2$, $ἡ$ is the regularization term also known as the shrinkage penalty which penalizes the complexity of the model and avoids the problem of overfitting. The tree pruning parameter $\mu$ regulates the depth of the tree and $\lambda$ is the regularization parameter that is associated with $l2$-norm of the scores vector, which is a way of evaluating the magnitude of scores. Including this norm, or any other similar expression, penalizes excessive sizes in the components of $w$.

It is noted that pruning is a machine learning technique which reduces the size of a decision tree by removing decision nodes whose corresponding features have little influence on the final prediction of the target variable. This procedure reduces the complexity of the model and, thus, corrects overfitting.

The $l2$-norm is used in the L2 or Ridge regularization method, while the $l1$-norm is used in the L1 or Lasso regularization method. Both methods can take the Tikhonov or the Ivanov form (see Tikhonov and Arsenin 1977; Ivanov et al. 2013).

### 2.2.1. A Closer Look at the XGBoost Minimization Algorithm

A loss function or a cost function like (4) measures how well a predictive algorithm fits the observed responses in a data set (for further details, see Friedman et al. 2001). For instance, in a binary classification problem, the logistic loss function is suitable because the probability score is bounded between 0 and 1. Then, by selecting a suitable threshold, a binary outcome prediction can be found. Various loss functions have been proposed in the literature, including: The square loss; the hinge loss (Steinwart and Christmann 2008); the logistic loss (Schapire and Freund 2012); the cross entropy loss (De Boer et al. 2005); and the exponential loss (Elliott and Timmermann 2003).

The intuition underpinning the regularization proposed in (4) involves reducing the magnitude of $w$, so that the procedure can avoid the problem of overfitting. The larger the $e$, the smaller the variability of the scores (Goodfellow et al. 2016).

The objective function at the $d$-th iteration is:

$$\mathcal{L}^{(d)} = \sum\nolimits_{i=1}^{n} \ell(Y_i,\ \hat{Y}_i^{(d-1)} + f_d(X_i)) + ἡ(f_d), \tag{5}$$

where $\hat{Y}_i^{(d-1)}$ is the prediction of the $i$-th observation at the $(d-1)$-th iteration. It is noted that $\ell(\cdot,\cdot)$ is generally a distance so its components can be swapped, i.e., $\ell(Y_i, \hat{Y}_i) = \ell(\hat{Y}_i,\ Y_i)$. Following Chen and Guestrin (2016), it is assumed that the loss function is a symmetric function.

Due to the non-linearities in the objective function to be minimized, the XGBoost is an algorithm that uses a second-order Taylor approximation of the objective function $\mathcal{L}$ in (5) as follows:

$$\mathcal{L}^{(d)} \cong \sum\nolimits_{i=1}^{n} [\ell(Y_i,\ \hat{Y}_i^{(d-1)}) + g_i f_d(X_i) + \frac{1}{2} h_i {f_d}^2(X_i)] + ἡ(f_d), \tag{6}$$

where $g_i = \partial_{\hat{y}_i^{(d-1)}}\ \ell(Y_i,\ \hat{Y}_i^{(d-1)})$ and $h_i = \partial^2_{\hat{Y}_i^{(d-1)}}\ \ell(Y_i,\ \hat{Y}_i^{(d-1)})$ denote the first and second derivatives of the loss function $\ell$ with respect to the component corresponding to the predicted classifier.

Since the authors minimized (6) with respect to $f_d$, this expression can be simplified by removing the constant terms as follows:

$$\mathcal{L}^{(d)} = \sum\nolimits_{i=1}^{n} [g_i f_d(X_i) + \frac{1}{2} h_i {f_d}^2(X_i)] + ἡ(f_d). \tag{7}$$

Substituting the shrinkage penalty $ἡ$ of (4) in (7), the authors obtained:

$$\mathcal{L}^{(d)} = \sum\nolimits_{i=1}^{n} [g_i f_d(X_i) + \frac{1}{2} h_i {f_d}^2(X_i)] + uT + \frac{1}{2}\lambda\|w\|_2^2. \tag{8}$$

The *l2*-norm shown in (8) is equivalent to the sum of the squared weights of all $T$ leafs. Therefore (8) is expressed as:

$$\mathcal{L}^{(d)} = \sum\nolimits_{i=1}^{n} [g_i f_d(X_i) + \frac{1}{2} h_i {f_d}^2(X_i)] + u\mathrm{T} + \frac{1}{2}\lambda \sum\nolimits_{j=1}^{T} w_j^2. \tag{9}$$

The definition of $I_j = \{i|q(X_i)\}$, $I_j$ is the set of observations that are classified into one leaf $j$, $j$ = 1, ..., $T$. Each $I_j$ receives the same leaf weight $w_j$. Therefore, $\mathcal{L}^{(d)}$ in (9) can also be seen as an objective function that corresponds to each set $I_j$. In this sense, the $f_d(X_i)$, which is assigned to the observations, corresponds to the weight $w_j$ that is assigned to each set $I_j$. Therefore (9) is expressed as:

$$\mathcal{L}^{(d)} = \sum\nolimits_{j=1}^{T} \left[ \left( \sum\nolimits_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left( \sum\nolimits_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + u\mathrm{T}. \tag{10}$$

In order to find the optimal leaf weight $w_j^*$, the authors derived (10) with respect to $w_j$, let the new equation be equal to zero, and cleared the value of $w_j^*$. Then the authors obtained:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}. \tag{11}$$

