
HOW MUCH TELEMATICS INFORMATION DO INSURERS NEED FOR CLAIM CLASSIFICATION?

Francis Duval

Chaire Co-operators en analyse des risques actuariels
Département des mathématiques
Université du Québec à Montréal
Montréal, QC H2X 3Y7
duval.francis.2@courrier.uqam.ca

Jean-Philippe Boucher

Chaire Co-operators en analyse des risques actuariels
Département des mathématiques
Université du Québec à Montréal
Montréal, QC H2X 3Y7
boucher.jean-philippe@uqam.ca

Mathieu Pigeon

Chaire Co-operators en analyse des risques actuariels
Département des mathématiques
Université du Québec à Montréal
Montréal, QC H2X 3Y7
pigeon.mathieu.2@uqam.ca

Thursday 27th May, 2021

ABSTRACT

It has been shown several times in the literature that telematics data collected in motor insurance help to better understand an insured's driving risk. Insurers that use this data reap several benefits, such as a better estimate of the pure premium, more segmented pricing and less adverse selection. The flip side of the coin is that collected telematics information is often sensitive and can therefore compromise policyholders' privacy. Moreover, due to their large volume, this type of data is costly to store and hard to manipulate. These factors, combined with the fact that insurance regulators tend to issue more and more recommendations regarding the collection and use of telematics data, make it important for an insurer to determine the right amount of telematics information to collect. In addition to traditional contract information such as the age and gender of the insured, we have access to a telematics dataset where information is summarized by trip. We first derive several features of interest from these trip summaries before building a claim classification model using both traditional and telematics features. By comparing a few classification algorithms, we find that logistic regression with lasso penalty is the most suitable for our problem. Using this model, we develop a method to determine how much information about policyholders' driving should be kept by an insurer. Using real data from a North American insurance company, we find that telematics data become redundant after about 3 months or 4,000 kilometers of observation, at least from a claim classification perspective.

Keywords motor insurance · telematics · supervised statistical learning · dichotomous response · claim classification · lasso logistic regression

1 Introduction

Usage-Based Insurance (UBI) is a type of motor insurance for which premiums are determined using information about the driving behaviour of the insured. This information is usually collected using telematics technology, most often through a device installed in the vehicle or a mobile application. Because of its multiple benefits, this type of insurance is increasingly promoted by insurers. It also seems to be more and more appreciated by consumers. A survey¹ conducted by *Willis Towers Watson* on 1,005 insurance consumers in the United States reports that 4 out of 5 drivers are in favour of sharing their recent driving information in exchange for a personalized insurance product. Among the benefits, it seems clear nowadays that the addition of telematics information into the insurance pricing models improves the precision of the pure premium (see for instance [Ayuso et al., 2019], [Pérez-Marín and Guillen, 2019], [Verbelen et al., 2017] and [Lemaire et al., 2015]). UBI also has many positive impacts on society. Indeed, because it encourages individuals to drive less and more safely, it helps making roads safer, reducing traffic congestion, limiting greenhouse gas emissions, and making insurance more affordable, among other things.

The idea of usage-based insurance was first articulated by William Vickrey, considered as the father of UBI. In [Vickrey, 1968], he criticizes the premium structure then used in motor insurance. He believes that the insurance premium should be modulated according to the use of the vehicle, and thus appear as a variable cost to the insured. In order to correct the premium structure that he considers deficient, Vickrey proposes in the late 1960s a new type of insurance where the premium increases with usage. In particular, he suggests pricing through a tax applied to gasoline or tires: insureds who consume more gasoline (or tires) would then have a higher premium. However, due to the lack of technology and organizational barriers, it was not until the mid-1990s that the first UBI program appeared in the United States. Nowadays, several major general insurance companies have their own UBI program, and this type of insurance is still growing in popularity. It is now a fact that UBI is collectively beneficial in many ways, and that is why it seems to be the future of motor insurance.

A fairly general section of the UBI literature discusses the feasibility, implementation, costs and benefits of UBI. [Litman, 2007] explores the practicability, pros and cons of different types of distance-based insurance, such as *Milage Rate Factor* and *Pay-at-the-Pump*. [Greenberg, 2009] establish that every mile driven insured with UBI rather than conventional insurance brings a significant benefit to the community, which they quantify at \$0.016. With this in mind, he proposes benefit-maximizing rules and incentives to increase the size of the UBI market. [Bordoff and Noel, 2008] estimate that the change from traditional insurance to UBI reduces mileage by 8% in average, resulting in yearly savings of \$658 per vehicle. Indeed, the fact that the premium increases with usage is a strong incentive to drive less. They state that most of these savings are attributable to reduced congestion and accidents.

In the recent literature on automobile insurance pricing, among all information you can get from a telematics device, several papers only focus on the distance driven, which is probably the most useful measure for ratemaking. [Boucher et al., 2017] simultaneously analyze the impact of distance driven and duration on claim frequency using Generalized Additive Models (GAMs) for cross-sectional count data. They find that neither distance nor duration is proportional to frequency, but that frequency tends rather to stabilize once a certain distance or duration has been reached. The authors invoke a “learning effect” and a “highway effect” to explain the fact that a policyholder travelling $2x$ miles is less than twice as dangerous as an insured travelling x miles. [Boucher and Turcotte, 2020] go further and analyse the link between distance driven and frequency using models for panel count data, including a Generalized Additive Model for Location, Scale and Shape (GAMLSS) and a GAM with fixed effects. They refute the idea of the “learning effect” to explain the non-linearity of frequency. Instead, they find that the relationship between frequency and distance driven is approximately linear, and that the apparent non-linearity is due to residual heterogeneity incorrectly captured by GAMs.

Since an insured rarely has more than one claim per year (see for instance [Boucher et al., 2007]²), assigning a probability of claiming is almost like assigning a premium. Therefore, a significant amount of studies related to UBI ratemaking have focused on claim classification. [Pesantez-Narvaez et al., 2019] compare the predictive performance of a non-penalized logistic regression and a boosting algorithm called XGBoost using classical and telematics information, including distance driven, fraction of driving at night, fraction of driving in urban areas and fraction of driving above speed limit. In [Huang and Meng, 2019], the authors compare the performance of various classification algorithms while proposing a way to bin continuous telematics variables in order to

¹www.willistowerswatson.com/en-US/Insights/2017/05/infographic-how-ready-are-consumers-for-connected-cars-and-usage-based-car-insurance

²Table 2 of this paper informs us that 99.5% of the policyholders have 1 claim or less.

create a finite number of risk classes and thus increase interpretability. It turns out that this discretization also increases the predictive power of the algorithms. [Paefgen et al., 2013] also compare multiple classification algorithms on real claim classification data and propose a novel way to aggregate telematics information into what they call an “aggregate risk factor”.

Moreover, the question of how to transform telematics data into useful information for pricing is still a thorny issue to this day. Indeed, it is not yet clear how to use raw telematics information in an optimal way. This falls within the field of feature engineering. Some simply create features manually from the raw data, while others contributed using techniques from machine learning, deep learning and pattern recognition ([Weidner et al., 2016], [Gao and Wüthrich, 2018], [Gao et al., 2018], [Gao and Wüthrich, 2019]).

Despite its many benefits, the growing popularity of UBI means that insurers are accumulating large amounts of sensitive data about their insureds. [Dewri et al., 2013] have shown that it is possible to deduce users’ destinations by analyzing their driving data, without even having access to the geographical coordinates of the trips. In recent years, concerns have been arising about the use of telematics technology, particularly with respect to confidentiality and data usage. In Canada and in other countries, insurance regulators make recommendations regarding UBI products. In particular, Financial Services Commission of Ontario (FSCO) and Financial Services Regulatory Authority of Ontario (FSRA) state that telematics data fall under the definition of “personal data”, and must therefore be handled according to the relevant legislation (see [Howell, 2013]). They also mention that the insurer must inform the consumer in several respects, such as the type of information collected and the use made of it. In spite of these recommendations, the fact remains that large amounts of personal data are stored by insurance companies. From an ethical, practical and economical point of view, it is nowadays important for an insurer to keep a minimal amount of personal data on its insureds. Notably, collecting less data reduces storage and manipulation costs in addition to facilitate data leakage prevention.

This raises the question of how much information an insurer should collect on the driving behavior of its insureds, which should be just enough to get a good idea of how the insured drives, but not too much for the reasons cited in the preceding paragraph. We explore this avenue through the binary claim classification framework, where the goal is to assign each insured a probability of claiming. In addition to data traditionally used in motor insurance pricing, we have access to telematics data in a format where we have one observation per trip, which allows us to derive interesting and representative features of the insured’s driving. We first build a classification model using both classical and telematics features. To this effect, we determine which classification algorithm is the most suited to our problem by comparing among a logistic regression with lasso penalty, a logistic regression with elastic-net penalty and a random forest, all this while accounting for interactions between features. It turns out that the lasso model is best suited to our task. Using this classification model, we then develop a method for determining how much telematics information an insurer should collect. For this purpose, we derive several classification datasets using increasing amounts of information about the insured’s driving. We then fit a lasso model on each of these datasets and compare performance: telematics information is considered redundant when it no longer substantially improves the classification score.

