Article

Predicting motor insurance claims using telematics data - XGBoost vs. logistic regression

Jessica Pesantez-Narvaez, Montserrat Guillen and Manuela Alcañiz

Dept. Econometrics, Riskcenter-IREA, Universitat de Barcelona; jessica.pesantez@ub.edu, (J.P-N) https://orcid.org/0000-0003-3161-7807; mguillen@ub.edu (M.G); https://orcid.org/0000-0002-2644-6268; malcaniz@ub.edu (M.A) https://orcid.org/0000-0002-5028-1926

* Correspondence: mguillen@ub.edu; Tel.: +34-934-037-039

Abstract:

XGBoost is recognized as an algorithm with exceptional predictive capacity. Models for a binary response indicating the existence of accident claims vs. no claims can be used to identify the determinants of traffic accidents. We compare the relative performances of logistic regression and XGBoost approaches for predicting the existence of accident claims using telematics data. The dataset contains information from an insurance company about individuals' driving patterns – including total annual distance driven and percentage of total distance driven in urban areas. Our findings show 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 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, we consider the recently proposed methods of XGBoost (Chen and Guestrin 2016) and logistic regression and compare their predictive performance in a sample of insured drivers, for whom we have telematic information.

We discuss the advantages and disadvantages of XGBoost compared to logistic regression and we show that a slightly improved predictive power is only obtained with the XGBoost method, but this complicates 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. We discuss, 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. We do not examine any other boosting methods here and remind readers that excellent descriptions can be found in Lee and Lin (2018), while extensions to high dimensional datasets are presented in Lee and Antonio (2016), both of which present cases studies of insurance applications. Many of those alternatives place 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, we use a real dataset comprising motor insurance policy holders and their telematics measurements, 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-Marin and Guillen (2019) show the potential benefits of analyzing telematics information when calculating motor insurance premiums. Further, Hultkrantz et al. (2012) highlight 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, we introduce the notation and outline the logistic regression and XGBoost methods. Second, we describe our dataset and provide some descriptive statistics. Third, we report the results of our comparisons in both a training and a testing sample. Finally, we conclude and offer some practical suggestions about the feasibility of applying new machine learning methods to the field of insurance.

2. Methodology description

Let us suppose that in a data set of n individuals and P covariates, we have 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, we say that $\text{Prob}(Y_i= t)=h_t(X_i)$, and that $\text{E}(Y_i)=\text{Prob}(Y_i= 1)=h_1(X_i)$.

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. What distinguishes a logistic regression from a classical linear regression model is primarily that the response variable is binary rather than continuous in nature.

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

$$ln\frac{h_1(X_i)}{h_0(X_i)} = ln\frac{\text{Prob}(Y_i= 1)}{\text{Prob}(Y_i= 0)} = \beta_0 + \sum_{p=1}^{P} X_{ip}\beta_p, \quad (1)$$

where β_0, $\beta_1,\ldots,\beta_P$ are the model coefficients[1], and $\text{Prob}(Y_i= 1)$ is the probability of observing the event in the response (response equal to 1), and $\text{Prob}(Y_i= 0)$ 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:

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

A logistic regression can be estimated by 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 model, a positive estimated coefficient indicates a positive association; thus, when the corresponding covariate increases, the probability of the event response also increases. Correspondingly, if the estimated coefficient is negative then the association is negative and, so, 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.

[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.

Apart from their interpretability, the popularity of logistic regression models is based on two characteristics: (i) 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 order 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:

$$\hat{Y}_i = \text{₣}(X_i) = \sum_{d=1}^{D} f_d(X_i), \quad f_d \in \text{F}, i = 1, \dots, n \tag{3}$$

where F is the function space[3] of the CART models, and each f_d corresponds to an independent CART structure which we denote 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. We denote this score as $w_{q(X)}$.

Thus, we can write q as a function q: $\mathbb{R}^P \rightarrow T$, where T is the total number of leafs of a tree and j is later used to denote a particular leaf, $j=1,\dots,T$. To calculate the final prediction for each individual, we sum the score of the leafs as in (3), where $\text{F} = \{f(\text{X}) = w_{q(X)}\}$, 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, $d=1,\dots,D$) 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:

$$\mathcal{L} = \sum_{i=1}^{n} \ell\left(Y_i, \hat{Y}_i\right) + \sum_{d=1}^{D} \acute{\eta}(f_d), \tag{4}$$

where ℓ is a convex loss function that measures the difference between the observed response Y_i and predicted response $\hat{Y}_i$ and $\acute{\eta} = \mu\text{T}+\frac{1}{2}\lambda\|w\|_2^2$, $\acute{\eta}$ 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 μ regulates the depth of the tree and λ 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.