The (10) was updated by replacing the new $w_j^*$. The next boosting iteration minimized the following objective function:

$$\begin{aligned} \hat{\mathcal{L}}^{(d)} &= \sum\nolimits_{j=1}^{T} \left[ \left( \sum\nolimits_{i \in I_j} g_i \right) \left( -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \right) + \frac{1}{2} \left( \sum\nolimits_{i \in I_j} h_i + \lambda \right) \left( -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} \right)^2 \right] + u\mathrm{T} \\ &= -\frac{1}{2} \sum\nolimits_{i=1}^{n} \frac{\left( \sum_{i \in I_j} g_i \right)^2}{\left( \sum_{i \in I_j} h_i + \lambda \right)} + u\mathrm{T}. \end{aligned} \tag{12}$$

Once the best objective function has been defined and the optimal leaf weights assigned to $I_j$, the best split procedure is considered. As (12) is derived for a wide range of functions, the authors were not able to identify all possible tree structures $q$ in each boosting iteration. This algorithm starts by building a single leaf and continues by adding new branches. Consider the following example:

Here, $I_L$ and $I_R$ are the sets of observations that are in the left and right parts of a node following a split. Therefore, $I = I_L + I_R$.

$$\hat{\mathcal{L}}^{(d)} = \frac{1}{2} \left[ -\sum\nolimits_{i=1}^{n} \frac{\left( \sum_{i \in I_j} g_i \right)^2}{\left( \sum_{i \in I_j} h_i + \lambda \right)} + \sum\nolimits_{i=1}^{n} \frac{\left( \sum_{i \in I_L} g_i \right)^2}{\left( \sum_{i \in I_L} h_i + \lambda \right)} + \sum\nolimits_{i=1}^{n} \frac{\left( \sum_{i \in I_R} g_i \right)^2}{\left( \sum_{i \in I_R} h_i + \lambda \right)} \right] - u, \tag{13}$$

$\hat{\mathcal{L}}^{(d)}$ of (13) is the node impurity measure, which is calculated for the $P$ covariates. The split is determined by the maximum value of (13). For example, in the case of CART algorithms, the impurity measure for categorical target variables can be information gain, Gini impurity or chi-square, while for continuous target variables it can be the Gini impurity.

Once the tree $f_d$ is completely built (i.e., its branches and leaf weights are established), observations are mapped on the tree (from the root to one corresponding leaf). Thus, the algorithms will update from (5) to (14) as many times as D boosting iterations are established and the final classification is the sum of the $D$ obtained functions which are shown in (3). Consequently, the XGBoost corrects the mistaken predictions in each iteration, as far as this is possible, and tends to overfit the data. Thus, to prevent overfitting, the regularization parameter value in the objective function is highly recommended.

### 2.2.2. Implementation

An example of R code is given in the Appendix A.

The implementation of XGBoost has proved to be quite effective for fitting real binary response data and a good method for providing a confusion matrix, i.e., a table in which observations and predictions are compared, with very few false positives and false negatives. However, since the final prediction of an XGBoost algorithm is the result of a sum of $D$ trees, the graphical representation and the interpretation of the impact of each covariate on the final estimated probability of occurrence may be less direct than in the linear or logistic regression models. For instance, if the final predictor is a combination of several trees, but each tree has a different structure (in the sense that each time the order of segmentation differs from that of the previous tree), the role of each covariate will depend on understanding how the covariate impacts the result in the previous trees and what the path of each observation is in each of the previous trees. Thus, in the XGBoost approach, it is difficult to isolate the effect on the expected response of one particular covariate compared to all the others.

Under certain circumstances, the XGBoost method can be interpreted directly. This happens when $f_d$ has analytical expressions that can easily be manipulated to compute $\sum_{d=1}^{D} f_d(X_i)$. One example is the linear booster, which means that each $f_d$ is a linear combination of the covariates rather than a tree-based classifier. In this case of a linear function, the final prediction is also a linear combination of the covariates, resulting from the sum of the weights associated with each covariate in each $f_d$.

The results for the true XGBoost predictive model classifier can easily be obtained in R with the xgboost package.

## 3. Data and Descriptive Statistics

Our case-study database comprised of 2767 drivers under 30 years of age who underwrote a pay-as-you-drive (PAYD) policy with a Spanish insurance company. Their driving activity was recorded using a telematics system. This information was collected from 1 January through 31 December 2011. The data set contained the following information about each driver: The insured's age (*age*), the age of the vehicle (*ageveh*) in years; the insured's gender (*male*); the driving experience (*drivexp*) in years; the percentage of total kilometers travelled in urban areas (*pkmurb*); the percentage of total kilometers travelled at night—that is, between midnight and 6 am (*pkmnig*); the percentage of kilometers above the mandatory speed limits (*pkmexc*); the total kilometers (*kmtotal*); and, finally, the presence of an accident claim with fault ($Y$) which was coded as 1 when, at least, one claim where the fault occurred in the observational period and was reported to the insurance company, and 0 otherwise. This study is interested in predicting Y using the aforementioned covariates. This data set has been extensively studied in Ayuso et al. (2014, 2016a, 2016b) and Boucher et al. (2017).

Table 1 shows the descriptive statistics for the accident claims data set. This highlighted that a substantial part of the sample did not suffer an accident in 2011, with just 7.05% of drivers reporting at least one accident claim. The insureds with no accident claim seemed to have travelled fewer kilometers than those presenting a claim. The non-occurrence of accident claims was also linked to a lower percentage of driving in urban areas and a lower percentage of kilometers driven above mandatory speed limits. In this dataset, 7.29% of men and 6.79% of women had an accident during the observation year.

The data set was divided randomly into a training data set of 1937 observations (75% of the total sample) and a testing data set of 830 observations (25% of the total sample). The function CreateDataPartion of R was used to maintain the same proportion of events (coded as 1) of the total sample in both the training and testing data sets.