In Section 2, we describe the classical and telematics data available to us in conjunction with details of how telematics features are derived. Section 3 follows, where we show how the classification datasets are built using varying amounts of information as well as other preprocessing steps. Then, in Section 4, the mathematical framework of supervised classification is introduced, in addition to explaining the functioning of the 3 preselected algorithms, namely the two penalized logistic regressions and the random forest. This is followed by Section 5, in which we first make a choice among the classification models presented in Section 4. We choose the lasso logistic regression for its good performance and for its simplicity. We also find that adding interactions does not substantially improve the performance of the models. Using our classification model as well as the datasets built in Section 3, we then develop a method to determine the right amount of telematics information to collect based on non-parametric bootstrapping. Using the data provided by the North American insurer, we find that telematics information no longer improves substantially classification after about 3 months, or 4,000 kilometers of observation. Finally, we conclude in Section 6.

2 Data

All the data is provided by a North American insurer and is related to personal auto insurance in the province of Ontario. We have access to a telematics database consisting of 210,854,092 summaries of trips made by 67,355 vehicles, of which an extract is shown in Table 1. The recording of a trip, made using an On-Board Diagnostics (OBD) device, begins when the vehicle is turned on and stops when the ignition is turned off. Each trip is summarized in 4 measurements: the datetime of departure and arrival, the distance driven and the

VIN	Trip number	Departure datetime	Arrival datetime	Distance	Maximum speed
A	1	2016-04-09 15:23:55	2016-04-09 15:40:05	10.0	72
A	2	2016-04-09 17:49:33	2016-04-09 17:57:44	4.5	68
⋮	⋮	⋮	⋮	⋮	⋮
A	3312	2019-02-11 18:33:07	2019-02-11 18:54:10	9.6	65
B	1	2016-04-04 06:54:00	2016-04-04 07:11:37	14.0	112
B	2	2016-04-04 15:20:19	2016-04-04 15:34:38	13.5	124
⋮	⋮	⋮	⋮	⋮	⋮
B	2505	2019-02-11 17:46:47	2019-02-11 18:19:22	39.0	130
C	1	2016-01-16 15:41:59	2016-01-16 15:51:35	3.3	65
⋮	⋮	⋮	⋮	⋮	⋮

Table 1: Extract from the telematics dataset. Dates are displayed in the yyyy-mm-dd format.

maximum speed reached. Trips are also associated with a vehicle via the vehicle identification number (VIN), but there is no column to identify the insured person, which makes it impossible to know who the driver is. Therefore, our analysis is based on vehicles rather than policyholders. For 57,671 of these vehicles, which are all observed during one or more insurance contracts, we have access to features traditionally used in motor insurance (gender, age, region, etc.) as well as claiming information. Among these features, that we will call “classical features”, 10 were selected and are described in Table 2. Only vehicles having at least one full-year

Classical feature name	Description	Type
annual_distance	Annual distance declared by the insured	Numeric
commute_distance	Distance to the place of work declared by the insured person	Numeric
conv_count_3_yrs_minor	Number of minor contraventions in the last 3 years	Numeric
gender	Gender of the policyholder	Categorical
marital_status	Marital status of the policyholder	Categorical
pmt_plan	Payment plan chosen by the policyholder	Categorical
veh_age	Vehicle age	Numeric
veh_use	Use of the vehicle	Categorical
years_claim_free	Number of years since last claim	Numeric
years_licensed	Number of years since driving licence	Numeric

Table 2: Classical features selected for the analysis.

contract are kept for the analysis, as well as only the first full-year contract for each vehicle, which means we end up with 29,799 vehicles, each associated with exactly one full-year contract. Keeping only one contract per vehicle allows us to get rid of the dependency between observations. In their observed year, 99.8% of the vehicles have made less than 5,000 trips, for an average of 1,581 trips per vehicle. The complete distribution is shown in Figure 1.

Based on the trip summaries of Table 1, we wish to derive telematics features that best depict the insured’s driving behavior by aggregating the trips for each vehicle. Indeed, we want these features, or covariates, to have a good predictive power when inputted into a supervised classification algorithm. This falls within the field of feature extraction, which is often a crucial step in machine learning. However, extracting (or creating) features from raw telematics data in an optimal way is not a simple task and is a research avenue in itself. This problem is addressed in several articles such as [Wüthrich, 2017] and [Gao and Wüthrich, 2018]. Features extracted in these two studies nevertheless require second-by-second data, which we do not have at hand. We are therefore largely inspired by features of the “usage”, “travel habits” and “driving performance” types derived in [Huang and Meng, 2019]. From the telematics dataset, we thus extract a total of 14 features, all described in Table 3 and for which the distribution is shown in Figure 2. In Table 3, we also display the average

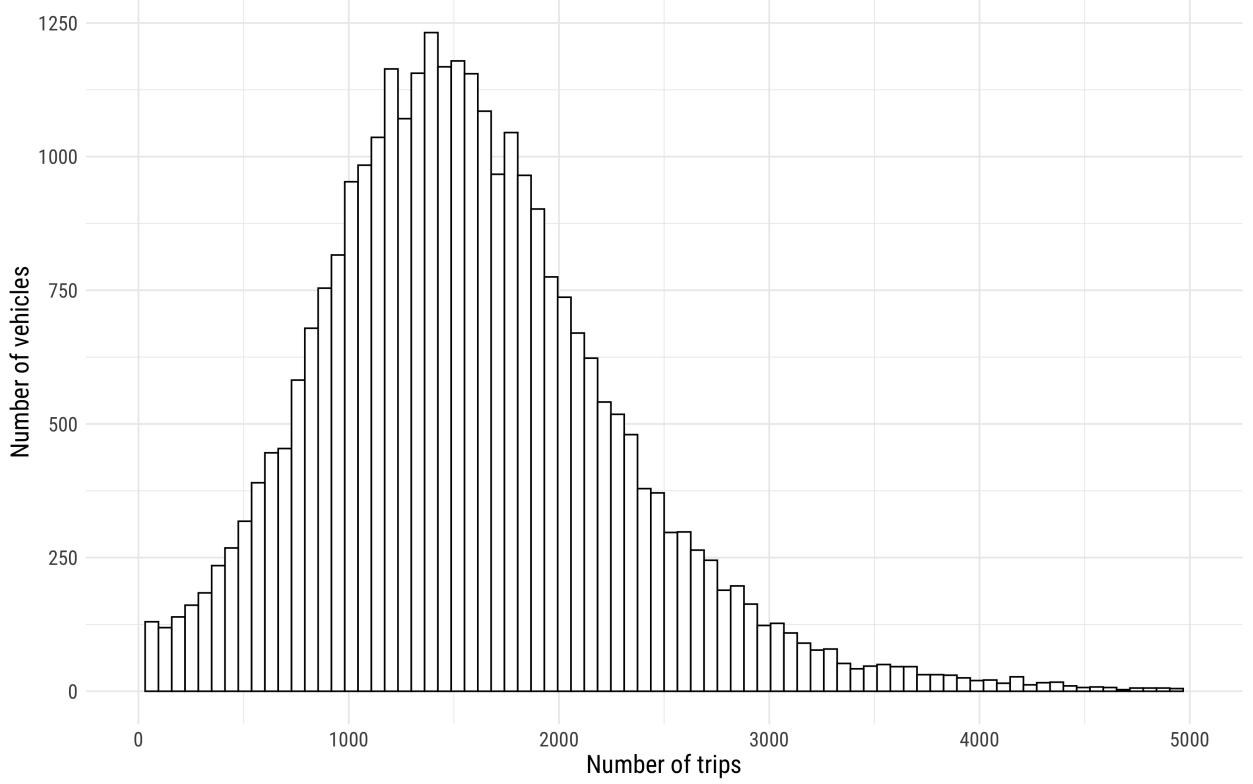


Figure 1: Histogram of the number of trips for the 29,799 vehicles in their observed year.

