UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MODÉLISATION DES SINISTRES EN ASSURANCE AUTOMOBILE AVEC L'UTILISATION DE DONNÉES TÉLÉMATIQUES : APPROCHES D'APPRENTISSAGE AUTOMATIQUE EN CLASSIFICATION ET RÉGRESSION DE COMPTAGE

THÈSE

PRÉSENTÉE

COMME EXIGENCE PARTIELLE

DU DOCTORAT EN MATHÉMATIQUES

PAR

FRANCIS DUVAL

MARS 2024

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MODELING AUTOMOBILE INSURANCE CLAIMS USING TELEMATICS DATA: MACHINE LEARNING APPROACHES IN CLASSIFICATION AND COUNT REGRESSION

THESIS

PRESENTED

AS PARTIAL REQUIREMENT

TO THE PH.D IN MATHEMATICS

BY

FRANCIS DUVAL

MARCH 2024

REMERCIEMENTS

Je remercie sincèrement tous ceux et celles qui ont contribué au succès de cette thèse. Tout d'abord, je suis infiniment reconnaissant envers mes deux co-directeurs de thèse, Jean-Philippe Boucher et Mathieu Pigeon. Merci d'avoir été des mentors attentifs, engagés et disponibles. Votre grande expertise et votre soutien constant ont grandement contribué à l'aboutissement de cette thèse. Merci pour vos précieux conseils et encouragements. Merci d'avoir répondu à toutes mes questions, même pendant vos jours de vacances. J'aimerais également souligner votre implication exceptionnelle au sein de la Chaire Co-operators en analyse des risques actuariels (CARA), où vous avez su créer un environnement d'étude et de recherche stimulant.

J'exprime mes sincères remerciements aux membres du jury, Marie-Hélène Descary, Arthur Charpentier et Frédéric Godin, pour avoir accepté d'évaluer ma thèse et pour le temps précieux qu'ils y ont consacré.

Mes remerciements vont également à ma copine Marie-Noëlle, qui m'a apporté son soutien et a su me faire rire tout au long de la rédaction de cette thèse. Sa présence et son soutien ont été d'une grande importance pour moi. Je tiens également à remercier mes parents, Brigitte et Simon, pour leur soutien inconditionnel. Leur amour, leurs encouragements et leur soutien constant ont été essentiels dans la réalisation de cette thèse. Merci pour tous les soupers du dimanche que vous m'avez préparés. Merci également à mes beaux-parents Marilène et Bruno pour leur soutien.

Je tiens ensuite à exprimer ma profonde gratitude envers Co-operators pour avoir fourni les données nécessaires à cette thèse, ainsi que pour leur généreux soutien financier. Votre contribution a été essentielle à la réalisation de cette recherche. Je souhaite particulièrement remercier Marc Morin de l'équipe d'intelligence d'affaires pour son aide précieuse dans la programmation du troisième chapitre de cette thèse. Son expertise et son assistance ont été d'une valeur inestimable et ont grandement contribué à la réussite de cette partie de mon travail.

Je souhaite également remercier le Conseil de recherches en sciences naturelles et en génie du Canada ainsi que les Fonds de recherche du Québec pour leur généreuse contribution financière tout au long de mes études doctorales. Les bourses octroyées par ces deux organismes m'ont permis de me consacrer pleinement à la rédaction de cette thèse.

Enfin, je tiens à exprimer mes sincères remerciements à mes collègues de la Chaire CARA. Votre présence et votre soutien ont été essentiels pour maintenir ma motivation tout au long de cette thèse. Les discussions enrichissantes et les échanges d'idées que nous avons eus ont grandement contribué à mon développement en tant que chercheur.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

RÉSUMÉ

Les produits d'assurance automobile ont traditionnellement été tarifés en fonction des attributs auto-déclarés fournis par les assurés. Ces attributs incluent généralement différents facteurs de risque tels que le sexe, l'âge, le lieu de résidence, le statut marital, l'utilisation du véhicule et l'historique des réclamations. Les assureurs se basent sur ces informations pour évaluer le niveau de risque associé à chaque contrat d'assurance et déterminer la prime appropriée. Récemment, les assureurs ont commencé à collecter des données sur la conduite de leurs assurés, ce qui ouvre de nouvelles possibilités en termes de tarification automobile et entraîne une véritable révolution dans le monde de l'assurance. Ces données télématiques comprennent des informations détaillées telles que la vitesse, l'accélération, le freinage, la distance parcourue et la position géographique. Une tarification basée sur ces données, connue sous le nom d'assurance basée sur l'usage, a été démontrée pour offrir de nombreux avantages. Cette approche permet une tarification plus précise et personnalisée en se basant sur les habitudes de conduite réelles de chaque assuré. Les avantages de l'assurance basée sur l'usage incluent également une meilleure équité dans la tarification, la promotion d'une conduite responsable, une amélioration de la sécurité routière et une réduction des émissions de gaz à effet de serre.

Cependant, cette nouvelle source de données présente de nombreux défis. L'un de ces défis est lié à la gestion et au stockage de ces données, qui sont souvent volumineuses. Un autre défi majeur réside dans le traitement et l'analyse de ces données. Afin de tirer pleinement parti des avantages offerts par ce nouveau paradigme, il est essentiel de développer des algorithmes avancés permettant d'extraire des informations pertinentes à partir de ces données. Cela implique notamment le développement de nouveaux algorithmes de tarification capables d'utiliser de manière optimale les données télématiques. Il est également important de comprendre ces données, notamment les relations entre les différentes variables télématiques et le risque d'accident.

En s'appuyant principalement sur des algorithmes d'apprentissage automatique, cette thèse a pour objectif d'améliorer les méthodes de tarification avec données télématiques en assurance automobile tout en approfondissant la compréhension du lien unissant les données télématiques et le niveau de risque des assurés. Au Chapitre 1, nous abordons la question de la quantité minimale de données télématiques requise pour obtenir une estimation précise du risque d'un assuré. En effet, il est dans l'intérêt d'un assureur de minimiser la quantité de données utilisées en raison de leur coût de stockage élevé et du temps de calcul considérable qu'elles exigent dans les algorithmes. Nous abordons cette question en utilisant un modèle de régression logistique avec régularisation lasso dans le contexte de la classification de réclamations, où le but est d'estimer la probabilité de réclamer pour chaque contrat d'assurance. Au Chapitre 2, nous développons une procédure basée sur des algorithmes non-supervisés de détection d'anomalies permettant d'extraire automatiquement des variables à partir des données télématiques. Cette méthode implique le calcul d'un score de « routine » et de « péculiarité » pour chaque trajet effectué par un véhicule. L'ensemble de ces scores constitue un profil de routine et de péculiarité pour chaque véhicule, à partir desquels des quantiles sont extraits pour être utilisés dans un algorithme de classification des réclamations. Nous proposons un modèle de classification incluant ces quantiles comme prédicteurs et utilisant une régularisation *elastic-net*, permettant une sélection automatique des variables. Enfin, au Chapitre 3, nous développons de nouveaux modèles de régression de comptage des réclamations utilisant des données télématiques. Notre approche repose sur une architecture de réseau neuronal spécifiquement conçue pour résoudre des problèmes actuariels, le *Combined Actuarial Neural Network* (CANN). Cette architecture combine un modèle de régression classique avec un réseau neuronal, offrant ainsi le meilleur des deux mondes. Alors que le modèle de régres-

sion classique fournit une base solide et interprétable, le réseau neuronal permet de capturer des relations complexes et des interactions non linéaires entre les variables. Cela signifie qu'il est capable d'extraire automatiquement des variables, ou représentation latentes, à partir des données télématiques dans ses couches cachées. Un aspect clé de ce chapitre est l'adaptation de l'architecture CANN à la spécification binomiale négative multivariée pour les données longitudinales.

ABSTRACT

Automobile insurance products have traditionally been priced based on self-reported attributes provided by insureds. These attributes typically include various risk factors such as gender, age, dwelling location, marital status, vehicle usage, and claim history. Insurers rely on this information to assess the level of risk associated with each insurance contract and determine the appropriate premium. Recently, insurers have started collecting telematics car driving data from their insureds, leading to new avenues in automobile pricing and revolutionizing the insurance industry. Telematics car driving data includes detailed information such as speed, acceleration, braking, distance traveled, and geographical location. Pricing based on this data, known as usage-based insurance, has proven to offer many benefits. This approach allows for more accurate and personalized pricing based on the actual driving behavior of each policyholder. The benefits of usage-based insurance also include improved fairness in pricing, promotion of responsible driving, enhanced road safety, and reduced greenhouse gas emissions.

However, this new data source presents several challenges. One of these challenges is related to the management and storage of the often large volumes of data. Another major challenge lies in processing and analyzing this data. To fully leverage the benefits offered by this new paradigm, it is essential to develop advanced algorithms capable of extracting relevant information from the data. This includes the development of new pricing algorithms that can effectively use telematics data. Understanding the data is also crucial, including the relationships between different telematics variables and the risk of accidents.

Relying mainly on machine learning algorithms, the goal of this thesis is to improve pricing methods using telematics data in automobile insurance while gaining a deeper understanding of the relationship between telematics data and insured risk levels. In Chapter 1, we address the question of the minimum amount of telematics data required to obtain an accurate estimation of an insured's risk. Minimizing the amount of data used is of interest to insurers due to the high cost of storage and considerable computational time required in algorithms. We tackle this question using a logistic regression model with lasso regularization in the context of claims classification, where the goal is to estimate the claiming probability for each insurance contract. In Chapter 2, we develop a procedure based on unsupervised anomaly detection algorithms that automatically extract features from telematics data. This method involves computing a "routine" and "peculiarity" score for each vehicle trip. The collection of these scores forms a routine and peculiarity profile for each vehicle, from which quantiles are extracted for use in a claim classification algorithm. We propose a classification model that includes these quantiles as predictors and use elastic-net regularization, enabling automatic variable selection. Finally, in Chapter 3, we develop new claim count regression models using telematics data. Our approach is based on a neural network architecture specifically designed for actuarial problems, the Combined Actuarial Neural Network (CANN). This architecture combines a classical regression model with a neural network, offering the best of both worlds. While the classical regression model provides a solid and interpretable foundation, the neural network captures complex relationships and nonlinear interactions between input variables. This means it can automatically extract features, or latent representations, from the telematics data in its hidden layers. One key aspect of this chapter is the adaptation of the CANN architecture to the multivariate negative binomial distribution specification for longitudinal data.

INTRODUCTION

Contexte

L'assurance joue un rôle de premier plan dans l'économie depuis plusieurs millénaires. La première forme connue de transfert de risque remonte à la civilisation babylonienne, vers le II^{e} millénaire avant notre ère, avec le « prêt à la grosse aventure » (Wikipedia contributors, nd), une forme de prêt à haut risque dans laquelle un prêteur octroyait des fonds à un capitaine de navire pour financer un voyage commercial. Selon ce type de prêt, si le voyage se déroulait comme prévu et que le navire revenait à bon port, le prêteur se faisait rembourser son prêt initial en plus de recevoir un intérêt supplémentaire. En revanche, si le navire coulait ou si la cargaison était endommagée ou perdue d'une quelconque manière, le prêteur devait renoncer à être remboursé et perdait donc son investissment initial. Le risque était alors partagé entre plusieurs prêteurs, atténuant ainsi le risque supporté par le capitaine. Cela procurait essentiellement deux avantages majeurs : d'une part, le capitaine était protégé d'une possible perte économique catastrophique, et d'autre part, les commerçants étaient davantage incités à entreprendre des voyages commerciaux, ce qui favorisait le développement économique. De nos jours, l'assurance revêt de nombreuses formes et constitue un pilier central dans notre économie de marché. En assumant une partie des risques liés aux événements incertains tels que les accidents, les maladies ou les catastrophes naturelles, l'assurance permet aux individus, aux entreprises et aux institutions de se prémunir contre des pertes financières potentiellement dévastatrices. Cela encourage l'entrepreneuriat, l'investissement et l'innovation en réduisant les risques associés à de nouvelles activités économiques, ce qui favorise la stabilité et la croissance économiques.

La présente thèse porte sur l'assurance automobile, qui vise à protéger les individus et les entreprises contre divers risques liés aux dommages matériels, aux blessures corporelles et à la responsabilité civile. Dans l'industrie de l'assurance, on observe ce qu'on appelle un « cycle de production inversé », signifiant que le coût d'un produit d'assurance n'est pas connu à l'avance, contrairement à la plupart des produits disponibles sur le marché. Par exemple, le coût de la tasse qui est, au moment d'écrire ces lignes, posée sur mon pupitre de travail, est connu par le manufacturier bien avant la vente de celle-ci. Pour un produit d'assurance, une partie significative des coûts sont aléatoires puisqu'ils dépendent de la fréquence et de la sévérité des réclamations futures de l'assuré, réputées pour être hautement imprévisibles. En conséquence, un agent, généralement un ou une actuaire, a l'important mandat de déterminer le coût des produits d'assurance. En général, les actuaires se concentrent à calculer la prime pure, c'est-à-dire le montant à charger pour couvrir les éventuelles réclamations liées aux sinistres. La prime pure est systématiquement moins élevée que la prime réellement chargée à l'assuré car elle n'inclut pas les diverses marges telles que les frais d'administration, les marges de profit, les taxes et autres frais associés à l'exploitation de l'assureur. Des modèles statistiques sont employés pour calculer la prime pure à charger à chaque assuré. Pour estimer le risque en assurance automobile, ces modèles s'appuient sur des caractéristiques du risque, telles que l'âge, le genre, la situation matrimoniale et le code postal de l'assuré, ainsi que sur des attributs liés au véhicule tels que le modèle et la marque. Ces caractéristiques sont sélectionnées selon leur pouvoir prédictif, c'est-à-dire selon la force de leur lien avec le risque à évaluer.

Jusqu'à récemment, les assureurs s'étaient contentés pendant plusieurs décennies de tarifer leurs produits d'assurance automobile en utilisant un ensemble restreint de ces caractéristiques du risque fournies à des modèles linéaires généralisés. Or, l'essor de la technologie télématique dans les dernières années a introduit une nouvelle source de données disponibles aux assureurs, ouvrant la voie à un changement de paradigme en assurance automobile, et même plus largement en assurance de dommages. « Télématique » est un mot-valise formé en combinant les mots « télécommunications » et « informatique ». Il est utilisé pour décrire le domaine qui se situe à l'intersection de ces deux branches, et fait référence, de manière fondamentale, à la transmission de données à distance. En assurance automobile, la technologie télématique est utilisée pour collecter des données sur la conduite des assurés via des dispositifs installés dans les véhicules ou des applications mobiles. Ces données comprennent notamment des informations telles que la vitesse de conduite, l'accélération, le freinage, les virages, la distance parcourue, les heures de conduite et la position géographique. Naturellement, les données télématiques ont un fort potentiel d'utilisation en tarification automobile car elles fournissent de l'information détaillée sur les habitudes et le style de conduite des assurés. Plusieurs assureurs collectent déjà ce genre de données auprès de leurs assurés et les utilisent pour établir leur tarification. Par exemple, certains offrent des rabais à ceux et celles qui répondent à leurs critères de conduite responsable. Cette approche est fréquemment désignée sous le terme d'assurance basée sur l'usage (UBI, acronyme anglais pour *Usage-Based Insurance*).

L'assurance basée sur l'usage est sur le point de révolutionner le monde de l'assurance en apportant des bénéfices considérables à de multiples entités impliquées, avec peu d'inconvénients. D'abord, elle pourrait avoir un impact considérable sur la sécurité des routes en encourageant une conduite plus responsable. De plus, en adoptant une tarification basée sur la distance parcourue, elle pourrait contribuer à la réduction des émissions de gaz à effet de serre en incitant une utilisation parcimonieuse du véhicule. Du point de vue de l'assureur, l'information télématique permet un calcul plus précis et plus segmenté des primes, permettant d'attirer les bons risques tout en évitant l'antisélection, ce qui confère un avantage concurrentiel. Finalement, les assurés bénéficient de ce type d'assurance puisqu'elle permet une tarification plus équitable et basée sur le comportement réel de conduite de chaque individu. Ainsi, plusieurs injustices seront résolues, notamment le cas d'un assuré appartenant à une classe de risque présentant un risque élevé, alors que son risque réel est en réalité faible. Les assurés jugés à haut risque à la lumière de leurs données télématiques auront quant à eux l'option d'améliorer leur comportement routier. Les données télématiques ont même le pouvoir de rendre obsolètes certaines variables traditionnellement utilisées en tarification automobile qui sont maintenant jugées éthiquement sensibles. D'ailleurs, certains pays ont déjà interdit ou restreint l'utilisation de certaines variables, telles que le genre et la région de l'assuré, pour le calcul tarifaire. L'assurance basée sur l'usage a ainsi le potentiel de réduire la discrimination fondée sur des critères personnels et immuables, en privilégiant plutôt une tarification basée sur le comportement de conduite.

Le concept d'assurance basée sur l'usage est attribué à l'économiste William Vickrey qui, en 1968, a remis en question le système traditionnel de tarification, soutenant que les facteurs de risque utilisés ne sont pas directement liés au risque d'accident et peuvent donc créer des biais et des injustices dans la tarification. Il a été le premier à suggérer aux assureurs d'utiliser des données télématiques pour établir le prix de leurs primes. Cependant, à l'époque, la technologie télématique était encore à ses balbutiements, rendant impossible l'application de ses idées. Le géo-positionnement par satellite (GPS), par exemple, n'a été pleinement opérationnel qu'en 1995. Ainsi, les chercheurs ont véritablement commencé à s'intéresser au domaine de l'assurance basée sur l'usage au début des années 2000, dès lors que les compagnies d'assurance ont commencé à collecter des données sur la conduite de leurs assurés. Depuis lors, de nombreuses recherches ont été conduites afin de comprendre différentes facettes de ces nouvelles données. Les chercheurs ont cherché à comprendre les multiples impacts de cette nouvelle méthode de tarification, notamment sur les plans économique, sociétal, environnemental, juridique et éthique. Parallèlement, des études ont été menées pour explorer les possibilités offertes par cette nouvelle source de données. Des chercheurs se sont ainsi penchés sur la modélisation de la conduite, essayant de comprendre les comportement et les habitudes de conduite des assurés. D'autres se sont focalisés à développer de nouveau modèles de tarification mettant à profit les données télématiques.

Problème posé

Autrefois, posséder des données était pratiquement synonyme de savoir. Cependant, nous avons aujourd'hui accès à une telle profusion de données qu'il ne suffit plus de les posséder: il est desormais essentiel de savoir quoi en faire et de quel angle les traiter. Les données télématiques, en raison de leur volume considérable, présentent un défi majeur en termes de gestion, de traitement et d'analyse. Les chercheurs se retrouvent confrontés à la tâche complexe de comprendre ces données riches en information et d'en tirer des connaissances exploitables, notamment pour la tarification en assurance automobile. Afin d'optimiser les avantages reliés à l'assurance basée sur l'usage discutés précedemment, il est important de mettre en place une tarification adéquate adaptée à ce nouveau paradigme. Les données de conduite des assurés contiennent de l'information précieuse pour évaluer leur risque individuel et pour déterminer une prime juste. Il a par exemple été démontré dans le cadre de plusieurs études que la distance parcourue, qui représente une mesure d'exposition au risque, est fortement liée au risque d'accident. Cependant, la distance parcourue n'est pas la seule information télématique pertinente pour évaluer le risque d'un assuré, ce qui fait que d'autres études ont été menées pour identifier les meilleurs moyens d'extraire des facteurs de risque à partir de ces données.

Parmi les études proposant des modèles de tarification basée sur l'usage, plusieurs proposent d'extraire des variables de tarification à partir des données de conduite en se basant sur le jugement humain. Par exemple, il est envisageable que la connaissance des périodes de la journée dans lesquelles un assuré conduit le plus souvent puisse contribuer à une meilleure évaluation de son risque. Une chercheuse peut alors obtenir, à partir des données de conduite, la proportion de la distance conduite dans différentes plages horaires, telles que les périodes de pointe et les périodes de faible affluence routière. En d'autres termes, la chercheuse essaie, en utilisant son jugement et son bon sens, d'extraire des variables qui, avec un peu de chance, seront corrélées avec le risque d'un assuré, permettant ainsi de mieux estimer son niveau de risque.

Ces dernières années, des approches dites « data driven » ont émergé, c'est-à-dire des approches qui mettent davantage l'accent sur les données. On pourrait dire que celles-ci permettent aux données de s'exprimer plus « librement », sans être contraintes par des hypothèses ou jugements subjectifs. Par rapport à l'approche décrite dans le précédent paragraphe, elles ont l'avantage d'être moins influencées par les failles et les biais humains. Elles nécessitent néanmoins des algorithmes plus sophistiqués capables d'extraire automatiquement des variables de tarification à partir de données télématiques très détaillées.

Il est désormais indéniable que les fournisseurs d'assurance doivent privilégier l'assurance basée sur l'usage. Au cours des dernières années, la recherche s'est attelée à déterminer comment tirer le meilleur parti des données télématiques. Cependant, ce domaine en est encore en pleine effervescence et il est essentiel de poursuivre les recherches afin d'explorer de nouvelles perspectives et approches permettant de réaliser le plein potentiel offert par la télématique.

Objectifs et structure de la thèse

La présente thèse a pour objectif d'améliorer les méthodes de tarification basée sur l'usage en assurance automobile ainsi que de mieux comprendre le lien unissant les données télématiques au risque. Pour parvenir à cet objectif, nous nous appuyons principalement sur des méthodes d'apprentissage automatique. Nous disposons de données télématiques sous la forme de résumés de trajets, ainsi qu'un jeu de données traditionnel d'assurance comprenant divers attributs liés à l'assuré, au véhicule et à l'expérience de sinistres. Une clé nous permet d'associer chaque résumé de trajet à un contrat d'assurance du jeu de données traditionnel.

La thèse est divisée en trois chapitres, chacun correspondant à un article scientifique. Au Chapitre 1, nous nous intéressons à la quantité minimale de données télématiques nécessaire pour obtenir une estimation précise du risque d'un assuré. Les données télématiques sont souvent très volumineuses et leur coût de stockage, élevé. De plus, elles requièrent un temps de calcul considérable lorsqu'elles sont utilisées dans des algorithmes de plus en plus exigeants en puissance de calcul. Il est donc dans l'intérêt de l'assureur de garder et d'utiliser un minimum de ces données. Nous abordons cette question dans le contexte de la classification de réclamations, où l'objectif principal est d'estimer la probabilité de réclamer pour chaque assuré. De plus, nous proposons un algorithme de classification qui utilise de la régularisation lasso et intègre des variables extraites manuellement à partir des données télématiques.

Au Chapitre 2, nous examinons si le niveau de routine et de particularité d'un assuré peut contribuer à l'estimation de son risque de réclamation. Nous développons une procédure pour extraire automatiquement des variables à partir des données télématiques. La procédure consiste à d'abord extraire un profile de « routine » et un profil de « péculiarité » pour chaque véhicule en utilisant des algorithmes de détection d'anomalies non supervisés. Des quantiles sont ensuite extraites de ces profils, et nous étudions leur pouvoir prédictif dans le contexte de la classification des réclamations.

Au Chapitre 3, nous développons des modèles transversaux et longitudinaux de régression de comptage des réclamations en utilisant une architecture spéciale de réseau de neurones appelée *Combined Actuarial Neural Network* (CANN). Cette architecture est spécifiquement conçue pour les problématiques actuarielles et permet de combiner un modèle de tarification classique avec un réseau de neurones. Alors que la partie classique du modèle fournit une base solide, le réseau de neurones nous permet d'analyser plus en profondeur les données afin d'identifier des signaux qui pourraient avoir été négligés par le modèle de tarification classique. Plus précisément, nous utilisons un perceptron multicouches pour traiter les données télématiques et en extraire automatiquement des variables utiles pour la tâche de régression. Pour la spécification du nombre de réclamations, nous considérons trois distributions : Poisson, binomiale négative et négative binomiale multivariée. Cette dernière distribution permet une analyse longitudinale plutôt que transversale et tient donc compte de la dépendance entre les contrats appartenant à un même véhicule. Cette approche nous permet de mieux modéliser les réclamations et d'obtenir des estimations plus précises du risque associé à chaque contrat.

Article associé au Chapitre 1

Duval, F., Boucher, J. P., & Pigeon, M. (2022). How Much Telematics Information Do Insurers Need for Claim Classification?. *North American Actuarial Journal*, 26(4), 570–590.

Article associé au Chapitre 2

Duval, F., Boucher, J. P., & Pigeon, M. (2023). Enhancing claim classification with feature extraction from anomaly-detection-derived routine and peculiarity profiles. *Journal of Risk and Insurance*, 90(2), 421–458.

Article associé au Chapitre 3

Duval, F., Boucher, J. P., & Pigeon, M. (2023). Telematics Combined Actuarial Neural Network for Cross-Sectional and Longitudinal Claim Count Data. *Soumis pour publication*.

CHAPTER 1
HOW MUCH TELEMATICS INFORMATION DO INSURERS NEED FOR CLAIM CLASSIFICATION?

1.1 Introduction

Il a été démontré à plusieurs reprises dans la littérature que les données télématiques collectées dans le cadre de l'assurance automobile aident à mieux comprendre le risque de conduite d'un assuré. Les assureurs qui utilisent ces données en tirent plusieurs avantages, tels qu'une meilleure estimation de la prime pure, une tarification plus segmentée et moins d'antisélection. Le revers de la médaille est que les informations télématiques collectées sont souvent sensibles et peuvent donc compromettre la vie privée des assurés. De plus, en raison de leur grand volume, ce type de données est coûteux à stocker et difficile à manipuler. Ces facteurs, combinés au fait que les régulateurs de l'assurance ont tendance à émettre de plus en plus de recommandations concernant la collecte et l'utilisation des données télématiques, rendent important pour un assureur de déterminer la bonne quantité d'informations télématiques à collecter. En plus des facteurs de risque traditionnels tels que l'âge et le sexe de l'assuré, nous avons accès à un jeu de données télématiques où les informations sont résumées par trajet. Nous dérivons d'abord plusieurs variables de tarification à partir de ces résumés de trajet avant de construire un modèle de classification des réclamations en utilisant à la fois des facteurs de risque traditionnels et télématiques. En comparant quelques algorithmes de classification, nous constatons que la régression logistique avec une pénalité lasso est la plus adaptée à notre problème. En utilisant ce modèle, nous développons une méthode pour déterminer la quantité d'informations sur la conduite des assurés qui doivent être conservées par un assureur. En utilisant des données réelles provenant d'une compagnie d'assurance nord-américaine, nous constatons que les données télématiques deviennent redondantes après environ 3 mois ou 4 000 kilomètres d'observation, du moins du point de vue de la classification des réclamations.

1.2 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

1.3 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 survey (Watson, 2017) conducted by *Willis Towers Watson* on 1005 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 et Guillen, 2019), (Verbelen *et al.*, 2017) and (Lemaire *et al.*, 2015)). UBI also has many positive impacts on society (see for instance (Greenberg, 2009) and (Bordoff et Noel, 2008)). 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, implemention, costs and benefits of UBI. (Litman, 2007) explores the practicability, pros and cons of differents types of distance-based insurance, such as *Mileage 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 et 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 et 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)[1]), 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 et Meng, 2019), the authors compare the performance of various classification algorithms while proposing a way to bin continuous telematics variables in order to 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 et Wüthrich, 2018), (Gao *et al.*, 2018), (Gao et 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

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

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 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 1.4, we describe the classical and telematics data available to us in conjunction with details of how telematics features are derived. Section 1.5 follows, where we show how the classification datasets are built using varying amounts of information as well as other preprocessing steps. Then, in Section 1.6, 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 1.7, in which we first make a choice among the classification models presented in Section 1.6. 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 1.5, 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 1.8.

1.4 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 between January 2016 and December 2018, for which an extract is shown in Table 1.1. 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 1.1: Extract from the telematics dataset. Dates are displayed in the yyyy-mm-dd format.

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 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 1.2. Distributions of the classical numeric features and those of the classical categorical features are shown in Figures 1.6 and 1.7 of Appendix 1.9, respectively. Only vehicles having at least one full-year contract are kept for the analysis, which means we end up with $29{,}799$ vehicles. In the following section, the telematics dataset will be aggregated by contract (which means the latter will go from one row

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 1.2: Classical features selected for the analysis.

per trip to one row per contract) and then merged with classical features and claim information (which are already on a contract basis). The resulting dataset will then be given as input to supervised learning algorithms. Since we know that supervised learning algorithms generally learn best on independent observations, we keep only the earliest one-year contract for each vehicle, which allows us to get rid of the dependency that exists between the different contracts associated with the same vehicle. 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 1.1.

Based on the trip summaries of Table 1.1, 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 et 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 et Meng, 2019). From the telematics dataset, we thus extract a total of 14

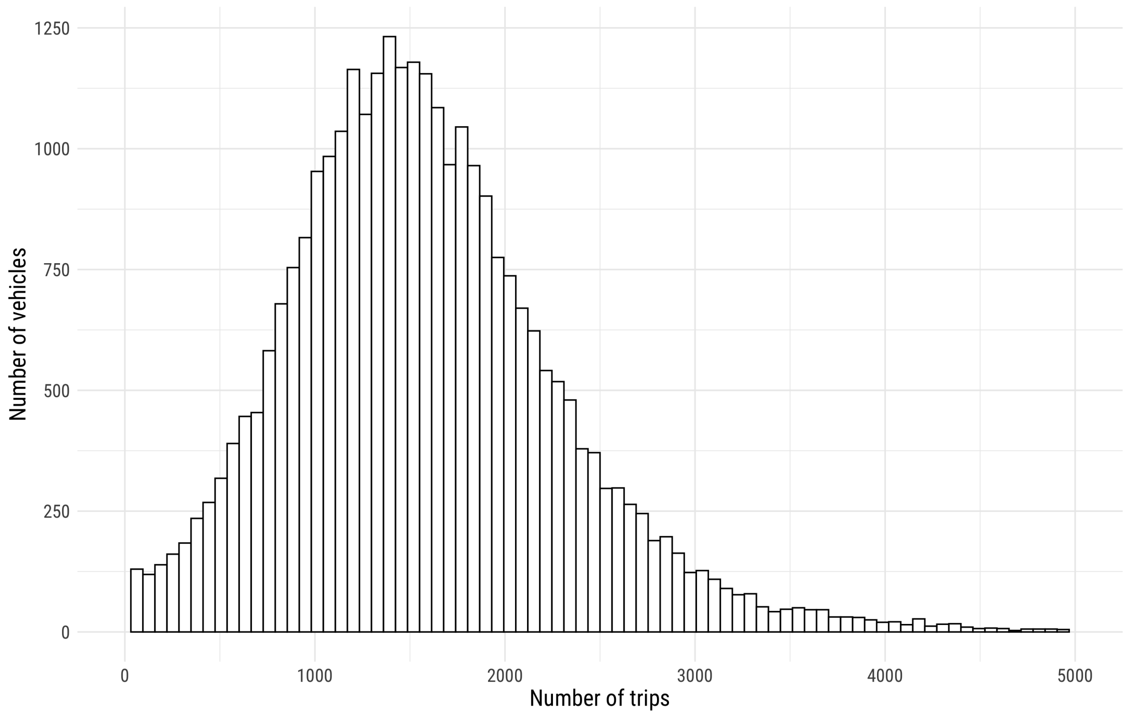


Figure 1.1: Histogram of the number of trips for the $29{,}799$ vehicles in their observed year.

features, all described in Table 1.3 and for which the distribution is shown in Figure 1.2. In Table 1.3, we also display the average 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 95% confidence level) for half of the 14 features, which we have highlighted in bold in Table 1.3. 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 mid-

[2] 0h-6h

[3] 11h-14h

[4] 20h-0h

[5] 7h-9h Monday to Friday

[6] 17h-20h Monday to Friday

		Mean value		
Feature name	**Description**	Non-claimants (94.7%)	Claimants (5.3%)	**p-value (t-test)**
avg_daily_distance	**Average daily distance**	$\mathbf{44.3}$	$\mathbf{50.1}$	$\mathbf{< 0.0001}$
avg_daily_nb_trips	**Average daily number of trips**	$\mathbf{4.3}$	$\mathbf{4.8}$	$\mathbf{< 0.0001}$
med_trip_avg_speed	**Median of the average speeds of the trips**	$\mathbf{28.8}$	$\mathbf{28.0}$	$\mathbf{< 0.0001}$
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**	$\mathbf{138.2}$	$\mathbf{141.9}$	$\mathbf{< 0.0001}$
prop_long_trip	Proportion of long trips (> 100km)	0.0108	0.0103	0.4114
frac_expo_night	Fraction of night driving[2]	0.0262	0.0276	0.1630
frac_expo_noon	**Fraction of midday driving[3]**	$\mathbf{0.210}$	$\mathbf{0.198}$	$\mathbf{< 0.0001}$
frac_expo_evening	**Fraction of evening driving[4]**	$\mathbf{0.0845}$	$\mathbf{0.0965}$	$\mathbf{< 0.0001}$
frac_expo_peak_morning	Fraction of morning rush hour driving[5]	0.0968	0.0985	0.4453
frac_expo_peak_evening	**Fraction of evening rush hour driving[6]**	$\mathbf{0.137}$	$\mathbf{0.143}$	$\mathbf{< 0.0001}$
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 1.3: 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.

day and more in the evening, especially in the rush hour. Note that we have at our disposal 2 “mileage” variables. The first one, `annual_distance`, is declared by the insured at the very beginning of the contract and corresponds to the number of kilometers that the insured estimates he/she will drive during the following year. The second, `avg_daily_distance`, is measured by the OBD device and is the actual number of kilometers the insured has driven during the year, divided by 365.25, which is the average number of days in a year. One could argue that these two features tell roughly the same thing about an insured and that we should get rid of one of them. However, insureds underestimate their distance travelled in a year by an average 1,399 kilometers. Indeed, the average of the `avg_daily_distance` feature times 365.25 is 16,374 kilometers while the average of `annual_distance` is 14,975 kilometers. Moreover, the Pearson correlation coefficient between these two is only 0.37, which means there is a considerable discrepancy between `annual_distance` and `avg_daily_distance`. We therefore keep both of these features in our analysis, since each one tells a different story about an insured. In order to determine which features (classical and telematics combined) are significant in predicting the occurrence of a claim, a non-penalized logistic regression was fitted to the data. The coefficients obtained for the 24 features and their standard deviations are

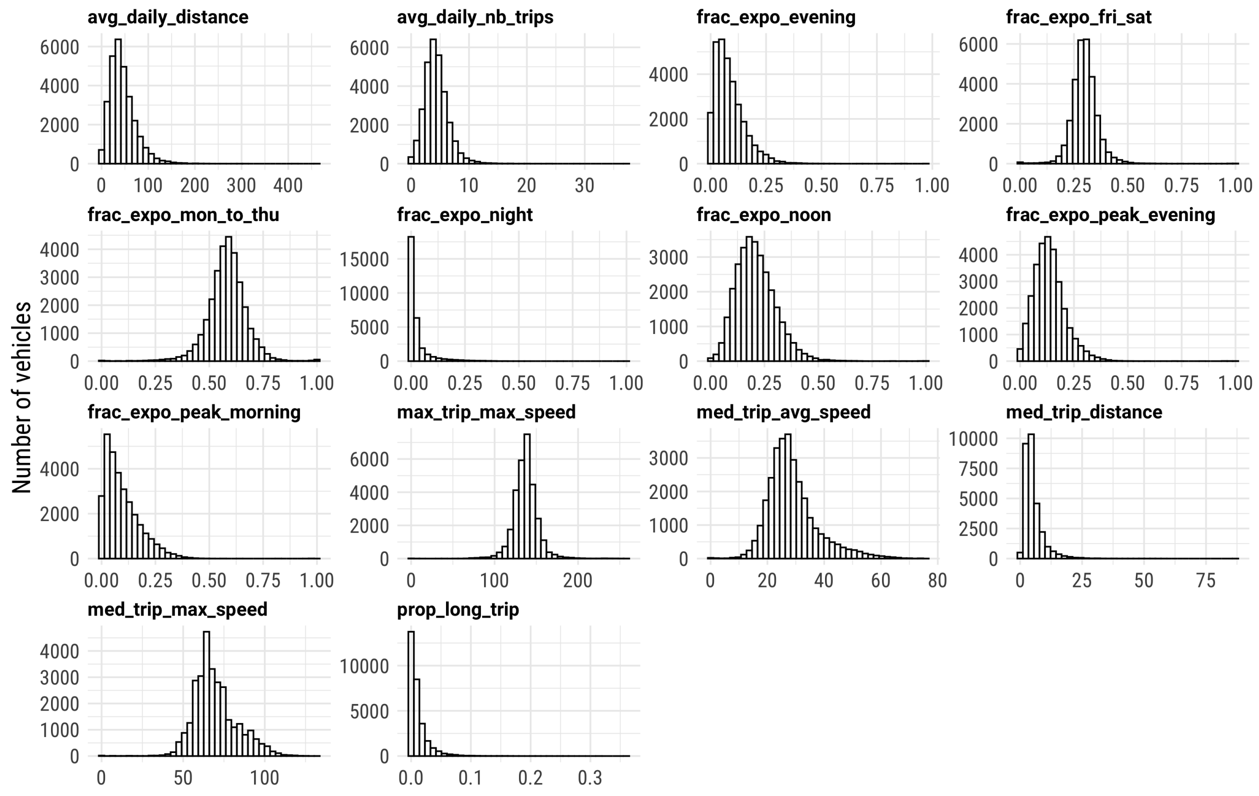


Figure 1.2: Distributions over the $29{,}799$ vehicles of the 14 features extracted from the telematics dataset.

shown in Figure 1.8 of Appendix 1.9. It turns out that 10 of the 24 coefficients, among which 7 are related to telematics features, are significant (using a 5% threshold).

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 1.4. The calculation of the telematics features related to this insured would then be done according to Table 1.5.*

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,

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

insurers were offering 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: the latter could only be reduced. 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.