Note 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

---

[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. 2002); 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2019 doi:10.20944/preprints201905.0122.v1

Peer-reviewed version available at Risks 2019, 7, 70; doi:10.3390/risks7020070

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 2003).

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. 2015) and the exponential loss (Elliott and Timmermann 2013).

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 ή, the smaller the variability of the scores (Goodfellow et al. 2016).

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

$$\mathcal{L}^{(d)} = \sum_{i=1}^{n} \ell\left(Y_i, \hat{Y}_i^{(d-1)} + f_d(X_i)\right) + \acute{\eta}(f_d), \qquad (5)$$

where $\hat{Y}_i^{(d-1)}$ is the prediction of the *i-th* observation at the *(d-1)-th* iteration. Note 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), we assume 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:

$$\mathcal{L}^{(d)} \cong \sum_{i=1}^{n}[\ \ell\left(Y_i, \hat{Y}_i^{(d-1)}\right) + g_i f_d\ (X_i) + \frac{1}{2} h_i\, {f_d}^2(X_i)] + \acute{\eta}(f_d), \qquad (6)$$

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

Since we minimize (6) with respect to f_d, we can simplify this expression by removing constant terms as follows:

$$\mathcal{L}^{(d)} = \sum_{i=1}^{n}[g_i\, f_d(X_i) + \frac{1}{2} h_i\, {f_d}^2(X_i)] + \acute{\eta}(f_d). \qquad (7)$$

Substituting the shrinkage penalty ή of (4) in (7), we obtain:

$$\mathcal{L}^{(d)} = \sum_{i=1}^{n}[g_i f_d(X_i) + \frac{1}{2} h_i\, {f_d}^2(X_i)] + \mu\mathrm{T} + \frac{1}{2}\lambda\|w\|_2^2. \qquad (8)$$

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

$$\mathcal{L}^{(d)} = \sum_{i=1}^{n}[g_i f_d(X_i) + \frac{1}{2} h_i\, {f_d}^2(X_i)] + \mu\mathrm{T} + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2. \qquad (9)$$

Now, let us define $I_j = \{i | q(X_i)\}$, I_j is the set of observations that are classified into one leaf j, $j=1,\ldots,T$. Each I_j receives the same leaf weight w_j. So $\mathcal{L}^{(d)}$ 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:

$$\mathcal{L}^{(d)} = \sum_{j=1}^{T}\left[\left(\sum_{i\epsilon I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\epsilon I_j} h_i + \lambda\right) w_j^2\right] + \mu\mathrm{T}. \qquad (10)$$

In order to find the optimal leaf weight w_j^*, we derive (10) with respect to w_j, let the new equation be equal to zero, and clear the value of w_j^*. Then we obtain:

$$w_j^* = -\frac{\sum_{i\,\epsilon\, I_j} g_i}{\sum_{i\,\epsilon\, I_j} h_i + \lambda}\ . \qquad (11)$$

So we update (10) by replacing the new w_j^*. The next boosting iteration will minimize the following objective function:

$$\hat{\mathcal{L}}^{(d)} = \sum_{j=1}^{T}\left[\left(\sum_{i\,\epsilon\, I_j} g_i\right)\left(-\frac{\sum_{i\,\epsilon\, I_j} g_i}{\sum_{i\,\epsilon\, I_j} h_i + \lambda}\right) + \frac{1}{2}\left(\sum_{i\,\epsilon\, I_j} h_i + \lambda\right)\left(-\frac{\sum_{i\,\epsilon\, I_j} g_i}{\sum_{i\,\epsilon\, I_j} h_i + \lambda}\right)^2\right] + \mu\mathrm{T} =$$

$$= -\frac{1}{2}\sum_{i=1}^{n}\frac{\left(\sum_{i\epsilon I_j} g_i\right)^2}{\left(\sum_{i\epsilon I_j} h_i+\lambda\right)} + \mu T. \quad (12)$$

Once the best objective function has been defined and the optimal leaf weights assigned to I_j, we next consider what the best split procedure will be. Because (12) is derived for a wide range of functions, we are 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:

Let I_L and I_R be the sets of observations that are in the left and right parts of a node following a split. So that $I = I_L + I_R$.