Feature name	Description	Mean value		p-value (t-test)
		Non-claimants	Claimants	
avg_daily_distance	Average daily distance	44.3	50.1	< 0.0001
avg_daily_nb_trips	Average daily number of trips	4.3	4.8	< 0.0001
med_trip_avg_speed	Median of the average speeds of the trips	28.8	28.0	< 0.0001
med_trip_distance	Median of the distances of the trips	5.2	5.2	0.9203
med_trip_max_speed	Median of the maximum speeds of the trips	69.6	70.0	0.2906
max_trip_max_speed	Maximum of the maximum speed of the trips	138.2	141.9	< 0.0001
prop_long_trip	Proportion of long trips (> 100km)	0.0108	0.0103	0.4114
frac_expo_night	Fraction of night driving ³	0.0262	0.0276	0.1630
frac_expo_noon	Fraction of midday driving⁴	0.210	0.198	< 0.0001
frac_expo_evening	Fraction of evening driving⁵	0.0845	0.0965	< 0.0001
frac_expo_peak_morning	Fraction of morning rush hour driving ⁶	0.0968	0.0985	0.4453
frac_expo_peak_evening	Fraction of evening rush hour driving⁷	0.137	0.143	< 0.0001
frac_expo_mon_to_thu	Fraction of driving on Monday to Thursday	0.582	0.582	0.6786
frac_expo_fri_sat	Fraction of driving on Friday and Saturday	0.299	0.300	0.5475

Table 3: Mean value of the 14 features extracted from the telematics dataset for claimants and non-claimant. Two-sample t-tests were conducted to determine whether the mean differs significantly between the two groups.

value of the telematics features for two groups of vehicles, namely the claimants (those who have claimed at least once during their observed year) and the non-claimants (those who have not claimed during their observed year), and a two-sample t-test is conducted for each of the features to determine whether the mean differs significantly between the two groups. It turns out that the difference in the mean is significant (at a

³0h-6h

⁴11h-14h

⁵20h-0h

⁶7h-9h Monday to Friday

⁷17h-20h Monday to Friday

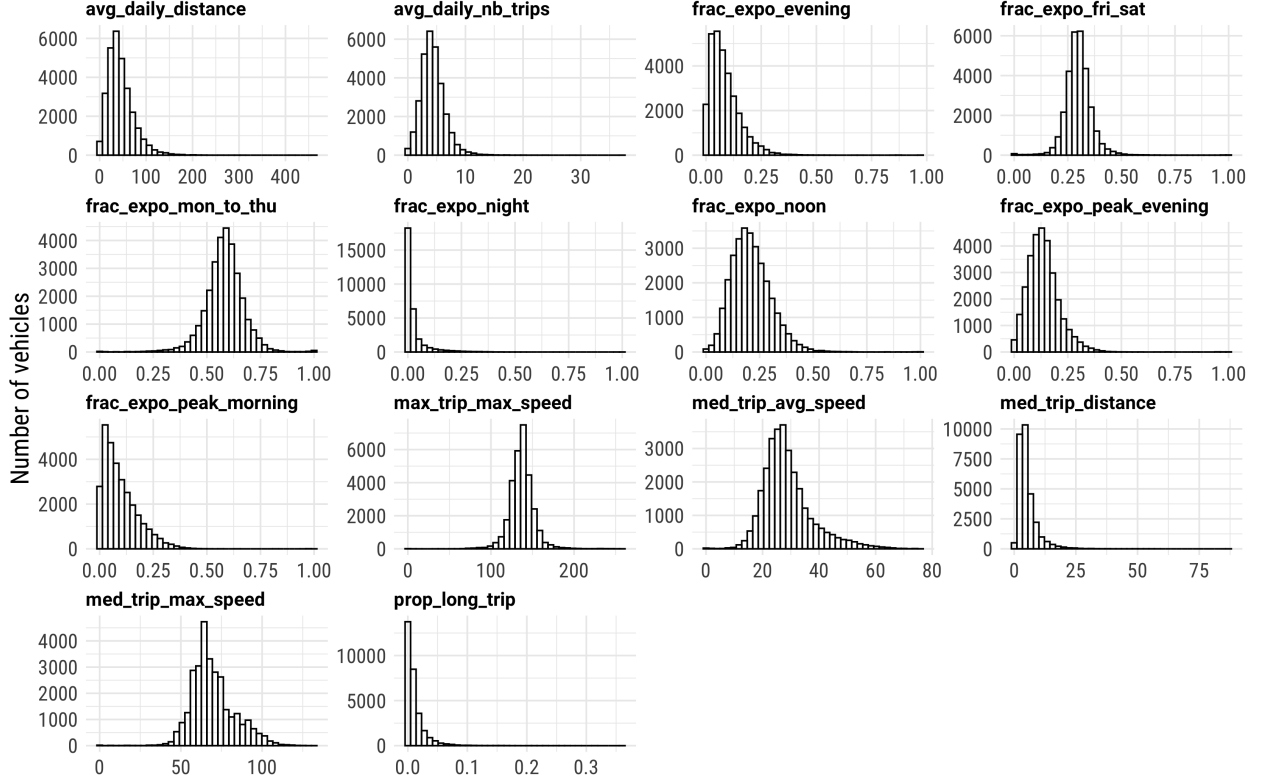


Figure 2: Distributions over the 29,799 vehicles of the 14 features extracted from the telematics dataset.

95% confidence level) for half of the 14 features, which we have highlighted in bold in Table 3. This suggests that these 7 features contain predictively relevant information. Note that claimant vehicles tend to travel more distance and make more trips, which seems natural. They also tend to have a lower average median speed. This may be due to the fact that claimant insureds have a higher propensity to drive in the city, where average speed is lower and the risk of collision, higher than elsewhere. Claimants also tend to have a higher maximum speed reached in their observed year, drive less in the midday and more in the evening, especially in the rush hour.

Example 1. In order to illustrate the exact calculation of the 14 telematics features, let us imagine that a insured, during a given contract that we assume lasts 7 days, made only 5 trips, summarized in Table 4. The calculation of

Trip number	Departure time	Arrival time	Weekday	Distance	Average speed	Maximum speed
1	18:20	18:28	Monday	8	60	73
2	17:40	17:54	Monday	9	39	70
3	09:35	09:48	Tuesday	17	78	102
4	07:30	07:37	Thursday	9	77	92
5	12:20	13:35	Saturday	109	88	120

Table 4: Summaries of the 5 trips made by a fictitious insured. Note that weekday and average speed can be easily derived from the telematics dataset.

the telematics features related to this insured would then be done according to Table 5.

One must be careful when analyzing telematics data. Indeed, insureds who have chosen to be observed telematically for insurance purposes do not correspond to the general population of insureds, which means our data cannot be considered a simple random sample of the company’s insured population. As far as we are concerned, 10% to 15% of the insureds in our North American insurance company’s portfolio are observed with a telematics device, and these are generally worse and younger drivers. This is because at the time the data was collected, in order to encourage policyholders to choose the telematics option, insurers were offering

Feature	Computation	Value
avg_daily_distance	$(8 + 9 + 17 + 9 + 109)/7$	21.7
avg_daily_nb_trips	$5/7$	0.71
med_trip_avg_speed	$\text{med}\{60, 39, 78, 77, 88\}$	77
med_trip_distance	$\text{med}\{8, 9, 17, 9, 109\}$	9
med_trip_max_speed	$\text{med}\{73, 70, 102, 92, 120\}$	92
max_trip_max_speed	$\text{max}\{73, 70, 102, 92, 120\}$	120
prop_long_trip (> 100km)	$1/5$	0.2
frac_expo_night	$0/(8 + 9 + 17 + 9 + 109)$	0
frac_expo_noon	$109/(8 + 9 + 17 + 9 + 109)$	0.72
frac_expo_evening	$0/(8 + 9 + 17 + 9 + 109)$	0
frac_expo_peak_morning	$9/(8 + 9 + 17 + 9 + 109)$	0.06
frac_expo_peak_evening	$(9 + 8)/(8 + 9 + 17 + 9 + 109)$	0.11
frac_expo_mon_to_thu	$(8 + 9 + 17 + 9)/(8 + 9 + 17 + 9 + 109)$	0.28
frac_expo_fri_sat	$109/(8 + 9 + 17 + 9 + 109)$	0.72

Table 5: Telematics features calculated for the fictitious insured of Table 4.

a 5% entry discount in addition to a renewal discount ranging from 0% to 25%. At that time, it was also not possible for insurers, at least in the region where the data was collected, to increase the premium based on telematics information. Because car insurance in Ontario is very expensive, any discount is welcomed by high-premium policyholders such as bad drivers and youths. As a consequence, an unexpectedly large proportion of bad/young drivers end up using the telematics option. However, this selection bias does not affect our analysis since the models and methods we develop apply only to telematically observed insureds. One must only be careful not to draw conclusions from these data and apply them to the general population of insureds.

3 Data preparation

3.1 Design of the classification datasets

With the data we have at hand, we wish to build several classification datasets using a varying amount of telematics information, or trip summaries. We will later on perform classification on each of them and compare performance, which will allow us to determine how much telematics information is needed to obtain a proper classifier. For this purpose, we compute telematics features of Table 3 in several versions, which we do in two different ways. The first way to proceed, called the “time leap” method (TL), consists in first calculating features using only one month of trip summaries for each vehicle, and then add one month worth of data to each subsequent version. Since vehicles are observed over one year, telematics features of Table 3 are derived in 12 different versions. In general, the k^{th} version is calculated using the first k months of telematics information related to a given vehicle, for $k = 1, \dots, 12$. The second way to proceed, called the “distance leap” method (DL), is quite similar to the time leap way, but uses 1,000-kilometer leaps instead of one-month leaps to jump from version to version. For the sake of uniformity, we also derive telematics features in 12 different versions using this second method. In general, the k^{th} version is calculated using the first $1,000 \times k$ kilometers of telematics information, for $k = 1, \dots, 12$. Note that 37% of the vehicles have accumulated less than 12,000 kilometers of driving during their observed year, which means that they end up with some identical versions of the telematics features for the distance leap method. For instance, a vehicle that has accumulated 6,000 kilometers of driving has its distance leap versions 6 to 12 of the telematics features calculated with the same amount of telematics information, namely with 6,000 kilometers of trip summaries. Each version of the telematics features can be represented by a $29,799 \times 14$ matrix, where each row corresponds to a vehicle and each column to a feature. The matrix containing the k^{th} version of the telematics features derived according to the time leap method is noted \mathbf{x}_k^{TL} , while the one derived according to the distance leap method is noted \mathbf{x}_k^{DL} . Let us also denote by \mathbf{x}^c the $29,799 \times 10$ matrix containing the classical features of Table 2 and by \mathbf{y} the response vector of length 29,799, which indicates whether or not each vehicle had a claim in its observed year. By using time leap versions of the telematics features, we then build 12 classification datasets, denoted $\mathcal{D}_1^{\text{TL}}, \dots, \mathcal{D}_{12}^{\text{TL}}$, all sharing the same classical features \mathbf{x}^c as well as the same response vector \mathbf{y} . The only difference between them is the version of the telematics features used or, in other words, the amount of trip summaries used to compute telematics