1.5 Data preparation

1.5.1 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 1.3 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 1.3 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 $k = 1, \ldots, 12$. The second way to proceed, called the “distance leap” method (DL), is quite similar to the time leap way, but uses 1,000-

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`	$\max\{73,70,102,92,120\}$	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 1.5: Telematics features calculated for the fictitious insured of Table 1.4.

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 of a telematics feature for vehicle i is calculated using the first $1{,}000 \times k$ kilometers of telematics information related to this vehicle, for $k = 1, \ldots, 12$. If it turns out that the vehicle has done less than $1{,}000k$ kilometers in its observed year, we simply use all available telematics information from this vehicle. 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 5,500 kilometers of driving has its distance leap versions 6 to 12 of the telematics fatures calculated with the same amount of telematics information, namely with 5,500 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 $\boldsymbol{x}_k^{\text{TL}}$, while the one derived according to the distance leap method is noted $\boldsymbol{x}_k^{\text{DL}}$. Let us also denote by $\boldsymbol{x}^c$ the $29{,}799 \times 10$ matrix containing the classical features of Table 1.2 and by $\boldsymbol{y}$ 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 $\mathcal{D}_1^{\mathsf{TL}}, \dots, \mathcal{D}_{12}^{\mathsf{TL}}$, all sharing the same classical features $\boldsymbol{x}^c$ as well as the same response vector $\boldsymbol{y}$. 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, $\mathcal{D}_k^{\mathsf{TL}}$, $k = 1, \dots, 12$, is the classification dataset built with the matrix of telematics features $\boldsymbol{x}_k^{\mathsf{TL}}$, which means it is obtained by concatenating $\boldsymbol{x}^c$, $\boldsymbol{x}_k^{\mathsf{TL}}$ and $\boldsymbol{y}$. In addition to these 12 classification datasets, we also create a dataset containing no telematics information, noted D_0^{TL}. The latter is therefore built by concatenating $\boldsymbol{x}^c$ and $\boldsymbol{y}$. Note that all 13 datasets describe the same vehicles and therefore have the same number of rows. In a similar fashion, 13 datasets $\mathcal{D}_0^{\mathsf{DL}}, \dots, \mathcal{D}_{12}^{\mathsf{DL}}$ 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 $m \in \{\mathsf{TL}, \mathsf{DL}\}$.

1.5.2 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 embeding 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 $\widehat{\beta}_1, \dots, \widehat{\beta}_k$. Then, each of the k categories is encoded with its corresponding coefficient. Hence, category j is encoded with the value $\widehat{\beta}_j$, for $j = 1, ..., k$. 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_lencode_glm` function of the `embed` package in the `R` programming language ((R Core Team, 2020)) 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 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`.

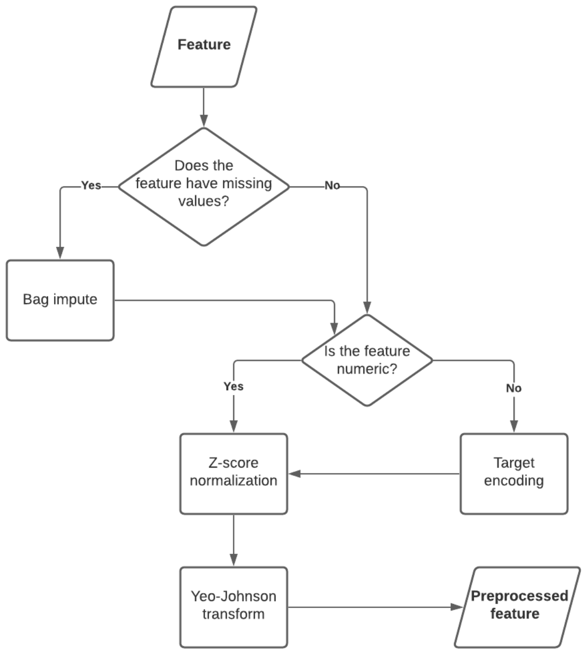


Figure 1.3: Flowchart of the feature preprocessing “recipe”, that is applied every time a model is trained or validated in this analysis.

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 $\boldsymbol{x} = (x_1, \ldots, x_n)$. The z-score normalized version of this vector is

$$\boldsymbol{x}^* = \left(\frac{x_1 - \overline{x}}{s}, \ldots, \frac{x_n - \overline{x}}{s} \right),$$

where $\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$ and $s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2}$ are the empirical mean and standard deviation of vector $\boldsymbol{x}$, respectively. In order to obtain more symmetric feature distributions, we use the Yeo-Johnson transformation (see (Yeo et 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 1.3 illustrates the data preprocessing "recipe".

1.6 Binary classification algorithms

1.6.1 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 $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})$ 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 $\mathbb{E}[Y|\boldsymbol{x}]$, the conditional mean of Y given the features, which can also be seen as the conditional probability $\mathbb{P}(Y = 1|\boldsymbol{x})$. 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.

1.6.2 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

$$p_i := \mathbb{P}(Y_i = 1|\boldsymbol{x}_i) = \sigma\left(\beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}\right), \tag{1.1}$$

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 $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)$, 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

$$L(\beta_0, \boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \ln(p_i) + (1 - y_i)\ln(1 - p_i). \tag{1.2}$$

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.

1.6.3 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 1.2. In lasso logistic regression, the optimization problem thus becomes

$$\min_{\beta_0,\boldsymbol{\beta}} \left\{ L(\beta_0,\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}, \tag{1.3}$$

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 1.3 is called the Lagrangian formulation of the lasso logistic regression optimization problem. It can be useful to rewrite this in the constrained form,

$$\min_{\beta_0,\boldsymbol{\beta}} \{L(\beta_0,\boldsymbol{\beta})\} \quad \text{s.t.} \quad \sum_{j=1}^{p} |\beta_j| \leq s. \tag{1.4}$$

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 $\boldsymbol{\beta}$, 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

$$\sum_{j=1}^{p} |\beta_j|^q \leq s, \tag{1.5}$$

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 Rigde 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 1.3 is not differentiable due to the absolute values in the penalty term, but it is convex, and a wide range of methods from convex optimzation 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).

1.6.4 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 et 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

$$\min_{\beta_0,\boldsymbol{\beta}} \left\{ L(\beta_0, \boldsymbol{\beta}) + \lambda \left[\frac{(1-\alpha)}{2} \left\| \boldsymbol{\beta} \right\|_2^2 + \alpha \left\| \boldsymbol{\beta} \right\|_1 \right] \right\}, \tag{1.6}$$

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 1.6 is convex and can be solved by a variety of algorithms such as coordinate descent.

1.6.5 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 embeded 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).