**Table 1.** The description of the variables in the accident claims data set [1].

| Variables | | Non-Occurrence of Accident Claims (Y = 0) | Occurrence of Accident Claims (Y = 1) | Total |
|---|---|---|---|---|
| Age (years) | | 25.10 | 24.55 | 25.06 |
| Gender | Female | 1263 (93.21%) | 92 (6.79%) | 1355 |
| | Male | 1309 (92.71%) | 103 (7.29%) | 1412 |
| Driving experience (years) | | 4.98 | 4.46 | 4.94 |
| Age of vehicle (years) | | 6.37 | 6.17 | 6.35 |
| Total kilometers travelled | | 7094.63 | 7634.97 | 7132.71 |
| Percentage of total kilometers travelled in urban areas | | 24.60 | 26.34 | 24.72 |
| Percentage of total kilometers above the mandatory speed limit | | 6.72 | 7.24 | 6.75 |
| Percentage of total kilometers travelled at night | | 6.88 | 6.66 | 6.86 |
| Total number of cases | | 2572 (92.95%) | 195 (7.05%) | 2767 |

[1] The mean of the variables according to the occurrence and non-occurrence of accident claims. The absolute frequency and row percentage is shown for the variable gender.

## 4. Results

In this section, the results obtained in the training and testing samples were compared when employing the methods described in Section 2.

### *4.1. Coefficient Estimates*

Table 2 presents the estimates obtained using the two methods. It is noted, however, that the values are not comparable in magnitude as they correspond to different specifications. The logistic regression uses its classical standard method to compute the coefficients of the variables and their standard errors. However, the boosting process of the XGBoost builds $D$ models in reweighted versions and, therefore, a historical record of the $D$ times $P + 1$ coefficient estimates was obtained. XGBoost can only obtain a magnitude of those coefficients if the base learner allows it, and this is not the case when $f_d$ are CART models.

The signs obtained by the logistic regression point estimate and the mean of the XGBoost coefficients are the same. The inspection of the results in Table 2 shows that older insureds are less likely to suffer a motor accident than younger policy holders[4]. In addition, individuals who travel more kilometers in urban areas are more likely to have an accident than those that travel fewer kilometers in urban areas. The authors were not able to interpret the coefficients of the XGBoost, but by inspecting the maximum and minimum values of the linear booster case, an idea of how the estimates fluctuate until iteration $D$ was obtained.

Only the coefficients of age and percentage of kilometers travelled in urban areas were significantly different from zero in the logistic regression model, but the authors preferred to keep all the coefficients of the covariates in the estimation results to show the general effect of the telematics covariates on the occurrence of accident at-fault claims in this dataset, and to evaluate the performance of the different methods in this situation.

[4] In general, this is only partially true. The relation of the variable age is typically non-linear, U-shaped, as (very) young drivers also cause a lot of accidents. The maximum age in this sample is 30 and so, even if models with age and age$^2$ were estimated, the results did not change substantially.

**Table 2.** The parameter estimates of the logistic regression and XGBoost with linear booster.

| Training Data Set | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Parameter Estimates** | **Logistic Regression** | | | | **XGBoost (Linear Booster)** | | |
| | **Lower Bound** | **Estimate** | **Upper Bound** | ***p*-Value** | **Minimum** | **Mean** | **Maximum** |
| Constant | −2.8891 | −0.5442 | 1.8583 | 0.6526 | −2.6760 | −2.6690 | −1.7270 |
| * age | −0.2059 | −0.0994 | 0.0011 | 0.0591 | −0.2573 | −0.2416 | −0.0757 |
| drivexp | −0.1285 | −0.0210 | 0.0906 | 0.7060 | −0.0523 | −0.0517 | −0.0069 |
| ageveh | −0.0786 | −0.0249 | 0.0257 | 0.3481 | −0.0897 | −0.0885 | −0.0220 |
| male | −0.3672 | 0.0039 | 0.3751 | 0.9837 | 0.0019 | 0.0020 | 0.0070 |
| kmtotal | −0.0203 | 0.0266 | 0.2505 | 0.0137 | 0.1164 | 0.1176 | |
| pkmnig | −0.0354 | −0.0046 | 0.0239 | 0.7625 | −0.0292 | −0.0290 | −0.0061 |
| pkmexc | −0.0122 | 0.0144 | 0.0385 | 0.2650 | 0.0180 | 0.1007 | 0.1016 |
| * pkmurb | 0.0002 | 0.0146 | 0.0286 | 0.0425 | 0.0436 | 0.2008 | 0.2023 |

In the logistic regression columns, the point estimates are presented with the lower and upper bound of a 95% confidence interval. In the XGBoost columns, the means of the coefficient estimates with a linear boosting of the *D* iterations are presented. Similarly, bounds are presented with the minimum and maximum values in the iterations. There are no regularization parameter values. * Indicates that the coefficient is significant at the 90% confidence level in the logistic regression estimation. The calculations were performed in R and scripts are available from the authors.

Figure 1 shows the magnitude of all the estimates of the XGBoost in 200 iterations. From approximately the tenth iteration, the coefficient estimates tend to become stabilized. Thus, no extreme changes were present during the boosting.

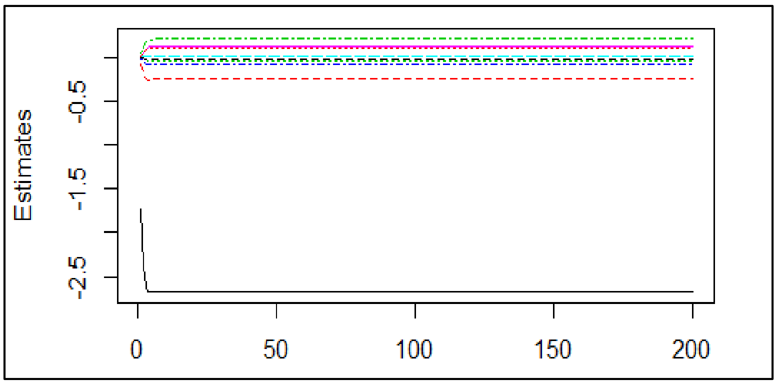

**Figure 1.** The magnitude of all the estimates in the $D = 200$ iterations. The different colors indicate each of the coefficients in the XGBoost iteration.