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

$\hat{\mathcal{L}}^{(d)}$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.

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 have 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.

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 comprises 2,767 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 January 1 through to December 31, 2011. The data set contains the following information about each driver: insured's age (*age*), age of the vehicle (*ageveh*) in years, insured's gender (*male*), driving experience (*drivexp*) in years, percentage of total kilometers travelled in urban areas (*pkmurb*), 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*), total kilometers (*kmtotal*), and, finally, the presence of an

accident claim with fault (Y) which was coded as 1 when, at least, one claim with fault occurred in the observational period and was reported to the insurance company, and 0 otherwise. We are 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 highlights 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. Insureds with no accident claim seem to have travelled fewer kilometers than those presenting a claim. The non-occurrence of accident claims is 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.

Table 1. *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	1,263 (93.21%)	92 (6.79%)	1,355
	Male	1,309 (92.71%)	103 (7.29%)	1,412
Driving experience (years)		4.98	4.46	4.94
Age of vehicle (years)		6.37	6.17	6.35
Total kilometers travelled		7,094.63	7,634.97	7,132.71
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		2,572 (92.95%)	195 (7.05%)	2,767

[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.

The data set is divided randomly into a training data set of 1,937 observations (75% of the total sample) and a testing data set of 830 observations (25% of the total sample). 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.

4. Results

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

4.1. Coefficient Estimates

Table 2 presents the estimates obtained using the two methods. Note, 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, so, we obtain a historical record of the D times $P+1$ coefficient estimates. 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. Inspection of the results in Table 2 shows that, in general, older insureds are less likely to suffer a motor accident than younger policy holders. 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. We are not able to interpret the coefficients of the XGBoost, but by inspecting the maximum and minimum values of the linear booster case, we obtain an idea of how the estimates fluctuate until iteration *D*.

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**	**Minimum**	**Mean**	**Maximum**
Constant	-2.8891	-0.5442	1.8583	-2.6760	-2.6690	-1.7270
*age	-0.2059	-0.0994	0.0011	-0.2573	-0.2416	-0.0757
drivexp	-0.1285	-0.0210	0.0906	-0.0523	-0.0517	-0.0069
ageveh	-0.0786	-0.0249	0.0257	-0.0897	-0.0885	-0.0220
male	-0.3672	0.0039	0.3751	0.0019	0.0020	0.0070
kmtotal	-0.0203	0.0266	0.0707	0.0137	0.1164	0.1176
pkmnig	-0.0354	-0.0046	0.0239	-0.0292	-0.0290	-0.0061
pkmexc	-0.0122	0.0144	0.0385	0.0180	0.1007	0.1016
*pkmurb	0.0002	0.0146	0.0286	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. Calculations were performed in R and scripts are available from the authors.

Only the coefficients of age and percentage of kilometers travelled in urban areas are significantly different from zero in the logistic regression model, but we have preferred to keep all the coefficients of the covariates in the estimation results so as 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.

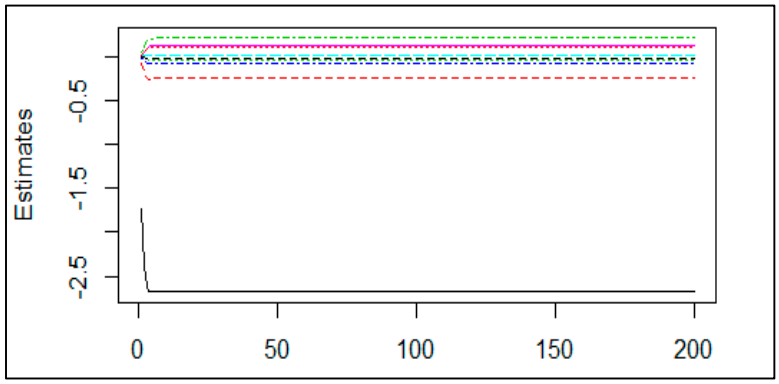

Figure 1. *The magnitude of all the estimates in the D=200 iterations. Different colors indicate each of the coefficients in the XGBoost iteration.*

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 are present during the boosting.