1.7 Analyzes

1.7.1 Claim classification model

In order to choose among the 3 classification models presented in Section 1.6 (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 $\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ containing $p$ features
  • Number of features to pick $1 \leq p^* \leq p$
  • Number of trees (or bootstrap samples) $B$
for $b = 1, \ldots, B$ 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 $\widehat{T}_b(\boldsymbol{x})$.
end
Output: Random forest classifier $\hat{f}_{\text{RF}}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} \widehat{T}_b(\boldsymbol{x})$
```

1.7.1.1 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 1.3.

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 $\mathcal{G}_\lambda$ as a candidate. It turns out that the best value for λ is 0.000231 (which is the value in $\mathcal{G}_\lambda$ 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 1.6). 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. $\mathcal{G}_\lambda$. 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 $\mathcal{G}_\lambda$ 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.

1.7.1.2 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 et 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 1.2 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 1.3). These 7 features are presumed to have good predictive power, and according to the principle of heredity (see (Kuhn et 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 1.7.1.1 is used.

1.7.1.3 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 1.6. 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 $\mathcal{V}_{12}^{\mathsf{TL}}$ to assess the performance of the 5 tuned models. The models are first trained on the full training dataset $\mathcal{T}_{12}^{\mathsf{TL}}$ before being scored on $\mathcal{V}_{12}^{\mathsf{TL}}$, which allows us to compute an AUC value for each of them, shown in Table 1.6. The AUC values on the testing set are slightly lower than those obtained by cross-validation, which is normal.

	Optimal hyperparameters					
Models	λ	α	p^*	n^*	**AUC (5-fold cross-validation)**	**AUC (testing set)**
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 1.6: Tuning results on the training set $\mathcal{T}_{12}^{\mathsf{TL}}$ and classification performance on the testing set $\mathcal{V}_{12}^{\mathsf{TL}}$. Numbers in superscript indicate standard deviations.

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 et Meng, 2019) and (Baecke et Bocca, 2017)).

1.7.1.4 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 $\widehat{\beta}_1$, its importance score is $|\widehat{\beta}_1|$. 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 1.4 for each of the 3 models. Looking at this figure, we can notice that the lasso and

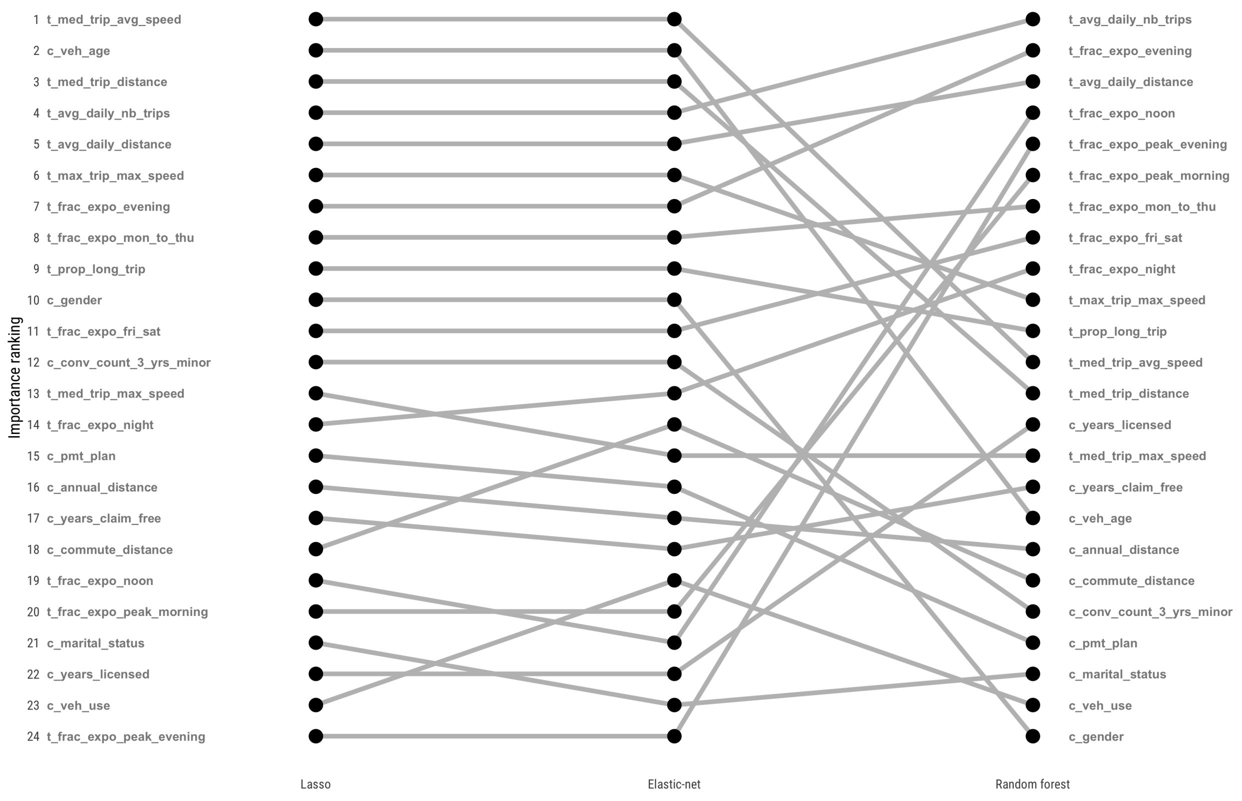
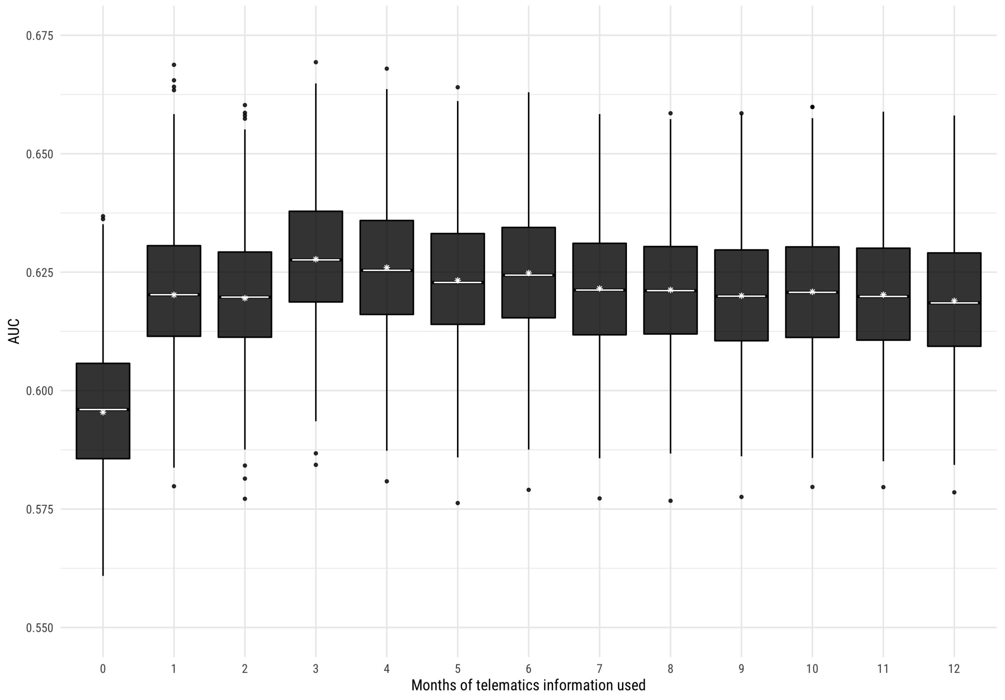


Figure 1.4: Ranking of the features according to their importance for each of the 3 models. Telematics features have been given the prefix “t_”, while classical features have the prefix “c_”.

the elastic-net 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, which is consistent with the literature. For instance, (Verbelen *et al.*, 2017) show using GAMs that telematics features have better predictive power than their classical counterparts. Indeed, their GAM model which only uses telematics outperforms their GAM using only classical factors, at least according to some criteria. In traditional pricing, i.e. in pricing without driving data, gender is an important factor in determining an insured's premium. Note that here, gender only ranks 10^{th}, and this is probably due to the fact that gender is a proxy for other factors related to driving habits. As pointed out by (Ayuso *et al.*, 2016b), it seems that the difference in the risk of accident between men and women is mainly due to the difference in driving intensity, which is captured by two of our telematics features, namely `avg_daily_nb_trips` and `avg_daily_distance`.

1.7.2 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 $\mathcal{T}_0^m, \ldots, \mathcal{T}_{12}^m$, $m \in \{\text{TL}, \text{DL}\}$ derived in Section 1.5. The lasso model tuned and trained on dataset $\mathcal{T}_k^m$ is denoted by $\mathcal{M}_k^m$, for $k = 0, \ldots, 12$ 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 datset $\mathcal{V}_k^m$, for $k = 0, \ldots, 12$ 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 $\mathcal{V}_0^m, \ldots, \mathcal{V}_{12}^m$, $m \in \{\text{TL}, \text{DL}\}$. 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 $\mathcal{M}_0^m, \ldots, \mathcal{M}_{12}^m$, $m \in \{\text{TL}, \text{DL}\}$, 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 1.5 for the 13 lasso models and for both approaches,



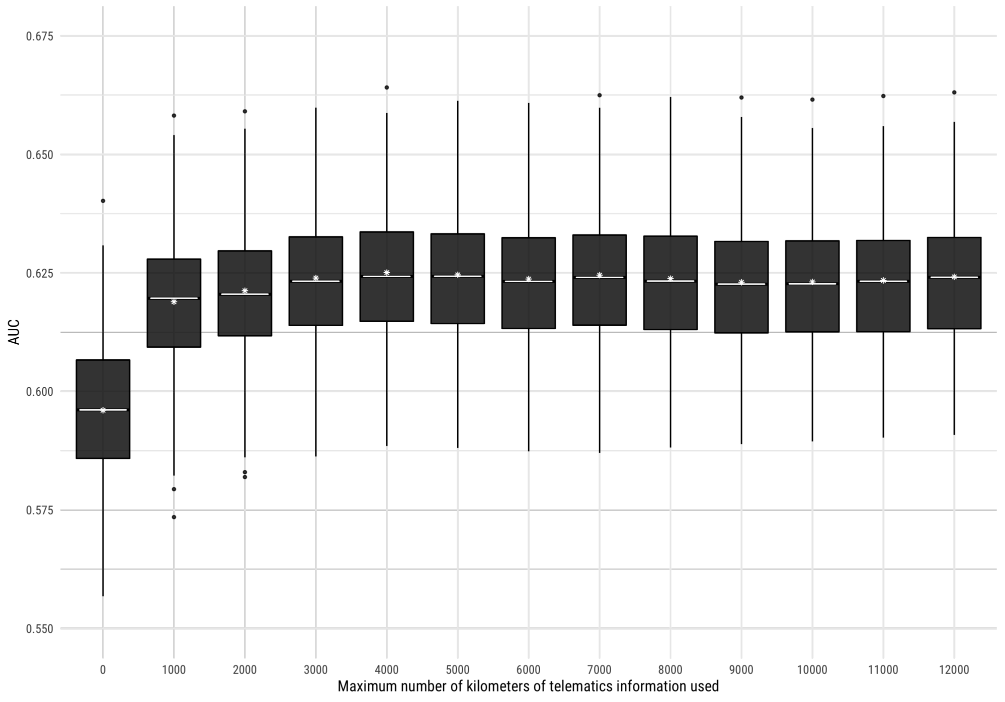


Figure 1.5: Distribution of the classification performance (AUC) obtained with non-parametric bootstrap for the 13 models and for both approaches. Black stars show the AUC value obtained on the original testing set.

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.

1.8 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 bootstraping, 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.

As reported in (Bolderdijk *et al.*, 2011), policyholders tend to adopt better driving habits in the early months of telematics observation when a financial incentive is given. Therefore, it is probably not a good idea to observe an insured's driving habits for 3 months (or 4,000 kilometers), determine a discount (or surcharge) and then apply that same discount (or surcharge) *ad vitam aeternam*. This could indeed result in pure premiums that are too low, since the observed behavior would be biased due to a lack of financial incentive to drive well after the observation period. A simple solution to this issue would be to monitor policyholders at random times during their contract, so as to eventually collect 3 months (or 4000 kilometers) of data. (Guillen *et al.*, 2021) address this issue by making pricing dynamic. Basically, they propose a pricing scheme in which the premium is updated weekly, which drives the insured to adopt good driving habits on an ongoing basis. They combine a baseline premium related to traditional risk factors with extra charges related to driving behaviors deemed unsafe captured by what they call "near-miss events", such as hard braking and smartphone use when driving. Moreover, they provide an alternative way to solve the problem of minimizing data storage since their method only requires keeping one week of telematics data. However, unlike our paper, they do not use information related to driving habits such as the proportion of night driving, average speed, proportion of long trips, etc., which have a demonstrated link to claim experience. Their method also requires collecting good near-miss data, which is not trivial. In this analysis, only collision coverage claims were considered, i.e. the target column given as input to the clasification 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.

1.9 Appendix A

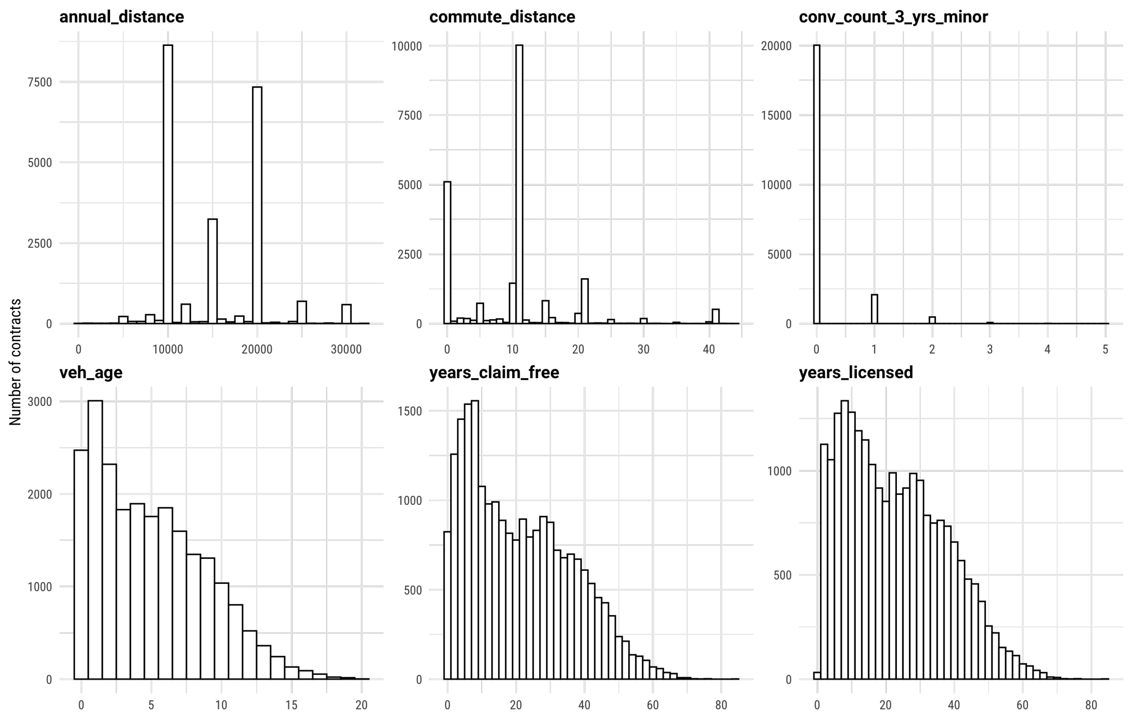


Figure 1.6: Distributions over the 29,799 vehicles of the 6 classical numeric features selected for the analysis.

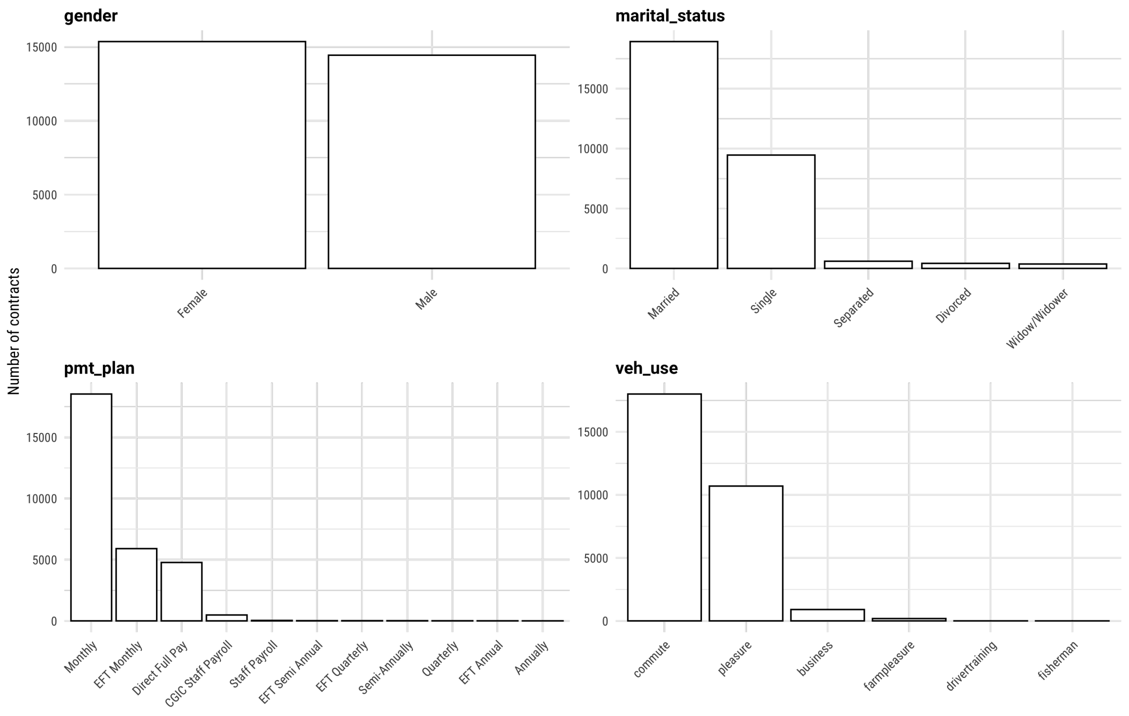


Figure 1.7: Distributions over the 29,799 vehicles of the 4 classical categorical features selected for the analysis.

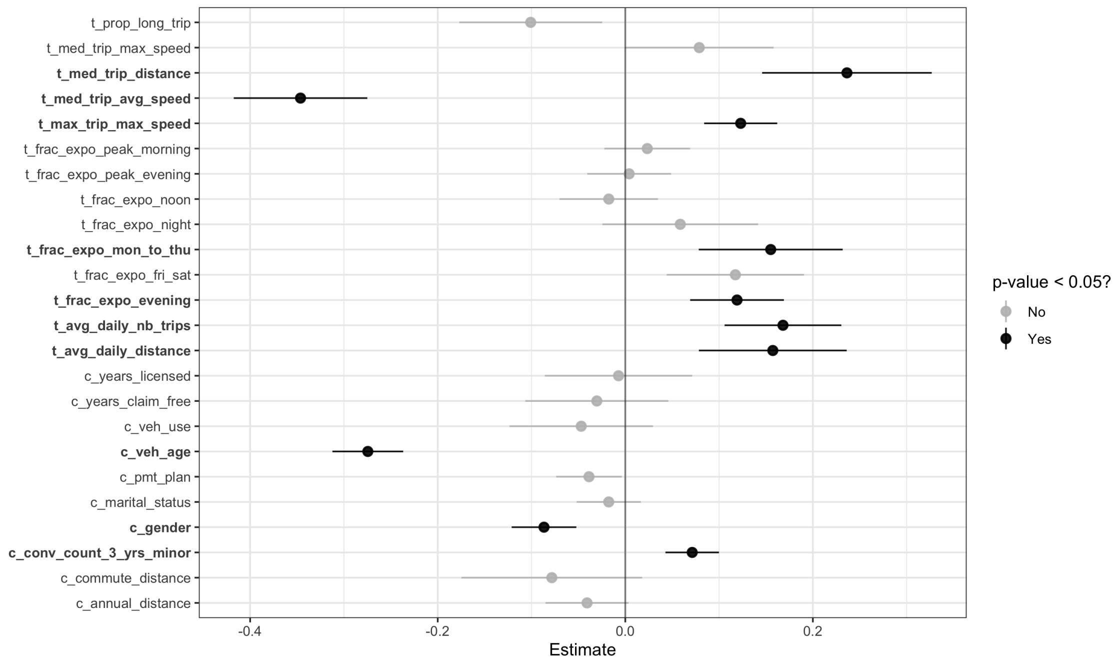


Figure 1.8: Non-penalized logistic regression coefficients and their standard deviation for classical and telematics features obtained with the 29,799 vehicles.

CHAPTER 2
ENHANCING CLAIM CLASSIFICATION WITH FEATURE EXTRACTION FROM ANOMALY-DETECTION-DERIVED ROUTINE AND PECULIARITY PROFILES

2.1 Introduction

L'assurance basée sur l'usage devient la nouvelle norme en matière d'assurance automobile: il est donc pertinent de trouver des moyens efficaces d'utiliser les données de conduite des assurés. En appliquant la détection d'anomalies (AD) aux résumés de trajets des véhicules, nous développons une méthode permettant de dériver un profil de « routine » et un profil de « singularité » pour chaque véhicule. À cette fin, des algorithmes de détection d'anomalies sont utilisés pour calculer un score d'anomalie de routine et un score d'anomalie de singularité pour chaque trajet effectué par un véhicule. Le premier mesure le degré d'anomalie du trajet par rapport aux autres trajets effectués par le véhicule concerné, tandis que le second mesure son degré d'anomalie par rapport aux trajets effectués par n'importe quel véhicule. Les vecteurs de scores d'anomalies résultants sont utilisés comme profils de routine et de singularité. Des caractéristiques sont ensuite extraites de ces profils, pour lesquelles nous étudions le pouvoir prédictif dans le cadre de la classification des réclamations. En utilisant des données réelles, nous constatons que les caractéristiques extraites du profil de singularité des véhicules améliorent la classification.

2.2 Abstract

Usage-based insurance is becoming the new standard in vehicle insurance; it is therefore relevant to find efficient ways of using insureds' driving data. Applying anomaly detection (AD) to vehicles' trip summaries, we develop a method allowing to derive a "routine" and a "peculiarity" anomaly profile for each vehicle. To this end, AD algorithms are used to compute a routine and a peculiarity anomaly score for each trip a vehicle makes. The former measures the anomaly degree of the trip compared with the other trips made by the concerned vehicle, while the latter measures its anomaly degree compared with trips made by any vehicle. The resulting anomaly scores vectors are used as routine and peculiarity profiles. Features are then extracted from these profiles, for which we investigate the predictive power in the claim classification framework. Using real data, we find that features extracted from the vehicles' peculiarity profile improve the classification.

Keywords: Automobile insurance, Telematics car driving data, Driving habits, Unsupervised anomaly detection, Supervised learning, Claim classification, Feature extraction

2.3 Introduction and Motivations

Although the idea dates back to (Vickrey, 1968), usage-based insurance (UBI) is a fairly new insurance scheme mostly used in vehicle insurance, in which the insured's premium is estimated by making use of their driving data, which is recorded by a mechanism, usually an on-board diagnostics (OBD) device or a smartphone application. Nowadays, the latter is favored by insurers over the OBD device because it is cheaper and more practical. Determining a fair pure premium for each insured is a crucial task for any sensible insurance company; traditionally, automobile insurers have relied mainly upon static attributes related to the vehicle or the insured, which are indirectly related to accident risk. With the rise of telematics technology, it is now feasible for insurers to offer a more customized premium that is more in line with an insured's risk, which may now be determined by considering the insured's volume, habits and style of driving. UBI, or pricing using telematics data, seems likely to be the future standard in automobile insurance and the market share of UBI products is currently growing quickly. According to a study[1] conducted by *Allied Market Research*, a market research firm based in Portland, Oregon, the worldwide UBI market is expected to expand at a compound annual growth rate of 25.1% from 2020 to 2027, reaching US$ 149.2 billion by 2027. The reason for this rapid expansion is that UBI products are valuable to both insurers and insureds. On one side, UBI benefits insurers because usage-based premiums tend to attract safer drivers[2], which lowers claim costs and increases profit margins. In other words, having more granular premiums allows them to avoid adverse selection and helps them stay competitive in a fierce market. On the other hand, it goes without saying that safer drivers and drivers with low driving volume are eager to buy this type of insurance product, which saves them money. UBI may even attract riskier drivers who wish to adjust their driving behavior in order to obtain a discount. Perhaps they are only risky drivers because they previously had few monetary incentives for safe driving. UBI products also seem to increase customer satisfaction because they allow customers to have more control over their premium. On top of that, usage-based premiums can help achieve societal goals because they promote less and safer driving, thereby reducing road congestion and pollution (see, for instance, (Litman, 2007) and (Bordoff et Noel, 2008)). They also help reduce discrimina-

[1] `https://www.alliedmarketresearch.com/usage-based-insurance-market`

[2] This is only true in a free market. In some regions, the UBI market is not free due to regulations that prevent insurers from applying a surcharge based on insureds' driving data.

tion by insurers based on immutable criteria, such as the gender of the insured. The COVID-19 pandemic has accelerated the shift toward telematics insurance. Indeed, with the recent boom in work-from-home culture, people are using their vehicles less and are therefore attracted to UBI, which is more beneficial to them.

For pricing purposes, insurers generally model claim risk using supervised learning models, typically generalized linear models (GLM; (Nelder et Wedderburn, 1972), (Dionne et Vanasse, 1989)), and use risk factors as independent variables. Traditionally, risk factors involved in pricing have been static insured- or vehicle-related features that are thought to be correlated with the policy risk, such as the insured's gender, the vehicle age, the bonus-malus level, etc. Such risk factors are easily used in prediction models because they can be encoded with just one entry per policy. With devices recording telematics data, each policy can now be linked with a complete history of driving information. One of the challenges insurers face is therefore to translate this data into relevant telematics-based risk factors. Perhaps the most obvious one that can be derived from driving data is the total distance driven during the policy, or mileage, which can be seen as a measure of risk exposure. Motor insurance in which pricing is done using mileage as a rate factor is often referred to as pay-as-you-drive (PAYD) insurance. It has been shown in several studies that distance driven is strongly linked with claim frequency (see for instance (Litman, 2005), (Ferreira Jr. et Minikel, 2010), (Boucher *et al.*, 2013) and (Lemaire *et al.*, 2016)). Consequently, some papers have focused upon this risk factor and have further investigated the relation between mileage and claim frequency: (Boucher *et al.*, 2017) make use of generalized additive models (GAM; (Hastie et Tibshirani, 1990)) and splines to model the non-linear effect of mileage on claim frequency and propose a pricing scheme that incorporates this information; more recently, (Boucher et Turcotte, 2020) find, using generalized additive models for location, scale and shape (GAMLSS; (Rigby et Stasinopoulos, 2005)), that while the apparent association between mileage and claim frequency seems non-linear, the true relationship is quite linear. Accordingly, they conclude that each additional kilometer driven by an insured increases the expected number of claims by about $\frac{1}{15{,}000}$, and state that this could be used to give a surcharge/rebate to an insured that drives more/less than expected.

Although distance driven is probably the best telematics-based feature for predicting claims, it does not tell the whole story about an insured's risk and therefore, further driving data may be useful for pricing. Driving data can be divided into two classes: driving-habits-related data and driving-style-related data. The former includes information on *when*, *where* and *how much* the insured drives, whereas the latter describes *how* they drive. In this regard, motor insurance in which the premium is computed using driving style-related data is often referred to as pay-how-you-drive (PHYD) insurance. Some studies focus on driving habits-based risk factors ((Paefgen *et al.*, 2013), (Paefgen *et al.*, 2014), (Guillen *et al.*, 2019)). (Verbelen *et al.*, 2017), besides using mileage and number of trips, compute compositional predictors based on information such as the distribution of distance driven over different time slots and types of road. (Ayuso *et al.*, 2019) propose a methodology in which driving habits-related risk factors are used as a correction to the premium calculated with traditional risk factors. Other studies, in conjunction with driving habits, investigate the predictive power of driving style-related data ((Wüthrich, 2017), (Gao *et al.*, 2018), (Gao *et al.*, 2019), (Gao et Wüthrich, 2019), (Huang et Meng, 2019), (Gao *et al.*, 2021), (So *et al.*, 2021)). (Guillen *et al.*, 2020) and (Guillen *et al.*, 2021) develop a pricing scheme in which near-miss events, situations in which an accident is "narrowly" averted, and which comprise harsh braking, harsh acceleration and smartphone usage events, are used to update a baseline premium on a weekly basis.

Table 2.14 of Appendix 2.13 gives an overview of the recent literature on UBI pricing. Of these studies, many use handcrafted telematics-based risk factors or, in other words, extract telematics features by means of human guesswork. Aside from total distance driven, some of the most popular handcrafted telematics features include the average distance per trip, the fraction of night and urban driving and the fraction of distance traveled above the speed limit (see, for instance, (Verbelen *et al.*, 2017), (Guillen *et al.*, 2019), (Ayuso *et al.*, 2016b), (Huang et Meng, 2019), (Ayuso *et al.*, 2019), (Ayuso *et al.*, 2014), (Ayuso *et al.*, 2016a), (Geyer *et al.*, 2020)). Although these handcrafted features often lead to a substantial improvement in claim risk modeling, some are arbitrary and subjective, which may favor or penalize some drivers. In addition, they may not be optimal for risk estimation. Take, for instance, the fraction of night driving. Where do you draw the line between night and day? If this line is at $00{:}00$, an insured who often drives between $11{:}30$ p.m. and $00{:}00$ will be favored, while another who often drives between $00{:}00$ and $00{:}30$ a.m. will be penalized, because night driving is often considered more dangerous than day driving. Regarding variables akin to the fraction of distance driven above the speed limit, it is not clear where to set the threshold. Is it really dangerous to drive a little faster than the speed limit? In this paper, we address this issue by proposing a human-guesswork-free procedure for feature extraction. Indeed, we believe that algorithms are better than humans at finding useful patterns in the data. Furthermore, most of the models developed in the studies of Table 2.14 are not robust to irrelevant of redundant features; most of them use algorithms that do not perform automatic feature selection or regularization and that do not handle collinearity well. On our end, we use a logistic regression model regularized by an *elastic-net* penalty to perform the classification task, which allows for automatic feature selection and for a proper handling of collinear features.

The core of this paper consists in the development of an automatic procedure allowing the extraction of telematics features from driving-habits-related data. As it has been stated, this procedure has the benefit of not requiring human guesswork and is therefore more objective and fair. For this purpose, both a **routine** and a **peculiarity profile** are first derived for each vehicle by means of anomaly detection algorithms. Such algorithms allow the computation of an anomaly score, which measures the level of "outlierness" for each observation in a dataset. These algorithms are used to derive both a routine and a peculiarity score for each vehicle trip. For this, anomaly detection algorithms are applied using two different approaches, which are referred to as the *local* and *global schemes*. Scores calculated according to the former scheme correspond to the *routine scores*, while those calculated with the latter scheme are referred to as the *peculiarity scores*. In the local scheme, algorithms are applied to each vehicle individually and in silo, which means the anomaly score of a given trip made by a vehicle x is defined only with respect to all the other trips made by vehicle x. Each driver thus defines their own normality against which their trips are compared. The routine-scores vector for a vehicle forms what we call its routine profile. In the global scheme, algorithms are applied to all trips by all vehicles at once, meaning that the anomaly score of a given trip made by vehicle x is defined with respect to information on trips made by vehicle x and all other vehicles. The peculiarity scores vector for a vehicle capture where its driving habits lie in relation to the other vehicles and is referred to as its peculiarity profile. We believe that the insureds' routine and peculiarity profiles can help differentiate between claimants and non-claimants. Our hypothesis is that more routine drivers, who know more about the trips they make, are less prone to claim than less routine ones. Indeed, an insured who always makes the same trips knows their way and how to cope with sensitive parts of it. On the other hand, we believe drivers who tend to make peculiar trips are more likely to claim than others because it is expected that peculiar trips, which are made, for instance, at unusual times of the day or speeds, are more dangerous. Features are subsequently extracted from these profiles, whereupon their predictive power is investigated through an elastic-net logistic regression model. In particular, we answer the question of how an insured's routine and peculiarity profile influences their likelihood of claiming, which has never been addressed before in the literature.

In addition to a traditional automobile insurance pricing dataset, we have at hand information on the driving habits of the vehicle's main driver in the format of trip summaries. Both datasets are described in Section 2.4. Before being fed to the anomaly detection algorithms, telematics data need to be preprocessed, which is done in Section 2.5, together with some data visualization. Section 2.6 follows, where the three anomaly detection algorithms used, namely Mahalanobis' method, the Local Outlier Factor and the Isolation Forest, are presented in detail. The claim classification model we use, which is a logistic regression model with elastic-net regularization, is then described in Section 2.7. We also detail all the preprocessing steps required for the input features. Both the anomaly detection algorithms and the classification model have hyperparameters that need to be calibrated, which is done in Section 2.8. In Section 2.9, models using features extracted from the routine and peculiarity profiles are scored on the testing set, whereupon their performance is compared to a baseline model. We conclude in Section 2.10.

2.4 Data Description

2.4.1 Telematics dataset

The data we have at hand is provided by a Canadian property and casualty insurer. We have access to a telematics dataset in which each entry matches a trip made with an insured vehicle. The recording of a trip, which is done using an OBD device, starts when the engine is turned on and ends when it is turned off. This type of recording is sometimes referred to as a *key-on key-off* event (see for instance (Verbelen *et al.*, 2017)). The dataset is comprised of 7,438,883 of these events reported by 4,834 vehicles, each of which is identifiable by its vehicle identification number (VIN). All of them were observed over a one-year period coinciding with a one-year insurance policy, starting somewhere between December 30^{th}, 2015 and January 1^{st}, 2019. Each trip is characterized by four quantities: the datetime of departure and arrival, the distance driven and the maximum speed reached. Along with the VIN and an identifier for trips, this translates into a six-column dataset, an extract of which is shown in Table 2.1.

2.4.2 Traditional dataset

We also have a 4,834-row dataset that provides us with information about the policy during the one-year observation period for each of the 4,834 vehicles in the telematics dataset. Each entry in this dataset depicts an insurance policy with ten risk factors that are traditionally leveraged in automobile insurance pricing, related to either the policyholder or the insured vehicle, all of which are detailed in Table 2.2. Most of these risk factors are commonly used in pricing models in the literature (see, for instance, (Guillen *et al.*, 2020), (Boucher *et al.*, 2017) and (Verbelen *et al.*, 2017)). This dataset, which we will refer to as the traditional dataset, also features the target variable that will later be used for claim classification, namely the indicator

VIN	Trip ID	Departure datetime	Arrival datetime	Distance	Maximum speed
A	1	2017-05-02 19:04:15	2017-05-02 19:24:24	25.0	104
A	2	2017-05-02 21:31:29	2017-05-02 21:31:29	6.4	66
⋮	⋮	⋮	⋮	⋮	⋮
A	2320	2018-04-30 21:17:22	2018-04-30 21:18:44	0.2	27
B	1	2017-03-26 11:46:07	2017-03-26 11:53:29	1.5	76
B	2	2017-03-26 15:18:23	2017-03-26 15:51:46	35.1	119
⋮	⋮	⋮	⋮	⋮	⋮
B	1485	2018-03-23 20:07:08	2018-03-23 20:20:30	10.1	92
C	1	2017-11-20 08:14:34	2017-11-20 08:40:21	9.7	78
⋮	⋮	⋮	⋮	⋮	⋮

Table 2.1: Extract from the telematics dataset. Dates are displayed in the yyyy-mm-dd format. The actual VINs have been hidden for privacy purposes.

of the occurrence of a claim during the coverage period. Finally, there is the VIN column, which we need to merge the two datasets. Our goal is to extend the traditional dataset with features extracted from the telematics dataset.

2.4.3 Training and testing sets

In order to properly assess the performance of the models on new instances, a portion of the data must be set aside. To this end, 70% of the VINs (that is, 3384 VINs) are used to build the training set, whereas the remaining 30% (that is, 1450 VINs) comprise the testing set. The training set will later be used to preprocess the data as well as to tune and train the models, while the testing set will only come into play at the very end of the modeling process to assess the generalization performance of the previously tuned and trained models.

2.5 Telematics data preprocessing and visualization

In Section 2.6, three popular anomaly detection (AD) algorithms are introduced, each of which is used to compute both a routine and a peculiarity score for each vehicle trip. Note that since the routine scores are

Variable name	Description	Type
vin	Unique vehicle identifier	ID
annual_distance	Annual distance declared by the insured	Numeric
commute_distance	Distance to the place of work declared by the insured	Numeric
conv_count_3_yrs_minor	Number of minor contraventions in the last three years	Numeric
gender	Gender of the insured	Categorical
marital_status	Marital status of the insured	Categorical
pmt_plan	Payment plan chosen by the insured	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 obtaining driver's license	Numeric
claim_ind	Indicator of the occurrence of a claim	Categorical

Table 2.2: Overview of the traditional dataset.

calculated in the local scheme, these are also referred to as *local anomaly scores*, while peculiarity scores, which are calculated through the global scheme, are also called *global anomaly scores*. Both local and global anomaly scores are calculated exclusively from information found in the telematics dataset of Table 2.1. However, we first preprocess the latter in order to extract useful information. Indeed, the raw telematics data we have are not in the ideal format to use as input to the AD algorithms. First of all, the departure datetime, which is stored as the time (in seconds) elapsed since an arbitrary point in time (January 1^{st}, 1970 in the R software), is not very meaningful for AD algorithms. It might be more useful to have information about the time of day and the day of the week the trip started. From the departure datetime, we thus create two new trip attributes, namely the time elapsed since midnight (in seconds) and the time elapsed since Monday midnight (in days), respectively referred to as the time-of-day and time-of-week trip attributes. These two newly created variables are cyclical in nature and need to be encoded accordingly. Let us consider the time-of-day attribute, which ranges from 0 to 86,400. The non-cyclical encoding of this attribute over five days as a function of the datetime is shown in Figure 2.1a. Looking at the latter, one can understand the concern with supplying non-properly encoded cyclical data into the algorithms: there are discontinuities in the graph at the end of each day (when time goes from $23{:}59{:}59$ to $00{:}00{:}00$). Indeed, $00{:}00{:}00$ is encoded as 0 whereas $23{:}59{:}59$ is encoded as 86,399. This create inconsistencies: without cyclical encoding, a trip starting at $23{:}59{:}59$ would be considered very “far” from another trip that starts at $00{:}00{:}00$ according to the AD algorithms. This is obviously an issue because the two trips are only one second apart. In order to address this, we use sine and cosine functions, which are cyclical, to encode both time-of-day and time-of-week attributes. To this end, we first normalize both of them so that a cycle (which is 86,400 seconds for the time-of-day attribute and seven days for the time-of-week attribute) is compressed between 0 and 2π, and then apply the sine/cosine function. Mathematically, the sine and cosine transformations of the time-of-day attribute are expressed with

$$\begin{aligned}\text{time\_of\_day}_{sin} &= \sin\left(\frac{2\pi}{86400}\times\text{time\_of\_day}\right),\\ \text{time\_of\_day}_{cos} &= \cos\left(\frac{2\pi}{86400}\times\text{time\_of\_day}\right).\end{aligned}$$

In a similar fashion, the time-of-week attribute is encoded with

$$\begin{aligned}\text{time\_of\_week}_{sin} &= \sin\left(\frac{2\pi}{7}\times\text{time\_of\_week}\right),\\ \text{time\_of\_week}_{cos} &= \cos\left(\frac{2\pi}{7}\times\text{time\_of\_week}\right).\end{aligned}$$

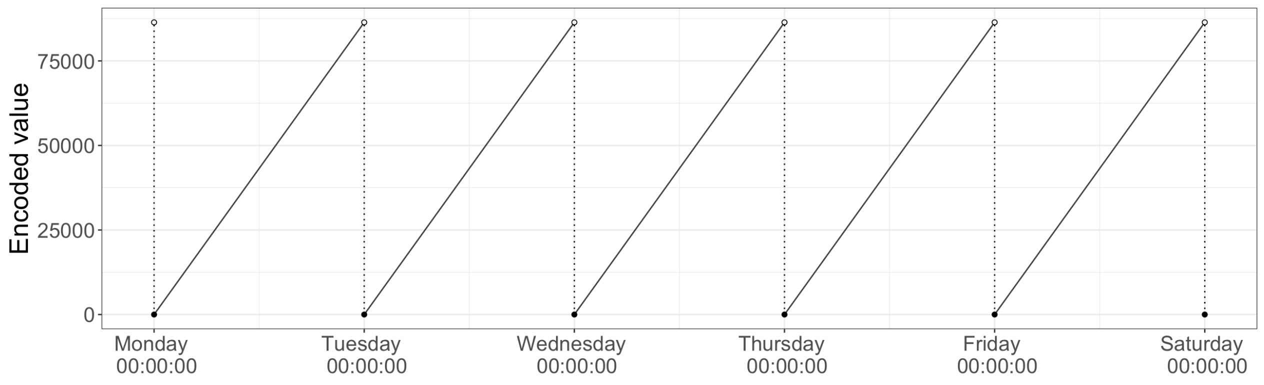
Sine and cosine encodings for the time-of-day attribute are shown in Figure 2.1b, where one sees that discontinuities have disappeared, which is desirable. It is worth noting that these cyclical attributes need to be encoded with two dimensions, that is, with both sine and cosine functions. Indeed, if they are encoded with only the sine (or cosine) transform, the encoding would not be unique. For instance, in the case of the time-of-day attribute, the sine transform for midnight (0 seconds since midnight) is the same as for noon (43,200 seconds since midnight), since

$$\sin\left(\frac{2\pi}{86400}\times 0\right) = \sin\left(\frac{2\pi}{86400}\times 43200\right) = 0.$$

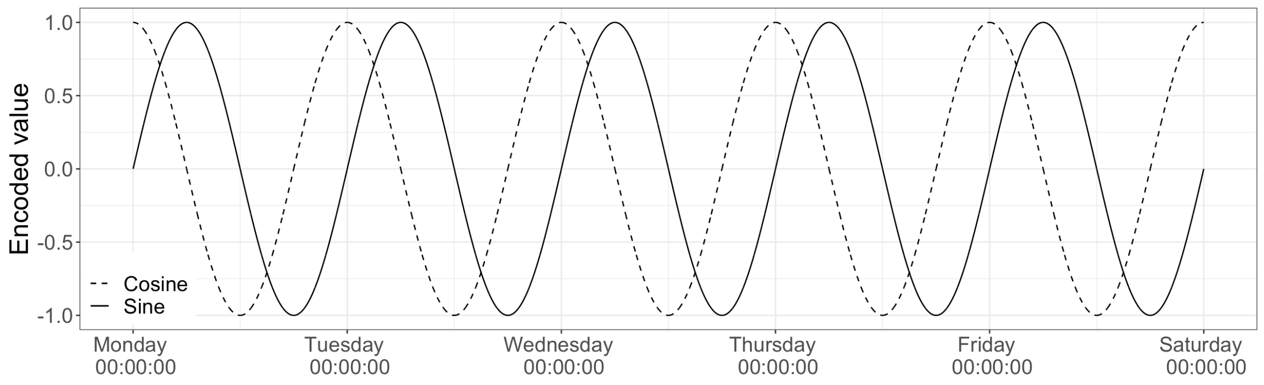
Secondly, two additional trip attributes are created from the four existing ones in Table 2.1, namely the average speed (in kilometers per hour) and the duration (in minutes) of the trip. Finally, because anomaly detection algorithms benefit from centered and scaled data, trip attributes undergo a z-score normalization. Consider the numerical vector $\boldsymbol{x} = (x_1, \ldots, x_n)$. The z-score normalized version of $\boldsymbol{x}$ is

$$\boldsymbol{x}^* = \left(\frac{x_1 - \overline{x}}{s}, \ldots, \frac{x_n - \overline{x}}{s}\right),$$

where $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$. In essence, AD algorithms are applied on the preprocessed version of the telematics dataset of Table 2.1, for which the eight attributes are described



(a) Non-cyclical encoding



(b) Sine and cosine encodings

Figure 2.1: Encoding of the time-of-day attribute (number of seconds elaspsed since midnight)

Trip attribute	**Description**
`duration`	Trip duration (in minutes)
`distance`	Trip distance (in kilometers)
`avg_speed`	Trip average speed (in kilometers per hour)
`max_speed`	Trip maximum speed (in kilometers per hour)
`time_of_day_sin`	Sine encoding of the number of seconds elapsed since midnight
`time_of_day_cos`	Cosine encoding of the number of seconds elapsed since midnight
`time_of_week_sin`	Sine encoding of the number of days elapsed since Monday midnight
`time_of_week_cos`	Cosine encoding of the number of days elapsed since Monday midnight

Table 2.3: Variables (or trip attributes) of the eight-dimensional dataset to which anomaly detection algorithms are applied. All eight variables undergo a z-score normalization.

in Table 2.3. Because the AD algorithms are applied to such a dataset, the anomaly degree of a trip is defined exclusively in terms of these attributes, which may be summarized as the driving habits. Therefore, AD algorithms seek to identify trips that exhibit an anomalous combination of duration, distance, average speed, maximum speed, time of day and time of week: the more anomalous the combination of attributes, the higher the anomaly score. In Figure 2.2, the distributions of the eight attributes of Table 2.3 over the whole telematics dataset are shown. In fact, only six distributions are plotted because for both the "time of day" and "time of week" attributes, the non-encoded version is shown to facilitate interpretation. As can be seen, trips of more than 50 km are fairly rare, as are night trips. The vast majority of trips are made at an average speed of less than 100 km/h and at a maximum speed of less than 150 km/h. Moreover, the insureds in our portfolio drive slightly less on Sunday than on any other day. It should be noted that all trips were made in an area where the maximum speed limit is 110 km/h.

2.6 Multivariate Anomaly Detection Algorithms

In this section, three popular anomaly detection algorithms are presented, each of which is used to derive both a routine and a peculiarity profile for each vehicle or, equivalently, both a local and a global anomaly score for each vehicle trips. Because the vehicles' trips are not labeled as anomalies or non-anomalies, we find ourselves more specifically in the framework of *non-supervised* anomaly detection. Therefore, the three algorithms used are selected from non-supervised anomaly detection algorithms, which do not re-

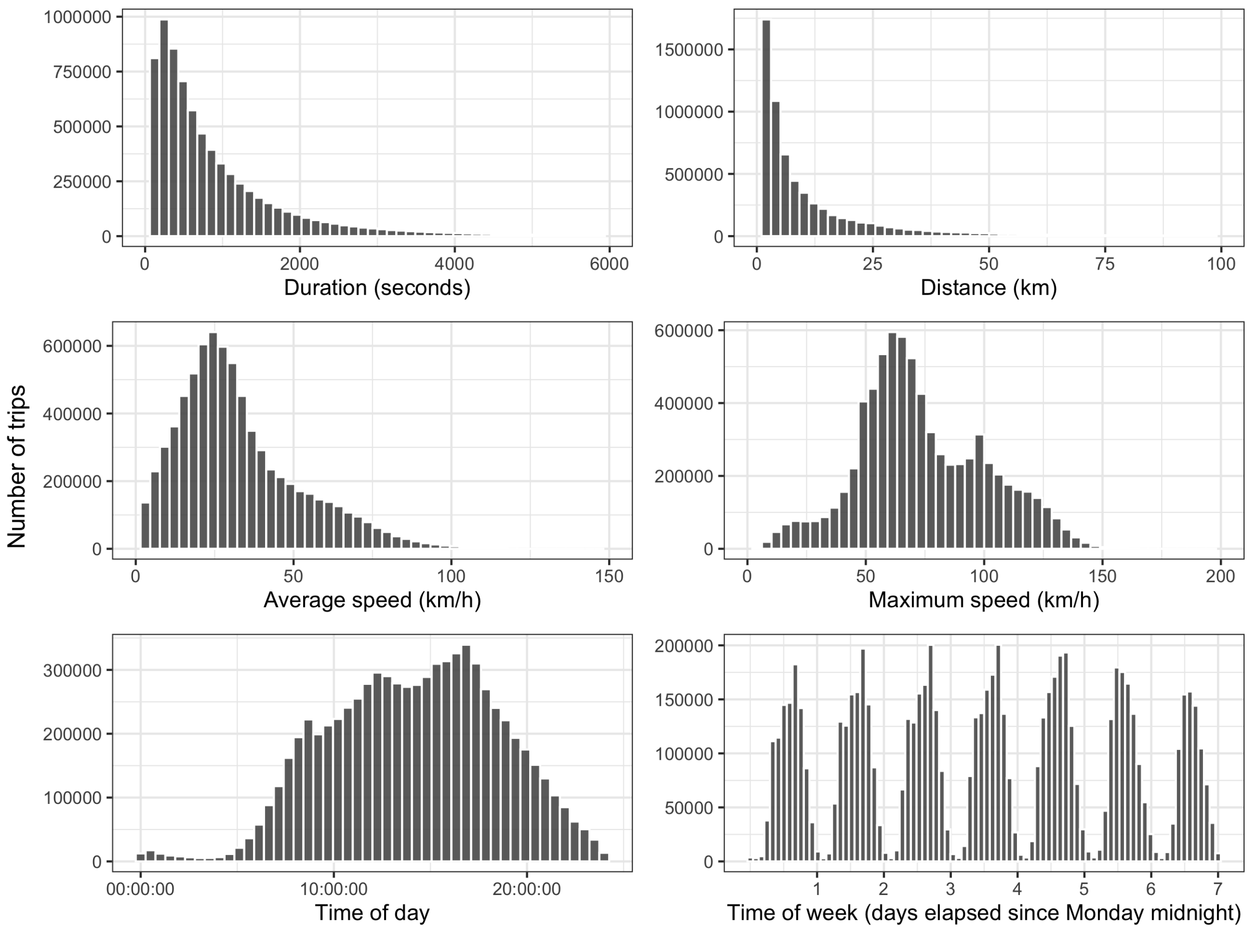


Figure 2.2: Distribution of the trip attributes.

quire a label to derive anomaly scores.

In the local scheme, the anomaly score of a trip, which we refer to as “local,” is computed by applying an AD algorithm to each vehicle separately, and thus reflects its anomaly degree with respect to all the other trips made by the vehicle in question. This means that a trip that seems very unlikely, such as a 200 kilometer trip made on Tuesday at 3 a.m., could be assigned a low local anomaly score, as long as the vehicle regularly takes this kind of trip. On the other hand, a trip that seems very common, for instance a ten kilometer trip on Monday at rush hour, could be assigned a high local anomaly score if the vehicle rarely makes this kind of trip. A trip with a high local anomaly score is one that bears little resemblance to trips typically made by the vehicle. The set of local anomaly scores for a vehicle’s trips forms a vector of real numbers, or empirical distribution, which we call the “routine profile.” A routine vehicle is one that does not have a wide variety of trips, while a non-routine vehicle has a diverse range of them. To illustrate, a cliché example of a routine vehicle would be one driven by someone who lives very simply and only uses their car to get food and essentials every Sunday at noon. A cliché example of a non-routine vehicle would be one driven by a pizza delivery person who has a changing schedule and is always making different kinds of trips at all times of the day and week. This being said, it is expected that a routine vehicle will have a small dispersion of local anomaly scores, while a non-routine one will have a large dispersion. To return to the illustration, the person who only uses their car to get food and essentials every Sunday at noon would then have a much less dispersed distribution of their scores than the pizza delivery person. This is why we refer to the local score vector as a “routine profile.”

In the global scheme, the anomaly score of a trip, which we refer to as “global,” is calculated by applying an AD algorithm to all vehicles in the telematics dataset at once. The global anomaly score of a trip thus reflects its anomaly degree with respect to all other trips in the dataset. In this case, a 200 kilometer trip made on Tuesday at 3 a.m. is likely to be assigned a high anomaly score because this kind of trip, using common sense and looking at Figure 2.2, is hardly ever made by the general population of vehicles. Conversely, a ten kilometer trip made on Monday at rush hour will probably be assigned a low global anomaly score. Along these lines, a trip with a high global anomaly score is one that bears little resemblance to trips typically made by the general population of vehicles. The set of global anomaly scores for a vehicle’s trips forms a vector of real numbers, or an empirical distribution, that we call the “peculiarity profile.” A somewhat peculiar vehicle is one that makes trips considered rather anomalous by the AD algorithm (i.e., it has high anomaly scores), while a somewhat non-peculiar vehicle makes trips considered rather ordinary (i.e., it has low anomaly scores). We believe that peculiar trips have a different risk than more "standard" trips. For instance, the AD algorithms could detect trips made at night, which have been shown to be more prone to cause a claim, long trips, where driver fatigue may be a risk factor, or trips made at a high speed, which can be risky. We therefore think that deriving a peculiarity profile for the vehicles can help better capture their risk.

Let us consider a dataset where n equally weighted observations (or instances) are described by p numeric variables. We denote by $\boldsymbol{X} = (x_{ij})_{n\times p}$ the data matrix, where x_{ij} is the value of the i^{th} observation for the j^{th} variable. We also denote by $\boldsymbol{x}_i = (x_{i1}, \dots, x_{ip}) \in \mathbb{R}^p$ the i^{th} observation (or row) of this matrix. Note that the observations $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ define a cloud of n points in $\mathbb{R}^p$. In the AD context, one is interested in determining an anomaly score, which is a real number that reflects the extent to which an observation is considered anomalous, for each observation.

2.6.1 Mahalanobis' method

A simple way of obtaining an anomaly score for each point $\boldsymbol{x}_i$ is to compute its distance from the centroid of the point cloud $\overline{\boldsymbol{x}} = 1/n \sum_{i=1}^{n} \boldsymbol{x}_i$. Indeed, the distance from the center of the distribution seems a natural way of measuring the anomaly level of an observation: the further an observation is from the center, the more it should be considered anomalous. For this purpose, standard Euclidean distance is not a suitable distance metric if the variables are on different scales. The Mahalanobis distance ((Mahalanobis, 1936)) is more advisable to use and is actually a fairly popular way of performing unsupervised AD. Mahalanobis' method computes the Mahalanobis distance between a point $\boldsymbol{x}$ and a point cloud with centroid $\overline{\boldsymbol{x}}$ and covariance matrix Σ, given by

$$d_M(\boldsymbol{x}) = \sqrt{(\boldsymbol{x} - \overline{\boldsymbol{x}})^\top \Sigma^{-1} (\boldsymbol{x} - \overline{\boldsymbol{x}})},$$

which can be directly used as an anomaly score.

2.6.2 Local Outlier Factor

The local outlier factor (LOF) algorithm ((Breunig *et al.*, 2000)) is based upon the concept of local density, where "local" is defined by the k nearest neighbors of a point. It is worth mentioning that k is a hyperparameter and thus, its value must be carefully chosen by the user. A point has a low (respectively high) local density if its neighborhood has a low (respectively high) concentration of points. To determine whether a point is an anomaly or not, one compares its local density with those of its neighbors. If the local density of the point is higher (respectively lower) than those of its neighbors, it will be given a low (respectively high) anomaly score.

To understand exactly how this method works, let us first introduce the concept of k-distance, which is simply the distance between a point and its k^{th} nearest neighbor. One typically uses the Euclidean distance metric on previously scaled data, which is equivalent to using the normalized Euclidean distance metric on non-scaled data. The k nearest neighbors of $\boldsymbol{x}$, whose set of indices is denoted with $\mathcal{N}_k(\boldsymbol{x})$, are the set of points that lie at a distance of k-distance$(\boldsymbol{x})$ or less from $\boldsymbol{x}$. Note that in case of a tie (i.e., if more than one point is at a distance of exactly k-distance$(\boldsymbol{x})$ from $\boldsymbol{x}$), the set $\mathcal{N}_k(\boldsymbol{x})$ actually contains more than k indices.

The *reachability distance* (RD) of a point $\boldsymbol{x}$ from another point $\boldsymbol{x}_i$ is defined as

$$\text{RD}_k(\boldsymbol{x} \text{ from } \boldsymbol{x}_i) = \max\left\{k\text{-distance}(\boldsymbol{x}_i), d(\boldsymbol{x}, \boldsymbol{x}_i)\right\},$$

where $d(\boldsymbol{x}, \boldsymbol{x}_i)$ is the distance between point $\boldsymbol{x}$ and point $\boldsymbol{x}_i$. It may perhaps be observed that “reachability distance” is a slight misuse of language because it is actually not a distance metric in the mathematical term because it lacks the symmetry property. In other words, the RD of $\boldsymbol{x}$ from $\boldsymbol{x}_i$ is the actual distance between the two points, but is at least the k-distance of $\boldsymbol{x}_i$. Therefore, if the two points are “sufficiently” close, the actual distance is replaced by the k-distance of $\boldsymbol{x}_i$. (Breunig *et al.*, 2000) state that using the RD instead of a real distance metric helps to achieve more stable results.

Using the RD, one can define the *local reachability density* (LRD) of a point $\boldsymbol{x}$, given by

$$\text{LRD}_k(\boldsymbol{x}) = \frac{1}{\operatorname{average}_{i \in \mathcal{N}_k(\boldsymbol{x})}\left\{\text{RD}_k(\boldsymbol{x} \text{ from } \boldsymbol{x}_i)\right\}}.$$

The LRD of a point is thus the inverse of the arithmetic average reachability distance from its neighbors. If a point is easily “reachable” by its neighbors (i.e., if the average reachability distance is low), it means that the neighbors will be relatively close to the point, and then, taking the inverse, the LRD will be relatively high. Conversely, if a point is not easily “reachable” by its neighbors, the LRD will be low. To sum it up, the LRD of a point measures the concentration of points in its neighboorhood.

Then, the LRD of a point $\boldsymbol{x}$ is compared to those of its neighbors, and its LOF anomaly score may be derived with

$$\text{LOF}_k(\boldsymbol{x}) = \frac{\operatorname{average}_{i \in \mathcal{N}_k(\boldsymbol{x})}\left\{\text{LRD}_k(\boldsymbol{x}_i)\right\}}{\text{LRD}_k(\boldsymbol{x})}.$$

The LOF score of a point is therefore the arithmetic average LRD of its neighbors divided by its own LRD. If point $\boldsymbol{x}$ has a low LRD relative to its neighbors, it means that its neighborhood is sparse compared to those of its neighbors, and that it is an outlier. In that case, its LOF will be greater than 1. Conversely, if point $\boldsymbol{x}$ has a high LRD relative to its neighbors, it is considered an inlier, and consequently, its LOF will be lower than 1.

2.6.3 Isolation Forest

The isolation forest (IF) algorithm, developed in (Liu *et al.*, 2008), detects anomalies by means of an ensemble of binary trees, called *isolation trees*, built upon the data matrix. What one calls a tree is in fact the representation of the recursive splitting (or partitioning) in two parts of the variable space $\mathbb{R}^p$. IF deviates from the AD mainstream, where normal observations are first modeled before anomalies can be identified as observations that do not conform to the modeled distribution. In contrast, IF directly detects anomalies using the concept of isolation. Indeed, anomalies are few in number and are different from other data points (which means they are found in sparse regions of the variable space), making them easier to isolate relative to normal data points.

In essence, an isolation tree works by recursively partitioning the variable space $\mathbb{R}^p$ into two parts until all observations are isolated. At each step, the splitting is done by randomly selecting a variable and a split value for that variable. Assuming all observations are distinct, the partitioning process thus ends when all observations are in their own leaf corresponding to a hyperrectangular region of $\mathbb{R}^p$ resulting from the partitioning. From then on, one can count the number of splitting steps required to isolate each observation i, denoted $h(\boldsymbol{x}_i)$, which can be interpreted as the path length within the tree to reach the leaf where observation i lies starting from the root node. The shorter the path length, the easier it is to isolate the observation, and the higher its degree of anomaly.

Because a single tree may yield unstable results, IF builds M isolation trees in order to create an ensemble (or forest), yielding M path lengths $h_1(\boldsymbol{x}_i), \dots, h_M(\boldsymbol{x}_i)$ for each observation i. One could compute the average path length for all M trees $\overline{h}(\boldsymbol{x}_i)$ for each observation i and use it directly as an anomaly score. However, the authors standardize the average path length to improve comparison and interpretation. For each observation i, they first divide $\overline{h}(\boldsymbol{x}_i)$ by the expected path length for any observation of the dataset, which depends only on the sample size n, given by

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}, \tag{2.1}$$

where $H(i)$ is the harmonic number, which can be estimated with $H(i) = \ln(i) + 0.5772156649$. Equation 2.1 originates from the field of binary search trees (BST). Indeed, isolation trees have a structure equivalent to BSTs, and thus the expected path length of an observation is the same as the expected length of an unsuccessful search in a BST. The ratio of $h(\boldsymbol{x}_i)$ over $c(n)$ is thereafter exponentiated in a way to obtain an anomaly score between 0 and 1. The anomaly score for observation i is hence given by

$$s(\boldsymbol{x}_i, n) = 2^{-\frac{\overline{h}(\boldsymbol{x}_i)}{c(n)}}. \tag{2.2}$$

The higher the score s, the more the observation is considered anomalous.

It is worth noting that trees that make up an IF are usually not built on the entire dataset. IF is in fact known to work well when the sample size is small. Consequently, one usually chooses the number of instances b to pick every time a tree is built.

2.7 Supervised Binary Classification Model