## *4.2. Prediction Performance*

The performance of the two methods was evaluated using the confusion matrix, which compares the number of observed events and non-events with their corresponding predictions. Usually, the larger the number of correctly classified responses, the better the model. However, the out-of-sample performance was even more important than in-sample results. This means that the classifier must be able to predict the observed events and non-events in the testing sample and not just in the training sample.

The predictive measures used to compare the predictions of the models were sensitivity, specificity, accuracy and the root mean square error (RMSE). Sensitivity measures the proportion of actual positives that are classified correctly as such, i.e., True positive/(True positive + False negative). Specificity measures the proportion of actual negatives that are classified correctly as such, i.e., True negative/(True negative + False positive). Accuracy measures the proportion of total cases classified correctly (True positive + True negative)/Total cases. RMSE measures the distance between the observed and predicted values of the response. It is calculated as follows:

$$\sqrt{\sum_{i=1}^{n} \frac{\left(Y_i - \hat{Y}_i\right)^2}{n}}, \tag{14}$$

The higher the sensitivity, the specificity and the accuracy, the better the models predict the outcome variable. The lower the value of RMSE, the better the predictive performance of the model.

Table 3 presents the confusion matrix and the predictive measures of the methods (the logistic regression, XGBoost with a tree booster and XGBoost with a linear booster) for the training and testing samples. The results in Table 3 indicated that the performance of the XGBoost with the linear booster (last column) was similar to that of the logistic regression both in the training and testing samples[5]. XGBoost using the tree approach provided good accuracy and a good RMSE value in the training sample, but did not perform as well as the other methods in the case of the testing sample. More importantly, XGBoost failed to provide good sensitivity. In fact, the XGBoost with the tree booster clearly overfitted the data, because while it performed very well in the training sample, it failed to do so in the testing sample. For instance, sensitivity was equal to 100% in the training sample for the XGBoost tree booster methods, but it was equal to only 7.9% in the testing sample.

**Table 3.** The confusion matrix and predictive measures of the logistic regression, XGBoost with a tree booster and XGBoost with a linear booster for the testing and training data sets.

| **Testing Data Set** | | | |
|---|---|---|---|
| **Predictive Measures** | **Logistic Regression** | **XGBoost (Tree Booster)** | **XGBoost (Linear Booster)** |
| $Y_i = 0, \hat{Y}_i = 0$ | 524 | 692 | 516 |
| $Y_i = 1, \hat{Y}_i = 0$ | 38 | 58 | 38 |
| $Y_i = 0, \hat{Y}_i = 1$ | 243 | 75 | 251 |
| $Y_i = 1, \hat{Y}_i = 1$ | 25 | 5 | 25 |
| Sensitivity | 0.3968 | 0.0790 | 0.3968 |
| Specificity | 0.6831 | 0.9022 | 0.6728 |
| Accuracy | 0.6614 | 0.8397 | 0.6518 |
| RMSE | 0.2651 | 0.2825 | 0.2651 |
| **Training Data Set** | | | |
| **Predictive Measures** | **Logistic Regression** | **XGBoost (Tree Booster)** | **XGBoost (Linear Booster)** |
| $Y_i = 0, \hat{Y}_i = 0$ | 1030 | 1794 | 1030 |
| $Y_i = 1, \hat{Y}_i = 0$ | 55 | 0 | 55 |
| $Y_i = 0, \hat{Y}_i = 1$ | 775 | 11 | 775 |
| $Y_i = 1, \hat{Y}_i = 1$ | 77 | 132 | 77 |
| Sensitivity | 0.5833 | 1.0000 | 0.5833 |
| Specificity | 0.5706 | 0.9939 | 0.5706 |
| Accuracy | 0.5715 | 0.9943 | 0.5715 |
| RMSE | 0.2508 | 0.0373 | 0.2508 |

The threshold used to convert the continuous response into a binary response is the mean of the outcome variable. The authors performed the calculations.

It cannot be concluded from the foregoing, however, that XGBoost has a poor relative predictive capacity. Model-tuning procedures have not been incorporated in Table 3. However, tuning offers the possibility of improving the predictive capacity by modifying some specific parameter estimates.

[5] This is not surprising because XGBoost (linear) is a combination of linear probability models.

The following are some of the possible tuning actions that could be taken: Fixing a maximum for the number of branches of the tree (maximum depth); establishing a limited number of iterations of the boosting; or fixing a number of subsamples in the training sample. The `xgboost` package in R denotes these tuning options as general parameters, booster parameters, learning task parameters, and command line parameters, all of which can be adjusted to obtain different results in the prediction.

Figure 2 shows the ROC curve obtained using the three methods on the training and testing samples. This study confirmed that the logistic regression and XGBoost (linear) have a similar predictive performance. The XGBoost (tree) presented an outstanding AUC in the case of the training sample, and the same value as the logistic regression in the testing sample. However, as discussed in Table 3, it failed to maintain this degree of sensitivity when this algorithm is used with new samples.