4.2. Prediction Performance

The performance of the two methods is 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, out-of-sample performance is 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 are 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:

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

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. *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.

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 indicate that the performance of the XGBoost with the linear booster (last column) is similar to that of the logistic regression both in the training and testing samples. XGBoost using the tree approach provides good accuracy and a good RMSE value in the training sample, but it does not perform as well as the other methods in the case of the testing sample. More importantly, XGBoost fails to provide good sensitivity. In fact, the XGBoost with the

tree booster clearly overfits the data, because while it performs very well in the training sample, it fails to do so in the testing sample. For instance, sensitivity is equal to 100% in the training sample for the XGBoost tree booster methods, but it is equal to only 7.9% in the testing sample.

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; yet, tuning offers the possibility of improving the predictive capacity by modifying some specific parameter estimates. 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. We confirm that the logistic regression and XGBoost (linear) have a similar predictive performance. The XGBoost (tree) presents 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 fails to maintain this degree of sensitivity when this algorithm is used with new samples.

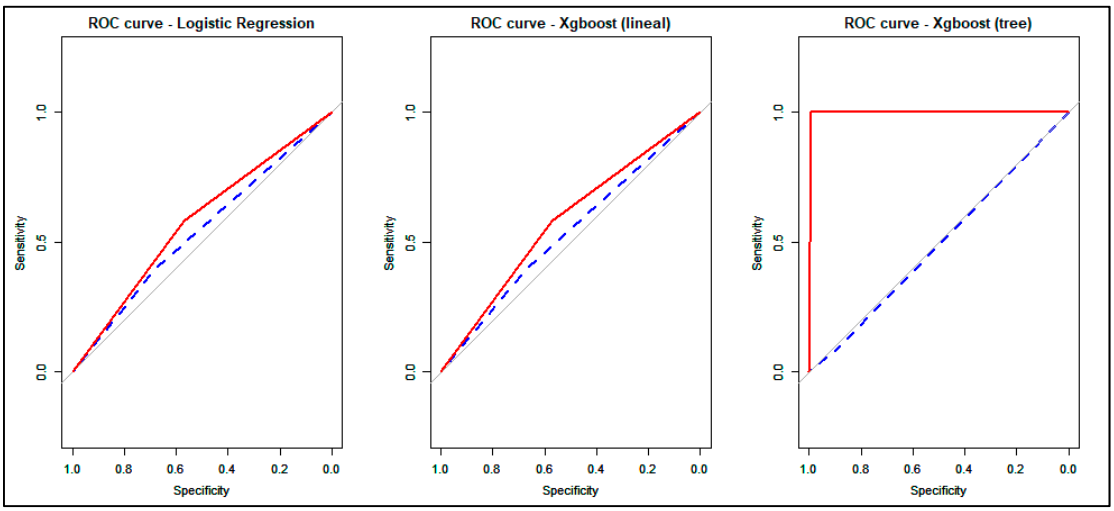

Figure 2. *The Receiver Operating Characteristics (ROC) curve obtained using the three methods on the training and testing samples.* The red solid line represents the ROC curve obtained by each method in the training sample, and the blue dotted line represents the ROC curve obtained by each method in 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, we propose a simple sensitivity analysis of the regularization parameters. In so doing, we evaluate the evolution of the following confusion matrix measures: accuracy, sensitivity and specificity – according to some given regularization parameter values for the training and the testing sample – and, finally, choose the regularization parameter that gives the highest predictive measures in the training and testing samples.

We consider two regularization methods. First, we consider the L2 (Ridge), which is Chen and Guestrin's (2016) original proposal and takes the $l2$-norm of the leaf weights. It has a parameter λ that multiplies that $l2$-norm. Second, we consider the L1 (Lasso) method, which is an additional

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 May 2019 doi:10.20944/preprints201905.0122.v1

Peer-reviewed version available at Risks 2019, 7, 70; doi:10.3390/risks7020070

implementation possibility of the `xgboost` package in R that takes the *l1*-norm of the leaf weights. It has a parameter α that multiplies that *l1*-norm. Consequently, λ and α calibrate the regularization term in (4). For simplicity, no tree pruning was implemented, so $\mu=0$ in (4).