Every supervised learning task requires n labeled observations (or examples, or samples) gathered in a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip}) \in \mathcal{X}$ is the p-dimensional input (or feature) vector for the i^{th} observation and $y_i \in \mathcal{Y}$, the corresponding response (or label, target, output). The sets $\mathcal{X}$ and $\mathcal{Y}$ are often referred to as the feature space and the output space, respectively. Supervised learning theory assumes that observations from dataset $\mathcal{D}$ are realizations of the random vector $(\boldsymbol{X}, Y)$, so that a joint probability distribution $p_{\boldsymbol{X},Y}(\boldsymbol{x}, y) = p_{\boldsymbol{X}}(\boldsymbol{x})p_{Y|\boldsymbol{X}}(y|\boldsymbol{x})$ exists. The goal for a new observation is to predict its response y given its features $\boldsymbol{x}$ as accurately as possible. This is accomplished by training a supervised learning algorithm on the training dataset, which will "learn" a function that maps the features to the response as best as possible. Mathematically, a supervised learning algorithm seeks to find a member $h : \mathcal{X} \to \mathcal{A}$ of a predefined hypothesis function space $\mathcal{H}$ such that $h(\boldsymbol{x})$ is as close as possible to y for all observations, where $\mathcal{A}$ is the set of all possible predictions. In order to define "closeness" and thus properly choose h, a *loss function* $\ell : \mathcal{A} \times \mathcal{Y} \to \mathbb{R}^{+}$ is defined, where $\ell(y, h(\boldsymbol{x}))$ measures the distance between the prediction $h(\boldsymbol{x})$ and the actual response y. The goal is usually to minimize the empirical risk over the training set, given by

$$\widehat{\mathcal{R}}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y, h(\boldsymbol{x})).$$

From then on, supervised learning algorithms use optimization algorithms to find a function h that minimizes, globally or locally, the empirical risk function. We find ourselves specifically in the context of *super-*

vised binary classification when the response can take two distinct values (often encoded with 0 and 1), namely when $\mathcal{Y} = \{0, 1\}$.

2.7.1 Logistic Regression

Logistic regression is among the most popular algorithms for supervised binary classification problems. It assumes that the conditional distribution of the response given the features is a Bernoulli distribution, which is

$$p_{Y_i|\boldsymbol{X}_i}(y_i|\boldsymbol{x}_i) = \pi_i^{y_i}(1-\pi_i)^{1-y_i}, \quad y_i = 0, 1,$$

where $\pi_i := \mathbb{P}(Y_i = 1|\boldsymbol{X}_i = \boldsymbol{x}_i)$ is the conditional probability of being in class "1" for observation i. It further assumes that π_i can be expressed as the sigmoid transform $\sigma(\cdot)$ of the linear predictor $\boldsymbol{x}_i\boldsymbol{\beta}$, where $\boldsymbol{\beta} = (\beta_0, \dots, \beta_p)$ is a parameter vector:

$$\pi_i = \frac{1}{1 + \exp(-\boldsymbol{x}_i\boldsymbol{\beta})} := \sigma(\boldsymbol{x}_i\boldsymbol{\beta}).$$

Therefore, logistic regression seeks to find a function, which takes the form of the sigmoid transform of a linear transformation of the predictors, mapping the features $\boldsymbol{x}$ to a probability. The hypothesis space is thus $\mathcal{H} = \{\boldsymbol{x} \mapsto \sigma(\boldsymbol{x}\boldsymbol{\beta}) | \boldsymbol{\beta} \in \mathbb{R}^p\}$, and the set of possible predictions, $\mathcal{A} = [0, 1]$. From then on, a function $h \in \mathcal{H}$ must be chosen, which is equivalent to choosing (or estimating) the parameters $\boldsymbol{\beta}$. This is often done by minimizing the empirical risk with binary cross-entropy loss, given by

$$\widehat{\mathcal{R}}(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n} y_i \ln(\pi_i) + (1 - y_i)\ln(1 - \pi_i). \tag{2.3}$$

It is noteworthy that minimizing the empirical risk in Equation 2.3 is equivalent to estimating the parameters by maximum likelihood, because the right-hand side of the equation is just the Bernoulli log-likelihood times minus one. The estimated parameter vector that minimizes (2.3), denoted as $\widehat{\boldsymbol{\beta}}^{\text{MLE}}$, can be computed with a variety of numerical optimization methods, such as the method of iteratively reweighted least squares. Once estimated, the parameters can be used to obtain a prediction for a new observation $i = 0$:

$$\widehat{\pi}_0 = \sigma\left(\boldsymbol{x}_0\widehat{\boldsymbol{\beta}}^{\text{MLE}}\right). \tag{2.4}$$

2.7.2 Elastic-Net Regularization

Having low bias and low variance predictions are two desirable properties of a supervised learning model. Indeed, one can break down the reducible error of a prediction model as the sum of bias and variance.

From the well-known *bias-variance tradeoff* in machine learning, we know that one can decrease the bias of a model by increasing its variance, and vice-versa. It can be shown that the parameter vector's maximum likelihood estimator $\widehat{\boldsymbol{\beta}}^{\text{MLE}}$ of Subsection 2.7.1 is the asymptotically unbiased estimator of the true parameter vector $\boldsymbol{\beta}$ that has the smallest variance. However, it rarely is the best estimator for prediction because it still exhibits large variance, as do the resulting predictions. Regularization adds a regularization (or penalty) term to the empirical risk, which reduces variance at the cost of increasing bias. In many cases, the decrease in variance more than offsets the increase in bias, which improves prediction performance.

In the case of Ridge-regularized logistic regression, a penalty term that is proportional to the sum of squared coefficients is added to the empirical risk. The optimization problem hence becomes

$$\widehat{\boldsymbol{\beta}}^{\text{RIDGE}} = \operatorname*{argmin}_{\boldsymbol{\beta}} \left\{ \widehat{\mathcal{R}}(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}, \tag{2.5}$$

where $\lambda \geq 0$ is a hyperparameter that controls the amount of regularization. Examining Equation 2.5, one notices that the algorithm must not only minimize the empirical risk, but also the magnitude of the parameters, which has the effect of shrinking them toward zero. The resulting estimates (and thus the predictions) are biased because they are being pushed toward zero from the unbiased maximum likelihood estimates. In return, predictions have a lower variance because they result from smaller parameters (in absolute value). The larger the λ hyperparameter, the more biased the estimators will be and the less their variance will be. The aim is to find a good tradeoff between bias and variance by tuning the value of λ. For a given value of λ, the optimization problem of Equation 2.5 is convex and can therefore easily be solved using numerical optimization methods.

Although Ridge regression shrinks the coefficients towards zero, it never actually sets them to zero, yielding non-parsimonious models. Put another way, Ridge regression has no built-in feature selection mechanism, which is a flaw because sparser models are more easily interpreted by humans. Lasso regression, on the other hand, has the ability to set coefficients to zero due to the nature of its regularization term, and thus allows for automatic feature selection. Lasso works in a similar way as the Ridge regression except that the penalty term is proportional to the sum of the *absolute values* of the coefficients, and the optimization problem thus becomes

$$\widehat{\boldsymbol{\beta}}^{\text{LASSO}} = \operatorname*{argmin}_{\boldsymbol{\beta}} \left\{ \widehat{\mathcal{R}}(\boldsymbol{\beta}) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \tag{2.6}$$

As with Ridge regression, one wants to find a good tradeoff between bias and variance by tuning the λ hyperparameter. Once the value is set for λ, the convex optimization problem in (2.6) can be solved fairly easily using convex optimization theory.

Equations (2.5) and (2.6) are called the Lagrangian formulations of Ridge and lasso logistic regressions, respectively. In order to better understand the impact of both types of penalties, it is useful to represent them both with a constrained optimization problem, given by

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \widehat{\mathcal{R}}(\boldsymbol{\beta}) \right\} \quad \text{s.t} \quad \sum_{j=1}^{p} |\beta_j|^q \leq s, \tag{2.7}$$

where $q = 1$ and $q = 2$ correspond respectively to the lasso and Ridge cases, and where s is a hyperparameter with a one-to-one correspondence to the λ hyperparameter of the Lagrangian formulation. It is easier to remark with this constrained formulation that regularization gives a coefficient "budget" to the model. Indeed, the aim is to minimize the empirical risk while staying within the budget of s for the ℓ_q-norm (raised to the power of q) $||\boldsymbol{\beta}||_{\ell_q}^q = \sum_{j=1}^{p} |\beta_j|^q$ of the parameter vector. The constraint in Equation 2.7 actually defines a hypersolid (or constraint region) in $\mathbb{R}^p$ in which the estimated parameter vector must lie. In the special case of Ridge regression (i.e. $q = 2$), the constraint region is actually a hyperball, while for lasso (i.e. $q = 1$), it is a polytope. In the logistic regression framework with cross-entropy as a loss function, the empirical risk is convex and thus has the shape of a p-dimensional infinite bowl. Consequently, the solution to the optimization problem will always lie on the boundary of the constraint region, which means one looks for the point $\widehat{\boldsymbol{\beta}}$ where the bowl-shaped empirical risk function intersects with the constraint region. With this in mind, the shape of the hypersolid can help in understanding the behavior of regularized models. Indeed, the rationale for Ridge not performing automatic feature selection is that its constraint region has no sharp corners, being a hyperball. Therefore, the intersection point will almost surely not touch one of the axes which means that no coefficient will be set to zero. In contrast, lasso performs feature selection because a polytope has sharp corners. In two dimensions, namely if $\boldsymbol{\beta} = (\beta_1, \beta_2) \in \mathbb{R}^2$, the border of the constraint region is a circle in the Ridge case and a rhombus in the lasso case, and they are illustrated in Figure 2.3.

Lasso regression, however, is known to handle groups of features that are highly correlated with one another poorly due to the sharp corners of its constraint region. Indeed, for a group of strongly correlated features, lasso tends to choose only one of them in the group and does not care which one is selected. In contrast, Ridge will select all features and share the coefficient "budget" approximately equally among them, which

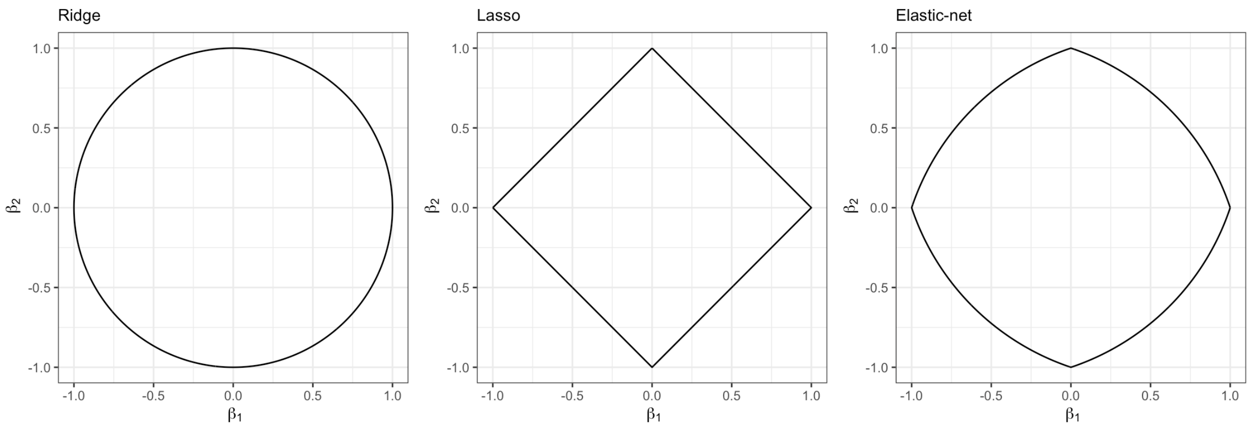


Figure 2.3: Boundary of the constraint regions in two dimensions (i.e. $\boldsymbol{\beta} = (\beta_1, \beta_2)$) for Ridge, lasso and elastic-net regressions, for $s = 1$. For the elastic-net regression, we set $\alpha = 0.5$.

is more desirable. Elastic-net regularization ((Zou et Hastie, 2003)) compromises between Ridge and lasso penalties, and solves the following optimization problem:

$$\widehat{\boldsymbol{\beta}}^{\text{E-N}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \widehat{\mathcal{R}}(\boldsymbol{\beta}) + \lambda \left[(1-\alpha) \sum_{j=1}^{p} \beta_j^2 + \alpha \sum_{j=1}^{p} |\beta_j| \right] \right\}, \tag{2.8}$$

where $\alpha \in [0, 1]$ is a "mixing" hyperparameter that controls the tradeoff between Ridge and lasso penalties. Alternatively, one can solve the following constrained optimization problem:

$$\widehat{\boldsymbol{\beta}}^{\text{E-N}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left\{ \widehat{\mathcal{R}}(\boldsymbol{\beta}) \right\} \quad \text{s.t} \quad (1-\alpha) \sum_{j=1}^{p} \beta_j^2 + \alpha \sum_{j=1}^{p} |\beta_j| \leq s, \tag{2.9}$$

where s is a hyperparameter analogous to the λ hyperparameter of Equation 2.8. In Figure 2.3, one notices that the elastic-net constraint region shares both Ridge and lasso properties, having both rounded edges and sharp corners. The former allows for feature selection, while the latter help to cope with highly correlated features by sharing the coefficient budget among them. Elastic-net regularization thus combines the best of both worlds. Moreover, it solves a convex program, which means the parameters can be estimated quite readily using convex optimization techniques.

In addition to yielding parsimonious and interpretable models as well as handling collinearity effectively, elastic-net regression has another major benefit. Indeed, elastic-net regression can be considered an "off-the-shelf" algorithm, meaning that it can be applied to data and quickly achieve good results. In point of fact, elastic-net regression requires very little data preprocessing and has very few hyperparameters (only two: α and λ), meaning they are easily tunable. In contrast, some neural networks and boosting algorithms can include dozens of hyperparameters, all of which must be carefully tuned. For further information about elastic-net regularization (or more generally, regularization), we refer to (Hastie *et al.*, 2015).

2.7.3 Preprocessing

2.7.3.1 Anomaly scores

First of all, let us recall that each of the AD algorithms presented in Section 2.6 is used to compute both a local and a global anomaly score for each vehicle trip in the telematics dataset. Once these scores are computed, the collection of which constitutes a vehicle's routine or peculiarity profile, the goal is to use them in conjunction with traditional risk factors and distance driven in an elastic-net logistic regression model to perform claim classification, as they are expected to convey important information about the occurrence of claims. Note that we use distance driven as a telematics feature since it is clearly related to the risk of claiming, as this is a measure of risk exposure. However, we do not arbitrarily handcraft additional telematics features since part of our research's goal is precisely to automate feature extraction. Obviously, not all vehicles completed the same number of trips during their one-year observation period, meaning their routine or peculiarity profile vector has a variable length. Consequently, the profile vectors cannot be entered directly as covariates in the elastic-net model, because the latter only accepts a rectangular design matrix as input. We therefore need a way to "translate" the profile vectors so that they are understood by the machine learning algorithm, i.e. the elastic-net model. In other words, it is required to extract features from these profile vectors. In this regard, we extract for each vehicle 11 evenly spread out quantiles of its profile vector, namely the $0^{\text{th}}, 10^{\text{th}}, \ldots, 90^{\text{th}}$ and 100^{th} percentiles. The 0^{th} and 100^{th} percentiles are also referred to as the minimum and the maximum of the profile vector, respectively. This way, each vehicle's telematics data is summarized with 11 real numbers, meaning each vehicle ends up with 11 extracted telematics features, which will later be used to enhance claim classification. One might ask why in particular we extract evenly spread out quantiles, or deciles. The reason for this choice is that we believe it is the least arbitrary and most neutral way to proceed. This way of proceeding also ensures that the machine learning model has the complete information on the distribution of the scores, and allows the model to find by itself the important information hidden in the profile vectors while reducing human intervention. This is desirable since the model has the benefit of receiving feedback from the response vector, as opposed to humans. In short, we think this is better than summarizing the score vectors with arbitrarily chosen statistics. The preprocessing of the anomaly scores is illustrated in Table 2.4.

VIN	Trip ID	Anomaly score
A	1	$\mathbb{R}$
A	2	$\mathbb{R}$
$\vdots$	$\vdots$	$\vdots$
A	2320	$\mathbb{R}$
B	1	$\mathbb{R}$
B	2	$\mathbb{R}$
$\vdots$	$\vdots$	$\vdots$
B	1485	$\mathbb{R}$
C	1	$\mathbb{R}$
$\vdots$	$\vdots$	$\vdots$

$$\xrightarrow[\text{percentiles}]{\text{extract}}$$

	Anomaly score's percentiles				
VIN	0^{th}	10^{th}	$\ldots$	90^{th}	100^{th}
A	$\mathbb{R}$	$\mathbb{R}$	$\ldots$	$\mathbb{R}$	$\mathbb{R}$
B	$\mathbb{R}$	$\mathbb{R}$	$\ldots$	$\mathbb{R}$	$\mathbb{R}$
C	$\mathbb{R}$	$\mathbb{R}$	$\ldots$	$\mathbb{R}$	$\mathbb{R}$

Table 2.4: Example of anomaly score preprocessing from the extract of Table 2.1.

2.7.3.2 Features

Although elastic-net logistic regression can be qualified as an off-the-shelf model, the features still need a few preprocessing steps before they can be entered into the classification algorithm, either for training or scoring. Features include the ten traditional risk factors from Table 2.2, the distance driven and the 11 telematics features extracted in Subsection 2.7.3.1, which makes for a total of 22 features. Each of them undergoes different preprocessing steps according to their nature. To carry out this preprocessing efficiently, the very handy `recipes` package from the `R` programming language is used. This package allows the definition of a preprocessing "recipe" (or workflow) consisting of multiple preprocessing steps to apply every time a model is either trained or scored. The `recipes` package can be used to center/scale features, impute missing values, numerically encode and group modalities for categorical features, etc. A key benefit of using this package is that a preprocessing recipe can be easily paired with a predictive model, thus avoiding data leakage. Indeed, it is not advisable to preprocess the whole dataset once at the beginning of the modeling pipeline because in such a case, information from the testing set may leak into the training set, which could lead to an overestimation of the model's performance. Similarly, information from the validation fold could leak into the training folds while cross-validating. In order to properly assess generalization power, the testing/validation set must indeed be completely disjoint from the training set. This is why it is best to make preprocessing an integral part of the modeling process by treating preprocessing and the prediction model as a single entity. That way, the preprocessing steps are performed every time a model is either trained or scored, and not using data that should be kept for testing/validation.

2.7.3.2.1 Categorical features

Just like many other supervised learning algorithms, elastic-net logistic regression is unable to deal with categorical features as is; they must first be numerically encoded. The categorical features used in the classification model include the four numerical traditional risk factors from Table 2.2, namely `gender`, `marital_status`, `pmt_plan` and `veh_use`. For this purpose, we choose a fairly standard way of transforming categorical features into numerical ones called "target encoding", which uses the target (or response) variable to assign a real number to each of the feature's categories. This is done by fitting a non-penalized logistic regression model without an intercept term on the response variable (which is the indicator of a claim in the observation period) using the feature to be encoded as the only covariate. This results in a coefficient for every category that is indicative of the claim risk for that category. These coefficients are then directly used as encoding values. This type of encoding is an alternative to the more common binary (or dummy) encoding, which creates several dummy variables from a categorical feature that indicates whether or not the observation belongs to each category. Conversely, target encoding has the benefit of not increasing the dimensionality of the dataset. Note that prior to target encoding, rare categories, namely those whose occurrence in the data is 5% of the observations or less, are pooled in an "other" category. This prevents from having categories with very few observations, which could lead to high prediction variability for these categories. Once they have been converted to numeric data, categorical features still need a few preprocessing steps related to numerical features, described in the next paragraph.

Example 2 (Target encoding) *Let us consider the small fictitious dataset in Table 2.5, with only one feature x and a binary response y. Suppose one wishes to target encode feature x, which is categorical with three categories: "blue", "white" and "red". Using non-penalized logistic regression without an intercept term, one first obtains the three coefficients by minimizing the empirical risk in Equation 2.3 or, equivalently, by maximizing likelihood: $\widehat{\beta}_{blue} = 0$, $\widehat{\beta}_{white} = -20.57$, $\widehat{\beta}_{red} = 20.57$. These estimated coefficients are then used to encode x.*

y	x	$x_{encoded}$
1	red	20.57
0	blue	0
1	red	20.57
1	blue	0
0	white	-20.57

Table 2.5: Example of target encoding with a non-penalized logistic regression on a fictitious dataset.

2.7.3.2.2 Numerical features

The numerical features used in the classification model include six of the features in Table 2.2 (`annual_distance`, `commute_distance`, `cov_count_3_yrs_minor`, `veh_age`, `years_claim_free` and `years_licensed`), distance driven, the four numerically encoded categorical features of Table 2.2 (`gender`, `marital_status`, `pmt_plan` and `veh_use`) and the 11 quantiles extracted from the routine and peculiarity profiles in Subsection 2.7.3.1. First, let us point out that the prediction function outputted by an elastic-net model is rather linear as the derived classification rule splits the hyperspace with hyperplanes. Therefore, in order to help break the linearity and better capture the vehicles' routine and peculiarity profiles, all $\binom{11}{2} = 55$ degree two interactions between the 11 anomaly scores-based features are first computed. All $6+4+11+55 = 76$ numerical features then undergo a Yeo-Johnson transformation followed by a z-score normalization. The Yeo-Johnson transformation reduces the skewness of the features' distributions and gets them closer to a normal distribution, which is usually beneficial for supervised learning algorithms. Mathematically, the Yeo-Johnson transformation ψ of $x \in \mathbb{R}$ is defined as

$$\psi(x, \lambda) = \begin{cases} ((x+1)^{\lambda} - 1)/\lambda & \text{if } \lambda \neq 0, x \geq 0 \\ \ln(x+1) & \text{if } \lambda = 0, x \geq 0 \\ -[(-x+1)^{2-\lambda} - 1]/(2-\lambda) & \text{if } \lambda \neq 2, x < 0 \\ -\ln(-x+1) & \text{if } \lambda = 2, x < 0, \end{cases} \tag{2.10}$$

where λ is a parameter that is usually optimized by maximum likelihood such that the empirical distribution of the transformed values is as close as possible to the normal distribution. Z-score normalization has the effect of centering and scaling the feature vectors. The centering allows us to omit the intercept parameter β_0 in the elastic-net model, while the scaling ensures that features all have equal importance in the mod-

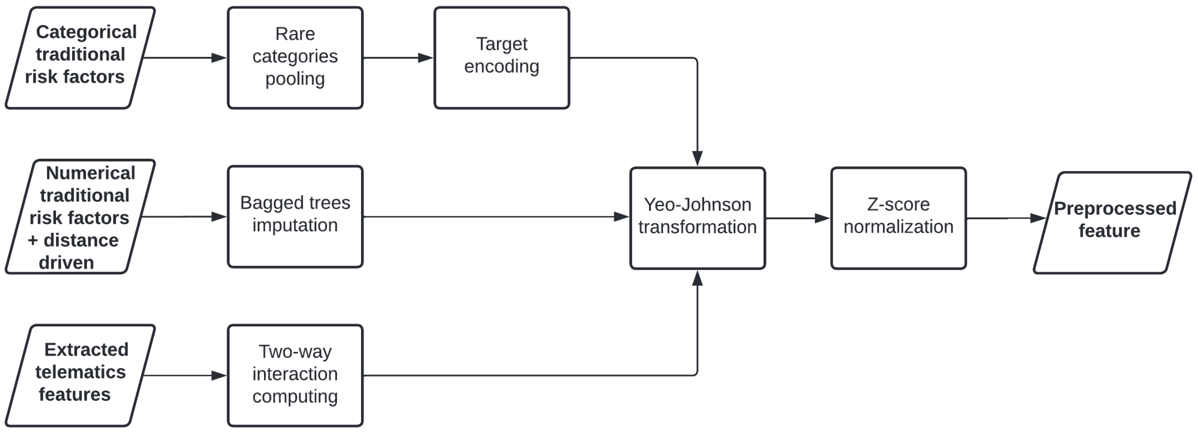

Categorical traditional risk factors
Rare categories pooling
Target encoding
Numerical traditional risk factors + distance driven
Bagged trees imputation
Yeo-Johnson transformation
Z-score normalization
Preprocessed feature
Extracted telematics features
Two-way interaction computing

Figure 2.4: Flowchart of the feature preprocessing workflow, which is applied every time a model is either trained or scored.

eling process. It is strongly recommended to scale features prior to feeding them into regularized logistic regression models because they are not scale invariant. Consider the feature vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ with empirical mean $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and standard deviation $s = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$. The z-score normalized version of $\boldsymbol{x}$ is

$$\boldsymbol{x}^* = \left(\frac{x_1 - \overline{x}}{s}, \ldots, \frac{x_n - \overline{x}}{s}\right). \tag{2.11}$$

Note that all features are complete except `commute_distance` from Table 2.2, which has a missing rate of 21.5%. We choose to impute these missing values with a rather popular imputation technique named “bagged trees imputation”, which first trains a committee of 25 regression trees on all instances having a non-missing value for `commute_distance` feature. It then substitutes missing values with the predictions produced by the tree committee. The complete data preprocessing workflow is illustrated in Figure 2.4.

2.8 Hyperparameter tuning

2.8.1 Anomaly Detection Techniques

Among the three AD algorithms presented in Section 2.6, LOF and IF have hyperparameters that need to be tuned. Indeed, with the LOF algorithm, one must choose the size of the neighborhood, defined by k, while in the case of IF, one must choose a value for the sampling size b. Our goal is to obtain the best possible claim classification performance and to this end, hyperparameter values are chosen in such a way to optimize the area under the receiver operating characteristic curve (AUC) when anomaly scores-based features are used alone in a non-penalized logistic regression model. Recall that anomaly scores are computed in two different ways, namely the local way, where AD is applied separately on each vehicle and the global way, where the latter is applied once on the whole portfolio. Therefore, four hyperparameters need optimization, namely k and b for both local and global schemes. Hyperparameter tuning is done using grid search, and AUC is obtained using five-fold cross-validation. It should be noted that only the training set is used to tune hyperparameters: the testing set is put aside until the very end of the modeling process, when we are ready for validation.

2.8.1.1 Local scheme

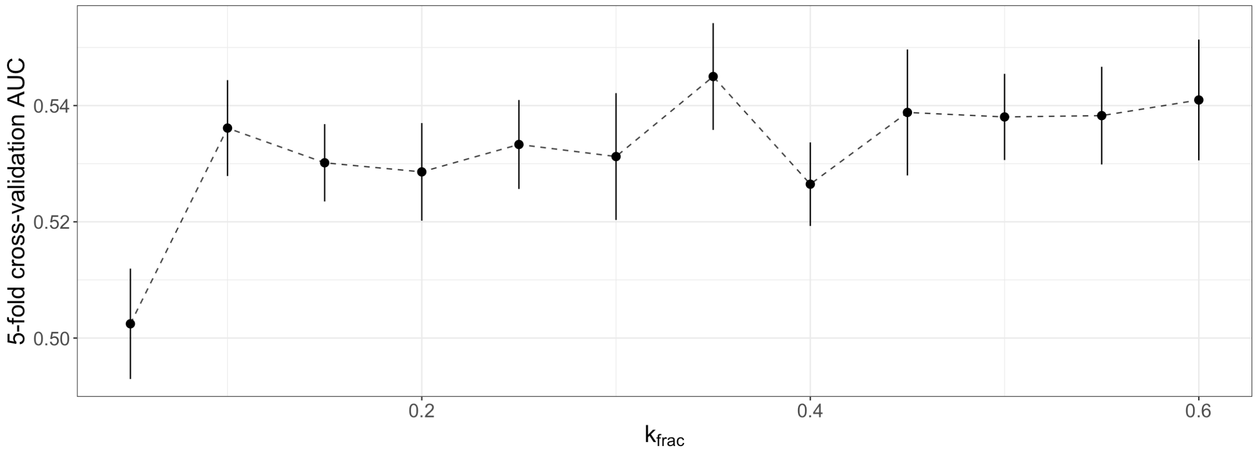
In the local scheme, the neighborhood size in the LOF algorithm and the sampling size in the IF algorithm are expressed as a fraction of the number of trips for each vehicle. Therefore, k_{frac} and b_{frac}, which represent respectively the fraction of trips used as the nearest neighbors in the LOF algorithm and the fraction of trips to draw every time an isolation tree is built in the IF algorithm, are tuned instead of k and b. The relation between k and k_{frac} as well as between b and b_{frac} are given by

$$k_i = k_{frac} \times n_i, \tag{2.12}$$

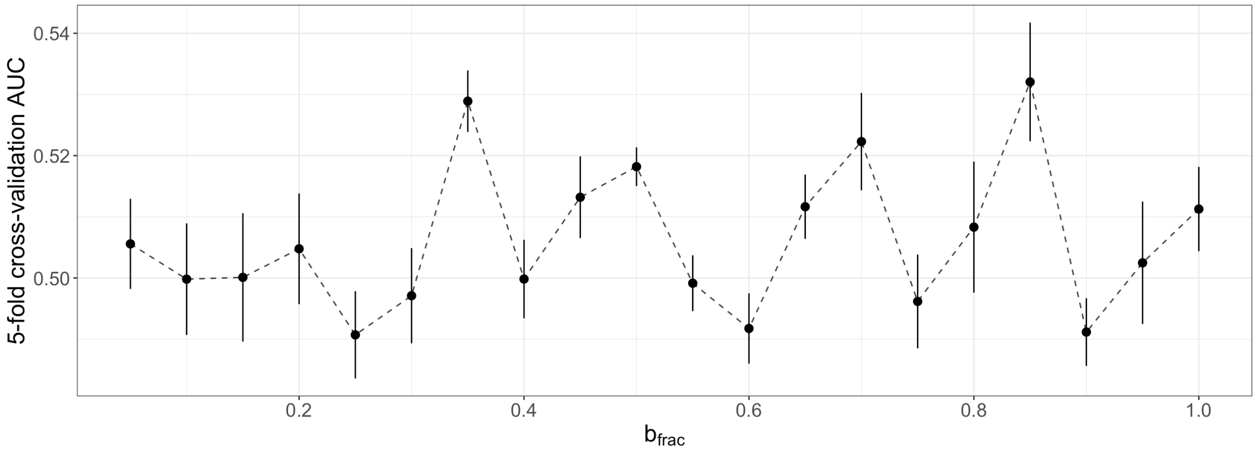
$$b_i = b_{frac} \times n_i, \tag{2.13}$$

where n_i is the number of trips made by the i^{th} vehicle. That way, algorithms are applied to each vehicle i using its corresponding value k_i (or b_i), which is proportional to its number of trips. The values tested for k_{frac} and b_{frac} go from 0.05 to 0.6 and from 0.05 to 1, respectively, in leaps of 0.05. Therefore, the grids used for the grid search are $\mathcal{G}_{k_{frac}} = \{0.05i\}_{i=1}^{12}$ and $\mathcal{G}_{b_{frac}} = \{0.05i\}_{i=1}^{20}$.

A cross-validation AUC value is obtained for each value in $\mathcal{G}_{k_{frac}}$ and $\mathcal{G}_{b_{frac}}$, whereupon the two chosen values (one for k_{frac} and one for b_{frac}) are those that maximize AUC. Note that the anomaly scores are preprocessed according to Table 2.4 and Figure 2.4. The tuning results for the local scheme are shown in Figure 2.5. It turns out that the optimal hyperparameter values that are found are $k_{frac} = 0.35$ for LOF and $b_{frac} = 0.85$ for IF, with respective AUCs of 0.5450 and 0.5320. These are the hyperparameter values that will be used from now on in the local scheme. It is worth noting that Mahalanobis' method, which has no hyperparameter, yields an AUC of 0.5200. The optimal hyperparameter for each local AD method and its corresponding AUC value are given in Table 2.6.



(a) Local LOF



(b) Local IF

Figure 2.5: Tuning results for local AD algorithms. The five-fold cross-validation AUC and its corresponding standard deviation (represented as the length of the vertical segment) is shown for each hyperparameter value.

2.8.1.2 Global scheme

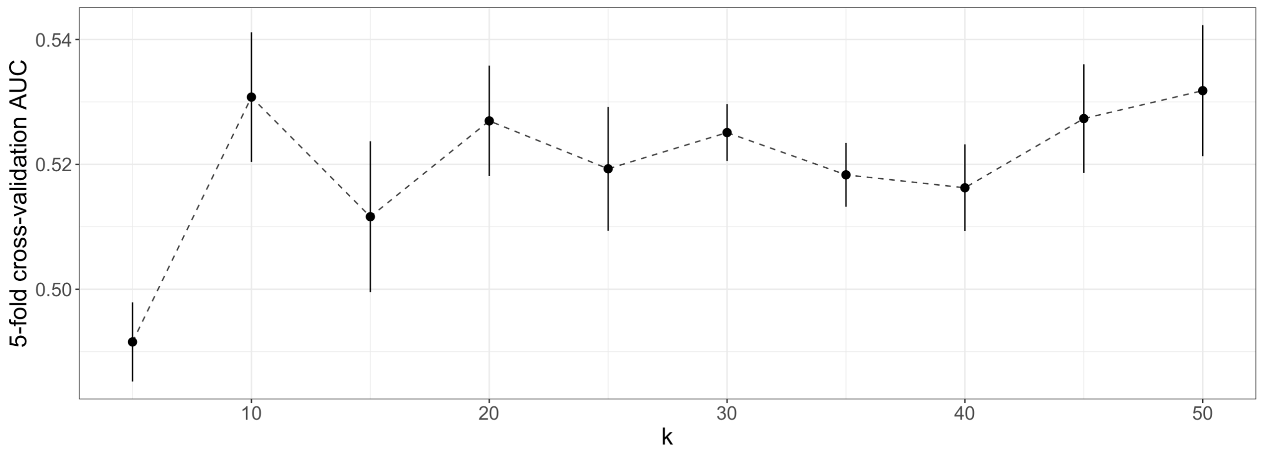
In the global scheme, hyperparameter tuning is done in a similar way as in the local scheme with the exception that the neighborhood size and the sampling size are not expressed as a fraction of the number of trips, but as an actual number of trips. The hyperparmeters k and b are thus tuned directly. Again, a grid search is used with $\mathcal{G}_k = \{5i\}_{i=1}^{10}$ and $\mathcal{G}_b = \{100i\}_{i=1}^{10}$ as tuning grids, and the chosen value for k (and for b) is the one matching the highest cross-validation AUC. The tuning results for the global scheme are shown in Figure 2.6. It turns out that the optimal hyperparameter values found are $k = 50$ for LOF and $b = 400$ for IF, with respective AUCs of 0.5318 and 0.5398. These are the hyperparameter values that will be used from now on in the global scheme. The optimal hyperparameter for each global AD method and its corresponding AUC value are given in Table 2.6.

	Hyperparameter value				
Anomaly detection algorithm	k_{frac}	b_{frac}	k	b	AUC
Local Mahalanobis	–	–	–	–	$0.5200^{(0.010079)}$
Local LOF	0.35	–	–	–	$0.5450^{(0.009192)}$
Local IF	–	0.85	–	–	$0.5320^{(0.009704)}$
Global Mahalanobis	–	–	–	–	$0.5481^{(0.006750)}$
Global LOF	–	–	50	–	$0.5318^{(0.010498)}$
Global IF	–	–	–	400	$0.5398^{(0.014215)}$

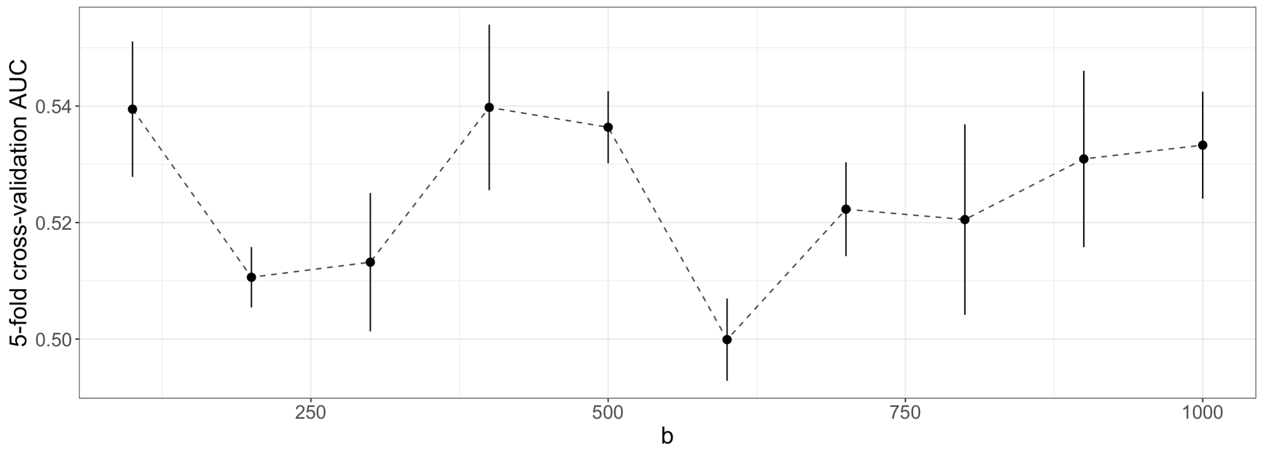
Table 2.6: Best hyperparameter value for each AD method and the resulting five-fold cross-validation AUC. Standard deviation is displayed as a superscript.

2.8.2 Elastic-Net Logistic Regression

Once LOF and IF are tuned for both local and global schemes, they, as well as Mahalanobis' method, are each employed to derive both a routine and a peculiarity profile for each vehicle, from which telematics features are extracted as quantiles according to Table 2.4. These telematics features, along with traditional risk factors (TRF) and distance driven, are then used together in an elastic-net logistic regression model. We have in total six different anomaly scores for each trip, six sets of telematics features are extracted. The classification performance of the six resulting models is compared to a baseline model that does not



(a) Global LOF



(b) Global IF

Figure 2.6: Tuning results for global AD algorithms. The five-fold cross-validation AUC and its corresponding standard deviation (represented as the length of the vertical segment) is shown for each hyperparameter value.

use AD at all, namely an elastic-net model that only uses the ten traditional risk factors and the distance driven as covariates. Remember that the elastic-net penalty has two hyperparameters, namely λ, which controls the severity of the penalty and α, which represents the fraction of the lasso-type penalty to be used. Tuning is done separately for each of the seven models using grid search as the hyperparameter space searching strategy and AUC as the performance metric. For the λ hyperparameter, a grid of 50 log-uniformly distributed values between 10^{-10} and 1, namely $\mathcal{G}_\lambda = \{10^{\frac{10}{49}i}\}_{i=0}^{49}$, is used. On the other hand, a coarse grid of five uniformly distributed values between 0 and 1 is considered for the mixing parameter α, namely $\mathcal{G}_\alpha = \{0, 0.25, 0.5, 0.75, 1\}$. For each of the seven models, cross-validation is employed on the training set to find the best combination of hyperparameters. For a given elastic-net model, five-fold cross-validation AUC is thus computed for each of the $|\mathcal{G}_\lambda| \times |\mathcal{G}_\alpha| = 250$ hyperparameter combinations. The values for λ and α that lead to the best cross-validation performance are thereafter chosen as the optimal pair. The optimal pair found for each of the seven models and the resulting AUC value are shown in Table 2.7.

	Optimal value		
Feature set used	λ	α	AUC
TRF + Distance driven (baseline)	2.33×10^{-2}	1	$0.6096^{(0.01300)}$
TRF + Distance driven + Local Mahalanobis	2.33×10^{-2}	1	$0.6095^{(0.01289)}$
TRF + Distance driven + Local LOF	2.22×10^{-3}	0.5	$0.6112^{(0.01392)}$
TRF + Distance driven + Local IF	2.33×10^{-2}	1	$0.6096^{(0.01299)}$
TRF + Distance driven + Global Mahalanobis	9.10×10^{-3}	1	$0.6149^{(0.01550)}$
TRF + Distance driven + Global LOF	3.56×10^{-3}	1	$0.6124^{(0.01410)}$
TRF + Distance driven + Global IF	9.10×10^{-3}	0	$0.6126^{(0.01498)}$

Table 2.7: Hyperparameter tuning results for the seven elastic-net models.

2.9 Analyses

2.9.1 Analysis of routine and peculiarity profiles

The AD algorithms have been tuned in Subsection 2.8.1 and are now ready to be applied to the processed telematics dataset in order to derive both a routine and a peculiarity profile for each vehicle, which we do. Let us first look at whether simple statistics calculated from these profiles can discriminate between claimants and non-claimants. In Table 2.8 is shown the mean value for 13 different statistics of routine and peculiarity profiles derived with Mahalanobis' method. The same results for the Local Outlier Factor and Isolation Forest algorithms are shown in Table 2.12 and Table 2.13 of Appendix 2.11, respectively. As can be seen, the mean and standard deviation of the profile vectors do not seem to be discriminating between the two groups. Also, we note that the peculiarity profiles are more successful in differentiating claimants from non-claimants than their routine counterparts, and mainly with the lower deciles of the distribution. This means that in order to estimate the claiming probability of a vehicle, one would look at how peculiar its least peculiar trips are. The peculiarity profiles derived with the tuned AD algorithms thus seem promising for creating useful telematics features for classification.

	Local Mahalanobis score (routine)			**Global Mahalanobis score (peculiarity)**		
	Mean value			**Mean value**		
Statistics	Claimants	Non-claimants	**p-value (t-test)**	Claimants	Non-claimants	**p-value (t-test)**
Mean	7.99	7.99	0.8588	8.34	9.33	0.6844
Standard Deviation	12.28	12.30	0.9423	40.17	78.40	0.6099
Minimum	2.58	2.55	0.0060*	2.93	2.92	$< 0.0001^*$
1st decile	3.71	3.69	0.0916	3.58	3.55	0.0006*
2nd decile	4.23	4.21	0.1367	3.98	3.94	0.0039*
3rd decile	4.73	4.71	0.1676	4.42	4.37	0.0287*
4th decile	5.27	5.26	0.1538	4.93	4.90	0.2124
5th decile	5.90	5.88	0.2072	5.55	5.52	0.4904
6th decile	6.65	6.64	0.7894	6.29	6.28	0.8658
7th decile	7.64	7.64	0.7350	7.22	7.24	0.6676
8th decile	9.13	9.15	0.2906	8.47	8.52	0.4780
9th decile	12.14	12.19	0.1457	10.62	10.67	0.6856
Maximum	286.57	293.76	0.3470	1009.76	2381.39	0.5556

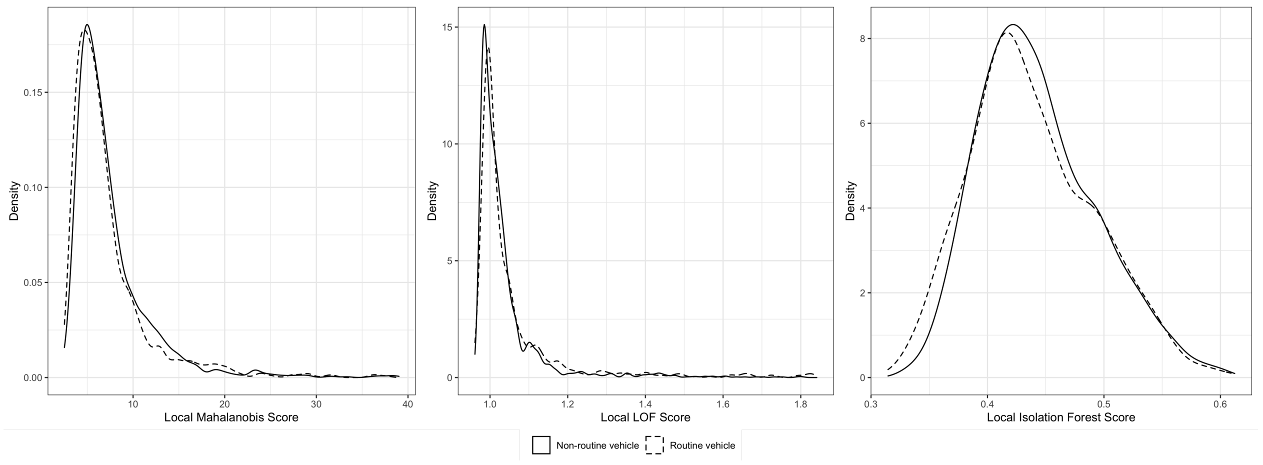
Table 2.8: Mean value of several statistics calculated from the profile vectors derived with Mahalanobis' method for claimants and non-claimants. Two-sample t-tests were conducted to determine whether the mean differs significantly between the two groups. A star indicates that the difference is significant at a 95% confidence level.

Ideally, we would like the derived profile vectors to help us differentiate between routine and non-routine vehicles, as well as between peculiar and non-peculiar ones. In other words, we would like the AD algorithms to have a similar concept of “routine” and “peculiarity” of a vehicle as a human observing the different trip attributes would have. By inspecting the distribution of the trip attributes for each vehicle separately, we thus select two vehicle pairs: a routine/non-routine one and a peculiar/non-peculiar one. The former is shown in Figure 2.9 of Appendix 2.12 and it is clear that the first vehicle (Figure 2.9a) is more routine than the second (Figure 2.9b) in terms of the six trip attributes. Indeed, the former tends to make trips of about 13 km concentrated around $11{:}00$ a.m. and $9{:}00$ p.m., while the latter makes trips of various distances at almost any time of the day. The attribute distributions of the selected peculiar/non-peculiar pair are presented in Figure 2.10 of Appendix 2.12, where the black density line corresponds to the global distribution of the attributes, i.e., the attributes’ distributions on the whole telematics dataset. We have deemed the first vehicle peculiar because the distribution of its attributes does not match the global distribution very well, unlike the second vehicle, which is deemed non-peculiar.

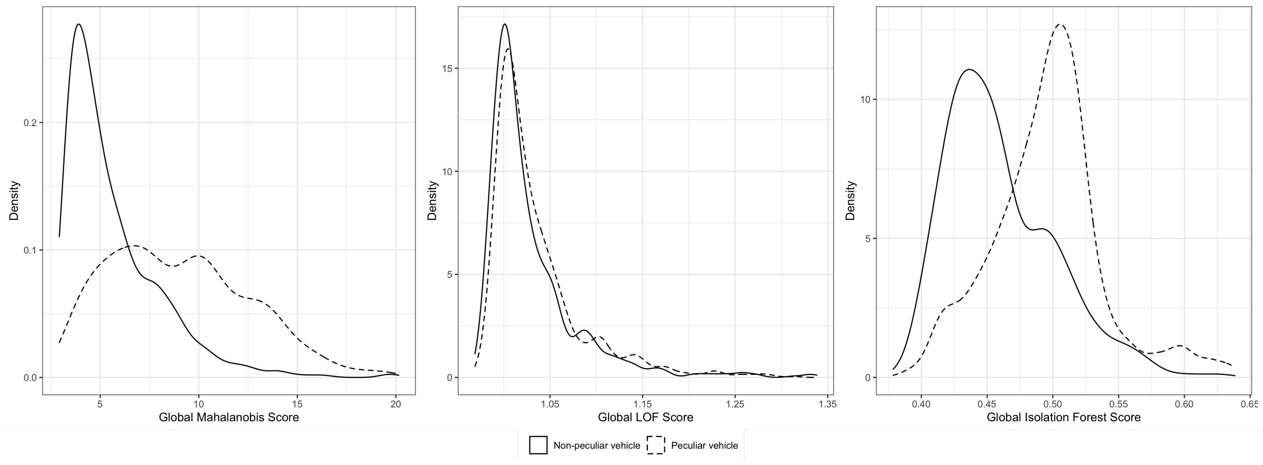
In Figure 2.7, we compare the density of the routine and peculiarity profiles derived using each of the three AD algorithms for both the routine/non-routine and the peculiar/non-peculiar vehicle pairs. Looking at Figure 2.7a, it is clear that AD algorithms applied locally fail to capture how routine a vehicle is. Indeed, the distribution of local anomaly scores for the selected routine vehicle coincides almost perfectly with that of the non-routine one, indicating that local anomaly scores computed with Mahalanobis’ method, LOF and Isolation Forest all fail to distinguish a routine vehicle from a non-routine one. Conversely, global Mahalanobis and Isolation Forest scores seem to capture the peculiarity profile of vehicles well, unlike global LOF scores. Indeed, one sees that the peculiar vehicle has a distribution of global scores much further to the right of the real numbers axis than the non-peculiar one, indicating more unusual trips. As a result, we expect the global anomaly scores from Mahalanobis’ method and the Isolation Forest to improve the ability to discriminate claimant from non-claimant vehicles, while local scores are expected to be of very little help.

2.9.2 Classification results

Elastic-net models, which were tuned in the previous section, are now ready to be trained on the full training set, which we do. They are then scored on the testing set, which has never been used before and is therefore totally unknown to the seven trained models. Classification performance is then measured with four



(a) Routine profile for the routine/non-routine vehicle pair.



(b) Peculiarity profile for the peculiar/non-peculiar vehicle pair.

Figure 2.7: Comparison of routine and peculiarity profiles for the routine/non-routine and the peculiar/non-peculiar vehicle pair.

metrics, namely AUC, accuracy, sensitivity and specificity. Accuracy is simply the number of correctly classified observations divided by the total number of observations. Sensitivity measures the ability of the model to correctly detect positive cases, i.e., claimants, while specificity measures the ability to correctly detect negative cases, i.e., non-claimants. The threshold used for the latter three metrics is 0.5, which means the hard prediction is 0 if the estimated probability of claiming is less than 0.5 and is 1 otherwise. In Table 2.9, the improvement of the six models that use anomaly detection-based features over the baseline model is displayed, as well as the baseline model's absolute performance. The latter achieves an AUC of 0.5948,

Anomaly detection algorithm	AUC	Accuracy	Sensitivity	Specificity
TRF + Distance driven (baseline)	0.5948	0.5634	0.5862	0.5407
TRF + Distance driven + Local Mahalanobis	0.0001	0	0	0
TRF + Distance driven + Local LOF	-0.0013	-0.0013	-0.0193	0.0165
TRF + Distance driven + Local IF	0	0	0	0
TRF + Distance driven + Global Mahalanobis	0.0184	0.0214	0.0110	0.0317
TRF + Distance driven + Global LOF	0.0006	0.0042	0.0028	0.0055
TRF + Distance driven + Global IF	0.0117	0.0063	-0.0372	0.0496

Table 2.9: Improvement of the six elastic-net models using AD over the baseline model and the baseline model's performance.

which is fine considering the small amount of data. In point of fact, the elastic-net models are tuned and trained on the training set, which has 3384 rows, and are scored on the testing set, which has 1450 rows. Moreover, AUC values are usually fairly low in claim classification due to the inherent randomness of the problem.

Clearly, features extracted from global anomaly scores, namely features extracted from the peculiarity profiles, are better candidates for classification than those extracted from local scores because they consistently yield superior results. In fact, the features extracted from local anomaly scores are deemed completely useless by the classification model because their inclusion does not improve (or even worsens) the performance metrics. This means that the routine profile we derived does not influence the risk of a claim. Conversely, global anomaly scores-based features nearly always improve classification metrics, regardless of whether Mahalanobis, LOF or IF is used to derive the anomaly scores. Global Mahalanobis performs best, with an

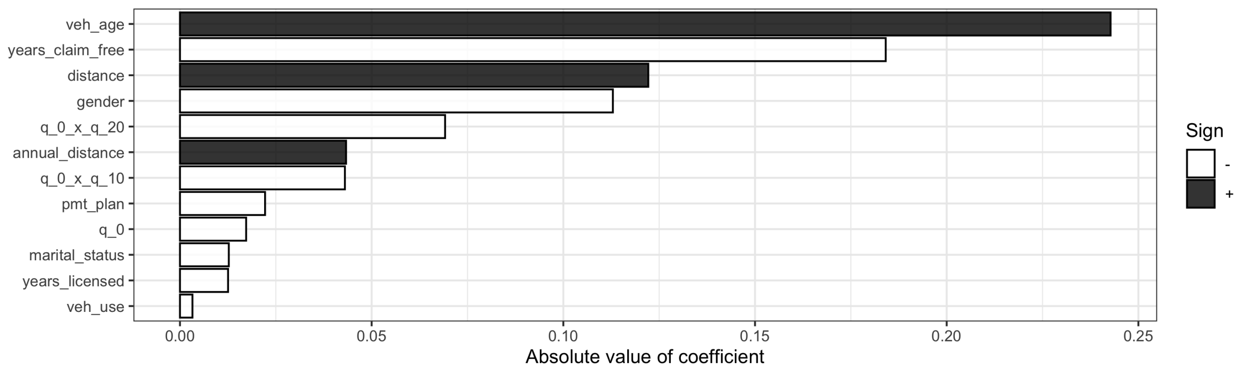


Figure 2.8: Elastic-net logistic regression non-zero coefficients obtained using features derived from global Mahalanobis anomay scores.

improvement in AUC, accuracy, sensitivity and specificity of 0.0184, 0.0214, 0.0110 and 0.0317 over the baseline model, respectively. This means that the peculiarity profiles we computed convey important information about the insureds' risks that traditional risk factors and distance driven (which is often considered one of the most predictive telematics features) do not capture. Although the improvement may seem modest, it has the potential to translate into thousands of dollars in additional profits for an insurer. Indeed, the improvement in classification suggests that with the newly extracted telematics features, insurers will be able to compute a more precise pure premium, which could help reduce adverse selection. LOF is the worst among global algorithms overall, showing only a slight improvement over the baseline model. We successfully extracted useful features from the global anomaly scores, indicating that the peculiarity profile of a vehicle impacts the probability of claiming. This is in line with the analysis performed in Subsection 2.9.1.

We shall now examine how the peculiarity profile of a vehicle influences the claim risk. To this end, we analyze the estimated coefficients of our best model, namely the one using features derived from global Mahalanobis scores. The sign and the absolute value of each estimated coefficient indicates how its related feature influences the probability of claiming. Furthermore, because features have been standardized prior to training the model, the absolute value of a coefficient can be interpreted as a measure of the importance of the underlying feature. In Figure 2.8, the sign and absolute value of every non-zero coefficient is displayed. Interestingly, out of the 77 features given as input to the model, only 12 are assigned a non-zero coefficient, which means the model deemed the remaining 65 features irrelevant for the classification task. The 12 features selected by the model include eight traditional risk factors (`veh_age`, `years_claim_free`, `gender`, `annual_distance`, `pmt_plan`, `marital_status`, `years_licensed` and `veh_use`), distance driven (`distance`) and three features extracted from global Mahalanobis scores (q0_x_q20, q0_x_q10 and q0). Interestingly enough, among the 11 extracted telematics features (q0, q10, ..., q90, q100) and their 55 two-fold interactions (q0_x_q10, q0_x_q20, ..., q80_x_q90, q90_x_q100), only features involving the 0^{th} percentile (i.e. the minimum of the anomaly score vector) and low percentiles are kept by the model, namely the 0^{th} percentile itself (q0), the interaction between the 0^{th} and 10^{th} percentiles (q0_x_q10) and the interaction between the 0^{th} and 20^{th} percentile (q0_x_q20). This indicates that lower global anomaly scores for a vehicle are predictive of the occurrence of a claim. Moreover, the coefficients of the abovementioned three anomaly score-based features are negative, which means, *ceteris paribus*, that the lower the percentile of the scores of a vehicle, the more likely it is to claim. This may seem counterintuitive because one might think that more anomalous trips must be more risky. However, when analyzing correlations between global Mahalanobis score and the eight trip attributes in Table 2.10, one notices that the anomaly score is more closely linked to duration, distance, avg_speed and max_speed than attributes related to time of day and time of week. This is because the four axes time_of_day_sin, time_of_day_cos, time_of_week_sin and

Trip attribute	**Spearman correlation**
duration	0.62
distance	0.79
avg_speed	0.35
max_speed	0.28
time_of_day_sin	0.09
time_of_day_cos	0.09
time_of_week_sin	-0.02
time_of_week_cos	0.02

Table 2.10: Spearman correlation of the eight trip attributes with the global Mahalanobis anomaly score.

time_of_week_cos have fewer anomalous values than the axes duration, distance, avg_speed and max_speed. Therefore, the anomalies are mostly determined using the first four attributes in Table 2.10. Given that the four attributes duration, distance, avg_speed and max_speed have a strong positive correlation with the anomaly score, this means that trips that are identified as anomalous (i.e., trips with a high anomaly score) are also trips with high values for the four abovementioned attributes. Naturally, trips with high values for these four attributes are most likely to have been taken on highways, which are usually safer than other types of roads. This would explain the negative coefficients associated with the global Mahalanobis scores-based features `q0`, `q0_x_q10` and `q0_x_q20`.

In order to economically assess the performance of a claim classification model, one can look at the pure premium it would charge to positive and negative instances in the testing set, where "pure premium" is understood as the claiming probability times the insured amount. In Table 2.11, we therefore computed the average pure premium for claimants and for non-claimants for both the baseline model and our best model using anomaly detection. We used an insured amount of 1,000 monetary units. As can be seen, both models charge more for claimants than for non-claimants, which is desirable. One also notices that our model using anomaly detection is able to better differentiate between the two at the premium level, charging 1.4 monetary units more for claimants and 1.4 monetary units less for non-claimants relative to the baseline model.

	Model	
	TRF + distance driven (baseline)	TRF + distance driven + global Mahalanobis
Claimants	83.6	85.0
Non-claimants	80.7	79.3

Table 2.11: Pure premium calculated with both the baseline model and the best model using anomaly detection, for claimants and for non-claimants. An insured amount of 1000 monetary units is assumed.

2.10 Conclusions

In this study, a novel method that allows the incorporation of telematics data into a claim classification model has been developed. The procedure, which utilizes anomaly detection algorithms, automatically extracts features from summarized information about trips made by insured vehicles. While most studies create telematics features by hand and subjectively, which may favor or disfavor certain insureds, the approach developed in this paper has the benefit of being objective and automatic. Our paper also addresses the lack of models doing automatic variable selection and regularization in the literature by incorporating elastic-net regularization into a logistic regression model. Although the method has been tested in the context of claim classification, it could easily be applied to the context of frequency or even severity modeling, because these are supervised learning problems as well. We found that anomaly scores computed globally can distinguish peculiar from non-peculiar vehicles and using a thorough machine learning methodology, we showed that these scores convey important information about the claiming risk of a vehicle. Indeed, features extracted from them allow the improvement of classification performance when added to the baseline model, especially when Mahalanobis' method is at play. We also found out that lower quantiles of the peculiarity profile vector are the most predictive of the claiming risk, and that these quantiles are negatively correlated with the probability of claiming. It seems that locally calculated anomaly scores, which is to say the routine profiles, do not help to improve classification, and this is probably due to the fact that such anomaly scores are unable to discern routine from non-routine vehicles.