features. In general, $\mathcal{D}_k^{\text{TL}}$, $k = 1, \dots, 12$, is the classification dataset built with the matrix of telematics features \mathbf{x}_k^{TL} , which means it is obtained by concatenating \mathbf{x}^c , \mathbf{x}_k^{TL} and \mathbf{y} . In addition to these 12 classification datasets, we also create a dataset containing no telematics information, noted $\mathcal{D}_0^{\text{TL}}$. The latter is therefore built by concatenating \mathbf{x}^c and \mathbf{y} . Note that all 13 datasets describe the same vehicles and therefore have the same number of rows. In a similar fashion, 13 datasets $\mathcal{D}_0^{\text{DL}}, \dots, \mathcal{D}_{12}^{\text{DL}}$ are built using the distance leap versions of the telematics features. In order to properly test models, it is common in machine learning to split the observations into training and testing sets. For future use, we thus randomly draw 70% of the 29,799 vehicles to make up the training set, and the remaining 30% forms the testing set. Training and testing datasets are respectively denoted by \mathcal{T}_k^m and \mathcal{V}_k^m , where $k = 0, \dots, 12$ and $m \in \{\text{TL}, \text{DL}\}$.

3.2 Preprocessing

Training and testing datasets need to be preprocessed every time they are being fed to the models, either for training or scoring purposes. The reasons for this are manifold. First, some of our features are categorical (gender, marital_status, pmt_plan and veh_use), and many supervised learning algorithms cannot handle this type of information, which means we need a way to encode them numerically. For this, we choose an embedding method called “target encoding”, which consists in replacing the value of each category, that is originally a string of characters, by a real number based on the response (or target) column. In the special case of mean target encoding, the value of each observation is replaced by the mean of the response variable for the category to which that observation belongs. For instance, imagine we have the feature “gender” with categories “woman” and “man” and that the mean of the response variable is 0.09 for women and 0.11 for men. Women would then be encoded with the value 0.09 while men would be assigned the value 0.11. What we use is similar to mean encoding, except that the encoded values are derived using a GLM rather than using the mean. Suppose we are in the context of supervised

binary classification and we want to encode the feature x , which is categorical with k categories. GLM target encoding consists in first fitting a logistic regression without intercept using only x to explain the binary response variable y , which yields coefficient estimates $\hat{\beta}_1, \dots, \hat{\beta}_k$. Then, each of the k categories is encoded with its corresponding coefficient. Hence, category j is encoded with the value $\hat{\beta}_j$, for $j = 1, \dots, k$. This means that the k -category categorical feature x becomes a numerical feature with k unique values. In order to perform target encoding, the `step_encode_glm` function of the `embed` package in the R programming language is used. Note that prior target encoding the categorical features, rare categories (i.e., those associated with 5% of the observations or less) are lumped together in a catch-all category.

Secondly, the `commute_distance` feature, which is numerical, is missing for 22.3% of the observations. Since many classification algorithms cannot handle missing values, we need a way to impute them. Virtually any prediction algorithm could be used to perform imputation, but some are more suitable than others. We choose an algorithm based on bagged decision trees (see [Breiman, 1996]) as they are known to be good at imputing missing data, partly because they generally have good accuracy and because they do not extrapolate beyond the bounds of the training data. Bagged decision trees are also preferable to random forest because they require fitting fewer decision trees to have a stable model. The idea is to first consider the feature to be imputed, namely `commute_distance`, as a response variable. Then, a bagged decision tree model is trained on

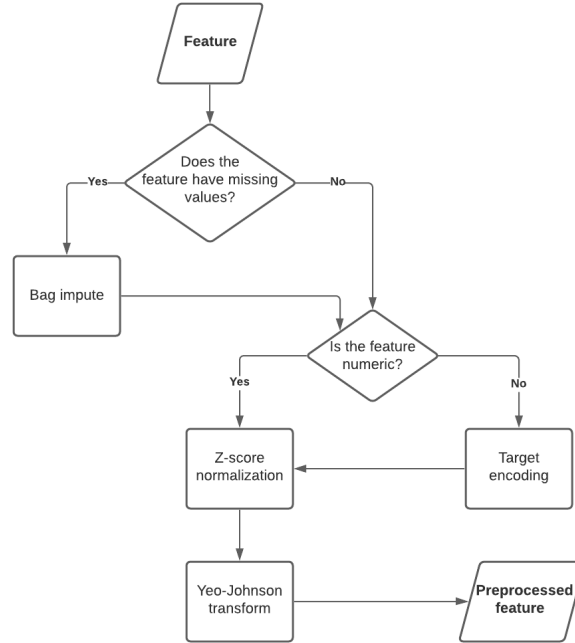


Figure 3: Flowchart of the feature preprocessing “recipe”, that is applied every time a model is trained or validated in this analysis.

all observations for which `commute_distance` is not missing, using all features except the latter as predictors. This fitted model is then scored on all observations with a missing `commute_distance` value, and the prediction is used as a replacement value. To implement bag imputation, we use the `step_bagimpute` function of the `recipe` package in R.

Thirdly, supervised learning algorithms generally learn best when the data is preprocessed in a certain way. For instance, they generally benefit from normalized and rather symmetrical feature distributions. To normalize features, we use the z-score normalization, which is a quite popular choice. Consider the numerical feature vector $\mathbf{x} = (x_1, \dots, x_n)$. The z-score normalized version of this vector is

$$\mathbf{x}^* = \left(\frac{x_1 - \bar{x}}{s}, \dots, \frac{x_n - \bar{x}}{s} \right),$$

where $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ and $s = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$ are the empirical mean and standard deviation of vector \mathbf{x} , respectively. In order to obtain more symmetric feature distributions, we use the Yeo-Johnson transformation (see [Yeo and Johnson, 2000]), which is similar to the Box-Cox transformation except that it allows for negative input values. Yeo-Johnson transformation also has the effect of making the data more normal distribution-like. The Yeo-Johnson transformation ψ applied on a real value x is defined as follows:

$$\psi(x, \lambda) = \begin{cases} ((x+1)^\lambda - 1)/\lambda & \text{if } \lambda \neq 0, x \geq 0 \\ \ln(x+1) & \text{if } \lambda = 0, x \geq 0 \\ -[(-x+1)^{2-\lambda} - 1]/(2-\lambda) & \text{if } \lambda \neq 2, x < 0 \\ -\ln(-x+1) & \text{if } \lambda = 2, x < 0, \end{cases}$$

where λ is a parameter that is optimized by maximum likelihood so that the resulting empirical distribution resembles a normal distribution as closely as possible. Note that the preprocessing steps are performed in a specific order and depends on the feature type. Figure 3 illustrates the data preprocessing “recipe”.

4 Binary classification algorithms

4.1 Binary classification framework and classification algorithm preselection

Let us be in the context of binary supervised classification, in which we have at our disposal n labeled examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$ and $y_i \in \{0, 1\}$ are respectively the p -dimensional vector containing the features and the label (or response variable) for observation i . The goal is to estimate $\mathbb{E}[Y|\mathbf{x}]$, the conditional mean of Y given the features, which can also be seen as the conditional probability $\mathbb{P}(Y = 1|\mathbf{x})$. Many algorithms have been developed to estimate this probability, including logistic regression, random forest, artificial neural networks, support vector machines, etc. For the analysis, we retain two supervised learning algorithms, i.e. penalized logistic regression and random forest. The reason for this choice is that the latter often give excellent classification results while requiring little preprocessing of the data. In fact, these two could be qualified as “off-the-shelf” algorithms because they perform implicitly feature selection and do not require much data preprocessing, unlike other algorithms such as neural networks. They are also quite easy to tune because they do not have too many hyperparameters. Note that some algorithms that require more care before inputting the data into the model can probably lead to better classification performance, but our goal is not really in that way. For the penalized logistic regression, we consider two specifications, namely the lasso penalty (also called ℓ_1 -penalty) and the elastic-net penalty.