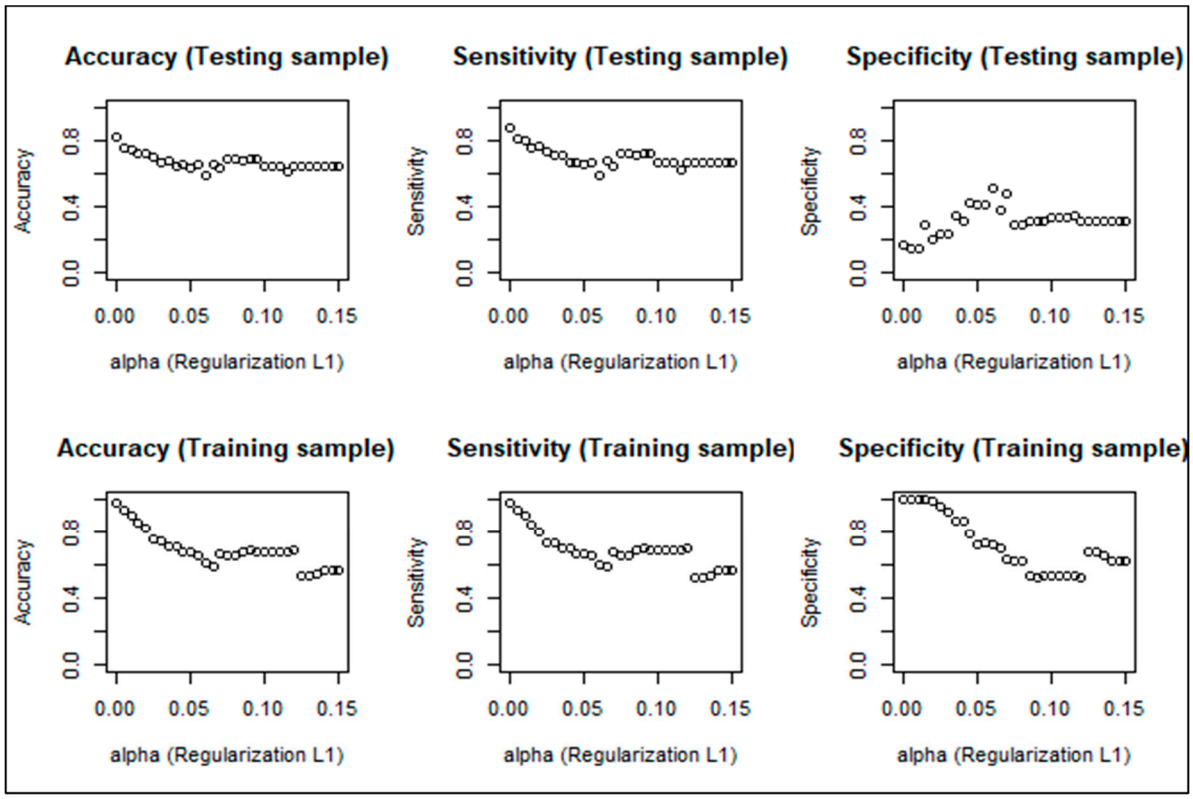

Figure 3. *The predictive measures according to α. L1 method applied to the training and testing samples*

The values of α and λ 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, we evaluate their changes in a small interval. Figure 3 shows the predictive measures for the testing and training samples according to the values of α when the L1 regularization method is implemented. When $\alpha = 0$, we obtain exactly the same predictive measure values as in Table 3 (column 3) because the objective function has not been regularized. As the value of α increases, the models' accuracy and sensitivity values fall sharply – to at least $\alpha \simeq 0.06$ in the training sample. In the testing sample, the fall in these values is not as pronounced; however, when α is lower than 0.06 the specificity performance is the lowest of the three measures. Moreover, selecting an α value lower than 0.05 results in higher accuracy and sensitivity measures, but lower specificity. In contrast, when α equals 0.06 in the testing sample, we obtain the highest specificity level of 0.5079, 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 are: 0.7227, 0.6086, and 0.6000, respectively. As a result when α is fixed at 0.06, the model performs similarly in both the testing and training samples.