In our method, each instance, or vehicle, is described with a distribution of local and global anomaly scores. From these distributions, quantiles are extracted and used as features in a penalized logistic regression model. While this type of model has a good performance/interpretability ratio, it is not the most appropriate for detecting complex interactions between features, which we think may be useful to properly characterize the anomaly scores' distribution and thus improve prediction. Such models also deal poorly with non-linear relationships between features and response. In fact, the current model only allows for linear links between features and log-odds, and uses the extracted telematics features as is, without trying to make them interact. For future research, we believe it would be wise to apply a neural network on the quantiles extracted from the routine and peculiarity profiles. Indeed, neural networks are known to automatically learn useful features from raw data. This way, instead of being arbitrary, the feature extraction process would be guided by the feedback of a loss function, which would probably enhance prediction. Alternatively, one could apply a neural network directly to the routine and peculiarity profile vectors. Furthermore, we still believe that the trips' degree of routine could possibly help to better classify the vehicles, but that the anomaly detection algorithms as implemented had a hard time capturing this degree of routine. Thus, for future research, one could try to find a way to better capture vehicles' routine profiles.

2.11 Appendix A: two-sample t-tests for LOF and IF algorithms

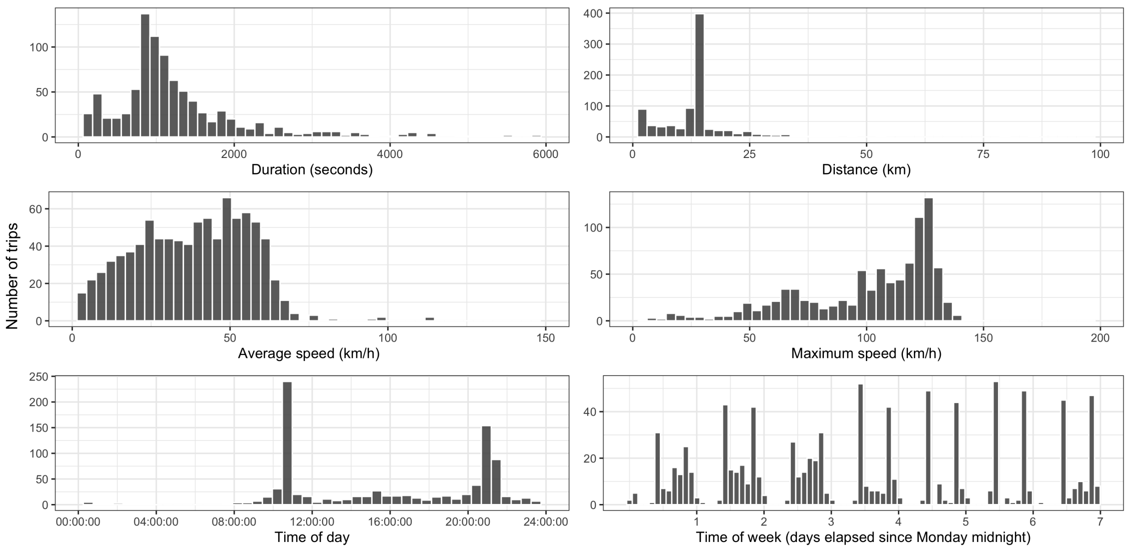
	Local LOF score (routine)			**Global LOF score (peculiarity)**		
	Mean value			**Mean value**		
Statistics	Claimants	Non-claimants	**p-value (t-test)**	Claimants	Non-claimants	**p-value (t-test)**
Mean	1.0601	1.0612	0.1454	1.0287	1.0292	0.4506
Standard Deviation	0.2167	0.2188	0.4537	0.0639	0.0647	0.9357
Minimum	0.9700	0.9694	0.0012*	0.9708	0.9709	0.2123
1st decile	0.9818	0.9816	0.0335*	0.9899	0.9901	0.0196*
2nd decile	0.9871	0.9870	0.0413*	0.9953	0.9957	0.0054*
3rd decile	0.9924	0.9923	0.4229	1.0002	1.0007	0.0060*
4th decile	0.9986	0.9986	0.7088	1.0054	1.0060	0.0093*
5th decile	1.0067	1.0070	0.1836	1.0116	1.0123	0.0225*
6th decile	1.0187	1.0191	0.3136	1.0194	1.0203	0.0352*
7th decile	1.0375	1.0378	0.5626	1.0304	1.0315	0.0761
8th decile	1.0703	1.0712	0.3871	1.0478	1.0490	0.1948
9th decile	1.1494	1.1540	0.0588	1.0831	1.0840	0.5396
Maximum	4.4216	4.5207	0.1062	1.8994	1.9434	0.8646

Table 2.12: Mean value of several statistics calculated from the profile vectors derived with LOF method for claimants and non-claimants. Two-sample t-tests were conducted to determine whether the mean differs significantly between the two groups. A star indicates that the difference is significant at a 95% confidence level.

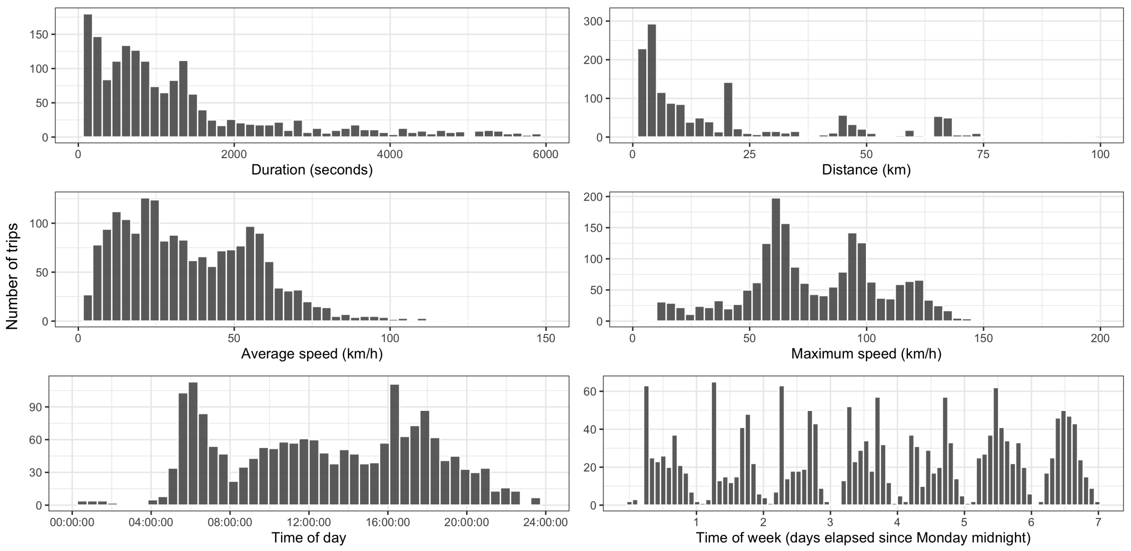
	Local IF score (routine)			Global IF score (peculiarity)		
	Mean value			Mean value		
Statistics	Claimants	Non-claimants	**p-value (t-test)**	Claimants	Non-claimants	**p-value (t-test)**
Mean	0.4423	0.4418	0.2391	0.4718	0.4717	0.8756
Standard Deviation	0.0593	0.0593	0.9131	0.0501	0.0504	0.3920
Minimum	0.3195	0.3183	0.1045	0.3841	0.3835	0.0001^*
1^{st} decile	0.3759	0.3754	0.3807	0.4173	0.4167	0.0447^*
2^{nd} decile	0.3934	0.3930	0.4578	0.4295	0.4287	0.0668
3^{rd} decile	0.4074	0.4069	0.3313	0.4403	0.4396	0.1543
4^{th} decile	0.4204	0.4199	0.2879	0.4513	0.4508	0.4168
5^{th} decile	0.4335	0.4329	0.2455	0.4630	0.4629	0.8404
6^{th} decile	0.4477	0.4471	0.2470	0.4761	0.4764	0.7312
7^{th} decile	0.4643	0.4638	0.2234	0.4911	0.4917	0.5090
8^{th} decile	0.4858	0.4853	0.2666	0.5094	0.5103	0.3887
9^{th} decile	0.5198	0.5194	0.3905	0.5364	0.5371	0.5594
Maximum	0.7415	0.7448	0.0625	0.7010	0.7018	0.6178

Table 2.13: Mean value of several statistics calculated from the profile vectors derived with Isolation Forest method for claimants and non-claimants. Two-sample t-tests were conducted to determine whether the mean differs significantly between the two groups. A star indicates that the difference is significant at a 95% confidence level.

2.12 Appendix B: routine/nonroutine and peculiar/nonpeculiar vahicle pairs

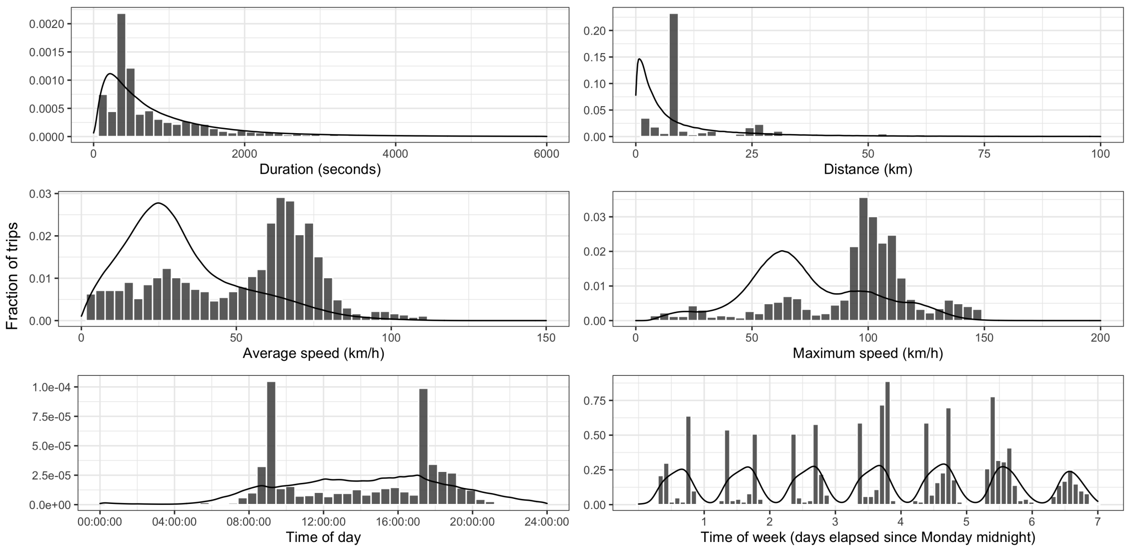


(a) Routine selected vehicle

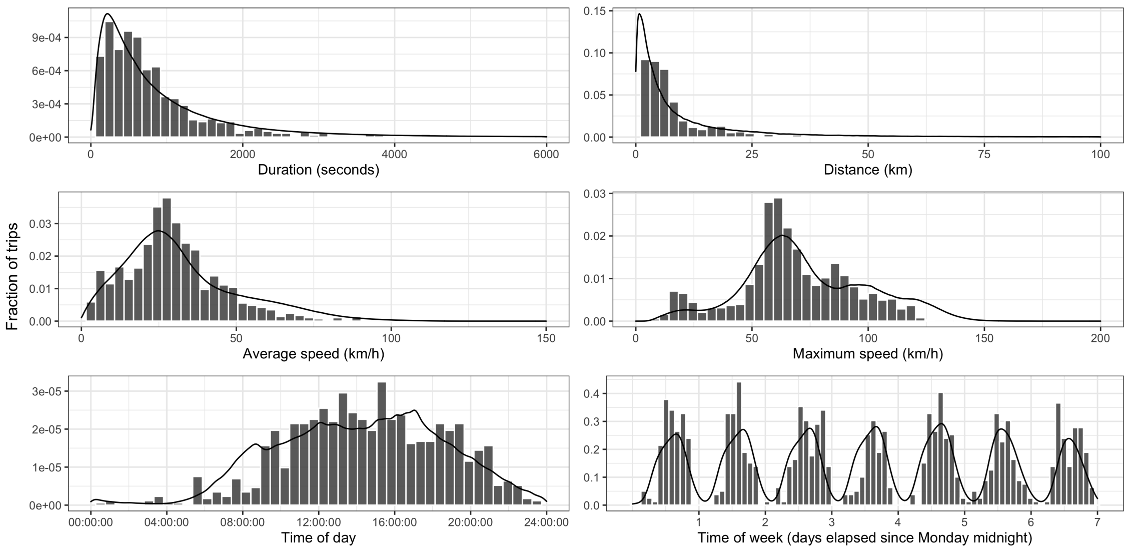


(b) Non-routine selected vehicle

Figure 2.9: Histograms of trip attributes for the routine/non-routine pair.



(a) Peculiar selected vehicle



(b) Non-peculiar selected vehicle

Figure 2.10: Histograms of trip attributes for the peculiar/non-peculiar pair. The black line shows the densities of the attributes for the whole telematics dataset.

2.13 Appendix C: literature review about claim prediction with telematics data

Reference	**Problem tackled**	**Telematics covariates**	**Modeling technique(s)**	**Some pros**	**Some cons**
(Paefgen *et al.*, 2013)	Claim classification	Mileage, road type, handcrafted features related to daytime, weekday and average velocity	GLM, neural network, decision tree	Flexible modeling	No feature selection or regularization, arbitrary choice of intervals for daytime, weekday and velocity variables
(Paefgen *et al.*, 2014)	Claim classification	Mileage, road type, handcrafted features related to daytime, weekday and velocity	GLM	Variable selection with a stepwise approach	Very few telematics feature engineering
(Ayuso *et al.*, 2016b)	Time to first claim modeling	Mileage, fraction of driving at night, over the speed limit and on urban roads	Weibull regression	Find that gender has little impact on claims experience once mileage is taken into account	Very few telematics feature engineering, no feature selection/regularization
(Boucher *et al.*, 2017)	Claim frequency modeling	Mileage	GAM	Flexible modeling	Use of mileage alone as telematics feature
(Verbelen *et al.*, 2017)	Claim frequency modeling	Mileage, number of trips, road type, handcrafted features related to daytime and weekday	GAM	Compositional features assessment, flexible modeling	Arbitrary choice of time slots for daytime and weekday variables
(Huang et Meng, 2019)	Claim classification, claim frequency modeling	Milage, handcrafted features related to travel habits, driving performance and critical incidents	GLM, SVM, random forest, XGBoost, neural network	Flexible modeling, many telematics features used	Very few telematics feature engineering, no feature selection/regularization
(Pesantez-Narvaez *et al.*, 2019)	Claim classification	Mileage, fraction of driving at night, over the speed limit and on urban roads	Logistic regression, XGBoost	Flexible modeling	No feature selection/regularization, very few telematics feature engineering
(Ayuso *et al.*, 2019)	Claim frequency modeling	Mileage, fraction of driving at night, over the speed limit and on urban roads	GLM	Use of telematics information as a correction of the premium calculated with classic covariates	No feature selection/regularization, very few telematics feature engineering
(Guillen *et al.*, 2019)	Claim frequency modeling	Fraction of driving at night, over the speed limit and on urban roads	Zero-inflated Poisson regression	Account for the excess of zeros	Very few telematics feature engineering, no feature selection/regularization
(Boucher et Turcotte, 2020)	Claim frequency modeling	Mileage	GAM and GAMLSS	Flexible modeling, longitudinal approach	Use of mileage alone as telematics feature
(Guillen *et al.*, 2020)	Near-miss frequency modeling	Mileage, urban and nighttime driving, speeding	GLM, GAM	Flexible modeling, more balanced dataset with near-misses	No feature selection or regularization
(Guillen *et al.*, 2021)	Claim frequency modeling	Mileage, near-misses (acceleration, braking and smartphone usage events)	GLM	Premium can be updated weekly	No feature selection/regularization
(Gao *et al.*, 2022)	Claim frequency modeling	Average driving time, velocity-acceleration heatmaps	GLM, feed-forward neural network, convolutional neural network	Automatic feature extraction with neural networks	No use of data related to driving habits

Table 2.14: Literature review about claim prediction with telematics data.

CHAPTER 3
TELEMATICS COMBINED ACTUARIAL NEURAL NETWORKS FOR CROSS-SECTIONAL AND LONGITUDINAL CLAIM COUNT DATA

3.1 Introduction

Nous présentons de nouveaux modèles transversaux et longitudinaux de comptage des réclamations pour l'assurance automobile, construits sur le cadre du *Combined Actuarial Neural Network* (CANN) proposé par (Wüthrich et Merz, 2019). L'approche CANN combine un modèle actuariel classique, tel qu'un modèle linéaire généralisé, avec un réseau neuronal. Cette fusion de modèles aboutit à un modèle à deux composantes comprenant un modèle de régression classique et une partie réseau neuronal. Le modèle CANN exploite les forces des deux composantes, fournissant une base solide et une bonne interprétabilité, tout en exploitant la flexibilité et la capacité à capturer les relations et interactions complexes offertes par le réseau neuronal. Dans nos modèles proposés, nous utilisons des modèles log-linéaires de régression de comptage des réclamations bien connus pour la partie de régression classique et un perceptron multicouche (MLP) pour la partie réseau neuronal. La partie MLP est utilisée pour traiter les données télématiques fournies sous forme de vecteur caractérisant les habitudes de conduite de chaque conducteur assuré. En plus des distributions de Poisson et binomiale négative pour les données transversales, nous proposons une procédure pour entraîner notre modèle CANN avec une spécification binomiale négative multivariée (MVNB). Ce faisant, nous introduisons un modèle longitudinal qui tient compte de la dépendance entre les contrats d'un même assuré. Nos résultats révèlent que les modèles CANN présentent des performances supérieures par rapport aux modèles log-linéaires qui reposent sur des facteurs de risque télématiques extraits manuellement.

3.2 Abstract

We present novel cross-sectional and longitudinal claim count models for vehicle insurance built upon the Combined Actuarial Neural Network (CANN) framework proposed by (Wüthrich et Merz, 2019). The CANN approach combines a classical actuarial model, such as a generalized linear model, with a neural network. This blending of models results in a two-component model comprising a classical regression model and a neural network part. The CANN model leverages the strengths of both components, providing a solid foundation and interpretability from the classical model while harnessing the flexibility and capacity to capture intricate relationships and interactions offered by the neural network. In our proposed models, we use well-known log-linear claim count regression models for the classical regression part and a multilayer perceptron (MLP) for the neural network part. The MLP part is used to process telematics car driving data given as a vector characterizing the driving behavior of each insured driver. In addition to the Poisson and negative binomial distributions for cross-sectional data, we propose a procedure for training our CANN model with a multivariate negative binomial (MVNB) specification. By doing so, we introduce a longitudinal model that accounts for the dependence between contracts from the same insured. Our results reveal that the CANN models exhibit superior performance compared to log-linear models that rely on manually engineered telematics features.

Keywords: Automobile insurance, Combined Actuarial Neural Network, Deep Learning, Claim count data, Multivariate negative binomial

3.3 Introduction and Motivations

Vehicle insurance products have traditionally been priced based on self-reported attributes provided by insureds. These attributes commonly include various risk factors, including gender, age, vehicle usage, and claim history. Insurers rely on this information to assess the level of risk associated with each insurance contract and determine appropriate premium rates. With the introduction of telematics technology, insurers can now collect a wide range of driving data through devices installed in the vehicles of policyholders or through mobile applications. This includes information such as vehicle speed, acceleration and braking behavior, mileage, location data, and factors like the time of day or types of roads frequently traveled. By leveraging this wealth of data, insurers can gain a more accurate and objective understanding of each individual's driving habits and style, enabling them to customize insurance offerings and pricing based on their actual driving behavior. This emerging paradigm, known as Usage-Based Insurance (UBI), revolutionizes the insurance landscape in various ways. For insurers, telematics data means more accurate risk assessment algorithms, which can often translate into a competitive advantage. For insureds, it means fairer premium rates that align more closely with their actual risk profiles rather than being computed based on broad demographic categories. It also means that they are priced based on risk indicators over which they have control. From a societal perspective, UBI also offers many advantages. One of the key benefits is the potential to improve road safety. By giving incentives for safe driving behavior and reduced mileage, UBI not only helps reduce the frequency and severity of accidents, ultimately saving lives and reducing the economic burden associated with road accidents, but also contributes to reducing greenhouse gas emissions. Additionally, telematics provide insurers with viable alternatives to sensitive risk factors, thereby helping to prevent unfair discrimination. For a more extensive overview of the benefits of UBI, we refer to the works of (Litman, 2007), (Bordoff et Noel, 2008) and (Ziakopoulos *et al.*, 2022).

One of the most prominent questions related to UBI is how to make the most out of the collected driving data. A significant subset of the literature has focused on incorporating mileage into pricing models due to its acknowledged importance as a risk factor in assessing risk and determining premium rates (see, for instance, (Boucher *et al.*, 2017), (Lemaire *et al.*, 2015), and (Turcotte et Boucher, 2023)). However, mileage alone fails to provide the whole story about an insured individual's driving behavior, prompting researchers to consider additional telematics information. One prevalent approach involves drawing upon domain knowledge to craft telematics features from raw data. By applying their expertise in the field, researchers can engineer features that capture critical aspects of driving behavior, specifically driving characteristics that are thought to be correlated with the risk of accident. Common examples of such features include harsh braking/acceleration events, cornering events, speeding, distracted driving, the fraction of driving during both different time slots (e.g., rush hour, late-night hours, weekdays), and on different road types (e.g., urban roads, highways), as well as the fraction of driving in different speed slots. While this approach captures signals missed by traditional risk factors and mileage (thereby improving pricing accuracy), it relies heavily on human judgment, with its inherent flaws and biases. With countless possible telematics features that can be engineered from raw telematics data, selecting the optimal ones for pricing is not straightforward. Furthermore, this process necessitates the setting of thresholds. For example, how should night driving or harsh braking be precisely defined?

The limitations of the aforementioned approach have motivated researchers to explore a new set of methods that rely more on data and decrease the need for human judgment. As highlighted in a recent study by (Embrechts et Wüthrich, 2022), the increasing amount of data available presents a challenge in manually designing features, leading actuaries to increasingly depend on tools like neural networks to learn and extract meaningful representations from the data. (Blier-Wong *et al.*, 2021) underline the importance of learning valuable representations from emerging data sources such as text, image, and sensor data. These sources, which include telematics car driving data, can enrich traditional data and offer improved insights for predicting future losses in insurance contracts. Neural networks are regarded as the most effective means for automatically extracting valuable features from raw data, which validates their practical application. In recent years, researchers have successfully applied the toolbox of deep learning, namely neural networks architectures with a large number of hidden layers, to handle telematics data and other types of unstructured data. In their work, (Wüthrich, 2017) introduce the speed-acceleration heatmap, a matrix representation that characterizes the driving style of an insured driver, which is well-suited for processing by deep learning algorithms. Subsequent studies ((Gao et Wüthrich, 2018), (Gao *et al.*, 2019), (Gao et Wüthrich, 2019), (Gao *et al.*, 2022)) have effectively leveraged these heatmaps by employing neural networks to learn representations from them. In (Meng *et al.*, 2022), the authors propose a supervised driving risk scoring convolutional neural network model that uses telematics car driving data to improve automobile insurance claims frequency prediction. (Blier-Wong *et al.*, 2020) propose a Convolutional Regional Autoencoder model for generating geographical risk encodings using convolutional neural networks. The resulting encodings, which aim to replace the traditional territory variable, proved beneficial for risk-related regression tasks.

In this paper, we present novel claim count models based on the Combined Actuarial Neural Network (CANN) approach, initially proposed by (Wüthrich et Merz, 2019). The CANN approach involves embedding a classical regression model, such as a generalized linear model (GLM; see (Nelder et Wedderburn, 1972) and (Dionne et Vanasse, 1989)), into a neural network, achieved by blending the regression functions of both models. Consequently, the resulting model comprises the two following components: the classical regression (or actuarial) model and the neural network. This blending process can be interpreted as a form of neural network boosting for the actuarial model, combining the strengths of both approaches. The calibration of the CANN neural network is performed using the classical actuarial model as the initial value in the gradient descent algorithm, with the negative log-likelihood of the specified distribution used as the loss function. One of the key benefits of this specific architecture is the solid foundation offered by the classical model, complemented by the network component's flexibility and pattern recognition capabilities. Neural networks excel in approximating highly nonlinear functions and possess the ability to compute valuable interactions between input variables automatically. Consequently, the CANN approach combines the best of both worlds, leveraging the interpretability and reliability of the classical model while capitalizing on the power of neural networks to capture complex relationships and patterns in the data. A few studies have successfully leveraged this approach: (Schelldorfer et Wuthrich, 2019) present a case study where a Poisson GLM for predicting claims frequencies is initially used, then enhanced through generalized additive models (GAMs) with natural cubic splines and finally combined with a neural network, resulting in a CANN approach. The study also explores the use of embedding layers for more efficient treatment of categorical variables; (Gabrielli *et al.*, 2020) boost an overdispersed Poisson model with a multilayer perceptron to improve individual loss reserving; (Tzougas et Kutzkov, 2023) use the CANN approach to enhance binary classification; (Laporta *et al.*, 2023) apply the CANN architecture in the context of quantile regression.

Our models employ a log-linear model for the actuarial model part and a multilayer perceptron (MLP) for the network part. Telematics information is incorporated into the MLP as a telematics vector, which is given as input to represent the driving behavior of each insured driver. The MLP part additionally includes traditional risk factors as inputs, enabling interactions between traditional and telematics inputs, while the log-linear part, constrained in estimating complex functions, is only given traditional risk factors. We explore three distinct distribution specifications for the claim count: Poisson and negative binomial for cross-sectional analysis, and multivariate negative binomial (MVNB; see (Hausman *et al.*, 1984) and (Boucher *et al.*, 2008)), also known as *negative multinomial*, for longitudinal analysis. The MVNB distribution is a popular choice for modeling longitudinal claim count data, as it captures the dependence between contracts from the same insured. However, to our knowledge, this specification has never been adapted to a neural network model for claim count regression. In this study, we extend the application of the MVNB distribution by incorporating it into the neural network framework, specifically the CANN architecture, for modeling longitudinal claim count data. This adaptation allows us to leverage the strengths of both the MVNB distribution and the neural network architecture. Our findings indicate that the CANN models perform better than their log-linear counterparts that rely on manually engineered telematics features. Furthermore, the CANN model using the MVNB specification exhibits a significant improvement compared to the two cross-sectional specifications.

In Section 3.4, we present the two datasets available to us: the contract dataset and the telematics dataset. Following that, in Section 3.5, we delve into the theory behind the CANN claim count models and also discuss the log-linear models that serve as benchmarks. Moving on to Section 3.6, we provide an explanation of how we apply the models on our specific dataset and show how we preprocess telematics data. In Section 3.7, we assess the performance of the models on a holdout sample and interpret the CANN models through permutation feature importance and partial dependence plots. Lastly, we conclude in Section 3.8.

3.4 Data

We have access to data from a Canadian property and casualty insurance company, which comes in two distinct datasets: the contract and the telematics dataset.

3.4.1 Contract dataset

In the contract dataset, each row represents a unique insurance contract. Contracts typically last for one year, but there are instances where their duration may be shorter or longer. Each vehicle is observed over one or more contracts; therefore, one vehicle can be represented by one or more rows in this dataset. Based on risk factors, a premium must be computed for each contract. When using a cross-sectional data model, contracts from the same vehicle are assumed to be independent of each other. On the other hand, a longitudinal data model assumes a dependence between contracts, allowing it to use information from previous contracts (including traditional risk factors, telematics data, past claims, etc.) to compute the premium. The contract dataset includes attributes commonly used in vehicle insurance pricing models. These traditional risk factors, displayed in Table 3.1, are recorded for 117,268 insurance contracts initiated between December 15th, 2015 and December 31st, 2018. In cases where multiple drivers are associated with

Variable name	**Description**	**Type**
`vin`	Unique vehicle identifier	ID
`annual_distance`	Annual distance declared by the insured	Numeric
`commute_distance`	Distance to the place of work declared by the insured	Numeric
`conv_count_3_yrs_minor`	Number of minor contraventions in the last three years	Numeric
`distance`	Real distance driven	Numeric
`expo`	Contract duration in years	Numeric
`gender`	Gender of the insured	Categorical
`marital_status`	Marital status of the insured	Categorical
`pmt_plan`	Payment plan chosen by the insured	Categorical
`veh_age`	Vehicle age	Numeric
`veh_use`	Use of the vehicle	Categorical
`years_licensed`	Number of years since obtaining driver's license	Numeric
`nb_claims`	Number of claims	Numeric

Table 3.1: Variables of the contract dataset.

a particular contract, attributes of the principal driver are used. Additionally, the dataset includes the vehicle identification number (VIN), allowing us to identify the insured vehicle accurately, alongside the reported claim count. As our goal is to perform claim count regression on contracts, the claim count variable will serve as the response for our supervised learning algorithms, namely the log-linear and the CANN models. The 117,268 contracts are associated with 49,671 distinct vehicles, resulting in an average of approximately 2,36 contracts per vehicle. The histogram of the number of contracts per vehicle is shown in Figure 3.1.

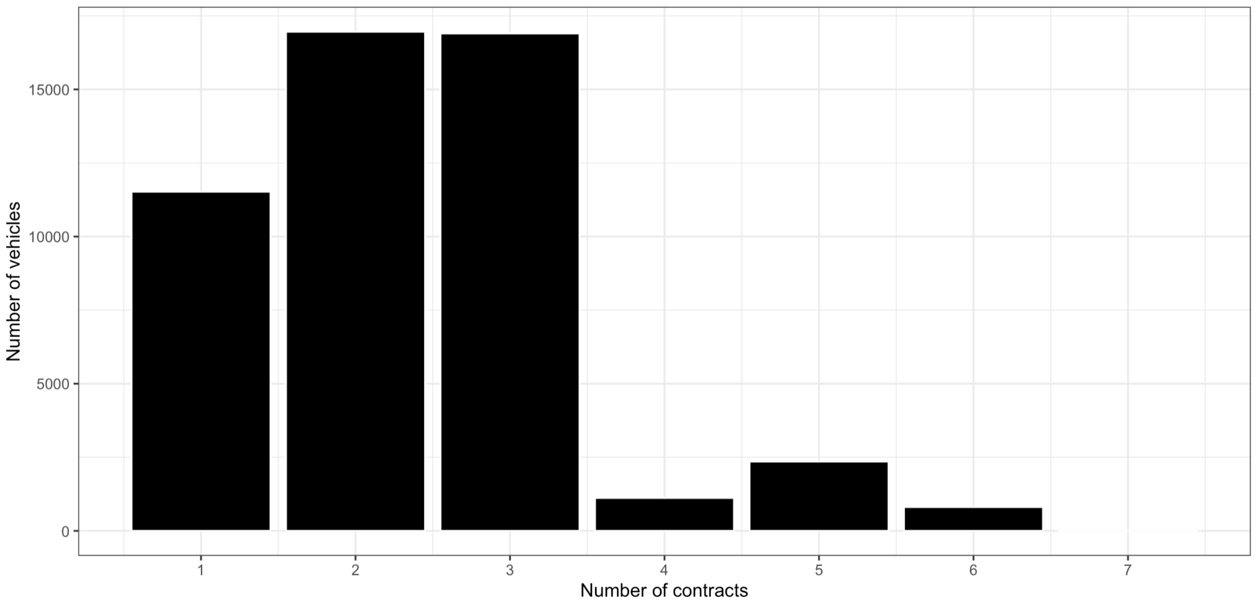


Figure 3.1: Number of contracts per vehicle.

3.4.2 Telematics dataset

All 117,268 contracts have been logged using an on-board diagnostics (OBD) device, capturing driving information. This data is stored as trip summaries in the telematics dataset, which comprises 117,566,259 trips. Each row in the dataset represents a specific trip, and every trip is described by 4 attributes: the departure and arrival date and time, the distance driven, and the maximum speed reached. Additionally, each trip is associated with a VIN, and with the date information, it is thus possible to link each trip with one of the 117,268 contracts. An extract from the telematics dataset is presented in Table 3.2.

3.4.3 Training, validation, and testing datasets

In supervised learning analysis, splitting the available data into training, validation, and testing sets is paramount for ensuring the reliability and ability to generalize of the learned model. The training set, which usually comprises the largest portion of the data, is used to train the model's parameters and optimize its performance. However, relying solely on the training set for performance assessment can lead to overfitting, par-

VIN	**Trip ID**	**Departure datetime**	**Arrival datetime**	**Distance**	**Maximum speed**
A	1	2017-05-02 19:04:15	2017-05-02 19:24:24	25.0	104
A	2	2017-05-02 21:31:29	2017-05-02 21:31:29	6.4	66
⋮	⋮	⋮	⋮	⋮	⋮
A	2320	2018-04-30 21:17:22	2018-04-30 21:18:44	0.2	27
B	1	2017-03-26 11:46:07	2017-03-26 11:53:29	1.5	76
B	2	2017-03-26 15:18:23	2017-03-26 15:51:46	35.1	119
⋮	⋮	⋮	⋮	⋮	⋮
B	1485	2018-03-23 20:07:08	2018-03-23 20:20:30	10.1	92
C	1	2017-11-20 08:14:34	2017-11-20 08:40:21	9.7	78
⋮	⋮	⋮	⋮	⋮	⋮

Table 3.2: Extract from the telematics dataset. Dates are displayed in the yyyy-mm-dd format. The actual VINs have been hidden for privacy purposes.

ticularly when the model has a high capacity. To address this, the validation set is used during the modeling process to assess the model's performance on unseen data. It plays an important role in tuning hyperparameters, selecting the optimal model architecture, and preventing overfitting. By assessing the model's performance on the validation set, one can obtain an estimate of its generalization performance and make necessary adjustments to improve its ability to generalize well to new, unseen examples. However, it is important to note that the back-and-forth process of evaluating the model on the validation set and adjusting its hyperparameters can introduce information leakage from the validation set into the training set. This can create an illusion of better performance than the model would exhibit in real-world scenarios. As a result, the testing set is reserved for the final evaluation of the learned model. It serves as an unbiased assessment of how well the model will perform on completely unseen data. This final evaluation provides an estimate of the model's true performance and helps determine its reliability in real-world scenarios. By keeping the testing set separate from the training and validation sets, we can ensure an unbiased evaluation and avoid any potential data leakage. We partition the data as outlined in Table 3.3 for our analysis. Approximately 60% of the vehicles are allocated for training, while approximately 20% is assigned to the validation and testing sets.

Set	Symbol	Number of vehicles	Number of contracts	Number of trips
Training	$\mathcal{T}_r$	30,000	70,451	71,416,560
Validation	$\mathcal{V}_a$	10,000	23,368	22,611,829
Testing	$\mathcal{T}_e$	9,671	23,449	23,537,870
Total	–	49,671	117,268	117,566,259

Table 3.3: Data partitioning

3.5 Count Regression Models

We consider a training dataset denoted as $\mathcal{T}_r$, which consists of $|\mathcal{T}_r|$ rows representing vehicle insurance contracts. Contracts are grouped by vehicle and each vehicle i is observed over T_i contracts. We define Y_{it} as a discrete random variable denoting the number of claims during the t^{th} contract of vehicle i. Furthermore, we have $\boldsymbol{x}_{it}$ a vector containing relevant predictor variables associated with the t^{th} contract of vehicle i. Importantly, we assume independence among all insured vehicles. In claim count regression, the ultimate goal is to estimate the probability mass function (PMF) of the number of claims, given all past and current information about the vehicle. Mathematically, we seek to estimate:

$$\mathbb{P}\left(Y_{it} = y_{it} | \boldsymbol{y}_{i,(1:t-1)}, \boldsymbol{x}_{i,(1:t)}\right), \quad y_{it} \in \mathbb{N}, \tag{3.1}$$

where $\boldsymbol{y}_{i,(1:t-1)} = (y_{i1}, \dots, y_{i,t-1})$ is the vector of past claims and $\boldsymbol{x}_{i,(1:t)} = \{\boldsymbol{x}_{i1}, \dots, \boldsymbol{x}_{it}\}$ is the set of past and current covariate vectors for vehicle i.

3.5.1 Cross-sectional models

In addition to assuming independence between vehicles, cross-sectional models also assume independence between contracts from the same vehicle. Consequently, these models do not use the history of a vehicle to estimate its future risk. The PMF of the number of claims can thus be written as:

$$\mathbb{P}\left(Y_{it} = y_{it} | \boldsymbol{y}_{i,(1:t-1)}, \boldsymbol{x}_{i,(1:t)}\right) = \mathbb{P}\left(Y_{it} = y_{it} | \boldsymbol{x}_{it}\right), \quad y_{it} \in \mathbb{N}. \tag{3.2}$$

3.5.1.1 Poisson regression

The Poisson distribution is widely used in supervised learning analysis for claim count data due to its good properties and simplicity. Under the Poisson specification, the PMF of the claim count for the t^{th} contract of vehicle i, denoted by Y_{it}, given its predictor vector, denoted by $\boldsymbol{x}_{it}$, is defined by

$$\mathbb{P}(Y_{it} = y_{it}|\boldsymbol{x}_{it}) = \frac{e^{-\mu(\boldsymbol{x}_{it})}\mu(\boldsymbol{x}_{it})^{y_{it}}}{y_{it}!}, \quad \text{for} \quad y_{it} \in \mathbb{N}, \tag{3.3}$$

with $\mathbb{E}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}] = \text{Var}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}] = \mu(\boldsymbol{x}_{it})$. The mean parameter $\mu(\boldsymbol{x}_{it})$ denotes the conditional expectation (and conditional variance) of Y_{it}. The regression function $\mu(\cdot)$ captures the relationship between the predictors $\boldsymbol{x}_{it}$ and the mean parameter in the Poisson distribution, indicating how the conditional expected count is influenced by the predictors. Subsequently, one must choose a specific functional form for $\mu(\cdot)$, which defines a hypothesis function space $\mathcal{H}$ that includes all the candidate functions for modeling $\mu(\cdot)$. The next step involves selecting the optimal function $\widehat{\mu} \in \mathcal{H}$, equivalent to estimating the parameters of the specified functional form based on the available data.

In order to define what constitutes a good regression function, it is necessary to select a suitable loss function that quantifies the dissimilarity between the estimated probability mass and the true label. The goal is to minimize this dissimilarity, improving the model's predictive performance. The cross-entropy loss, also known as the negative log-likelihood loss, is a commonly chosen option. For a specific observation i, the cross-entropy loss is given by $-\ln(p_i)$, where p_i is the estimated probability of observing the true label y_i. This loss function assigns a higher penalty to larger discrepancies between the true label and the predicted probability, incentivizing the model to converge towards more accurate predictions. To estimate the parameters, we typically aim to minimize the average loss function over the training set, also called the *empirical risk*. In the case of Poisson regression, this involves minimizing the average Poisson cross-entropy by solving the following optimization problem:

$$\widehat{\mu} = \underset{\mu \in \mathcal{H}}{\text{argmin}} \left\{ -\frac{1}{|\mathcal{T}_r|} \sum_{(i,t)\in\mathcal{T}_r} y_{it} \ln[\mu(\boldsymbol{x}_{it})] - \mu(\boldsymbol{x}_{it}) - y_{it}! \right\}. \tag{3.4}$$

Note that this is equivalent to maximizing the likelihood function. For some specifications of $\mu(\cdot)$, notably the log-linear specification, the criterion in Equation (3.4) is convex, which enables various convex optimization techniques to be applied. Alternative estimation techniques can also be used. One common option is regularization techniques, including lasso, Ridge, and elastic-net regressions. Instead of solely minimizing the average cross-entropy, these methods involve minimizing a modified objective function that includes a penalty term. Regularization is particularly beneficial for addressing common issues such as multicollinearity and overfitting.

3.5.1.1.1 Log-linear Poisson regression.

In the Poisson regression context, one notable specification for the regression function is the log-linear form, where the mean parameter is expressed as the exponential of a linear function of the predictors:

$$\mu^{\mathsf{LL}}(\boldsymbol{x};\boldsymbol{\beta}) = \exp\left\{\langle \boldsymbol{x}, \boldsymbol{\beta}\rangle\right\}, \tag{3.5}$$

where $\boldsymbol{\beta}$ denotes a vector of parameters, and $\langle \boldsymbol{x}, \boldsymbol{\beta}\rangle$ stands for the inner product between the predictor vector $\boldsymbol{x}$ and the coefficient vector $\boldsymbol{\beta}$. The use of the exponential function ensures that the mean parameter remains positive. Log-linear Poisson regression has favorable properties, notably its interpretability stemming from the quasi-linearity of the link function $\mu(\cdot)$. Moreover, when maximum likelihood is used for parameter estimation, this regression model falls within the framework of generalized linear models. GLMs provide valuable properties, such as the asymptotic Gaussian distribution of the parameters $\boldsymbol{\beta}$, allowing for the estimation of standard errors, hypothesis testing, and construction of confidence intervals.

However, log-linear regression does have a significant drawback – its regression function, being linear, lacks flexibility. To address this limitation, various techniques can be employed. In fact, any supervised learning technique could be used for the specification of $\mu(\cdot)$. One simple approach to incorporating non-linearity involves adding polynomial terms of the predictors to the model alongside the linear terms. Splines, on the other hand, offer a flexible and powerful method for modeling non-linear relationships. Instead of fitting a single global function, splines divide the predictor range into smaller intervals and fit separate polynomial functions within each interval. This approach enables more localized and flexible modeling of the relationship between the predictors and the mean parameter.

3.5.1.1.2 CANN Poisson regression.

In some cases, the supervised learning problem may require even more flexibility, and neural networks are particularly useful in such scenarios. Neural networks are formidable function approximation machines, well-known for their ability to estimate a wide range of highly non-linear multivariate functions. One of the key advantages of neural networks is their ability to handle raw and unstructured data effectively. Because we deal with detailed telematics data, this capability forms the basis for adopting the Combined Actuarial Neural Network (CANN) approach of (Wüthrich et Merz, 2019), which embeds a classical actuarial model into a neural network architecture. A CANN model consists of two distinct components: the classical regression model component and the neural network component. This architecture offers great flexibility, allowing for seamless integration of any classical model whose regression function is compatible with a neural network architecture. Likewise, the neural network component can employ various types of supervised architectures, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and other architectures tailored to the specific problem at hand. The classical regression model provides good initial estimations and serves as a guide for the neural network component. It offers a starting point for the network's optimization process, enabling faster convergence. The neural network component, in turn, refines the initial estimations, capturing additional signals and uncovering patterns that may have been missed by the classical model alone.

In our specific case, we use log-linear count regression as the classical model and a multilayer perceptron (MLP) as the neural network component in the CANN model. As a result, we have the following specification for the regression function:

$$\mu^{\text{CANN}}(\boldsymbol{x};\boldsymbol{\beta},\boldsymbol{\theta}) = \mu^{\text{LL}}(\boldsymbol{x};\boldsymbol{\beta}) \times \mu^{\text{MLP}}(\boldsymbol{x};\boldsymbol{\theta}), \tag{3.6}$$

where $\mu^{\text{MLP}}(\cdot)$ is the regression function learned by a multilayer perceptron parametrized with $\boldsymbol{\theta}$. In a nutshell, an MLP consists of interconnected layers, including an input layer, hidden layers, and an output layer. Each layer applies an affine transformation to the inputs it receives, followed by a non-linear activation function. This combination of linear transformations and non-linear activations allows MLPs to model complex non-linear relationships in the data. To delve into the mathematical description of an MLP, we can break down its structure starting from the input layer and progressing towards the output layer:

1. **Input layer ($l = 0$):** The input layer consists of n_0 nodes representing the input variables $\boldsymbol{x} = [x_1, x_2, \ldots, x_{n_0}]$.

2. **First hidden layer ($l = 1$):** The first hidden layer contains n_1 nodes, connected to the nodes from the input layer ($l = 0$) and the nodes in the subsequent layer ($l = 2$). The computations in the first hidden layer involve an affine transformation of the input variables followed by the application of a non-linear activation function, which introduces non-linearity into the network. Let us denote the weight matrix between layers $l = 0$ and $l = 1$ as $\boldsymbol{W}^{(1)}$ with dimensions (n_1, n_0) and the bias vector as $\boldsymbol{b}^{(1)}$ with dimensions $(n_1, 1)$. The activation function applied to the transformed inputs is denoted as ϕ. The computations in the first hidden layer can then be expressed as:

$$\boldsymbol{a}^{(1)} = \boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}, \quad \boldsymbol{z}^{(1)} = \phi\left(\boldsymbol{a}^{(1)}\right), \tag{3.7}$$

where $\boldsymbol{a}^{(1)}$ represents the preactivation values in the first hidden layer, and $\boldsymbol{z}^{(1)}$ represents the post-activation values. It is worth noting that the activation function ϕ is applied element-wise on the preactivation vector $\boldsymbol{a}^{(1)}$.

3. **Subsequent hidden layers ($l = 2, 3, \ldots, L-2$):** Each subsequent hidden layer l contains n_l nodes, connected to the nodes from the previous layer ($l-1$) and the nodes in the following layer ($l+1$). Similar to the first layer, the computations in the subsequent hidden layers involve an affine transformation of the inputs $\boldsymbol{z}^{(l-1)}$ followed by the application of the non-linear activation function ϕ. Let us denote the weight matrix between layers $l-1$ and l as $\boldsymbol{W}^{(l)}$ with dimensions (n_l, n_{l-1}) and the bias vector as $\boldsymbol{b}^{(l)}$ with dimensions $(n_l, 1)$. The computations in the hidden layers can then be expressed as:

$$\boldsymbol{a}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{z}^{(l-1)} + \boldsymbol{b}^{(l)}, \quad \boldsymbol{z}^{(l)} = \phi\left(\boldsymbol{a}^{(l)}\right), \tag{3.8}$$

   where $\boldsymbol{a}^{(l)}$ represents the preactivation values in the l^{th} hidden layer, and $\boldsymbol{z}^{(l)}$ represents the post-activation values.

4. **Output layer ($l = L-1$):** The output layer consists of n_{L-1} nodes, representing the final output(s) of the MLP. Similar to the hidden layers, the output layer involves an affine transformation followed by an activation function g. We denote the weight matrix between layers $L-2$ (last hidden layer) and $L-1$ as $\boldsymbol{W}^{(L-1)}$ with dimensions (n_{L-1}, n_{L-2}) and the bias vector as $\boldsymbol{b}^{(L-1)}$ with dimensions $(n_{L-1}, 1)$. The computations within the output layer can be expressed as:

$$\boldsymbol{a}^{(L-1)} = \boldsymbol{W}^{(L-1)}\boldsymbol{z}^{(L-2)} + \boldsymbol{b}^{(L-1)}, \quad \boldsymbol{z}^{(L-1)} = g\left(\boldsymbol{a}^{(L-1)}\right). \tag{3.9}$$

Note that the number of output neurons n_{L-1} should match the number of modeled distribution parameters. In the context of Poisson regression, where we are modeling a single parameter μ, only one output neuron is necessary. The choice of the output activation function $g(\cdot)$ is important and should be aligned with the specific problem being tackled since it determines the range and properties of the output values. For instance, in the classic case of a multi-class classification problem (where the multinoulli distribution is used as a specification for the target variable), each output neuron represents a class, and the predicted probabilities for each class should be positive and sum up to 1. In this scenario, a common choice for the activation function is the softmax function, which normalizes the outputs and ensures they are positive and sum up to 1. In our case, we need to ensure that the parameter μ, which represents the expected count, is always positive. While the exponential function is a natural choice to enforce positivity, it can sometimes lead to numerical instability, especially for large input values. As a better alternative, we choose to use the softplus function as the activation function for the output layer, defined as $\zeta(x) = \log(1 + \exp(x))$. The softplus function is well-behaved even for large input values, mitigating the issue of numerical instability that can arise with the exponential function. The parameters of the generic MLP described above, consisting of weight matrices and bias vectors, can be denoted as:

$$\boldsymbol{\theta} = \left\{ \boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}, \ldots, \boldsymbol{W}^{(L-1)}, \boldsymbol{b}^{(L-1)} \right\}. \tag{3.10}$$

Naturally, these parameters must be estimated. However, the criterion in Equation (3.4) is typically not convex, making it challenging to find the global minimum of the empirical risk. In practice, the goal is to find a "good enough" local minimum that yields satisfactory performance on the task. Gradient descent algorithms, such as stochastic gradient descent (SGD) and its variants, are commonly employed to update iteratively the parameters $\boldsymbol{\theta}$ in the direction of the steepest descent. The backpropagation algorithm efficiently computes the gradients and propagates them through the network, enabling parameter updates. With the introduced notation for the MLP, we can now express the specification in (3.6) as

$$\mu^{\text{CANN}}(\boldsymbol{x}; \boldsymbol{\beta}, \boldsymbol{\theta}) = \zeta \left\{ \langle \boldsymbol{x}, \boldsymbol{\beta} \rangle + \boldsymbol{a}^{(L-1)}(\boldsymbol{x}; \boldsymbol{\theta}) \right\}. \tag{3.11}$$

The corresponding computational graph for $L = 5$ (i.e., 3 hidden layers) is shown in Figure 3.2. Like a standard MLP, the network parameters $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$ can be estimated using gradient descent. A simplified pseudo-algorithm for the training of the Poisson CANN model is provided in Algorithm 2. The learning rate η is a hyperparameter that determines the step size taken every time a gradient descent step is performed. In other words, it controls how quickly or slowly the network parameters are updated during training. A higher learning rate allows for larger steps, which can lead to faster convergence. However, an excessively high learning rate may cause the optimization process to overshoot or oscillate around the minimum, hindering convergence. Conversely, a very low learning rate might result in slow convergence, requiring more iterations to reach an acceptable solution. Finding the right learning rate is important and is typically an empirical process that requires experimentation and tuning. In practice, mini-batch gradient descent is commonly used for training neural networks. It works by dividing the training data into smaller subsets, called mini-batches, and computing the gradients and parameter updates based on these mini-batches. This approach offers computational efficiency and improved generalization compared to regular gradient descent, making it a preferred choice in practice. For a comprehensive understanding of neural networks, we refer to the excellent book (Goodfellow *et al.*, 2016).

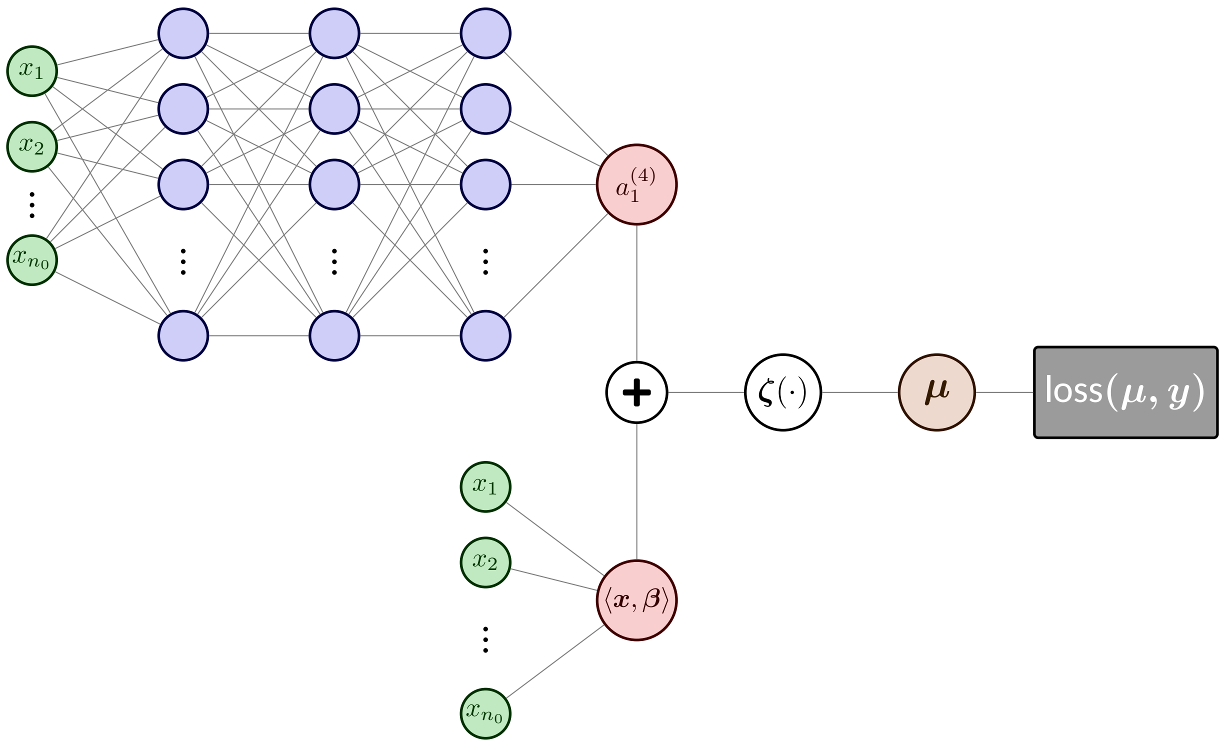

x_1
x_2
x_{n_0}
$a_1^{(4)}$
$+$
$\zeta(\cdot)$
$\boldsymbol{\mu}$
loss$(\boldsymbol{\mu}, \boldsymbol{y})$
x_1
x_2
x_{n_0}
$\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$

Figure 3.2: CANN architecture for the Poisson specification. The MLP's preactivation output value $a_1^{(4)}$ is added to the log-linear model's preactivation output value $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$ before being transformed with the softplus function $\zeta(\cdot)$. The resulting μ value is compared to the ground truth y using Poisson cross-entropy loss. The architecture shown employs a 3-hidden-layer MLP, but can be customized with any number of layers.

3.5.1.2 Negative binomial regression

One issue with the Poisson distribution is its equidispersion assumption. Indeed, we have that $\mu(\boldsymbol{x}) = \mathbb{E}[Y_{it}|\boldsymbol{X} = \boldsymbol{x}] = \text{Var}[Y_{it}|\boldsymbol{X} = \boldsymbol{x}]$. In practice, claim count data often exhibit overdispersion, where the observed variance of the claim count is greater than the mean. To address this limitation, alternative distributions allowing for overdispersion can be used. Among them, the negative binomial distribution (see, for instance, (Denuit *et al.*, 2007) and (Cameron et Trivedi, 2013)) stands out as a common choice. Under the negative binomial specification, the PMF of the claim count for the t^{th} contract of vehicle i (Y_{it}), given its predictor vector ($\boldsymbol{x}_{it}$), can be written as

$$\mathbb{P}(Y_{it} = y_{it}|\boldsymbol{x}_{it}) = \frac{\Gamma(y_{it} + \phi)}{y_{it}!\Gamma(\phi)} \left(\frac{\phi}{\phi + \mu(\boldsymbol{x}_{it})}\right)^{\phi} \left(\frac{\mu(\boldsymbol{x}_{it})}{\mu(\boldsymbol{x}_{it}) + \phi}\right)^{y_{it}}, \quad \text{for} \quad y_{it} \in \mathbb{N}, \tag{3.12}$$