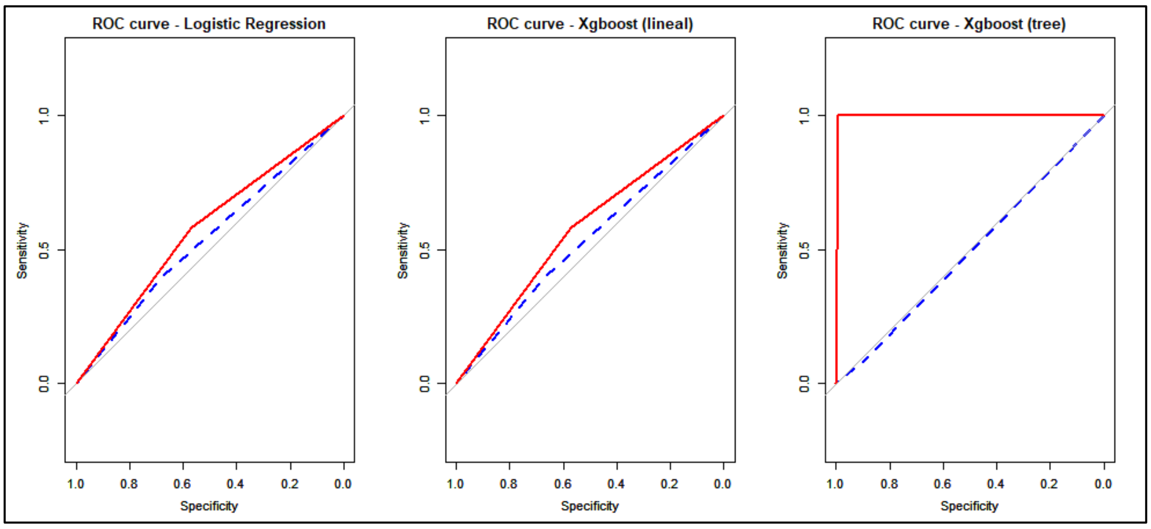

**Figure 2.** The receiver operating characteristics (ROC) curve obtained using the three methods on the training and testing samples. The red solid line represents the ROC curve obtained by each method in the training sample, and the blue dotted line represents the ROC curve obtained by each method in the testing sample. The area under the curve (AUC) is 0.58 for the training sample (T.S) and 0.49 for the testing sample (Te.S) when logistic regression is used; 0.58 for the T.S and 0.53 for the Te.S when XGBoost (linear booster) is used; and, 0.997 for the T.S and 0.49 for the Te.S when the XGBoost (tree booster) is used.

## 4.3. *Correcting the Overfitting*

One of the most frequently employed techniques for addressing the overfitting problem is regularization. This method shrinks the magnitude of the coefficients of the covariates in the modelling as the value of the regularization parameter increases.

In order to determine whether the XGBoost (tree booster) can perform better than the logistic regression model, a simple sensitivity analysis of the regularization parameters was proposed. In so doing, the evolution of the following confusion matrix measures was evaluated: Accuracy, sensitivity and specificity—according to some given regularization parameter values for the training and the testing sample—and, finally, the regularization parameter was chosen that gives the highest predictive measures in the training and testing samples.

Two regularization methods were considered. First, the L2 (Ridge) was considered, which is Chen and Guestrin (2016) original proposal and takes the $l2$-norm of the leaf weights. It has a parameter $\lambda$ that is multiplied to the $l2$-norm. Second, the L1 (Lasso) method was considered, which is an additional implementation possibility of the `xgboost` package in R that takes the $l1$-norm of the leaf weights. It has a parameter $\alpha$ that is multiplied to the $l1$-norm. Consequently, $\lambda$ and calibrated the regularization term in (4). For simplicity, no tree pruning was implemented, so $\mu = 0$ in (4).

The values of $\alpha$ and $\lambda$ should be as small as possible, because they add bias to the estimates, and the models tend to become underfitted as the values of the regularization parameters become larger. For this reason, their changes were evaluated in a small interval. Figure 3 shows the predictive measures for the testing and training samples according to the values of $\alpha$ when the L1 regularization method was implemented. When $\alpha = 0$, exactly the same predictive measure values as in Table 3 (column 3) were obtained because the objective function had not been regularized. As the value of $\alpha$ increased, the models' accuracy and sensitivity values fell sharply—to at least $\alpha \simeq 0.06$ in the training sample. In the testing sample, the fall in these values was not as pronounced. However, when $\alpha$ was lower than 0.06, the specificity performance was the lowest of the three measures. Moreover, selecting an $\alpha$ value lower than 0.05 resulted in higher accuracy and sensitivity measures, but lower specificity. In contrast, when $\alpha$ equaled 0.06 in the testing sample, the highest specificity level of 0.5079 was obtained, with corresponding accuracy and sensitivity values of 0.5892 and 0.5988, respectively. In the training sample, when $\alpha = 0.06$ the specificity, accuracy and sensitivity were: 0.7227, 0.6086, and 0.6000, respectively. As a result, when was fixed at 0.06, the model performed similarly in both the testing and training samples.