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

Peer-reviewed version available at Risks 2019, 7, 70; doi:10.3390/risks7020070

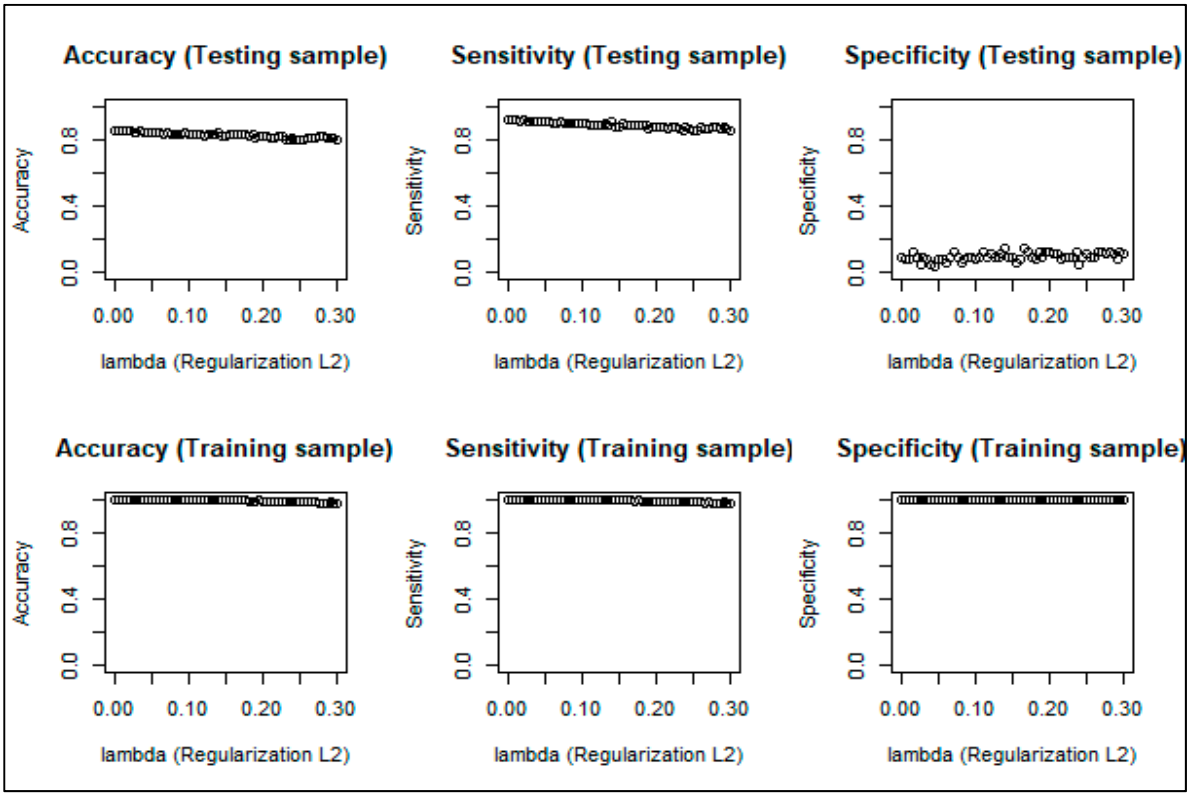

Figure 4. *The predictive measures according to λ. L2 method applied to the training and testing samples.*

Figure 4 shows the predictive measures for the testing and training samples according to the values of λ when the L2 regularization method is implemented. From $\lambda = 0$ to $\lambda = 0.30$ all predictive measures are around 100% in the training sample; however, very different results are recorded in the testing sample. Specifically, accuracy and sensitivity fall slowly, but specificity is low – there being no single λ that makes this parameter exceed at least 20%. As such, no λ can help improve specificity in the testing sample. The L2 regularization method does not seem to be an effective solution to correct the problem of overfitting in our case study data set.

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) explain 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. This method is of great relevance in tree models because it helps identify the order in which the leafs 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 appear 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 appear 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, so, 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) is much higher than that of 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 is similar to that obtained by the logistic regression. This means additional efforts have to be taken to tune the XGBoost model so as 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 provide a straightforward interpretation of the coefficient estimates. Moreover, the method offers a relatively high predictive performance considering that only two coefficients are 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, we obtained different conclusions for the two methods employed. Thus, given that the predictive performance of XGBoost was not much better than that of the logistic regression, even after careful regularization, we conclude 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.