```
Algorithm 2: Parameter estimation procedure – Poisson CANN model
Input: Training dataset $\{(\boldsymbol{x}_{it}, y_{it})\}_{(i,t)\in\mathcal{T}_r}$, learning rate $\eta$, number of epochs $E$
Output: Trained model parameters $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\beta}}$ and $\hat{w}_\phi$
Initialize model parameters $\hat{\boldsymbol{\theta}} = \mathbf{0}$, $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}^{\text{MLE}}$ and $\hat{w}_\phi$.
for epoch $\leftarrow$ 1 to $E$ do
    1. For each contract $(i,t)$, apply the CANN regression function with current network parameters to
       compute the current estimated mean parameter $\hat{\mu}_{it}$:
           $\hat{\mu}_{it} = \mu^{\text{CANN}}(\boldsymbol{x}_{it}; \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}).$
    2. Compute the empirical risk over the training dataset:
           $\mathcal{R} = -\frac{1}{|\mathcal{T}_r|} \sum_{(i,t)\in\mathcal{T}_r} y_{it} \ln[\hat{\mu}_{it}] - \hat{\mu}_{it} - y_{it}!.$
    3. Perform backpropagation to compute the gradients of $\mathcal{R}$ with respect to the network parameters:
           $\nabla_{\boldsymbol{\beta}}\mathcal{R}$ and $\nabla_{\boldsymbol{\theta}}\mathcal{R}$.
    4. Perform gradient descent using the learning rate:
           $\hat{\boldsymbol{\beta}} \leftarrow \hat{\boldsymbol{\beta}} - \eta \nabla_{\hat{\boldsymbol{\beta}}}\mathcal{R},$
           $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} - \eta \nabla_{\hat{\boldsymbol{\theta}}}\mathcal{R}.$
end
```

where $\phi > 0$ is a dispersion parameter. This can be seen as a generalization of the Poisson distribution. Indeed, the Poisson distribution is recovered when $\frac{1}{\phi} \to 0$. The first two centered moments are given by:

$$\mathbb{E}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}] = \mu(\boldsymbol{x}_{it}) \quad \text{and} \quad \text{Var}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}] = \mu(\boldsymbol{x}_{it}) + \frac{\mu(\boldsymbol{x}_{it})^2}{\phi}. \tag{3.13}$$

As can be seen, the negative binomial specification assumes overdispersion since $\text{Var}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}] > \mathbb{E}[Y_{it}|\boldsymbol{X}_{it} = \boldsymbol{x}_{it}]$. Once the specification for the regression function $\mu(\cdot)$ has been chosen, which defines a set of candidate functions $\mathcal{H}$, one can estimate the parameters of the regression function $\mu(\cdot)$ along with the dispersion parameter ϕ by maximum likelihood or, equivalently, by minimizing the empirical risk over the training set:

$$\{\widehat{\mu}, \widehat{\phi}\} = \underset{\mu\in\mathcal{H},\phi>0}{\text{argmin}} \left\{ -\frac{1}{|\mathcal{T}_r|} \sum_{(i,t)\in\mathcal{T}_r} \ln\left[\frac{\Gamma(y_{it}+\phi)}{y_{it}!\Gamma(\phi)}\right] + \phi \ln\left[\frac{\phi}{\phi+\mu(\boldsymbol{x}_{it})}\right] + y_{it} \ln\left[\frac{\mu(\boldsymbol{x}_{it})}{\mu(\boldsymbol{x}_{it})+\phi}\right] \right\}. \tag{3.14}$$

3.5.1.2.1 Log-linear negative binomial regression.

As in the Poisson case, a common specification for $\mu(\cdot)$ is the log-linear form, defined in Equation (3.5). In this case, the criterion in (3.14) is convex, and convex optimization can be used to estimate $\boldsymbol{\beta}$ and ϕ.

3.5.1.2.2 CANN negative binomial regression.

Similar to the approach used for the Poisson case, a CANN architecture can be used to model the mean parameter in the negative binomial distribution. The specification for the regression function $\mu(\cdot)$ remains identical to the Poisson case, as defined in Equation (3.11). In order to incorporate the extra distribution parameter ϕ, an additional output neuron is introduced in the network. This output neuron is connected to a neural network weight $w_\phi \in \mathbb{R}$ through the softplus function, ensuring that ϕ remains positive, i.e., $\phi = \zeta(w_\phi)$. It is important to highlight that the distribution parameter ϕ is not directly connected to the input variables $\boldsymbol{x}$. As a result, no heterogeneity is incorporated into this parameter, and a common estimated value $\widehat{\phi}$ is used for all observations. The exact architecture for the negative binomial CANN model is depicted in Figure 3.3. The network parameters $\boldsymbol{\beta}$, $\boldsymbol{\theta}$, and w_ϕ can be learned by minimizing the criterion in Equation (3.14) using the procedure described in Algorithm 3.

3.5.2 Longitudinal models

Cross-sectional models assume independence between all contracts. However, in our case, the data exhibits clustering due to contracts being grouped by vehicle. While it is reasonable to assume independence between contracts from distinct vehicles, this assumption is less valid for contracts from the same vehicle. In reality, the claim counts of contracts within the same vehicle may be influenced by shared vehicle-specific characteristics, unobserved risk factors, or policy-level effects, resulting in dependence between observations within each vehicle cluster. To appropriately address this dependence, we transition from cross-sectional to longitudinal models, enabling the introduction of within-vehicle dependence. In the case of claim count data, a longitudinal model can efficiently leverage the history of the vehicles to refine the risk estimation for future contracts.

While various models are available to analyze longitudinal data, such as random effects models, fixed effects models, generalized estimating equations (GEE), and autoregressive models (AR), among others, empirical evidence in the context of claim count regression supports the effectiveness of random (or mixed) effects

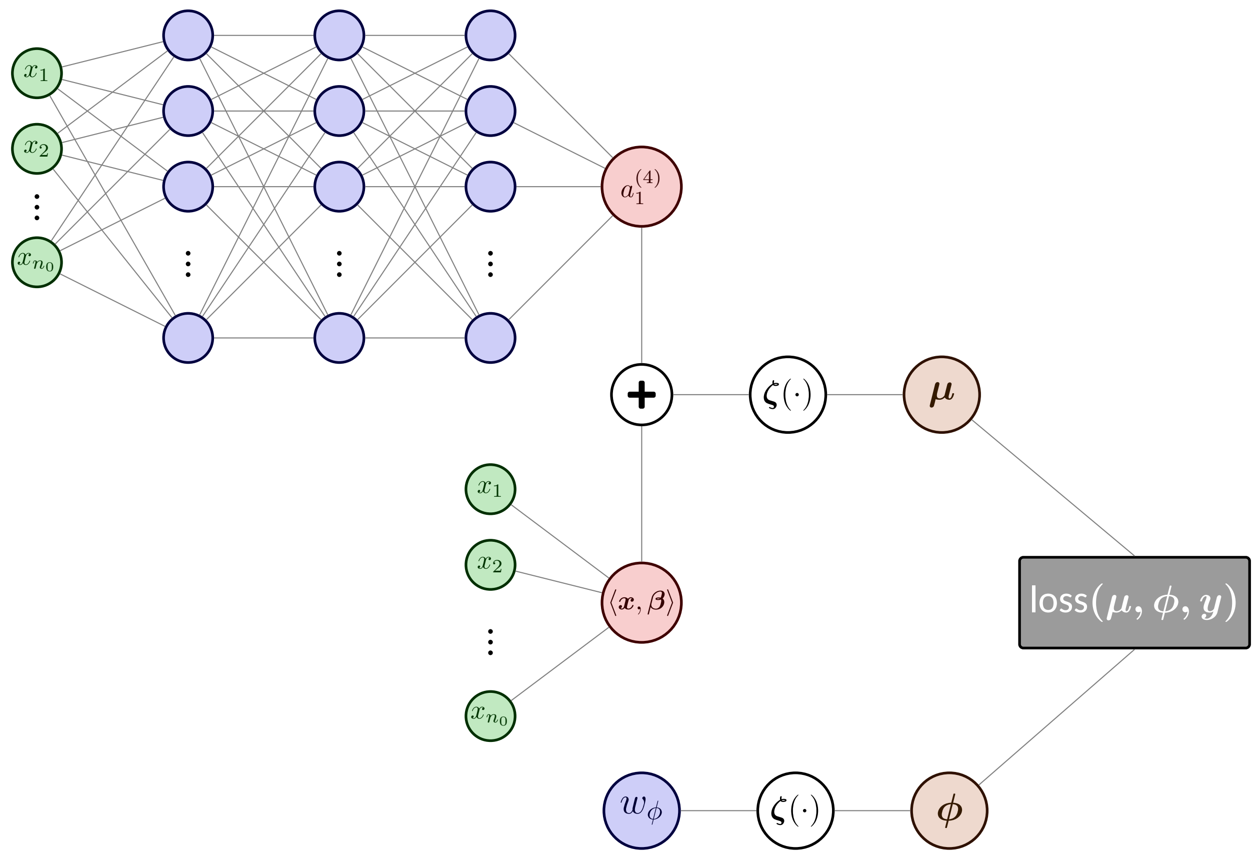


Figure 3.3: CANN architecture for the negative binomial specification. The MLP's preactivation output value $a_1^{(4)}$ is added to the log-linear model's preactivation output value $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$ before being transformed with the softplus function $\zeta(\cdot)$ to obtain the μ value of the negative binomial distribution. The ϕ value is obtained by transforming a real-valued parameter w_ϕ through the softplus function. The resulting parameters μ and ϕ are then compared to the ground truth y using negative binomial cross-entropy loss. The architecture shown employs a 3-hidden-layer MLP, but can be customized with any number of layers.

models (see (Boucher *et al.*, 2008)). In these models, a random effect, which is a random variable, is introduced in the specified distribution. For instance, in the case of count data, the specified distribution could be the Poisson distribution. The random effect is assumed to follow a certain distribution, such as a normal, gamma, or another appropriate distribution. The inclusion of the random effect allows for capturing the unobserved heterogeneity or individual-specific effects that cannot be accounted for by the observed covariates. It introduces additional variability into the model and accounts for the dependence within clusters. In longitudinal analysis, we need, for each vehicle i, to model the random vector of claim counts

Algorithm 3: Parameter estimation procedure – Negative binomial CANN model