4.2 Logistic regression

In logistic regression, the goal is to approximate the conditional probability of having a positive case ($Y = 1$) by applying the sigmoid function over a linear transformation of the features. The model can therefore be expressed as

$$p_i := \mathbb{P}(Y_i = 1|\mathbf{x}_i) = \sigma \left(\beta_0 + \sum_{j=1}^p \beta_j x_{ij} \right), \quad (1)$$

where σ , the sigmoid function, ensures that the output is a real number between 0 and 1. The model is parametrized by the unknown parameter vector $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)$, and the intercept β_0 . These parameters

are often estimated by maximum likelihood, which leads to asymptotically unbiased estimates. Maximizing likelihood is equivalent to minimizing cross-entropy loss, given by

$$L(\beta_0, \beta) = -\frac{1}{n} \sum_{i=1}^n y_i \ln(p_i) + (1 - y_i) \ln(1 - p_i). \quad (2)$$

There is no closed formula for maximum likelihood parameter estimates in the logistic regression framework, but a variety of numerical optimization methods can be used. Most of the time, the method of iteratively reweighted least squares (IRLS) is used.

4.3 Lasso logistic regression

Maximum likelihood estimator is the asymptotically unbiased estimator with the smallest variance. However, it is rarely the best for prediction accuracy. Indeed, although it has a low bias, it has a rather large variance. In 1996, [Tibshirani, 1996] proposes a new method called *least absolute shrinkage and selection operator* (lasso) for estimating parameters in linear regression that reduces the variance of the parameters at the cost of increased bias. In practice, this decrease in variance more than offsets the increase in bias, thus improving predictive performance. Although this method was originally used for models using the least squares estimator, it generalizes quite naturally to generalized linear models. In the case of logistic regression, a penalty proportional to the sum of the absolute values of the parameters is added to the cross-entropy loss of Equation 2. In lasso logistic regression, the optimization problem thus becomes

$$\min_{\beta_0, \beta} \left\{ L(\beta_0, \beta) + \lambda \sum_{j=1}^p |\beta_j| \right\}, \quad (3)$$

where $\lambda \geq 0$ is the penalty hyperparameter. In the special case where $\lambda = 0$, the penalty term disappears and we recover the conventional non-penalized logistic regression. This penalty hyperparameter is not directly optimized by the algorithm and must therefore be chosen by the user, for instance with cross-validation and grid search. Equation 3 is called the Lagrangian formulation of the lasso logistic regression optimization problem. It can be useful to rewrite this in the constrained form,

$$\min_{\beta_0, \beta} \{L(\beta_0, \beta)\} \quad \text{s.t.} \quad \sum_{j=1}^p |\beta_j| \leq s. \quad (4)$$

Note that there is a one-to-one correspondance between λ and s . This formulation allows to realize that the model gives itself a “budget” of parameters. Indeed, the sum of the absolute values of the coefficients, which is the ℓ_1 norm of the parameter vector β , must always be less than or equal to the constant s set by the user. This has the effect of shrinking and even zeroing some of the logistic regression coefficients. The lasso logistic regression fits into the more general framework where the constraint on the parameter vector is given by

$$\sum_{j=1}^p |\beta_j|^q \leq s, \quad (5)$$

where $q \geq 0$ is a fixed real number. In particular, setting $q = 1$ retrieves the lasso constraint, whereas $q = 0$ and $q = 2$ correspond respectively to best subset selection and Ridge logistic regressions. Best subset selection and Ridge have both their pros and cons. Ridge regression only shrinks coefficients: it never sets them to zero, which makes interpretation more difficult. In general, one prefers to have a sparse model: according to Occam’s Razor Principle, a simple explanation of a phenomenon is to be preferred to a more complex one. Best subset selection leads to sparse models, but it involves resolving a nonconvex and combinatorial optimization problem, and becomes infeasible above about 50 features. Lasso regression attempts to retain good features from both subset selection and Ridge: it leads to sparse models while being a convex optimization problem. In fact, we can show that $q = 1$ is the smallest value that leads to a convex problem, and this partly explains why lasso regression is so popular. The loss function to be minimized in Equation 3 is not differentiable due to the absolute values in the penalty term, but it is convex, and a wide range of methods from convex optimization theory have been developped to compute the solution, including coordinate descent, subgradient and proximal gradient-based methods. In this paper, lasso logistic regression is fit using the `glmnet` library of the R programming language. This library uses a proximal Newton algorithm, which consists in making a quadratic approximation of the log-likelihood part of the loss function and then applying a weighted coordinate descent, iteratively. For more details about lasso logistic regression, we refer to [Friedman et al., 2010], [Hastie et al., 2015] and [Hastie et al., 2016].

4.4 Elastic-net logistic regression

Even though lasso regression often performs very well on tabular data, it has a few drawbacks. Among other things, when there is a group of features that are highly correlated with each other, the lasso tends to select only one feature in the group and does not care which one is selected. This can be a problem for us, as some of the telematics features we have created (or even classical features) may be highly correlated with each other (e.g. `avg_daily_distance` and `avg_daily_nb_trips`). [Zou and Hastie, 2005] address this problem by proposing a new regularization and variable selection method called “elastic-net”. The elastic-net regression combines the penalties of Ridge and lasso regressions and thereby retains the best of both worlds. Ridge regression is known to share parameters among highly correlated features, which often improves performance, while lasso yields sparse models and thus performs feature selection, which is desirable. Elastic-net regression thus yields sparse models while improving the treatment of highly correlated features. More precisely, elastic-net regression includes a penalty term in its loss function that is a linear combination of the ℓ_1 (lasso) and ℓ_2 (Ridge) penalties. In the particular case of binary classification, elastic-net coefficients are therefore obtained by solving

$$\min_{\beta_0, \beta} \left\{ L(\beta_0, \beta) + \lambda \left[\frac{(1 - \alpha)}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right] \right\}, \quad (6)$$

where a new “mixing” hyperparameter $0 \leq \alpha \leq 1$ appears. Ridge and lasso regressions are in fact special cases of elastic-net regression, when $\alpha = 0$ and $\alpha = 1$, respectively. If α and λ are known, Criterion 6 is convex and can be solved by a variety of algorithms such as coordinate descent.

4.5 Random forest

Random forest classifier was first formalized by Leo Breiman in [Breiman, 2001] and enjoy great popularity for its many strengths, including a usually high accuracy on structured data. It consist in building several decision trees on slightly modified versions of the original dataset. The final prediction is then obtained by aggregating all the trees, often by taking the mean on the individual predictions. The trees built are usually very deep and therefore have little bias, but have a large variance. Aggregating them allows to drastically decrease the variance and to obtain a much more flexible prediction function, increasing the predictive power compared to a single tree. The main advantage of random forest over logistic regression is that it can approximate a wider range of functions since it is a non-parametric algorithm, making fewer assumptions about the form of the underlying data generating function. Another benefit is that it automatically takes into account the interactions between features due to the tree structure of its components. Like lasso and elastic-net logistic regressions, random forest has an embedded feature selection mechanism. On the other hand, random forest is harder to interpret, so for equal performance, logistic regression is preferred. Note that since logistic regression assumes a linear relationship between the features and the log-odd of a positive case, random forest usually outperforms it when this relationship is rather non-linear. In order to train a random forest model, one first generates bootstrap samples of the training dataset on which decision trees will later be built. This is done by drawing observations with replacement, and usually, as many observations as there are in the original training set are drawn. Then, for each of these bootstrap samples, $1 \leq p^* \leq p$ features are randomly picked, p^* being a previously chosen hyperparameter. This last step allows the decision trees to be built on different subsets of features, which has the effect of decorrelating the predictions, thus improving performance. Finally, a CART-like classification tree (see [Breiman et al., 1984]) is built on each of these datasets, each one yielding an estimated probability of having a positive case for every point of the feature space. The criterion we use to build the trees is the impurity of the nodes, measured by the Gini index. Every time the feature space is split in two, we thus choose the splitting point that decreases the Gini index the most. Other criteria are also possible. For a given point, a final prediction is obtained by averaging the individual predictions of all trees. More details about the general procedure are given in Algorithm 1. For more information about random forest, we refer to [Breiman, 2001], [Hastie et al., 2015] and [Hastie et al., 2016].

5 Analyzes

5.1 Claim classification model

In order to choose among the 3 classification models presented in Section 4 (lasso logistic regression, elastic-net logistic regression and random forest), we make them compete on the dataset whose telematics features are computed with all available trip summaries of the observed year for each vehicle, namely $\mathcal{D}_{12}^{\text{TL}}$.

Algorithm 1: Random forest binary classifier**Inputs :**

- Training dataset $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ containing p features
- Number of features to pick $1 \leq p^* \leq p$
- Number of trees (or bootstrap samples) B

for $b = 1, \dots, B$ **do**

1. Generate a bootstrap sample \mathcal{T}^* by drawing with replacement n observations from \mathcal{T} .
2. Pick at random p^* of the p features.
3. Build a CART classification tree on \mathcal{T}^* using only the p^* features previously picked, yielding the prediction function $\hat{T}_b(\mathbf{x})$.

end**Output :** Random forest classifier $\hat{f}_{\text{RF}}(\mathbf{x}) = \frac{1}{B} \sum_{b=1}^B \hat{T}_b(\mathbf{x})$ **5.1.1 Hyperparameter tuning**

First of all, the 3 models need to be tuned since they all involve hyperparameters that are not directly optimized by their respective algorithm. The general idea behind tuning is to test several combinations of hyperparameters and evaluate the out-of-sample performance for each of them, which is often done using a validation set or cross-validation on the training set. One then usually chooses the combination of hyperparameters yielding the best performance. We choose to use 5-fold cross-validation with the Area Under the receiver operating characteristic Curve (AUC) as the metric for evaluating classification performance. This metric is often used in binary classification, notably because it does not depend on the threshold used for classification and because it works well on unbalanced datasets (our dataset is highly unbalanced since there are many more non-claimant vehicles than claimant ones). Different methods have been developed to choose which combinations of hyperparameters to try out, including grid search, random search, Bayesian optimization, gradient-based optimization and evolutionary optimization. For penalized logistic regression, it is generally not necessary to use a sophisticated tuning algorithm, and one usually proceeds with a simple grid search, which we do. For the random forest, we choose a more refined method, namely a Bayesian optimization method, who have been shown to yield better results than grid search and random search (see for instance [Snoek et al., 2012]).