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.

References

Ayuso, Mercedes, Guillen, Montserrat, and Pérez-Marín, Ana-María. 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. doi: 10.1016/j.aap.2014.08.017

Ayuso, Mercedes, Guillen, Montserrat, and Pérez-Marín, Ana-María. 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. doi: 10.1016/j.trc.2016.04.004

Ayuso, Mercedes, Guillen, Montserrat, and Pérez-Marín, Ana-María. 2016b. Telematics and gender discrimination: some usage-based evidence on whether men's risk of accident differs from women's. *Risks* 4: 10. doi: 10.3390/risks4020010

Boucher, Jean-Phillippe, Côté, Steven, and Guillen, Montserrat. 2017. Exposure as duration and distance in telematics motor insurance using generalized additive models. *Risks* 5(4): 54. doi: 10.3390/risks5040054

Chen, Tianqi, and Guestrin, Carlos. 2016. XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 785-94. doi: 10.1145/2939672.2939785

de Boer, Pieter-Tjerk, Kroese, Dirk, Mannor, Shier, and Rubinstein, Reuven Y. 2005. A tutorial on the Cross Entropy Method. *Annals of Operations Research* 134(1): 19-67. doi: 10.1007/s10479-005-5724-z

Dietterich, Thomas G., Domingos, Pedro, Geetor, Lise, Muggleton, Stephen, and Tadepalli, Prasad. 2008. Structured machine learning: the next ten years. *Machine Learning* 73: 3-23. doi: 10.1007/s10994-008-5079-1

Elliot, Graham, and Timmermann, Allan. 2003. *Handbook of Economic Forecasting*. Elsevier.

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

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

Gomez-Verdejo, Vanessa, Arenas-Garcia, Jerónimo, Ortega-Moral, M, and Figueiras-Vidal, Anibal R. 2005. Designing RBF classifiers for weighted boosting. *In Proceedings. IEEE International Joint Conference on Neural Networks* 2: 1057-62. doi: 10.1109/IJCNN.2005.1555999

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

Guillen, Montserrat, Nielsen, Jens Perch, Ayuso, Mercedes, and Pérez-Marin, Ana-María. 2019. The use of telematics devices to improve automobile insurance rates. *Risk Analysis* 39(3): 662-72. doi: 10.1111/risa.13172

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

Huang, Jianhua Z., and Yang, Lijian. 2004. Identification of non-linear additive autoregressive models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 66(2): 463-77. doi: 10.1111/j.1369-7412.2004.05500.x

Hultkrantz, Lars, Nilsson, Jan-Eric, and Arvidsson, Sara. 2012. Voluntary internalization of speeding externalities with vehicle insurance. *Transportation Research Part A: Policy and Practice* 46(6): 926-937. doi: 10.1016/j.tra.2012.02.011

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

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

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

Lee, Simon C. K., and Lin, Sheldon. 2018. Delta boosting machine with application to general insurance. *North American Actuarial Journal* 22(3): 405-25. doi: 10.1080/10920277.2018.1431131

Lee, Simon, and Antonio, Katrien. 2015. Why high dimensional modeling in actuarial science?. In IACA Colloquia. https://pdfs.semanticscholar.org/ad42/c5a42642e75d1a02b48c6eb84bab87874a1b.pdf (Accessed on May 8, 2019)

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

Nasrabadi, Nasser M. 2007. Pattern recognition and machine learning. *Journal of Electronic Imaging* 16(4): 049901. doi: 10.1117/1.2819119

Natekin, Alexey, and Knoll, Alois. 2013. Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics* 7: 21. doi: 10.3389/fnbot.2013.00021

Pérez-Marín, Ana-María, and Guillen, Montserrat. 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. doi: 10.1016/j.aap.2018.11.005

Schapire, Roberte, and Freund, Yoav. 2012. *Boosting: Foundations and Algorithms*. MIT press.

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

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

Verbelen, Roel, Antonio, Katrien, and Claeskens, Gerda. 2018. Unraveling the predictive power of telematics data in car insurance pricing. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 67(5): 1275–304. doi: 10.1111/rssc.12283