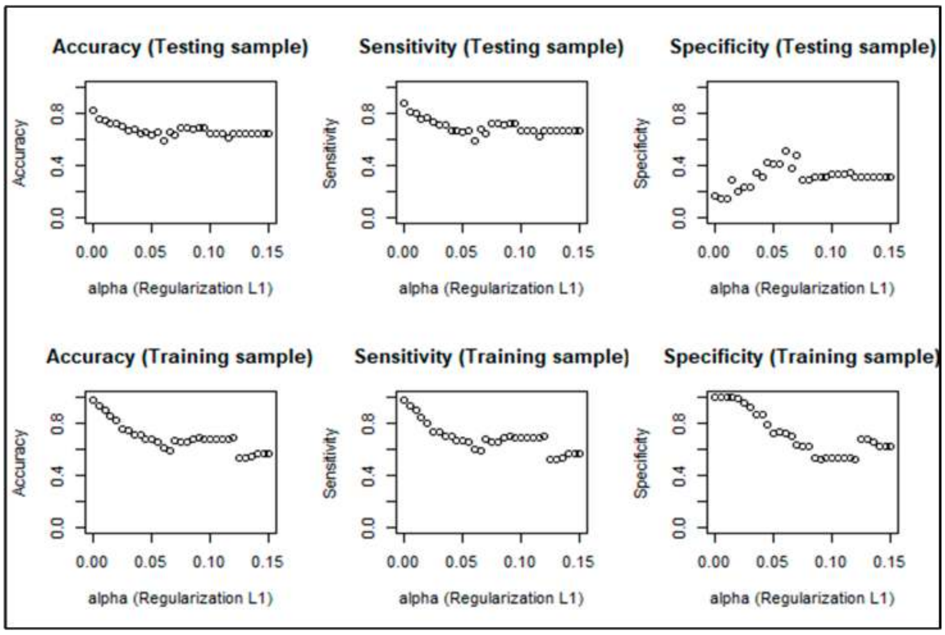

**Figure 3.** The predictive measures according to $\alpha$. L1 method applied to the training and testing samples.

Thus, with the L1 regularization method ($\alpha = 0.06$), the new model recovered specificity, but lost some sensitivity when compared with the performance of the first model in Table 3, for which no regularization was undertaken. Thus, the authors concluded that $\alpha = 0.06$ which can be considered as providing the best trade-off between correcting for overfitting while only slightly reducing the predictive capacity.

Figure 4 shows the predictive measures for the testing and training samples according to the values of $\lambda$ when the L2 regularization method is implemented. From $\lambda = 0$ to $\lambda = 0.30$. all predictive measures were approximately 100% in the training sample. However, very different results were recorded in the testing sample. Specifically, accuracy and sensitivity fell slowly, but specificity was low—there being no single $\lambda$ that made this parameter exceed at least 20%. Therefore, no $\lambda$ could help improve specificity in the testing sample. The L2 regularization method did not seem to be an effective solution to correct the problem of overfitting in our case study data set.

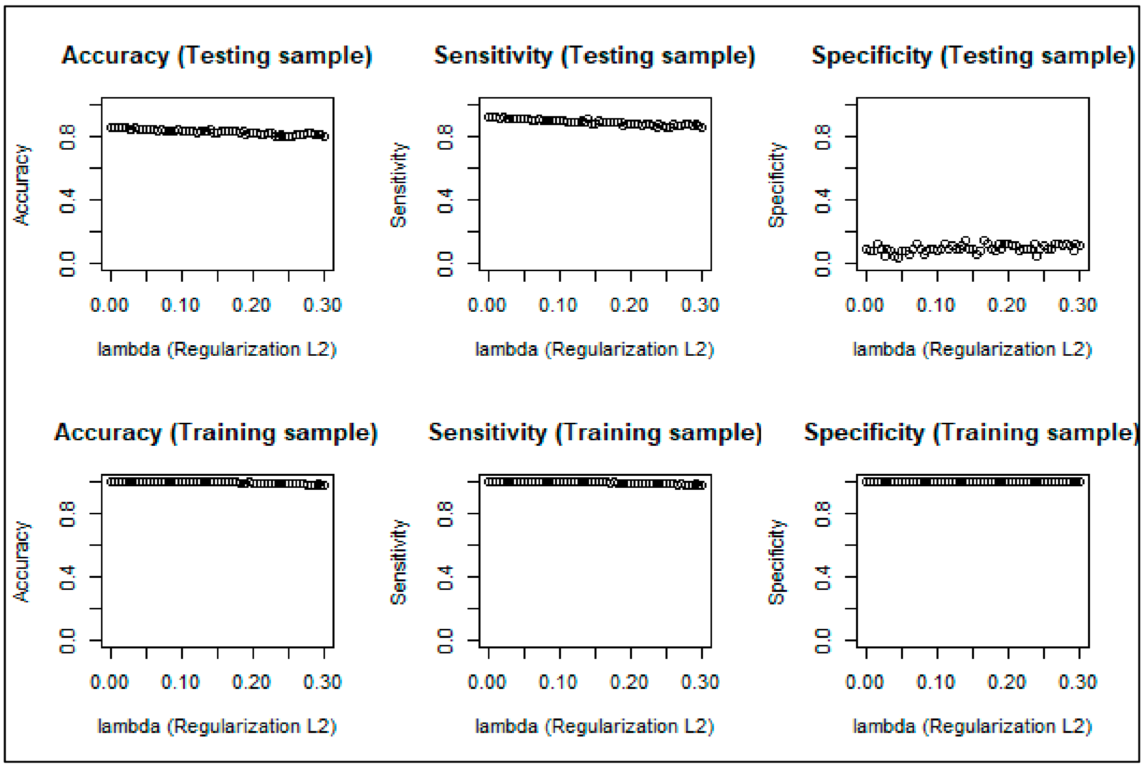

**Figure 4.** The predictive measures according to $\lambda$. L2 method applied to the training and testing samples.

The difference in outcomes recorded between the L1 and L2 regularization approaches might also be influenced by the characteristics of each regularization method. Goodfellow et al. (2016) and Bishop (2007) explained that L1 penalizes the sum of the absolute value of the weights, and that it seems to be robust to outliers, has feature selection, provides a sparse solution, and is able to give simpler but interpretable models. In contrast, L2 penalizes the sum of the square weights, has no feature selection, is not robust to outliers, is more able to provide better predictions when the response variable is a function of all input variables, and is better able to learn more complex models than L1.

### 4.4. Variable Importance

Variable importance or feature selection is a technique that measures the contribution of each variable or feature to the final outcome prediction based on the Gini impurity. This method is of great relevance in tree models because it helps identify the order in which the leaves appear in the tree. The tree branches (downwards) begin with the variables that have the greatest effect and end with those that have the smallest effect (for further details see, for example, Kuhn and Johnson 2013).

Table 4 shows the three most important variables for each method. The two agree on the importance of the percentage of total kilometers travelled in urban areas as a key factor in predicting the response variable. Total kilometers driven and age only appeared among the top three variables in the case of logistic regression, while the percentage of kilometers travelled over the speed limits and the percentage of kilometers driven at night appeared among the most important variables in the case of the XGBoost method.