The lasso requires the tuning of only one hyperparameter, which is the penalty parameter λ of Equation 3. We first create a grid of 100 penalty values ranging from 10^{-10} to 1 uniformly distributed on a logarithmic scale, namely the set

$$\mathcal{G}_\lambda = \left\{ 10^{-10 + \frac{i-1}{9.9}} \right\}_{i=1}^{100}.$$

Then, 5-fold cross-validation AUC is assessed on the training set $\mathcal{T}_{12}^{\text{TL}}$ using each of the values in \mathcal{G}_λ as a candidate. It turns out that the best value for λ is 0.000231 (which is the value in \mathcal{G}_λ associated with $i = 64$), yielding an AUC of 0.6373. For the elastic-net model, the mixing parameter α must also be tuned in addition to the penalty parameter (see Equation 6). With a grid search, one usually uses a coarse uniform grid of values for α . We thus choose to use 5 values uniformly distributed between 0 and 1 inclusively, namely the grid $\mathcal{G}_\alpha = \{0, 0.25, 0.5, 0.75, 1\}$. For λ , we use the same grid as for lasso, i.e. \mathcal{G}_λ . The performance of the $|\mathcal{G}_\lambda| \times |\mathcal{G}_\alpha| = 100 \times 5 = 500$ possible combinations of hyperparameters is thereafter evaluated and it turns out that $\lambda = 0.00298$ (which is the value in \mathcal{G}_λ associated with $i = 75$) and $\alpha = 0$ is the best choice, with an AUC value of 0.6377, slightly better than lasso.

Regarding the random forest, two hyperparameters are tuned, namely the number of features drawn every time a tree is built p^* (see Algorithm 1) and the minimum number of observations required to make a further split in any leaf n^* . Note that for simplicity, the latter does not appear in Algorithm 1. The total number of trees B must also be chosen, but it is not strictly speaking a hyperparameter. It only needs to be large enough for the performance to stabilize. We choose $B = 1000$, which is plenty for the number of observations we have. Note that B cannot be too large since a random forest can never overfit due to too many trees. However, increasing the number of trees obviously increases the computation time. Bayesian optimization is used to find the best possible pair (p^*, n^*) . Basically, it consists in treating the unknown function that maps

hyperparameter values to the loss function evaluated on a test set (or with cross-validation) as a random function. An *a priori* distribution, which captures beliefs about the behavior of this function, is defined. Then, as combinations of hyperparameters are tested and evaluated, the *a priori* distribution is updated, yielding the *a posteriori* distribution. The latter is thereafter used to find the next combination of hyperparameters to try out. So unlike grid and random search, Bayesian optimization leverages past evaluations to find the most promising candidates faster. To implement this procedure, we use the `tune_bayes` function of the `tune` package in R, which uses a Gaussian process to model the probability distribution over the function. One can think of the Gaussian process as a generalization of the normal distribution concept to functions. It turns out that $(p^*, n^*) = (1, 39)$ is the best pair that has been tested, yielding an AUC value of 0.6004. This means only one feature is picked every time a tree is built and that the growth of a tree stops when all its leaves have less than 39 observations.

5.1.2 Interactions

A limitation of logistic regression is that it does not naturally take into account interactions between features, unlike random forest. Fortunately, this can be overcome by manually calculating interactions. According to the interaction hierarchy principle (see [Kuhn and Johnson, 2019]), lower-level interactions are more likely than higher-level ones to explain variation in the response variable. For instance, level 2 interactions are more likely to be predictive than level 3 interactions, which are more likely to be predictive than level 4 interactions, and so on. Therefore, in order to keep computation time reasonable, we only consider level 2 (or pairwise) interactions. We also limit ourselves to calculating the interactions between the 10 classical features of Table 2 as well as telematics features whose mean value is significantly different between claimant and non-claimant vehicles (i.e. the 7 bolded features in Table 3). These 7 features are presumed to have good predictive power, and according to the principle of heredity (see [Kuhn and Johnson, 2019]), they have a higher probability than other features of creating interactions that also have good predictive power. This entails calculating $\binom{17}{2} = 136$ pairwise interactions. Next, lasso and elastic-net regressions are tuned on the training dataset $\mathcal{T}_{12}^{\text{TL}}$ expanded with the 136 interactions as new columns. For this purpose, the same grid search procedure described in Section 5.1.1 is used.

5.1.3 Out-of-sample performance comparison

Optimal hyperparameter(s) found as well as the 5-fold cross-validation AUC value for each of the 5 tuned models are shown in Table 6. The interaction-free elastic-net model has the best cross-validation score, with an AUC of 0.6377. However, it does not outperform the lasso model, which has an AUC of 0.6373, enough to justify the extra complexity. Indeed, an elastic-net model takes more time to tune since it has an additional hyperparameter, and also takes longer to fit. Note also that with or without interactions, the two logistic regressions perform similarly considering the variability of the AUC. Since one always prefer the simplest model for equal performance, we reject the two models including interactions. Finally, the random forest is the worst model, with an AUC of only 0.6004, which is considerably lower than the penalized logistic regressions. We therefore reject the latter. Note that the Bayesian optimization algorithm has found that the optimal value for the hyperparameter p^* is 1, reinforcing the belief that interactions between features do not carry useful information for classification. Indeed, the fact that $p^* = 1$ means that the decision trees that make up the random forest are built with only one feature at a time, thus eliminating the possibility of including interactions.

Ideally, in order to properly estimate performance in supervised learning, a model should be evaluated on samples that have not yet been used to build or fine-tune it. We therefore use the testing dataset $\mathcal{V}_{12}^{\text{TL}}$ to assess the performance of the 5 tuned models. The models are first trained on the full training dataset $\mathcal{T}_{12}^{\text{TL}}$ before being scored on $\mathcal{V}_{12}^{\text{TL}}$, which allows us to compute an AUC value for each of them, shown in Table 6. The AUC values on the testing set are slightly lower than those obtained by cross-validation, which is normal. In fact, the relatively close values indicate that we did not leak too much information into the models. The lasso model with interactions have the best testing set AUC value (0.6214), but we believe it is not enough to justify the addition of the 136 interaction columns to process. From now on, we will consequently use the lasso logistic regression model without interactions. Note that the AUC values obtained are around 0.6, which is in concordance with the literature on claim classification (see for instance [Huang and Meng, 2019] and [Baecke and Bocca, 2017]).

Models	Optimal hyperparameters				AUC (5-fold cross-validation)	AUC (testing set)
	λ	α	p^*	n^*		
Lasso	2.31×10^{-4}	–	–	–	0.6373 ^(0.0052)	0.6189
Elastic-net	2.98×10^{-3}	0	–	–	0.6377 ^(0.0049)	0.6176
Random forest	–	–	1	39	0.6004 ^(0.0064)	0.5889
Lasso (with interactions)	1.18×10^{-3}	–	–	–	0.6350 ^(0.0050)	0.6214
Elastic-net (with interactions)	1.52×10^{-2}	0	–	–	0.6359 ^(0.0046)	0.6198

Table 6: Tuning results on the training set \mathcal{T}_{12}^{TL} and classification performance on the testing set \mathcal{V}_{12}^{TL} . Numbers in superscript indicate standard deviations.