```
Input: Training dataset $\{(\boldsymbol{x}_{it}, y_{it})\}_{(i,t)\in\mathcal{T}_r}$, learning rate $\eta$, number of epochs $E$
Output: Trained model parameters $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\beta}}$ and $\hat{w}_\phi$
Initialize model parameters $\hat{\boldsymbol{\theta}} = \boldsymbol{0}$, $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}^{\text{MLE}}$ and $\hat{w}_\phi$.
for epoch ← 1 to $E$ do
    1. For each contract $(i, t)$, apply the CANN regression function with current network parameters to
       compute the current estimated mean parameter $\hat{\mu}_{it}$:
           $\hat{\mu}_{it} = \mu^{\text{CANN}}(\boldsymbol{x}_{it}; \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}}).$
    2. Compute the current estimated parameter $\hat{\phi}$:
           $\hat{\phi} = \zeta(\hat{w}_\phi).$
    3. Compute the empirical risk over the training dataset:
           $\mathcal{R} = -\frac{1}{|\mathcal{T}_r|} \sum_{(i,t)\in\mathcal{T}_r} \ln\left[\frac{\Gamma(y_{it}+\hat{\phi})}{y_{it}!\Gamma(\hat{\phi})}\right] + \hat{\phi}\ln\left[\frac{\hat{\phi}}{\hat{\phi}+\hat{\mu}_{it}}\right] + y_{it}\ln\left[\frac{\hat{\mu}_{it}}{\hat{\mu}_{it}+\hat{\phi}}\right].$
    4. Perform backpropagation to compute the gradients of $\mathcal{R}$ with respect to the network parameters:
           $\nabla_{\boldsymbol{\beta}}\mathcal{R}, \quad \nabla_{\boldsymbol{\theta}}\mathcal{R} \quad \text{and} \quad \nabla_{w_\phi}\mathcal{R}.$
    5. Perform gradient descent using the learning rate:
           $\hat{\boldsymbol{\beta}} \leftarrow \hat{\boldsymbol{\beta}} - \eta\nabla_{\hat{\boldsymbol{\beta}}}\mathcal{R},$
           $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} - \eta\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{R},$
           $\hat{w}_\phi \leftarrow \hat{w}_\phi - \eta\nabla_{\hat{w}_\phi}\mathcal{R}.$
end
```

$\boldsymbol{Y}_{i,(1:T_i)} = (Y_{i1}, \ldots, Y_{i,T_i})$. The joint PMF can be expressed with

$$\mathbb{P}\left(\boldsymbol{Y}_{i,(1:T_i)} = \boldsymbol{y}_{i,(1:T_i)} | \boldsymbol{x}_{i,(1:T_i)}\right) = \int_{-\infty}^{\infty} \left(\prod_{t=1}^{T_i} \mathbb{P}(Y_{it} = y_{it} | \boldsymbol{x}_{i,(1:T_i)}, \theta_i)\right) f(\theta_i) d\theta_i, \tag{3.15}$$

where $f(\theta_i)$ is the PDF of the ramdom effect.

3.5.2.1 Multivariate negative binomial regression

A multivariate negative binomial regression model is obtained by introducing a gamma-distributed random effect in the mean parameter of the Poisson distribution. Specifically, we assume that the conditional distribution of Y_{it}, given $\Theta_i = \theta_i$, follows a Poisson distribution with mean $\mu(\boldsymbol{x}_{it})\theta_i$, where Θ_i is a gamma-distributed random variable with mean 1 and variance $1/\phi$. The density of Θ_i is given by

$$f_{\Theta_i}(\theta_i) = \frac{\phi^\phi}{\Gamma(\phi)}\theta_i^{\phi-1}e^{-\phi\theta_i}, \quad \theta_i > 0. \tag{3.16}$$

By using Equation (3.15), one can derive the joint distribution for the vector of claim counts:

$$\mathbb{P}\left(\boldsymbol{Y}_{i,(1:T_i)} = \boldsymbol{y}_{i,(1:T_i)}|\boldsymbol{x}_{i,(1:T_i)}\right) = \prod_{t=1}^{T_i}\left(\frac{\mu(\boldsymbol{x}_{it})^{y_{it}}}{y_{it}!}\right)\frac{\Gamma(y_{i\bullet}+\phi)}{\Gamma(\phi)}\left(\frac{\phi}{\mu_{i\bullet}+\phi}\right)^\phi\left(\frac{1}{\mu_{i\bullet}+\phi}\right)^{y_{it}}, \tag{3.17}$$

where $\mu_{i\bullet} = \sum_{t=1}^{T_i}\mu_{it}$ and $y_{i\bullet} = \sum_{t=1}^{T_i} y_{it}$. This joint distribution is commonly referred to as the multivariate negative binomial (MVNB) or negative multinomial distribution. Note that the Poisson distribution is retrieved when $\frac{1}{\phi} \to 0$. Furthermore, given the past claim history denoted as $\boldsymbol{y}_{i,(1:t-1)} = (y_{i1}, \dots, y_{i,t-1})$ as well as current and past covariate vectors denoted as $\boldsymbol{x}_{i,(1:t)} = (\boldsymbol{x}_{i1}, \dots, \boldsymbol{x}_{it})$, one can show that the number of claims at time (or contract) t follows a negative binomial distribution. The probability of observing y_{it} claims at time t, given the past claim history as well as past and current covariate vectors, is thus expressed with

$$\mathbb{P}(Y_{it} = y_{it}|\boldsymbol{y}_{i,(1:t-1)}, \boldsymbol{x}_{i,1:t}) = \frac{\Gamma(y_{it}+\alpha_{it})}{y_{it}!\Gamma(\alpha_{it})}\left(\frac{\gamma_{it}}{\gamma_{it}+\mu(\boldsymbol{x}_{it})}\right)^{\alpha_{it}}\left(\frac{\mu(\boldsymbol{x}_{it})}{\mu(\boldsymbol{x}_{it})+\gamma_{it}}\right)^{y_{it}}, \quad t = 1, 2, \dots, T_i, \tag{3.18}$$

where $\alpha_{it} = \phi + \Sigma_{it}^{(y)}$ and $\gamma_{it} = \phi + \Sigma_{it}^{(\mu)}$. $\Sigma_{it}^{(y)} = \sum_{t'=1}^{t-1} y_{it'}$ and $\Sigma_{it}^{(\mu)} = \sum_{t'=1}^{t-1}\mu(\boldsymbol{x}_{it'})$ represent the number of past claims and the sum of past μ values for contract (i, t), resepctively. In the special case when $t = 1$, there is no past history and we set $\Sigma_{it}^{(y)} = \Sigma_{it}^{(\mu)} = 0$, which yields $\alpha_{i1} = \gamma_{i1} = \phi$. The expected claim count, given the past history, is given by:

$$\mathbb{E}\Big[Y_{it}|\boldsymbol{y}_{i,(1:t-1)}, \boldsymbol{x}_{i,1:t}\Big] = \mu(\boldsymbol{x}_{it})\left(\frac{\phi+\Sigma_{it}^{(y)}}{\phi+\Sigma_{it}^{(\mu)}}\right) \tag{3.19}$$

$$= \mu(\boldsymbol{x}_{it})\left(\frac{\alpha_{it}}{\gamma_{it}}\right). \tag{3.20}$$

Fitting an MVNB model, therefore, amounts to fitting a negative binomial model, where the parameters α_{it} and γ_{it} depend on the vehicle's history. Once the specification for the regression function $\mu(\cdot)$ is chosen, the parameter ϕ and the parameters in the regression function $\mu(\cdot)$ can be estimated by minimizing the empirical risk over the training set. This can be achieved through the following optimization problem:

$$\{\widehat{\mu}, \widehat{\phi}\} = \underset{\mu\in\mathcal{H},\phi>0}{\operatorname{argmin}}\left\{-\frac{1}{|\mathcal{T}_r|}\sum_{(i,t)\in\mathcal{T}_r}\ln\left[\frac{\Gamma(y_{it}+\alpha_{it})}{y_{it}!\Gamma(\alpha_{it})}\right] + \alpha_{it}\ln\left[\frac{\gamma_{it}}{\gamma_{it}+\mu(\boldsymbol{x}_{it})}\right] + y_{it}\ln\left[\frac{\mu(\boldsymbol{x}_{it})}{\mu(\boldsymbol{x}_{it})+\gamma_{it}}\right]\right\}. \tag{3.21}$$

3.5.2.1.1 Log-linear multivariate negative binomial regression.

If the specification for $\mu(\cdot)$ is the log-linear form, defined in Equation (3.5), the criterion in (3.21) is convex, and convex optimization can be used to estimate $\boldsymbol{\beta}$ and ϕ.

3.5.2.1.2 CANN multivariate negative binomial regression.

In the MVNB case, the CANN architecture, as defined in Equation (3.11), can also be used as a specification for the regression function $\mu(\cdot)$. To incorporate the additional distribution parameters α_{it} and γ_{it}, two additional output neurons are introduced in the network, as depicted in Figure 3.4. The distribution parameters α_{it} and γ_{it} stem from a common parameter $\phi > 0$, and for the t^{th} contract of vehicle i, we have $\alpha_{it} = \phi + \Sigma_{it}^{(y)}$ and $\gamma_{it} = \phi + \Sigma_{it}^{(\mu)}$. The neuron representing ϕ is connected to a network weight $w_\phi \in \mathbb{R}$ through the softplus function, i.e., $\phi = \zeta(w_\phi)$. An MVNB CANN model can be trained with backpropagation and gradient descent, as outlined in Algorithm 4. Notice that for a vehicle i, the parameter γ_{it} depends on the μ parameter values for its past contracts. As the training procedure of the CANN model is iterative, the estimated μ values change at each iteration. Hence, it is crucial to update $\Sigma_{it}^{(\mu)}$ for each contract (i, t) at every iteration. This updating procedure is carried out in step 2 of Algorithm 4.

3.6 Pratical Application with Telematics Data

In this section, we explain how our CANN regression models are applied to our datastet. Additionally, we describe the application of the log-linear models, which serve as benchmark models in our analysis.

3.6.1 Log-linear models

The Poisson, negative binomial, and MVNB log-linear models are benchmarks for the Poisson, negative binomial, and MVNB CANN models. These models incorporate all 11 traditional risk factors from Table 2.2, including the real distance driven (although not strictly classified as a traditional risk factor). For each contract (i, t), these traditional risk factors are denoted by the vector $\boldsymbol{x}_{it}^{(\text{trad})}$. Notice that among the 11 traditional risk factors, 4 are categorical: `gender`, `marital_status`, `pmt_plan`, and `veh_use`. For these risk factors, the approach involves initially grouping all rare categories, defined as those representing 5% or less of the total number of observations, and labeling them as "others." We then encode them numerically using dummy encoding. All the resulting traditional covariates are then centered and scaled. Moreover, `commute_distance` contains missing values, which we fill in using median imputation.

Unlike neural networks, log-linear models do not have the ability to learn features directly from raw data. As a result, we must manually engineer features from the telematics data used by these models. These 13 telematics features, described in Table 3.4, were specifically engineered from the telematics dataset as risk factors potentially correlated with the claiming risk. For each contract (i, t), these numerical handcrafted telematics features are denoted by the vector $\boldsymbol{x}_{it}^{(\text{hand})}$. Note that these handcrafted telematics features are also centered and scaled prior to being input into the log-linear models. The regression function for the μ parameter can thus be written as

$$\mu\left(\boldsymbol{x}^{(\text{trad, hand})}; \boldsymbol{\beta}\right) = \exp\left(\langle \boldsymbol{x}^{(\text{trad, hand})}, \boldsymbol{\beta} \rangle\right), \tag{3.22}$$

where $\boldsymbol{x}^{(\text{trad, hand})}$ is the concatenation of $\boldsymbol{x}^{(\text{trad})}$ and $\boldsymbol{x}^{(\text{hand})}$. The parameters estimated on the training set are shown in Table 3.5. Notably, when using telematics information, the estimated ϕ parameter in the MVNB log-linear model is higher. A higher ϕ value brings the correcting factor in Equation 3.20 closer to one, indicating reduced importance on past experience when telematics features are used. This underscores the relevance of the engineered telematics features.

3.6.2 CANN models

For the CANN regression models, we extract low-level descriptor vectors that are specifically designed to accurately describe the driving patterns within a particular contract, at least with the dataset we have. We expect the MLP component within the CANN models to learn meaningful high-level features from these low-level vectors. The hope is that the learned features in the hidden layers will be more relevant than the handcrafted features of Table 3.4. Each contract (i, t) is described by the following descriptor vectors,

[1] 20h-0h

[2] 0h-6h

[3] 11h-14h

[4] 17h-20h Monday to Friday

[5] 7h-9h Monday to Friday

which provide a summary of its telematics information:

$$
\begin{aligned}
\boldsymbol{x}_{it}^{(h)} &= \left(x_{it,1}^{(h)}, \dots, x_{it,24}^{(h)}\right) \in \mathbb{R}^{24}, \\
\boldsymbol{x}_{it}^{(d)} &= \left(x_{it,1}^{(d)}, \dots, x_{it,7}^{(d)}\right) \in \mathbb{R}^{7}, \\
\boldsymbol{x}_{it}^{(a)} &= \left(x_{it,1}^{(a)}, \dots, x_{it,14}^{(a)}\right) \in \mathbb{R}^{14}, \\
\boldsymbol{x}_{it}^{(m)} &= \left(x_{it,1}^{(m)}, \dots, x_{it,16}^{(m)}\right) \in \mathbb{R}^{16}, \\
\boldsymbol{x}_{it}^{(k)} &= \left(x_{it,1}^{(k)}, \dots, x_{it,10}^{(k)}\right) \in \mathbb{R}^{10}.
\end{aligned}
$$

- The elements in vector $\boldsymbol{x}_{it}^{(h)}$ represent the fraction of driving during each of the 24 hours of the day. Therefore, $x_{it,j}^{(h)}$ is the fraction of driving during the jth hour of the day for contract (i, t).
- The elements in vector $\boldsymbol{x}_{it}^{(d)}$ represent the fraction of driving during each of the 7 days of the week. Therefore, $x_{it,j}^{(d)}$ is the fraction of driving during the jth day of the week for contract (i, t). Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday are denoted by $j = 1, 2, 3, 4, 5, 6, 7$, respectively.
- The elements in vector $\boldsymbol{x}_{it}^{(a)}$ represent the fraction of trips made in different average speed slots. For instance, $x_{it,j}^{(a)}$ denotes the fraction of trips made at an average speed between $10(j - 1)$ and $10j$ kilometers per hour.
- The elements in vector $\boldsymbol{x}_{it}^{(m)}$ represent the fraction of trips made in different maximum speed slots. For instance, $x_{it,j}^{(m)}$ denotes the fraction of trips made where the maximum speed reached falls between $10(j - 1)$ and $10j$ kilometers per hour.
- The elements in vector $\boldsymbol{x}_{it}^{(k)}$ represent the fraction of trips made in different distance slots. For instance, $x_{it,j}^{(k)}$ denotes the fraction of trips between $5(j - 1)$ and $5j$ kilometers.

These descriptor vectors capture specific aspects of the driving patterns, such as hourly, weekly, average speed, and maximum speed distribution, providing valuable information for the MLPs. Since MLPs can only accept vectors as input, we concatenate these four vectors into a global telematics vector:

$$\boldsymbol{x}_{it}^{(tele)} = \left(\boldsymbol{x}_{it}^{(h)}, \boldsymbol{x}_{it}^{(d)}, \boldsymbol{x}_{it}^{(a)}, \boldsymbol{x}_{it}^{(m)}, \boldsymbol{x}_{it}^{(k)}\right).$$

We incorporate this telematics vector into the MLP component of the CANN models, together with the traditional risk factors $\boldsymbol{x}^{(trad)}$, enabling interactions between telematics and traditional inputs. In contrast, the log-linear part of the CANN models only includes the traditional risk factors due to the difficulty of processing low-level information. The regression function for the μ parameter can thus be written as

$$\mu^{\text{CANN}}\left(\boldsymbol{x}^{(\text{trad, tele})};\boldsymbol{\beta},\boldsymbol{\theta}\right)=\zeta\left\{\langle\boldsymbol{x}^{\text{trad}},\boldsymbol{\beta}\rangle+\boldsymbol{a}^{(L-1)}(\boldsymbol{x}^{\text{trad,tele}};\boldsymbol{\theta})\right\}. \tag{3.23}$$

where $\boldsymbol{x}^{(\text{trad, tele})}$ is the concatenation of $\boldsymbol{x}^{(\text{trad})}$ and $\boldsymbol{x}^{(\text{tele})}$.

The CANN models are trained using the `torch` library in the `R` programming language, using mini-batch gradient descent with 256 observations per batch. The optimizer we use to perform gradient descent is the Adam optimizer, which is a fairly popular choice for training neural networks. Additionally, we use the reduce-on-plateau learning rate scheduler, which dynamically adjusts the learning rate based on the model's performance, automatically reducing it when the improvement plateaus, allowing for better optimization and convergence during training. For the MLP component of our CANN models, we opt for 3 hidden layers $(L = 5)$ with 128, 64, and 32 hidden units, respectively $(n_1 = 128, n_2 = 64, n_3 = 32)$. We choose the rectified linear unit (ReLU) as the activation function $\phi(\cdot)$ used in the hidden layers. Additionally, we add batch normalization and dropout layers in-between fully connected layers. Batch normalization applies a normalization transformation to the input of a layer by subtracting the mini-batch mean and dividing by the mini-batch standard deviation. By maintaining a stable mean and variance throughout the network, it can mitigate the vanishing or exploding gradients problem, enabling more effective and efficient training. The dropout layers, on the other hand, serve as a regularization technique that helps prevent overfitting. During training, dropout randomly sets a fraction of the hidden units of a given hidden layer to zero at each iteration, which forces the network to learn redundant representations and reduces the reliance on specific features. This regularization technique improves the model's ability to generalize well to unseen data.

3.6.3 CANN hyperparameter tuning

To maximize the performance of our CANN models, we use grid search for hyperparameter tuning, with the average loss observed on the validation dataset $\mathcal{V}_a$ as our optimization criterion. Additionally, we incorporate a regularization technique known as "early stopping" to determine the best number of epochs. This approach allows us to prevent overfitting and select the optimal number of epochs based on the lowest average loss achieved during training. We focus on three key hyperparameters: `p`, which represents the probability of dropout in the dropout layers, `l_start`, denoting the initial learning rate used in the reduce-on-plateau learning rate scheduler, and `factor`, indicating the factor by which the learning rate is multiplied upon reaching a plateau. A plateau is the point where there is no observed improvement in the validation loss for two consecutive epochs. We compute the average validation loss for all 45 combinations derived from the following hyperparameter values:

- `l_start`: $0.00001, 0.00005, 0.0001, 0.0005, 0.001$
- `factor`: $0.3, 0.4, 0.5$
- `p`: $0.2, 0.3, 0.4$.

Remember that the network parameters in the classical components of the CANN models are initialized with the maximum likelihood estimators of the corresponding log-linear model, which is why we use relatively small learning rates. At the initialization stage, the network already produces reasonable predictions, reducing the need for large gradient descent steps. The validation loss for each of the 45 combinations and the three specifications is presented in Table 3.6. It is worth noting that each model is trained for 30 epochs, and as early stopping is employed, the displayed average validation loss is based on the optimal number of epochs, which can be less than 30. As can be seen, for all learning rates higher than 0.00001, the minimum average validation loss is achieved after a very small number of epochs, indicating that the network learns too quickly. Although the negative binomial and MVNB models perform best at a learning rate of 0.0001, we believe that with more epochs, we could achieve a lower average loss with a learning rate of 0.00001. This is particularly true since the average losses are quite similar for `lr_start` = 0.00001 and `lr_start` = 0.0001. When examining the first 9 rows of Table 3.6, it becomes apparent that the `factor` hyperparameter has a negligible effect on the validation loss. On the other hand, the `p` hyperparameter only seems to have an impact on the validation loss for the Poisson model, performing best when `p` = 0.4. Although the dropout rate does not significantly affect the performance for both the negative binomial and MVNB models, we also choose `p` = 0.4 for these two models since the best performance is achieved at a high number of epochs (29 and 30 epochs, respectively). This suggests that with more epochs, there is potential for further performance improvement. Therefore, we select `lr_start` = 0.00001, `factor` = 0.3, and `p` = 0.4 as the hyperparameters for all three specifications. We train the models again on the training set, this time for 100 epochs. The performance of the three models on the validation set is displayed in Table 3.7. As can be seen, all 3 specifications require 35 epochs to minimize the average validation loss.

3.7 Analyzes

3.7.1 Performance assessment on the testing set

After carefully tuning the hyperparameters of our CANN models, we have at hand promising claim count models that are now nearing implementation. The next crucial step is to estimate their generalization capabilities accurately. To achieve this, we cannot rely on the validation set, as it has been extensively used during the hyperparameter tuning process. Instead, we assess the models' generalization performance using the testing set $\mathcal{T}_e$, which has remained untouched until now. Using this independent dataset, we can estimate the models' predictive performance on unseen data points and determine their suitability for real-world applications. Furthermore, we perform a comparative analysis between the CANN and the benchmark models, namely the log-linear models that use telematics information in the form of handcrafted telematics features. This comparative assessment allows us to evaluate our CANN models' relative performance and effectiveness against established approaches. In order to fully capture the value of telematics data, we also evaluate the performance of all 6 models (Poisson, negative binomial and MVNB log-linear and CANN models) using only the 11 traditional risk factors as covariates. In the CANN models, the MLP component therefore only comprises the 11 traditional risk factors. This analysis helps us understand the contribution of telematics information in improving the predictive power of the models.

All 12 models are trained on the learning set, and their performance is evaluated on the testing set. To assess the performance, we employ 3 different scoring rules, namely the Poisson deviance, the logarithmic score, and the squared error. For each scoring rule, we compute the average value on the testing set. To assess the magnitude of the achieved performance, we begin by calculating the average scoring rule values for a baseline model. This baseline model is defined as a homogeneous Poisson log-linear model, where the estimation of the mean (and variance) parameter μ is estimated by the average number of claims per contract observed in the learning set:

$$\hat{\mu} = \frac{1}{|\{\mathcal{T}_r, \mathcal{V}_a\}|} \sum_{(i,t)\in\{\mathcal{T}_r,\mathcal{V}_a\}} y_{it}. \tag{3.24}$$

The average scoring rule values for this baseline model on the testing set are reported in Table 3.8. We can then evaluate the performance of each of the 6 models in terms of percentage improvement over the baseline model, as shown in Table 3.9. As can be seen, our CANN models consistently outperform their corresponding log-linear benchmark models across all scoring rules. Moreover, our longitudinal MVNB CANN model offers a significative improvement over both Poisson and negative binomial distributions, suggesting a substantial dependence among contracts within a vehicle.

3.7.2 Permutation feature importance and partial dependence plots

One substantial drawback of neural networks is their difficulty of interpretation. However, researchers have developed tools to shed light on the inner workings of these black box algorithms. Two particularly useful tools in this context are permutation feature importance and partial dependence plots.

Permutation feature importance is a model-agnostic technique that computes an importance score for each input (or variable) in a supervised learning algorithm. It achieves this by randomly permuting the values of a specific input while holding the other inputs constant and observing the resulting effect on the model's performance. By comparing the original model's performance with the permuted performance, we can determine the variable's importance relative to the chosen performance metric. Suppose we have a trained model and a holdout sample for evaluation purposes. We can initially score the model on this sample and measure its performance using a chosen metric, such as the average loss. Let us denote the average loss obtained with the original holdout sample as ℓ_{original}. To assess the importance of input j in the prediction process, we randomly shuffle the values of input j in the holdout sample and rescore the model. This process yields a new average loss, denoted as $\ell^{(j)}_{\text{permuted}}$, where the superscript j indicates that input j has been permuted. If input j is indeed important for the model's prediction, the permuted average loss $\ell^{(j)}_{\text{permuted}}$ is expected to be greater than the original average loss ℓ_{original}. This suggests that permuting the values of input j has a detrimental effect on the model's performance. To obtain an importance score for input j, we can compute the difference between the permuted average loss and the original average loss, resulting in the feature importance score FI_j:

$$\text{FI}_j = \ell^{(j)}_{permuted} - \ell_{original}.$$

To obtain a more reliable estimate of the importance score, this procedure can be repeated a certain number of times for input j, creating a distribution of the increase or decrease in the average loss. The whole procedure can then be repeated for all inputs. In Figure 3.5, the importance scores of the 20 most important variables for our best model, the MVNB CANN, are visualized using boxplots. Please note that the names used for the telematics inputs in Figure 3.5 differ from the introduced notation. However, a translation table is provided in Table 3.10 of Appendix 3.9 to clarify the correspondence between the names used and the introduced notation. Each boxplot represents the distribution of the 100 importance scores assigned to a specific input obtained by shuffling and assessing the model 100 times. The performance metric used is the average cross-entropy loss. The analysis reveals interesting findings regarding the claim count model. As can be seen, the top 5 most important variables are from our set of 11 traditional risk factors. Notably, `veh_age`, `distance`, and `expo` play a significant role in the model's performance. When it comes to telematics inputs, those related to maximum speed demonstrate a substantial impact on the model's performance. Particularly, `vma_16`, representing the fraction of trips made at a maximum speed exceeding 150 kilometers per hour, stands out as the most important input. In general, the fraction of trips made at high maximum speeds, such as `vma_14`, `vma_15`, and `vma_16`, proves to be valuable for predicting claims. Additionally, it is interesting to observe that `h_22` and `h_2`, which represent the fraction of driving during night hours, contribute substantially to the assessment of risk. Importantly, the `gender` variable, often used by insurers as a risk factor, is rendered useless in the presence of telematics inputs. It ranks as the 70th most important variable (not showed in Figure 3.5), indicating its insignificance in the model's predictive power.

Partial Dependence Plots (PDP) are valuable tools for understanding the relationship between a specific input variable and the output of a supervised learning model. PDPs are also model-agnostic, meaning they can be applied to different types of models. They provide insights into how changes in a particular input variable influence the model's predictions while keeping all other variables at fixed values. In other words, they illustrate the marginal effect of an input variable on the predicted outcome. To compute a PDP for a specific input variable j, the process involves the following steps. First, a grid of values is defined to cover the entire or plausible range of the variable's values. Next, while holding all other variables fixed, the input vector in the holdout dataset is sequentially replaced with each value from the defined grid. Subsequently, predictions are obtained using the trained model on the modified holdout dataset for each value. By plotting the input variable values on the x-axis and the corresponding average prediction on the y-axis, the resulting PDP visually showcases the relationship between the input variable and the model's predictions. Figure 3.6 displays the PDPs of the 8 most important telematics inputs in the MVNB CANN model. The plots reveal that the risk, expressed as the expected number of claims, appears to increase in a linear fashion with the proportion of trips made at high maximum speeds, indicated by the input variables `vma_14`, `vma_15`, and `vma_16`. Additionally, there appears to be a positive linear association between the expected number of claims and the proportion of driving taking place during nighttime hours, specifically between 9 p.m. and 10 p.m. (h_22) and between 1 a.m. and 2 a.m. (h_2). It is important to emphasize that when interpreting partial dependence plots, caution must be exercised, as the procedure assumes that the input variables are independent of each other. In particular, the interpretation of the PDPs related to the fraction of driving on Tuesdays (`p_2`) is challenging due to the correlation between the proportions of driving on different days of the week. For instance, if an insured individual drives in smaller proportions on Tuesdays, they will systematically drive in larger proportions on other days of the week.

3.8 Conclusions

In this study, we developed three novel claim count regression models leveraging telematics data in the form of trip summaries. Our models are based on the Combined Actuarial Neural Network architecture, specifically designed to address actuarial problems and harness rich and complex information such as data provided by telematics technology. One key aspect of our work is the adaptation of the CANN architecture to accommodate the MVNB distribution specification. This adaptation allows us to effectively capture the time dependence between insurance contracts, which is important for accurately modeling claim counts. Furthermore, our findings highlight the importance of telematics inputs related to the maximum speed reached during trips in the claim count models. With partial dependence plots, we found that claim frequency is positively correlated with the fraction of trips made at high maximum speeds. Overall, the new approaches developed in this article represent a significant advancement in accurately modeling claim counts and enhancing the performance of predictive models in the context of usage-based insurance. Remarkably, the CANN regression models consistently outperform traditional log-linear models using handcrafted telematics features, as demonstrated by the superior performance across three performance metrics. These results are further supported by the use of a proper machine learning methodology that effectively prevents data leakage and mitigates the risk of producing falsely optimistic results.

While the available telematics data has been instrumental in improving our claim count models, we believe that further improvement can be achieved with access to richer data. For instance, if second-by-second data or additional information such as harsh acceleration/braking and distracted driving were accessible, we believe the performance could be further improved. Depending on the data format, different types of neural networks, such as convolutional and recurrent neural networks, could be used as the network component in the CANN models. Additionally, we acknowledge that with more time and computational power, a more comprehensive fine-tuning process of the CANN models could yield even better results than what we achieved. Notably, we were constrained in adjusting the number of hidden layers and units in the MLP components of the CANN models due to time and computational limitations. Moreover, a more advanced tuning method, beyond the grid search approach used in this study, could be employed to optimize model performance. In this study, we used the MVNB distribution as our longitudinal specification.

However, alternative longitudinal specifications, such as the beta-binomial distribution, exist and could be easily implemented as they share similarities with the MVNB specification. Finally, it would be interesting to conduct further research investigating the impact of using a longitudinal model on telematics variables. It is expected that the importance of certain telematics variables would decrease when considering past claim history, as this historical data can provide insights into the claiming risk of an insured.

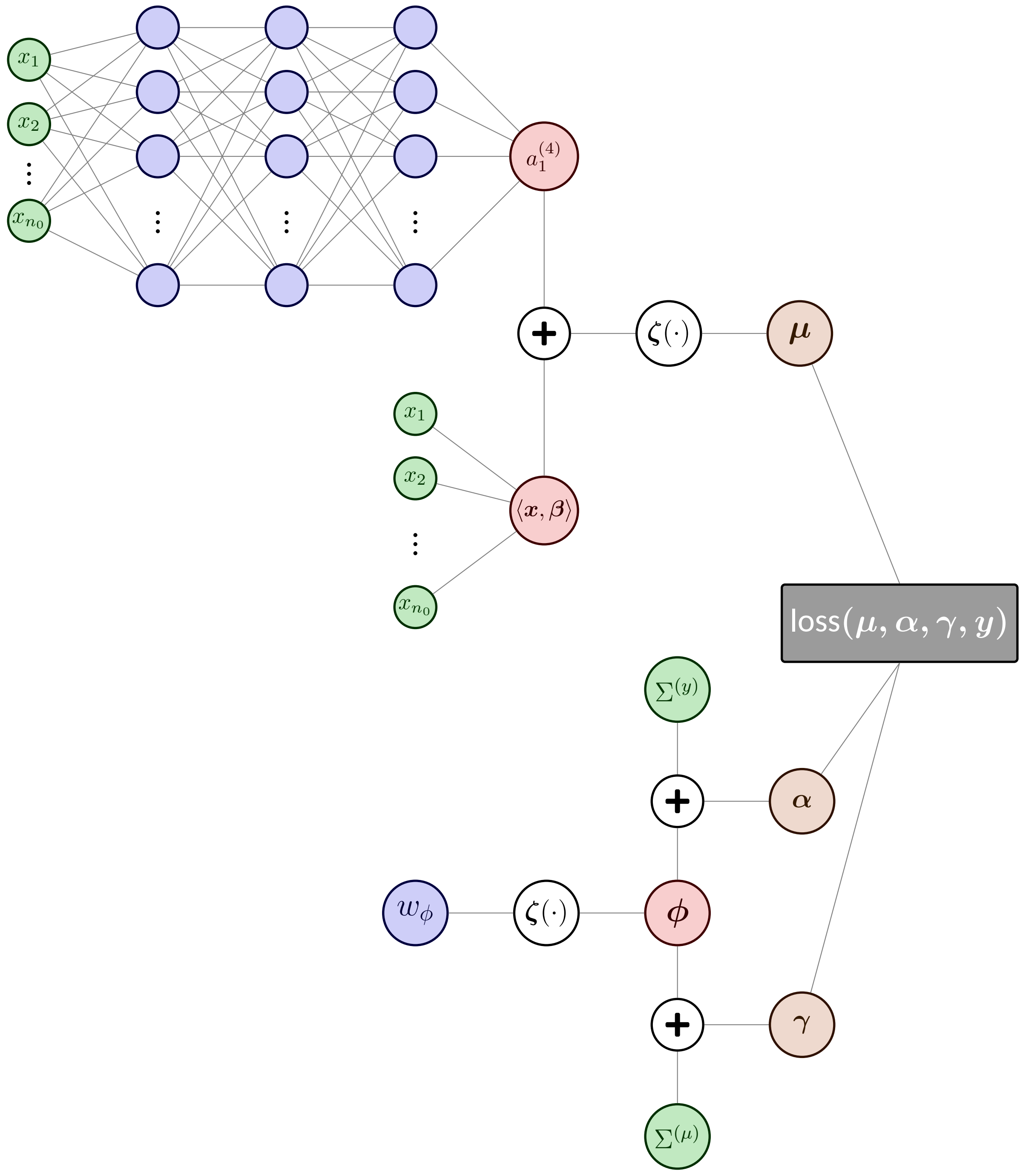


Figure 3.4: CANN architecture for the MVNB specification. The MLP's preactivation output value $a_1^{(4)}$ is added to the log-linear model's preactivation output value $\langle \boldsymbol{x}, \boldsymbol{\beta} \rangle$ before being transformed with the softplus function $\zeta(\cdot)$ to obtain the μ value of the negative binomial distribution of Equation (3.18). The ϕ value is obtained by transforming a real-valued parameter w_ϕ through the softplus function. To obtain α, the sum of past claims $\Sigma^{(y)}$ is added to the ϕ parameter, while for γ, the sum of past μ values $\Sigma^{(\mu)}$ is added to the same ϕ parameter. The resulting distribution parameters μ, α and γ are then compared to the ground truth y using negative binomial cross-entropy loss. The architecture shown employs a 3-hidden-layer MLP, but can be customized with any number of layers.

Algorithm 4: Parameter estimation procedure – MVNB CANN model

Input: Training dataset $\{(\boldsymbol{x}_{it}, y_{it})\}_{(i,t)\in\mathcal{T}_r}$, learning rate η, number of epochs E

Output: Trained model parameters $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\beta}}$ and $\hat{w}_\phi$