**Table 4.** Variable Importance. The most relevant variables of the different methods.

| Level of Importance | Logistic Regression | XGBoost (Tree Booster) |
|---|---|---|
| First | percentage of total kilometers travelled in urban areas | percentage of kilometers above the mandatory speed limits |
| Second | age | percentage of total kilometers travelled in urban areas |
| Third | total kilometers | percentage of total kilometers travelled at night |

## 5. Conclusions

XGBoost, and other boosting models, are dominant methods today among machine-learning algorithms and are widely used because of their reputation for providing accurate predictions. This novel algorithm is capable of building an ensemble model characterized by an efficient learning method that seems to outperform other boosting-based predictive algorithms. Unlike the majority of machine learning methods, XGBoost is able to compute coefficient estimates under certain circumstances and, therefore, the magnitude of the effects can be studied. The method allows the analyst to measure not only the final prediction, but also the effect of the covariates on a target variable at each iteration of the boosting process, which is something that traditional econometric models (e.g., generalized linear models) do in one single estimation step.

When a logistic regression and XGBoost compete to predict the occurrence of accident claims without model-tuning procedures, the predictive performance of the XGBoost (tree booster) was much higher than the logistic regression in the training sample, but considerably poorer in the testing sample. Thus, a simple regularization analysis has been proposed here to correct this problem of overfitting. However, the improvement in predictive performance of the XGBoost following this regularization was similar to that obtained by the logistic regression. This means additional efforts have to be taken to tune the XGBoost model to obtain a higher predictive performance without overfitting the data. This might be considered as the trade-off between obtaining a better performance, and the simplicity it provides for interpreting the effect of the covariates.

Based on our results, the classical logistic regression model can predict accident claims using telematics data and provided a straightforward interpretation of the coefficient estimates. Moreover, the method offered a relatively high predictive performance considering that only two coefficients were significant at the 90% confidence level. These results are not bettered by the XGBoost method.

When the boosting framework of XGBoost is not based on a linear booster, interpretability becomes difficult, as a model's coefficient estimates cannot be calculated. In this case, variable importance can be used to evaluate the weight of the individual covariates in the final prediction. Here, different conclusions were obtained for the two methods employed. Thus, given that the predictive performance of XGBoost was not much better than the logistic regression, even after careful regularization, the authors concluded that the new methodology needs to be adopted carefully, especially in a context where the number of event responses (accident) is low compared to the opposite response (no accident). Indeed, this phenomenon of unbalanced response is attracting more and more attention in the field of machine learning, because it is known that machine learning algorithms do not work well in datasets with imbalanced responses (He and Garcia 2008). XGBoost might perform better in other problems, especially when the number of events and no events are balanced. The reputation of XGBoost (tree booster) may be due to its capacity of accuracy. In our case study, XGBoost has proven the highest accuracy in the testing and training data sets, but it does not seem to be effective for sensitivity.

For future work, it would be interesting to see bigger datasets with thousands of explanatory variables to conclude whether or not XGBoost has better predictive performance than a regularized version of logistic regression. Similarly, it would also be interesting to see comparative studies for other machine learning approaches using this dataset, including but not limited to neural network approaches.

**Author Contributions:** All authors contributed equally to the conceptualization, methodology, software, validation, data curation, writing—review and editing, visualization, and supervision of this paper.

**Acknowledgments:** We thank the Spanish Ministry of Economy, FEDER grant ECO2016-76203-C2-2-P. The second author gratefully acknowledges financial support received from ICREA under the ICREA Academia Program.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Appendix A

**R Code**

```
# Loading data
    load("data.Rdata")
    x<-data
# Training and test data sets
# We divide 70% of the data set as training, and 30% as testing
    library(caret)
    part<-createDataPartition(x$Y,p = 0.70, list = F)
    train.set<-x[part,] # training data set
    train.set<-train.set()[−1]
    testing.set<- x[-part,] # testing data set
    testing.set<-testing.set()[−1]
## First Method: Logistic Regression
    logistic1 <- glm(factor(train.set$Y) ~ x2+I(x2^2)+x3+x4+factor(x1)+x5+x6+x7+x8,
    data = train.set,family = binomial(link = 'logit'))
    summary(logistic1)
# Predicting the output with the testing data set
    predicted.log.test <- predict(logistic1,testing.set, type = 'response')
# Predicting the output with the training data set
    predicted.log1.train<- predict(logistic1,train.set, type = 'response')
# Variable Importance
    varImp(logistic1)
## Second Method: XGBoost (tree booster)
    library(xgboost)
    library(Matrix)
# Function xgboost requires sparsing data first
    sparse_xx.tr<- sparse.model.matrix(Y ~ x2+I(x2^2)+x3+x4+factor(x1)+x5+x6+x7+x8, data = train.set)
    sparse_xx.te<- sparse.model.matrix(Y ~ x2+I(x2^2)+x3+x4+factor(x1)+x5+x6+x7+x8,
    data = testing.set)
    xgboost_reg <- xgboost(data = sparse_xx.tr, label = train.set$Y, objective = "binary:logistic",
    nrounds = 100, verbose = 1)
# Predicting the output with testing data set
    pred.xgboost.test<- predict(xgboost_reg,sparse_xx.te, outputmargin = F)
# Predicting the output with training data set
    pred.xgboost.train<-predict(xgboost_reg,sparse_xx.tr, outputmargin = F)
# Variable Importance
    importance <- xgb.importance(feature_names = sparse_xx.tr@Dimnames[(2)],
    model = xgboost_reg)
```

## References

Ayuso, Mercedes, Montserrat Guillén, and Ana María Pérez-Marín. 2014. Time and distance to first accident and driving patterns of young drivers with pay-as-you-drive insurance. *Accident Analysis and Prevention* 73: 125–31. [CrossRef] [PubMed]