5.1.4 Feature importance

Once the models are trained, in addition to the performance, it is interesting to look at which features contributed the most to classify observations. For the 3 models compared, it is possible to calculate an importance score for each feature. For lasso and elastic-net logistic regressions, since the models are trained with normalized versions of the features, the absolute value of the estimated parameter may be used for this purpose. For instance, if the estimated parameter associated with `avg_daily_distance` is β_1 , its importance score is $|\beta_1|$. For the random forest, there are many ways to compute feature importance. We choose to use the mean decrease of the Gini index. This method consists in assessing for each feature how much it has contributed to decrease the impurity of the tree nodes, measured with the Gini index. Once the importance score is obtained for all the features, we can order them from the most to the least important, which we did in Figure 4 for each of the 3 models. Looking at this figure, we can notice that the lasso and the elastic-net

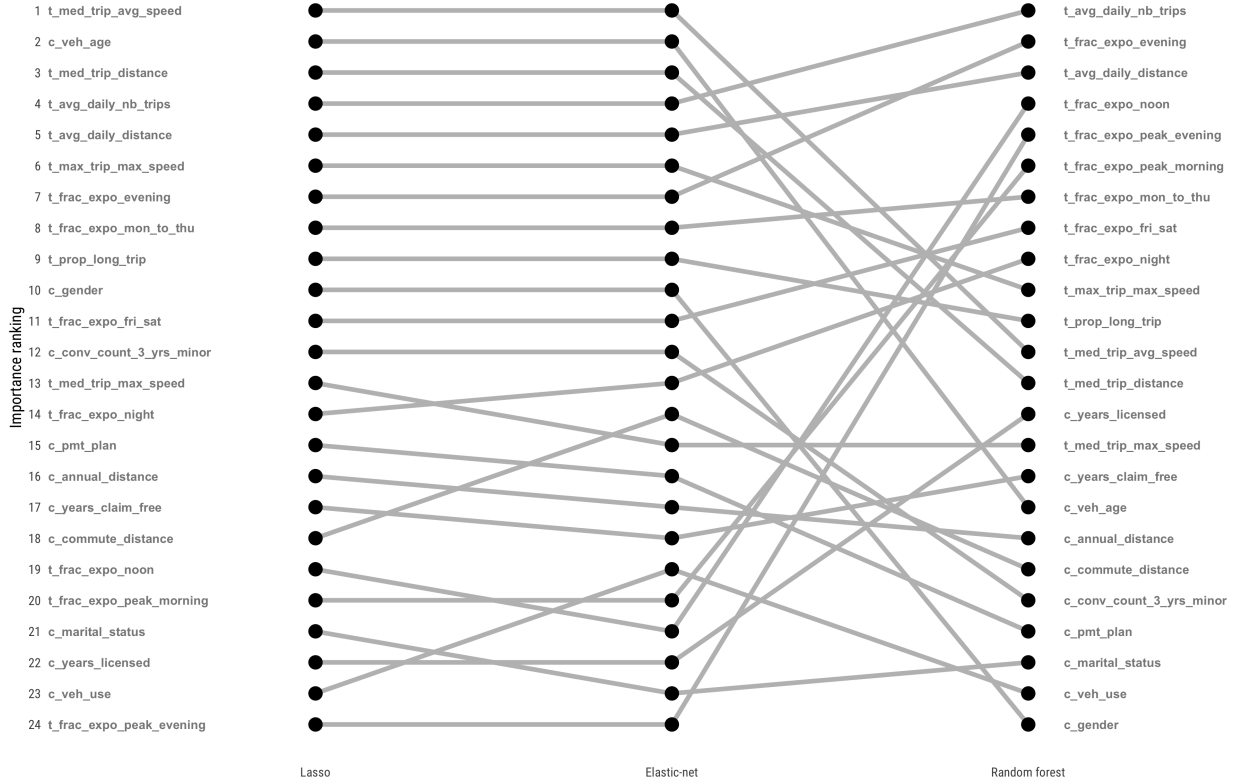


Figure 4: Ranking of the features according to their importance for each of the 3 models. Telematics features have been given the prefix “t_”, while classical features have the prefix “c_”.

models have a high degree of agreement since they consider the same 12 most important features and that the links intersect very little. This is not so surprising as these two logistic regression models work in a similar

way. On the other hand, the random forest has a lower degree of agreement with the latter two since the links intersect a lot. This is probably because a random forest makes no assumptions about the nature of the relationship between the features and the response variable, whereas logistic regression assumes that this relationship is logistic-linear. The random forest therefore probably leverages better the features with rather non-linear relationships with the response variable, in contrast to logistic regression. However, all 3 models agree that `avg_daily_nb_trips`, `avg_daily_distance`, `max_trip_max_speed`, `frac_expo_evening` and `frac_expo_mon_to_thu` are important features for prediction, all ranked in the top 10 most important features. One last important thing to note is that telematics features are considered more important by the models than their classical counterparts. Indeed, telematics features have an average ranking of 9.45, while for classical features, this value is 16.77. This makes us believe that telematics data tells a better story about an insured's risk than traditional ratemaking information.

5.2 Classification performance assessment on the 13 classification datasets

Remember that our main objective is to develop a method to estimate the amount of telematics information that an insurer should collect from its policyholders. The method we propose consists in first tuning and training a lasso model on each of the training datasets $\mathcal{T}_0^m, \dots, \mathcal{T}_{12}^m$, $m \in \{\text{TL}, \text{DL}\}$ derived in Section 3. The lasso model tuned and trained on dataset \mathcal{T}_k^m is denoted by \mathcal{M}_k^m , for $k = 0, \dots, 12$ and $m \in \{\text{TL}, \text{DL}\}$. We then assess the performance of these fitted models on their corresponding testing dataset. For instance, the fitted model \mathcal{M}_k^m is assessed on the testing dataset \mathcal{V}_k^m , for $k = 0, \dots, 12$ and $m \in \{\text{TL}, \text{DL}\}$. The performance is evaluated using the AUC and in order to obtain a distribution of the latter, a non-parametric bootstrap strategy is used. Non-parametric bootstrap is a method used to estimate the distribution of any statistic and consists in generating new samples (or datasets) called “bootstrap samples”. A bootstrap sample is simply obtained by drawing with replacement as many observations as there are in the original sample. An empirical distribution is then obtained by calculating the desired statistic on each bootstrap sample. Therefore, we generate $b = 500$ bootstrap samples for each of the 13 testing dataset $\mathcal{V}_0^m, \dots, \mathcal{V}_{12}^m$, $m \in \{\text{TL}, \text{DL}\}$. Then, in order to obtain an AUC distribution for model \mathcal{M}_k^m , we score the latter on each of the 500 bootstrap samples related to \mathcal{V}_k^m , noted $\{^{(j)}\mathcal{V}_k^m\}_{j=1}^{500}$ and we derive the 500 corresponding AUC values, which form the empirical distribution. Once the empirical distribution of the AUC has been obtained for each of the models $\mathcal{M}_0^m, \dots, \mathcal{M}_{12}^m$, $m \in \{\text{TL}, \text{DL}\}$, it is thereafter possible to inspect them and determine at what point telematics information becomes redundant or, in other words, at what point the addition of telematics information in the lasso model no longer meaningfully improve the classification performance. The AUC distributions are shown in Figure 5 for the 13 lasso models and for both approaches, namely the time leap approach (upper panel) and the distance leap approach (bottom panel). The first thing that can be noticed when looking at the boxplots is that the addition of telematics features into the model significantly improves its performance, which is in line with the literature. Indeed, the first boxplot from the left in each of the upper and bottom panels, which corresponds to the classical model, is lower than all the other ones. Looking at the upper panel, we realize that with the addition of as little as one month's worth of trip summaries, we can drastically improve classification performance. Similarly, adding just 1,000 kilometers of telematics data into the classical model also improves performance substantially (bottom panel). After 1 month or 1,000 kilometers, although the marginal improvement is less substantial, the AUC slightly increases but stabilizes fairly quickly, after about 3 months or 4,000 kilometers. After this point, telematics information collected seems redundant and no longer improves classification. This suggests that telematics features calculated with 3 months of trip summaries have about the same predictive power as those calculated with 1 complete year of data.

6 Conclusions

In this paper, we first created several relevant telematics features from trip summaries in order to incorporate telematics information into supervised classification models. Then, using these features in conjunction with features traditionally used in automobile insurance, and by adequately preprocessing the data, we compared the performance of 3 popular classification algorithms, namely a lasso logistic regression, an elastic-net logistic regression and a random forest. We found that random forest, which often gives good results in classification tasks, performs the worst, while the two logistic regressions are on equal footing. However, we chose the lasso as our classification model because of its greater simplicity. We also considered interactions between features, and found that they contain little or no predictive power since their addition into the models does not improve out-of-sample classification performance. Then, based on the lasso model and thus remaining within the framework of supervised classification, we developed a novel method for determining when information on the insured's driving becomes redundant. A great strength of our method is that it requires little computational

time and does not require second-by-second trip data, which are large and therefore time-consuming to process and difficult to manipulate. Also, it can be used by any insurance company that has access to a dataset similar to the one used in the analysis, namely a telematics dataset where each observation corresponds to a trip. Using real data from a North American insurance company, we found out, using non-parametric bootstrapping, that after about 3 months or 4,000 kilometers of observation, telematics no longer help achieving a better classification performance, at least if measured with the AUC. This means it is probably not worth for this insurer to observe its policyholders during long periods of time. Rather, it is better off observing them during a predetermined short period. Indeed, people generally do not enjoy being tracked, and telematics data is both costly and risky to store and manipulate. In addition, collecting less telematics information can help to meet the recommendations of insurance regulators and facilitate the prevention of data leakage.

In this analysis, only collision coverage claims were considered, i.e. the target column given as input to the classification models is the indicator of a collision claim, at-fault or not. One could repeat the analysis by considering at-fault and non at-fault collision claims separately, and see whether either type needs more of less telematics history to have a good estimate of the claiming probability. The analysis could also be performed again using collision-free claims, including theft, vandalism and fire, and see if we come to similar conclusions. Lastly, we could generalize the approach for count data. Therefore, instead of having the indicator of a claim as the response variable, we would instead have the number of claims, moving us into a counting regression context.

Acknowledgement

The authors gratefully acknowledge The Co-operators for both financial support and for providing the data used in this paper through the Co-operators Chair in Actuarial Risk Analysis. The authors also thank the Natural Sciences and Engineering Research Council of Canada for funding.