```
Initialize model parameters $\hat{\boldsymbol{\theta}} = \mathbf{0}$, $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}^{\mathsf{MLE}}$ and $\hat{w}_\phi$.
Compute the number of past claims for each contract $(i,t)$: $\Sigma_{it}^{(y)} = \sum_{t'=1}^{t-1} y_{it'}$.
for epoch ← 1 to $E$ do
    1. For each contract $(i,t)$, apply the CANN regression function with current network parameters to
       compute the current estimated mean parameter $\hat{\mu}_{it}$:
           $\hat{\mu}_{it} = \mu^{\mathsf{CANN}}(\boldsymbol{x}_{it}; \hat{\boldsymbol{\beta}}, \hat{\boldsymbol{\theta}})$.
    2. Initialize or update the sum of past $\mu$ values for each contract $(i,t)$: $\hat{\Sigma}_{it}^{(\mu)} = \sum_{t'=1}^{t-1} \hat{\mu}_{it}$.
    3. Compute the current estimated parameter $\hat{\phi}$:
           $\hat{\phi} = \zeta(\hat{w}_\phi)$.
    4. Compute the current estimated parameter $\hat{\alpha}_{it}$ and $\hat{\gamma}_{it}$:
           $\hat{\alpha}_{it} = \hat{\phi} + \Sigma_{it}^{(y)}$,
           $\hat{\gamma}_{it} = \hat{\phi} + \hat{\Sigma}_{it}^{(\mu)}$.
    5. Compute the empirical risk over the training dataset:
           $\mathcal{R} = -\frac{1}{|\mathcal{T}_r|} \sum_{(i,t)\in\mathcal{T}_r} \ln\left[\frac{\Gamma(y_{it}+\hat{\alpha}_{it})}{y_{it}!\Gamma(\hat{\alpha}_{it})}\right] + \hat{\alpha}_{it} \ln\left[\frac{\hat{\gamma}_{it}}{\hat{\gamma}_{it}+\hat{\mu}_{it}}\right] + y_{it} \ln\left[\frac{\hat{\mu}_{it}}{\hat{\mu}_{it}+\hat{\gamma}_{it}}\right]$.
    6. Perform backpropagation to compute the gradients of $\mathcal{R}$ with respect to the network parameters:
           $\nabla_{\boldsymbol{\beta}}\mathcal{R}$,   $\nabla_{\boldsymbol{\theta}}\mathcal{R}$   and   $\nabla_{w_\phi}\mathcal{R}$.
    7. Perform gradient descent using the learning rate:
           $\hat{\boldsymbol{\beta}} \leftarrow \hat{\boldsymbol{\beta}} - \eta\nabla_{\hat{\boldsymbol{\beta}}}\mathcal{R}$,
           $\hat{\boldsymbol{\theta}} \leftarrow \hat{\boldsymbol{\theta}} - \eta\nabla_{\hat{\boldsymbol{\theta}}}\mathcal{R}$,
           $\hat{w}_\phi \leftarrow \hat{w}_\phi - \eta\nabla_{\hat{w}_\phi}\mathcal{R}$.
end
```

Feature name	**Description**
`avg_daily_nb_trips`	Average daily number of trips
`frac_expo_evening`	Fraction of evening driving[1]
`frac_expo_fri_sat`	Fraction of driving on Friday and Saturday
`frac_expo_mon_to_thu`	Fraction of driving on Monday to Thursday
`frac_expo_night`	Fraction of night driving[2]
`frac_expo_noon`	Fraction of midday driving[3]
`frac_expo_peak_evening`	Fraction of evening rush hour driving[4]
`frac_expo_peak_morning`	Fraction of morning rush hour driving[5]
`max_trip_max_speed`	Maximum of the maximum speed of the trips
`med_trip_avg_speed`	Median of the average speeds of the trips
`med_trip_distance`	Median of the distances of the trips
`med_trip_max_speed`	Median of the maximum speeds of the trips
`prop_long_trip`	Proportion of long trips (> 100km)

Table 3.4: Handcrafted telematics features extracted from the telematics dataset.

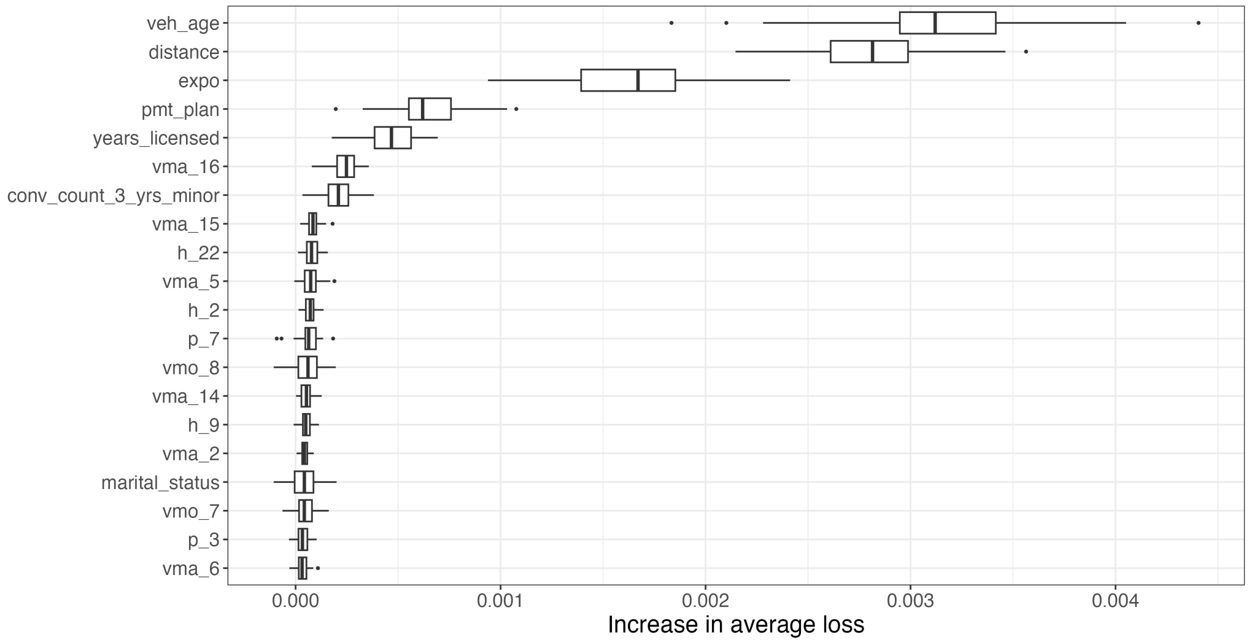


Figure 3.5: Importance scores of the 20 most important variables obtained for the MVNB CANN model.

Parameters	No telematics Poisson	No telematics Negative binomial	No telematics MVNB	With telematics Poisson	With telematics Negative binomial	With telematics MVNB
Intercept	-2.8310	-2.8311	-2.8281	-2.8454	-2.8456	-2.8430
annual_distance	0.0273	0.0280	0.0288	0.0353	0.0359	0.0367
commute_distance	0.0055	0.0055	0.0056	0.0159	0.0159	0.0157
conv_count_3_yrs_minor	0.0470	0.0474	0.0469	0.0381	0.0384	0.0380
distance	0.1697	0.1706	0.1681	0.1244	0.1252	0.1232
expo	0.1945	0.1943	0.1957	0.1812	0.1813	0.1834
gender_Male	-0.0234	-0.0238	-0.0236	-0.0409	-0.0415	-0.0415
marital_status_Single	0.0241	0.0243	0.0243	0.0194	0.0194	0.0192
marital_status_other	0.0342	0.0341	0.0341	0.0299	0.0297	0.0298
pmt_plan_EFT.Monthly	0.0963	0.0965	0.0969	0.0828	0.0830	0.0833
pmt_plan_Monthly	0.0856	0.0854	0.0850	0.0773	0.0771	0.0768
pmt_plan_other	0.0134	0.0135	0.0131	0.0111	0.0111	0.0107
veh_age	-0.1552	-0.1543	-0.1540	-0.1433	-0.1425	-0.1422
veh_use_other	-0.0085	-0.0083	-0.0084	-0.0100	-0.0098	-0.0100
veh_use_pleasure	-0.0025	-0.0023	-0.0027	-0.0014	-0.0013	-0.0018
years_licensed	-0.1061	-0.1064	-0.1076	-0.0538	-0.0539	-0.0547
avg_daily_nb_trips	-	-	-	0.0428	0.0424	0.0411
frac_expo_evening	-	-	-	0.0734	0.0738	0.0741
frac_expo_fri_sat	-	-	-	0.0290	0.0288	0.0294
frac_expo_mon_to_thu	-	-	-	0.0854	0.0852	0.0857
frac_expo_night	-	-	-	0.0192	0.0193	0.0198
frac_expo_noon	-	-	-	0.0103	0.0100	0.0092
frac_expo_peak_evening	-	-	-	0.0049	0.0047	0.0046
frac_expo_peak_morning	-	-	-	0.0072	0.0073	0.0071
max_trip_max_speed	-	-	-	0.1084	0.1087	0.1079
med_trip_avg_speed	-	-	-	-0.1465	-0.1470	-0.1472
med_trip_distance	-	-	-	0.0082	0.0088	0.0081
med_trip_max_speed	-	-	-	0.0725	0.0723	0.0736
prop_long_trip	-	-	-	0.0310	0.0314	0.0322
ϕ	-	2.8397	3.4868	-	3.1193	3.9119

Table 3.5: Estimated parameters of the log-linear models on the training set.

Hyperparameter values			Average validation loss			Number of epochs		
`l_start`	`factor`	`p`	Poisson	Negative binomial	MVNB	Poisson	Negative binomial	MVNB
0.00001	0.3	0.2	0.2357	0.2351	0.2350	8	16	17
0.00001	0.3	0.3	0.2354	0.2350	0.2350	10	24	22
0.00001	0.3	0.4	0.2353	0.2351	0.2349	18	29	30
0.00001	0.4	0.2	0.2358	0.2351	0.2350	8	16	17
0.00001	0.4	0.3	0.2355	0.2350	0.2350	10	24	22
0.00001	0.4	0.4	0.2353	0.2351	0.2349	18	29	30
0.00001	0.5	0.2	0.2360	0.2351	0.2350	8	16	17
0.00001	0.5	0.3	0.2356	0.2350	0.2350	10	24	22
0.00001	0.5	0.4	0.2354	0.2351	0.2349	18	29	30
0.00005	0.3	0.2	0.2355	0.2351	0.2349	2	4	4
0.00005	0.3	0.3	0.2355	0.2351	0.2349	3	5	5
0.00005	0.3	0.4	0.2363	0.2352	0.2351	4	8	7
0.00005	0.4	0.2	0.2355	0.2351	0.2349	2	4	4
0.00005	0.4	0.3	0.2355	0.2351	0.2349	3	5	5
0.00005	0.4	0.4	0.2365	0.2352	0.2351	4	8	7
0.00005	0.5	0.2	0.2355	0.2351	0.2349	2	4	4
0.00005	0.5	0.3	0.2355	0.2351	0.2349	3	5	5
0.00005	0.5	0.4	0.2369	0.2352	0.2351	4	8	7
0.0001	0.3	0.2	0.2354	0.2349	0.2349	1	3	3
0.0001	0.3	0.3	0.2354	0.2350	0.2348	2	3	3
0.0001	0.3	0.4	0.2371	0.2352	0.2350	2	5	4
0.0001	0.4	0.2	0.2354	0.2349	0.2349	1	3	3
0.0001	0.4	0.3	0.2354	0.2350	0.2348	2	3	3
0.0001	0.4	0.4	0.2374	0.2352	0.2350	2	5	4
0.0001	0.5	0.2	0.2354	0.2349	0.2349	1	3	3
0.0001	0.5	0.3	0.2354	0.2350	0.2348	2	3	3
0.0001	0.5	0.4	0.2378	0.2352	0.2350	2	5	4
0.0005	0.3	0.2	0.2356	0.2350	0.2350	1	1	1
0.0005	0.3	0.3	0.2354	0.2352	0.2350	2	1	1
0.0005	0.3	0.4	0.2358	0.2352	0.2350	2	4	3
0.0005	0.4	0.2	0.2356	0.2350	0.2350	1	1	1
0.0005	0.4	0.3	0.2354	0.2352	0.2350	2	1	1
0.0005	0.4	0.4	0.2358	0.2352	0.2350	2	4	3
0.0005	0.5	0.2	0.2356	0.2350	0.2350	1	1	1
0.0005	0.5	0.3	0.2354	0.2352	0.2350	2	1	1
0.0005	0.5	0.4	0.2358	0.2352	0.2350	2	4	3
0.001	0.3	0.2	0.2358	0.2353	0.2351	1	1	1
0.001	0.3	0.3	0.2362	0.2352	0.2349	1	1	1
0.001	0.3	0.4	0.2362	0.2350	0.2349	2	2	2
0.001	0.4	0.2	0.2358	0.2353	0.2351	1	1	1
0.001	0.4	0.3	0.2362	0.2352	0.2349	1	1	1
0.001	0.4	0.4	0.2362	0.2350	0.2349	2	2	2
0.001	0.5	0.2	0.2358	0.2353	0.2351	1	1	1
0.001	0.5	0.3	0.2362	0.2352	0.2349	1	1	1
0.001	0.5	0.4	0.2362	0.2350	0.2349	2	2	2

Table 3.6: Coarse hyperparameter tuning for the CANN models. The training process is stopped after 30 epochs. The provided validation loss corresponds to the optimal number of epochs, consistent with the early stopping procedure.

Specification	Average validation loss	Number of epochs
Poisson	0.2352	35
Negative binomial	0.2351	35
MVNB	0.2349	35

Table 3.7: Optimal CANN models' performance on the validation set.

Scoring rule	Baseline model
Poisson deviance	0.3682
Logarithmic score	0.2470
Squared error	0.0697

Table 3.8: Performance of the baseline model on the testing set.

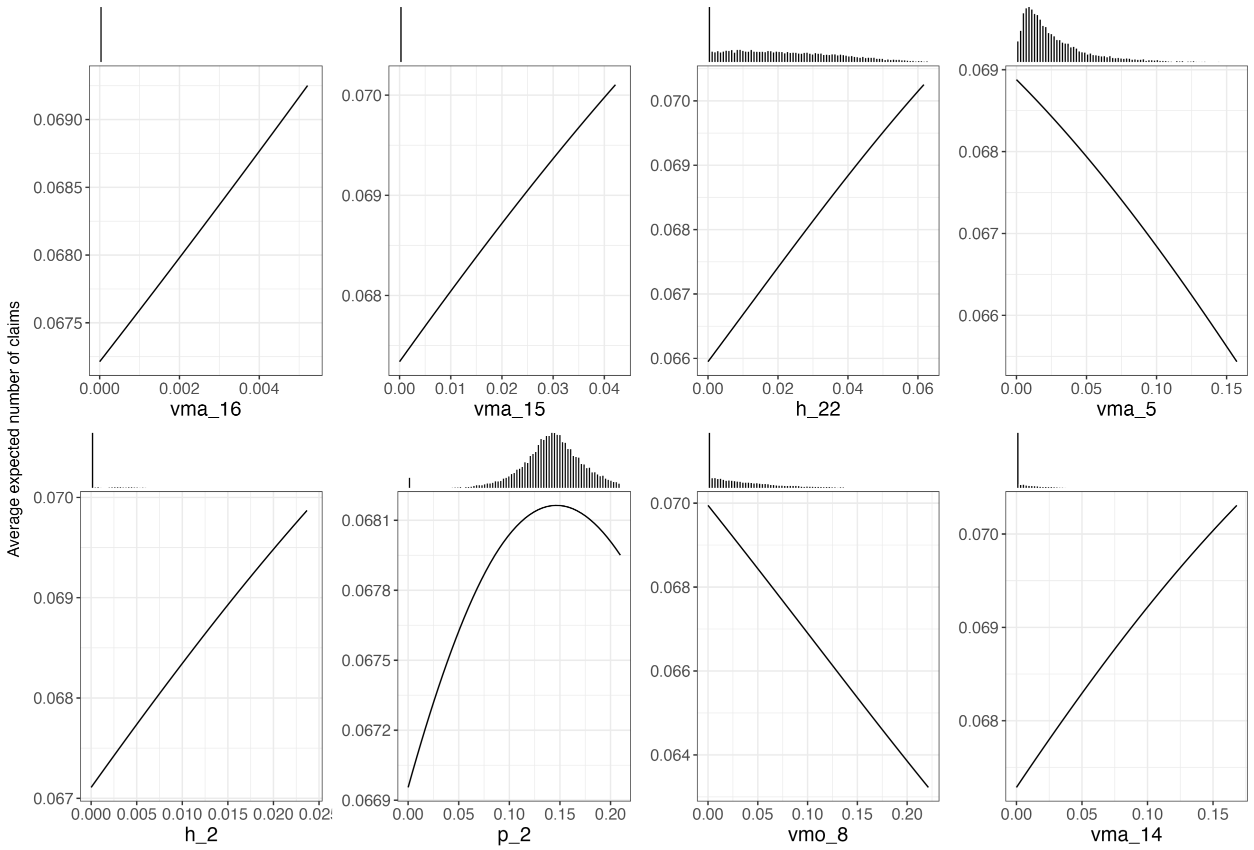


Figure 3.6: Partial dependence plots showcasing the 8 most important telematics inputs in the MVNB CANN model. The histogram above each line plots shows the input's distribution.

	No telematics		**With telematics**	
Scoring rule	Log-linear model	CANN model	Log-linear model	CANN model
	Poisson			
Poisson deviance	5.23 %	5.53 %	5.68 %	5.78 %
Logarithmic score	3.90 %	4.12 %	4.23 %	4.31 %
Squared error	2.10 %	2.26 %	2.30 %	2.38 %
	Negative binomial			
Poisson deviance	5.24 %	5.58 %	5.68 %	5.81 %
Logarithmic score	3.99 %	4.24 %	4.31 %	4.41 %
Squared error	2.10 %	2.27 %	2.30 %	2.37 %
	MVNB			
Poisson deviance	5.36 %	5.65 %	5.79 %	5.90 %
Logarithmic score	4.07 %	4.27 %	4.38 %	4.46 %
Squared error	2.13 %	2.29 %	2.34 %	2.41 %

Table 3.9: Performance comparison of the CANN models and their corresponding log-linear benchmark model on the testing set.

3.9 Appendix A: Telematics Input Names Translation

Introduced notation	Notation in the plots	Description
$x^{(h)}_{it,1}$	h_1	Fraction of driving between midnight and 1 a.m.
$x^{(h)}_{it,2}$	h_2	Fraction of driving between 1 a.m. and 2 a.m.
$\vdots$	⋮	⋮
$x^{(h)}_{it,24}$	h_24	Fraction of driving between 11 p.m. and midnight
$x^{(d)}_{it,1}$	p_1	Fraction of driving on Mondays
$x^{(d)}_{it,2}$	p_2	Fraction of driving on Tuesdays
$\vdots$	⋮	⋮
$x^{(d)}_{it,7}$	p_7	Fraction of driving on Sundays
$x^{(a)}_{it,1}$	vmo_1	Fraction of trips with average speed between 0 and 10 kph
$x^{(a)}_{it,2}$	vmo_2	Fraction of trips with average speed between 10 and 20 kph
$\vdots$	⋮	⋮
$x^{(a)}_{it,14}$	vmo_14	Fraction of trips with average speed exceeding 130 kph
$x^{(m)}_{it,1}$	vma_1	Fraction of trips with maximum speed between 0 and 10 kph
$x^{(m)}_{it,2}$	vma_2	Fraction of trips with maximum speed between 10 and 20 kph
$\vdots$	⋮	⋮
$x^{(m)}_{it,16}$	vma_16	Fraction of trips with maximum speed exceeding 150 kph
$x^{(k)}_{it,1}$	d_1	Fraction of trips with distance between 0 and 5 km
$x^{(k)}_{it,2}$	d_2	Fraction of trips with distance between 5 and 10 km
$\vdots$	⋮	⋮
$x^{(k)}_{it,10}$	d_10	Fraction of trips with distance exceeding 45 km

Table 3.10: Telematics Inputs Names Translation

CONCLUSION

Dans cette thèse, nous avons proposé des méthodes novatrices pour améliorer la tarification basée sur l'usage en assurance automobile et approfondir notre compréhension de l'impact des données télématiques sur le niveau de risque des assurés. En utilisant des techniques d'apprentissage automatique, nous avons développé des modèles prédictifs capables d'exploiter efficacement l'information télématique.

Au Chapitre 1, nous avons proposé plusieurs modèles de classification des réclamations utilisant les données télématiques sous forme de variables créées manuellement, à savoir une régression logistique avec régularisation lasso, une régression logistique avec régularisation elastic-net et une forêt aléatoire. Nous avons constaté que le modèle lasso est le mieux adapté pour cette tâche. En utilisant ce modèle, nous avons développé une procédure pour estimer la quantité minimale d'informations télématiques nécessaire afin d'évaluer de manière précise le risque d'un assuré. Nos résultats montrent qu'après environ 3 mois, ou 4000 kilomètres d'observation télématique, l'ajout d'informations supplémentaires n'améliore pas de manière substantielle la performance de l'algorithme de classification. Cette information est précieuse pour les chercheurs et les assureurs, car elle leur permet de réduire l'utilisation de ces données volumineuses dans des algorithmes gourmands en puissance de calcul, tout en maintenant une estimation précise du risque.

Au Chapitre 2, nous avons formulé l'hypothèse selon laquelle le degré de « routine » et de « péculiarité » d'un assuré est lié à son risque de réclamation. En utilisant des algorithmes de détection d'anomalies non-supervisés appliqués aux données télématiques, nous avons calculé des profils de routine et de péculiarité pour chaque assuré. À partir de ces profils, nous avons extrait des variables sous forme de quantiles, lesquels ont ensuite été évalués en termes de pouvoir prédictif via un modèle de régression logistique avec une pénalité elastic-net. Les résultats, obtenus grâce à une méthodologie appropriée d'apprentissage automatique, ont démontré que les quantiles issus des profils de péculiarité des assurés permettent d'améliorer l'estimation du risque. Cette méthode présente l'avantage d'être objective, car elle minimise l'utilisation du jugement humain, parfois sujet à des biais, dans l'extraction de représentations utiles à partir des données télématiques.

Au Chapitre 3, nous avons exploité un perceptron multicouches pour automatiser et optimiser l'extraction de représentations à partir des données télématiques brutes. Dans cette optique, nous avons développé des modèles de régression de comptage des réclamations basés sur une architecture de réseau de neurones appelée *Combined Actuarial Neural Network* (CANN). Cette architecture hybride combine les avantages d'un modèle de régression classique avec ceux d'un réseau de neurones, nous permettant ainsi de tirer profit des deux approches. Grâce au perceptron multicouches, nous avons pu extraire automatiquement des représentations utiles des données télématiques en utilisant une approche supervisée, où le modèle apprend en se basant sur la variable réponse. Un élément clé de ce chapitre réside dans l'adaptation du modèle CANN à la spécification binomiale négative multivariée, ce qui permet une modélisation longitudinale des réclamations et donc une prise en compte de l'historique des réclamations. Les résultats obtenus démontrent que nos modèles CANN, utilisant les données télématiques brutes, surpassent les modèles classiques qui utilisent des variables créées manuellement à partir de ces données. En analysant notre modèle CANN longitudinal à l'aide d'un graphique de l'importance des variables par permutation et de graphes de dépendance partielle, nous avons constaté que la proportion de trajets effectués à des vitesses maximales élevées ainsi que la proportion de conduite à des heures tardives jouent un rôle important dans la détermination du niveau de risque d'un assuré, et qu'elles sont positivement corrélées au risque.

En bref, cette thèse a permis de faire avancer les connaissances et les méthodes dans le domaine de la tarification basée sur l'usage en assurance automobile. Les différents modèles proposés ont ouvert la voie à des approches plus efficaces et plus précises pour l'évaluation du risque. Au fil de nos analyses, plusieurs constatations ont émergé. Parmi les variables traditionnelles de tarification utilisées, l'âge du véhicule (`veh_age`), la durée du contrat (`expo`), le plan de paiement (`pmt_plan`) et le nombre d'années depuis l'obtention du permis de conduire (`years_licensed`) se sont avérés être des facteurs clés pour modéliser la sinistralité. Les variables télématiques, en particulier la distance parcourue, jouent également un rôle déterminant dans cette modélisation, tout comme les variables liées à la vitesse maximale atteinte lors des trajets, qui s'avèrent assez utiles pour la tarification. Dans le cadre de cette thèse, nous avons procédé à une évaluation de divers algorithmes d'apprentissage supervisé: les modèles linéaires généralisés, la régularisation Lasso, la régularisation *elastic-net*, les forêts aléatoires et les réseaux de neurones. Les résultats exposés dans le Chapitre 1 ont montré que les modèles linéaires généralisés régularisés surpassent les performances des forêts aléatoires. Par conséquent, nous préconisons l'utilisation de la régularisation dans le cadre de ces données, car elle améliore considérablement les performances prédictives tout en facilitant la sélection automatique des variables les plus pertinentes. En outre, en raison de leur meilleure interprétabilité, nous privilégions les modèles linéaires généralisés régularisés par rapport aux modèles basés sur des arbres de décision. Au Chapitre 3, nous avons démontré que l'application de réseaux de neurones sur les données télématiques peut améliorer la précision de la tarification. Toutefois, étant donné leur difficulté d'interprétation, nous recommandons de les utiliser uniquement lorsque les données le justifient. En effet, nous estimons que les réseaux de neurones sont particulièrement utiles lorsqu'il s'agit de données complexes qui ne peuvent pas être correctement traitées par des algorithmes plus simples, tels que les modèles linéaires généralisés ou l'expertise humaine. Cependant, pour des données plus simples, il est probable que les gains de performance soient négligeables. Dans ce cas, nous préconisons l'utilisation d'algorithmes plus interprétables, tels que les modèles linéaires généralisés ou les modèles additifs généralisés.

Enfin, diverses possibilités d'extension qui pourraient faire l'objet de recherches futures sont identifiées. D'une part, toutes les approches proposées pourraient être généralisées en utilisant des modèles de fréquence-sévérité, au lieu de se limiter à des modèles de classification ou de régression de comptage des réclamations. Cela permettrait une analyse plus complète des risques en calculant directement la prime pure. D'autre part, les approches pourraient être adaptées à des données télématiques plus détaillées. Dans cette thèse, nous nous sommes limités à l'utilisation de résumés de trajets pour nos analyses. Cependant, l'utilisation de données télématiques plus détaillées, telles que des observations collectées seconde par seconde, pourrait nous fournir une compréhension plus approfondie des habitudes de conduite et des comportements des assurés, tout en fournissant de l'information plus riche aux modèles prédictifs. Bien sûr, cela nécessiterait une adaptation des algorithmes pour prendre en compte la nature différente et le volume plus important de ces données.

ANNEXE A

CARACTÉRISTIQUES DE L'ORDINATEUR UTILISÉ POUR LES CALCULS EFFECTUÉS DANS CETTE THÈSE

Marque	Apple
Modèle	MacBook Pro 2016
Processeur	2,6 GHz Intel Core i7 quatre cœurs
Mémoire RAM	16 Go 2133 MHz LPDDR3
Stockage	256 Go SSD
Carte graphique	Intel HD Graphics 530 1536 Mo
Écran	Rétina 15,4 pouces (2880 × 1800)
Systèmes d'exploitation	MacOS Mojave, Catalina, Big Sur et Monterey
Numéro de série	C02TG07MGTFL

Table A.1: Spécifications de l'ordinateur

BIBLIOGRAPHIE

Ayuso, M., Guillen, M. et Marín, A. M. P. (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: emerging technologies*, *68*, 160–167.

Ayuso, M., Guillen, M. et Nielsen, J. P. (2019). Improving automobile insurance ratemaking using telematics: incorporating mileage and driver behaviour data. *Transportation*, *46*(3), 735–752.

Ayuso, M., Guillén, M. et Pérez-Marín, A. M. (2014). Time and distance to first accident and driving patterns of young drivers with pay-as-you-drive insurance. *Accident Analysis & Prevention*, *73*, 125–131.

Ayuso, M., Guillen, M. et Pérez-Marín, A. M. (2016b). Telematics and gender discrimination: Some usage-based evidence on whether men's risk of accidents differs from women's. *Risks*, *4*(2), 10.

Baecke, P. et Bocca, L. (2017). The value of vehicle telematics data in insurance risk selection processes. *Decision Support Systems*, *98*, 69–79.

Blier-Wong, C., Baillargeon, J.-T., Cossette, H., Lamontagne, L. et Marceau, E. (2020). Encoding neighbor information into geographical embeddings using convolutional neural networks. Dans *The Thirty-Third International Flairs Conference*.

Blier-Wong, C., Baillargeon, J.-T., Cossette, H., Lamontagne, L. et Marceau, E. (2021). Rethinking representations in P&C actuarial science with deep neural networks. *arXiv preprint arXiv:2102.05784*.

Bolderdijk, J. W., Knockaert, J., Steg, E. et Verhoef, E. T. (2011). Effects of pay-as-you-drive vehicle insurance on young drivers' speed choice: Results of a dutch field experiment. *Accident Analysis & Prevention*, *43*(3), 1181–1186.

Bordoff, J. E. et Noel, P. J. (2008). *The impact of pay-as-you-drive auto insurance in California*. Brookings Institution.

Boucher, J.-P., Côté, S. et Guillen, M. (2017). Exposure as duration and distance in telematics motor insurance using generalized additive models. *Risks*, *5*(4), 54.

Boucher, J.-P., Denuit, M. et 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, J.-P., Denuit, M. et Guillén, M. (2008). Models of insurance claim counts with time dependence based on generalization of poisson and negative binomial distributions. *Variance*, *2*(1), 135–162.

Boucher, J.-P., Pérez-Marín, A. M. et Santolino, M. (2013). Pay-as-you-drive insurance: the effect of the kilometers on the risk of accident. Dans *Anales del Instituto de Actuarios Españoles*, volume 19, 135–154. Instituto de Actuarios Españoles.

Boucher, J.-P. et Turcotte, R. (2020). A longitudinal analysis of the impact of distance driven on the probability of car accidents. *Risks*, *8*(3), 91.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Breiman, L., Friedman, J., Stone, C. J. et Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Breunig, M. M., Kriegel, H.-P., Ng, R. T. et Sander, J. (2000). Lof: identifying density-based local outliers. Dans *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 93–104.

Cameron, A. C. et Trivedi, P. K. (2013). *Regression analysis of count data*, volume 53. Cambridge university press.

Denuit, M., Maréchal, X., Pitrebois, S. et Walhin, J.-F. (2007). *Actuarial modelling of claim counts: Risk classification, credibility and bonus-malus systems*. John Wiley & Sons.

Dewri, R., Annadata, P., Eltarjaman, W. et Thurimella, R. (2013). Inferring trip destinations from driving habits data. Dans *Proceedings of the 12th ACM workshop on privacy in the electronic society*, 267–272.

Dionne, G. et Vanasse, C. (1989). A generalization of automobile insurance rating models: the negative binomial distribution with a regression component. *ASTIN Bulletin: The Journal of the IAA*, *19*(2), 199–212.

Embrechts, P. et Wüthrich, M. V. (2022). Recent challenges in actuarial science. *Annual Review of Statistics and Its Application*, *9*, 119–140.

Ferreira Jr., J. et Minikel, E. (2010). Pay-as-you-drive auto insurance in massachusetts. *A risk assessment and report on consumer, industry and environmental benefits, Commissioned by: Conservation Law Foundation & Environmental Insurance Agency*.

Friedman, J., Hastie, T. et Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, *33*(1), 1.

Gabrielli, A., Richman, R. et Wüthrich, M. V. (2020). Neural network embedding of the over-dispersed poisson reserving model. *Scandinavian Actuarial Journal*, *2020*(1), 1–29.

Gao, G., Meng, S. et Wüthrich, M. V. (2019). Claims frequency modeling using telematics car driving data. *Scandinavian Actuarial Journal*, *2019*(2), 143–162.

Gao, G., Wang, H. et Wüthrich, M. V. (2021). Boosting poisson regression models with telematics car driving data. *Machine Learning*, 1–30.

Gao, G., Wang, H. et Wüthrich, M. V. (2022). Boosting poisson regression models with telematics car driving data. *Machine Learning*, 1–30.

Gao, G. et Wüthrich, M. V. (2018). Feature extraction from telematics car driving heatmaps. *European*

Actuarial Journal, *8*(2), 383–406.

Gao, G. et Wüthrich, M. V. (2019). Convolutional neural network classification of telematics car driving data. *Risks*, *7*(1), 6.

Gao, G., Wuthrich, M. V. et Yang, H. (2018). Driving risk evaluation based on telematics data. *Available at SSRN 3288347*.

Geyer, A., Kremslehner, D. et Muermann, A. (2020). Asymmetric information in automobile insurance: Evidence from driving behavior. *Journal of Risk and Insurance*, *87*(4), 969–995.

Goodfellow, I., Bengio, Y. et Courville, A. (2016). *Deep learning*. MIT press.

Greenberg, A. (2009). Designing pay-per-mile auto insurance regulatory incentives. *Transportation research part D: transport and environment*, *14*(6), 437–445.

Guillen, M., Nielsen, J. P., Ayuso, M. et Pérez-Marín, A. M. (2019). The use of telematics devices to improve automobile insurance rates. *Risk analysis*, *39*(3), 662–672.

Guillen, M., Nielsen, J. P. et Pérez-Marín, A. M. (2021). Near-miss telematics in motor insurance. *Journal of Risk and Insurance*, *88*(3), 569–589.

Guillen, M., Nielsen, J. P., Pérez-Marín, A. M. et Elpidorou, V. (2020). Can automobile insurance telematics predict the risk of near-miss events? *North American Actuarial Journal*, *24*(1), 141–152.

Hastie, T., Qian, J. et Tay, K. (2016). An introduction to glmnet.

Hastie, T. et Tibshirani, R. (1990). Generalized additive models. *Monographs on statistics and applied probability*, *43*.

Hastie, T., Tibshirani, R. et Wainwright, M. (2015). *Statistical learning with sparsity: the lasso and generalizations*. CRC press.

Hausman, J., Hall, B. et Griliches, Z. (1984). Econometric models for count data with an application to the patents-r&d relationship. *Econometrica*, *52*, 909–938.

Howell, P. (2013). Usage-based automobile insurance pricing in ontario (2013). `https://www.fsrao.ca/industry/auto-insurance-sector/guidance/archived/usage-based-automobile-insurance-pricing-ontario-2013`.

Huang, Y. et Meng, S. (2019). Automobile insurance classification ratemaking based on telematics driving data. *Decision Support Systems*, *127*, 113156.

Kuhn, M. et Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*. CRC Press.

Laporta, A. G., Levantesi, S. et Petrella, L. (2023). Neural networks for quantile claim amount estimation: a quantile regression approach. *Annals of Actuarial Science*, 1–21.

Lemaire, J., Park, S. C. et Wang, K. (2015). The use of annual mileage as a rating variable. *ASTIN Bulletin*, *46*(1), 39.

Lemaire, J., Park, S. C. et Wang, K. C. (2016). The use of annual mileage as a rating variable. *ASTIN Bulletin*, *46*(1), 39–69.

Litman, T. (2005). Pay-as-you-drive pricing and insurance regulatory objectives. *Journal of Insurance Regulation*, *23*(3).

Litman, T. (2007). Distance-based vehicle insurance feasibility, costs and benefits. *Victoria*, *11*.

Liu, F. T., Ting, K. M. et Zhou, Z.-H. (2008). Isolation forest. Dans *2008 eighth ieee international conference on data mining*, 413–422. IEEE.

Mahalanobis, P. C. (1936). Generalized distances in statistics. *Proceedings of the National Institute of Sciences of India*, *2*, 49–55.

Meng, S., Wang, H., Shi, Y. et Gao, G. (2022). Improving automobile insurance claims frequency prediction with telematics car driving data. *ASTIN Bulletin*, *52*(2), 363–391.

Nelder, J. A. et Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, *135*(3), 370–384.

Paefgen, J., Staake, T. et Fleisch, E. (2014). Multivariate exposure modeling of accident risk: Insights from pay-as-you-drive insurance data. *Transportation Research Part A: Policy and Practice*, *61*, 27–40.

Paefgen, J., Staake, T. et 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, A. M. et 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, J., Guillen, M. et Alcañiz, M. (2019). Predicting motor insurance claims using telematics data—xgboost versus logistic regression. *Risks*, *7*(2), 70.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria

Rigby, R. A. et Stasinopoulos, D. M. (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *54*(3), 507–554.

Schelldorfer, J. et Wuthrich, M. V. (2019). Nesting classical actuarial models into neural networks. *Available at SSRN 3320525*.

Snoek, J., Larochelle, H. et Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*.

So, B., Boucher, J.-P. et Valdez, E. A. (2021). Cost-sensitive multi-class adaboost for understanding driving behavior based on telematics. *ASTIN Bulletin*, *51*(3), 719–751.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Turcotte, R. et Boucher, J.-P. (2023). Gamlss for longitudinal multivariate claim count models. *North American Actuarial Journal*, 1–24.

Tzougas, G. et Kutzkov, K. (2023). Enhancing logistic regression using neural networks for classification in actuarial learning. *Algorithms*, *16*(2), 99.

Verbelen, R., Antonio, K. et Claeskens, G. (2017). Unraveling the predictive power of telematics data in car insurance pricing. *Available at SSRN 2872112*.

Vickrey, W. (1968). Automobile accidents, tort law, externalities, and insurance: An economist's critique. *Law and Contemporary Problems*, *33*(3), 464–487.

Watson, W. T. (2017). Infographic: How ready are consumers for connected cars and usage-based car insurance? `https://www.wtwco.com/en-US/Insights/2017/05/infographic-how-ready-are-consumers-for-connected-cars-and-usage-based-car-insurance`.

Weidner, W., Transchel, F. W. et Weidner, R. (2016). Classification of scale-sensitive telematic observables for riskindividual pricing. *European Actuarial Journal*, *6*(1), 3–24.

Wikipedia contributors (n.d.). Prêt à la grosse aventure. `https://fr.wikipedia.org/wiki/Pr%C3%AAt_%C3%A0_la_grosse_aventure`. Accessed: February 26, 2024.

Wüthrich, M. V. (2017). Covariate selection from telematics car driving data. *European Actuarial Journal*, *7*(1), 89–108.

Wüthrich, M. V. et Merz, M. (2019). Yes, we cann! *ASTIN Bulletin*, *49*(1), 1–3.

Yeo, I.-K. et Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, *87*(4), 954–959.

Ziakopoulos, A., Petraki, V., Kontaxi, A. et Yannis, G. (2022). The transformation of the insurance industry and road safety by driver safety behaviour telematics. *Case studies on transport policy*, *10*(4), 2271–2279.

Zou, H. et Hastie, T. (2003). Regression shrinkage and selection via the elastic net, with applications to microarrays. *Journal of the Royal Statistical Society: Series B*, *67*, 301–20.

Zou, H. et 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.