Ayuso, Mercedes, Montserrat Guillén, and Ana María Pérez-Marín. 2016a. Using GPS data to analyse the distance travelled to the first accident at fault in pay-as-you-drive insurance. *Transportation Research Part C* 68: 160–67. [CrossRef]

Ayuso, Mercedes, Montserrat Guillén, and Ana María Pérez-Marín. 2016b. Telematics and gender discrimination: some usage-based evidence on whether men's risk of accident differs from women's. *Risks* 4: 10. [CrossRef]

Bishop, Christopher M. 2007. Pattern recognition and machine learning. *Journal of Electronic Imaging* 16: 049901. [CrossRef]

Boucher, Jean-Philippe, Steven Côté, and Montserrat Guillen. 2017. Exposure as duration and distance in telematics motor insurance using generalized additive models. *Risks* 5: 54. [CrossRef]

Chen, Tianqi, and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York: ACM, pp. 785–94. [CrossRef]

De Boer, Pieter-Tjerk, Dirk P. Kroese, Shie Mannor, and Reuven Y. Rubinstein. 2005. A tutorial on the Cross Entropy Method. *Annals of Operations Research* 134: 19–67. [CrossRef]

Dietterich, Thomas G., Pedro Domingos, Lise Getoor, Stephen Muggleton, and Prasad Tadepalli. 2008. Structured machine learning: The next ten years. *Machine Learning* 73: 3–23. [CrossRef]

Elliott, Graham, and Allan Timmermann. 2003. *Handbook of Economic Forecasting*. Amsterdam: Elsevier.

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. 2001. *The Elements of Statistical Learning*. New York: Springer.

Gao, Guangyuan, and Mario V. Wüthrich. 2018. Feature extraction from telematics car driving heatmaps. *European Actuarial Journal* 8: 383–406. [CrossRef]

Gao, Guangyuan, and Mario V. Wüthrich. 2019. Convolutional neural network classification of telematics car driving data. *Risks* 7: 6. [CrossRef]

Gao, Guangyuan, Shengwang Meng, and Mario V. Wüthrich. 2019. Claims frequency modeling using telematics car driving data. *Scandinavian Actuarial Journal* 2019: 143–62. [CrossRef]

Gomez-Verdejo, Vanessa, Jeronimo Arenas-Garcia, Manuel Ortega-Moral, and Anıbal R. Figueiras-Vidal. 2005. Designing RBF classifiers for weighted boosting. *IEEE International Joint Conference on Neural Networks* 2: 1057–62. [CrossRef]

Goodfellow, Ian, Bengio Yoshua, and Courville Aaron. 2016. *Deep Learning*. Chenai: MIT Press.

Greene, William. 2002. *Econometric Analysis*, 2nd ed. New York: Chapman and Hall.

Guillen, Montserrat, Jens Perch Nielsen, Mercedes Ayuso, and Ana M. Pérez-Marín. 2019. The use of telematics devices to improve automobile insurance rates. *Risk Analysis* 39: 662–72. [CrossRef] [PubMed]

Hastie, Trevor, Rob Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Prediction, Inference and Data Mining*. New York: Springer.

He, Haibo, and Edwardo A. Garcia. 2008. Learning from imbalanced data. *IEEE Transactions on Knowledge & Data Engineering* 9: 1263–84. [CrossRef]

Huang, Jianhua Z., and Lijian Yang. 2004. Identification of non-linear additive autoregressive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 66: 463–77. [CrossRef]

Hultkrantz, Lars, Jan-Eric Nilsson, and Sara Arvidsson. 2012. Voluntary internalization of speeding externalities with vehicle insurance. *Transportation Research Part A: Policy and Practice* 46: 926–37. [CrossRef]

Ivanov, Valentin K., Vladimir V. Vasin, and Vitalii P. Tanana. 2013. *Theory of Linear Ill-Posed Problems and Its Applications*. Zeist: VSP.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. New York: Springer, vol. 112, p. 18.

Kuhn, Max, and Kjell Johnson. 2013. *Applied Predictive Modeling*. New York: Springer, vol. 26.

Lee, Simon, and Katrien Antonio. 2015. Why High Dimensional Modeling in Actuarial Science? Paper presented at Actuaries Institute ASTIN, AFIR/ERM and IACA Colloquia, Sydney, Australia, August 23–27. Available online: https://pdfs.semanticscholar.org/ad42/c5a42642e75d1a02b48c6eb84bab87874a1b.pdf (accessed on 8 May 2019).

Lee, Simon CK, and Sheldon Lin. 2018. Delta boosting machine with application to general insurance. *North American Actuarial Journal* 22: 405–25. [CrossRef]

Natekin, Alexey, and Alois Knoll. 2013. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics* 7: 21. [CrossRef] [PubMed]

McCullagh, Peter, and John Nelder. 1989. *Generalized Linear Models*, 2nd ed. New York: Chapman and Hall.

Pérez-Marín, Ana M., and Montserrat Guillén. 2019. Semi-autonomous vehicles: Usage-based data evidences of what could be expected from eliminating speed limit violations. *Accident Analysis and Prevention* 123: 99–106. [CrossRef] [PubMed]

Schapire, Robert E., and Yoav Freund. 2012. *Boosting: Foundations and Algorithms*. Cambridge: MIT Press.

Steinwart, Ingo, and Andreas Christmann. 2008. *Support Vector Machines*. New York: Springer Science & Business Media.

Tikhonov, Andrej-Nikolaevich, and Vasiliy-Yakovlevich Arsenin. 1977. *Solutions of Ill-Posed Problems*. New York: Wiley.

Verbelen, Roel, Katrien Antonio, and Gerda Claeskens. 2018. Unraveling the predictive power of telematics data in car insurance pricing. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 67: 1275–304. [CrossRef]

Wüthrich, Mario V. 2017. Covariate selection from telematics car driving data. *European Actuarial Journal* 7: 89–108. [CrossRef]