References

- [Ayuso et al., 2019] Ayuso, M., Guillen, M., and Nielsen, J. P. (2019). Improving automobile insurance ratemaking using telematics: incorporating mileage and driver behaviour data. *Transportation*, 46(3):735–752.
- [Baecke and Bocca, 2017] Baecke, P. and Bocca, L. (2017). The value of vehicle telematics data in insurance risk selection processes. *Decision Support Systems*, 98:69–79.
- [Bordoff and Noel, 2008] Bordoff, J. E. and Noel, P. J. (2008). *The impact of pay-as-you-drive auto insurance in California*. Brookings Institution.
- [Boucher et al., 2017] Boucher, J.-P., Côté, S., and Guillen, M. (2017). Exposure as duration and distance in telematics motor insurance using generalized additive models. *Risks*, 5(4):54.
- [Boucher et al., 2007] Boucher, J.-P., Denuit, M., and Guillén, M. (2007). Risk classification for claim counts: a comparative analysis of various zeroinflated mixed poisson and hurdle models. *North American Actuarial Journal*, 11(4):110–131.
- [Boucher and Turcotte, 2020] Boucher, J.-P. and Turcotte, R. (2020). A longitudinal analysis of the impact of distance driven on the probability of car accidents. *Risks*, 8(3):91.
- [Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.
- [Breiman, 2001] Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.
- [Breiman et al., 1984] Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.
- [Dewri et al., 2013] Dewri, R., Annadata, P., Eltarjaman, W., and Thurimella, R. (2013). Inferring trip destinations from driving habits data. In *Proceedings of the 12th ACM workshop on privacy in the electronic society*, pages 267–272.
- [Friedman et al., 2010] Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.
- [Gao and Wüthrich, 2018] Gao, G. and Wüthrich, M. V. (2018). Feature extraction from telematics car driving heatmaps. *European Actuarial Journal*, 8(2):383–406.

- [Gao and Wüthrich, 2019] Gao, G. and Wüthrich, M. V. (2019). Convolutional neural network classification of telematics car driving data. *Risks*, 7(1):6.
- [Gao et al., 2018] Gao, G., Wuthrich, M. V., and Yang, H. (2018). Driving risk evaluation based on telematics data. *Available at SSRN 3288347*.
- [Greenberg, 2009] Greenberg, A. (2009). Designing pay-per-mile auto insurance regulatory incentives. *Transportation Research Part D: Transport and Environment*, 14(6):437–445. The interaction of environmental and traffic safety policies.
- [Hastie et al., 2016] Hastie, T., Qian, J., and Tay, K. (2016). An introduction to glmnet.
- [Hastie et al., 2015] Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical learning with sparsity: the lasso and generalizations*. CRC press.
- [Howell, 2013] Howell, P. (2013). Usage-based automobile insurance pricing in ontario (2013).
- [Huang and Meng, 2019] Huang, Y. and Meng, S. (2019). Automobile insurance classification ratemaking based on telematics driving data. *Decision Support Systems*, 127:113156.
- [Kuhn and Johnson, 2019] Kuhn, M. and Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*. CRC Press.
- [Lemaire et al., 2015] Lemaire, J., Park, S. C., and Wang, K. (2015). The use of annual mileage as a rating variable. *Astin Bulletin*, 46(1):39.
- [Litman, 2007] Litman, T. (2007). Distance-based vehicle insurance feasibility, costs and benefits. *Victoria*, 11.
- [Paefgen et al., 2013] Paefgen, J., Staake, T., and Thiesse, F. (2013). Evaluation and aggregation of pay-as-you-drive insurance rate factors: A classification analysis approach. *Decision Support Systems*, 56:192–201.
- [Pérez-Marín and Guillen, 2019] Pérez-Marín, A. M. and Guillen, M. (2019). Semi-autonomous vehicles: Usage-based data evidences of what could be expected from eliminating speed limit violations. *Accident Analysis & Prevention*, 123:99–106.
- [Pesantez-Narvaez et al., 2019] Pesantez-Narvaez, J., Guillen, M., and Alcañiz, M. (2019). Predicting motor insurance claims using telematics data—xgboost versus logistic regression. *Risks*, 7(2):70.
- [Snoek et al., 2012] Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*.
- [Tibshirani, 1996] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.
- [Verbelen et al., 2017] Verbelen, R., Antonio, K., and Claeskens, G. (2017). Unraveling the predictive power of telematics data in car insurance pricing. *Available at SSRN 2872112*.
- [Vickrey, 1968] Vickrey, W. (1968). Automobile accidents, tort law, externalities, and insurance: An economist’s critique. *Law and Contemporary Problems*, 33(3):464–487.
- [Weidner et al., 2016] Weidner, W., Transchel, F. W., and Weidner, R. (2016). Classification of scale-sensitive telematic observables for riskindividual pricing. *European Actuarial Journal*, 6(1):3–24.
- [Wüthrich, 2017] Wüthrich, M. V. (2017). Covariate selection from telematics car driving data. *European Actuarial Journal*, 7(1):89–108.
- [Yeo and Johnson, 2000] Yeo, I.-K. and Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959.
- [Zou and Hastie, 2005] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.

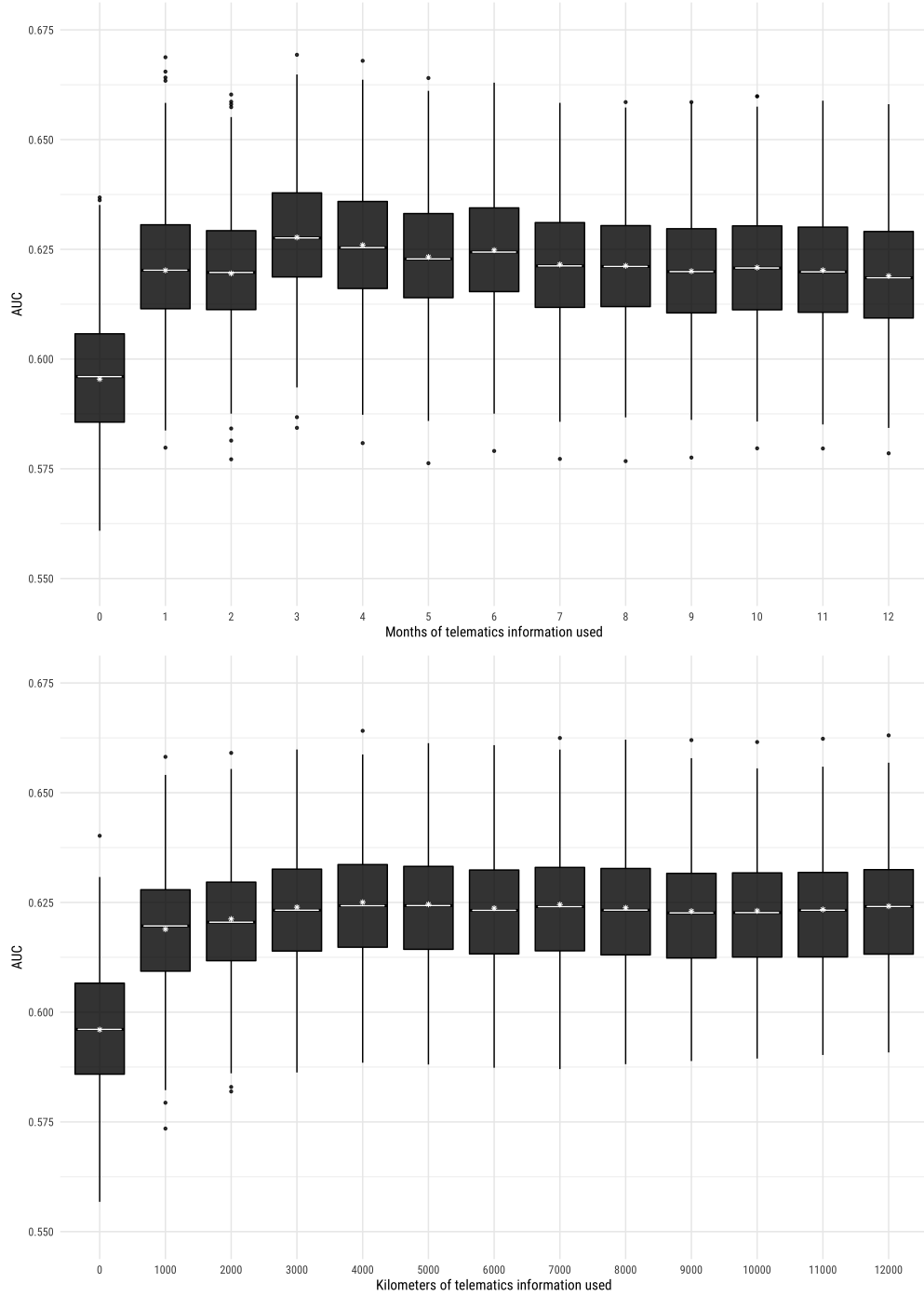


Figure 5: Distribution of the classification performance (AUC) obtained with non-parametric bootstrap for the 13 models and for both approaches. White stars show the AUC value obtained on the original testing set.