Université de Montréal

Estimating the probability of a fleet vehicle accident: A deep learning approach using Conditional Variational Auto-Encoders

par

Marie-Ève Malette-Campeau

Département d'informatique et de recherche operationnelle
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Discipline

November 21, 2020

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Estimating the probability of a fleet vehicle accident: A deep learning approach using Conditional Variational Auto-Encoders

présenté par

Marie-Ève Malette-Campeau

a été évalué par un jury composé des personnes suivantes :

Liam Paull
(président-rapporteur)

Guillaume Rabusseau
(directeur de recherche)

Fabian Bastin
(codirecteur)

Georges Dionne
(codirecteur)

Yoshua Bengio
(membre du jury)

Abstract

Risk is the possibility of a negative or undesired outcome. In our work, we evaluate the risk of a fleet vehicle accident using the 1998 and 1999 records from the files of the Societe d'assurance automobiles du Quebec (SAAQ), where each observation in the data set corresponds to a truck carrier of merchandise, and where the number of accidents during the following year it had. For each vehicle, we have useful information such as the number and type of violations it had, as well as some of its characteristics like the number of axles or the number of cylinders. With our objective in mind, we propose a new approach using conditional variational auto-encoders (CVAE) considering two distributional assumptions, Negative Binomial and Poisson, to model the distribution of a fleet vehicle accident. Our main motivation for using a CVAE is to capture the joint distribution between the number of accidents of a fleet vehicle and the predictor variables of such accidents, and to extract latent features that help reconstruct the distribution of the number of fleet vehicle accidents. We compare the CVAE with other probabilistic methods, such as a simple MLP model that learns the distribution of the number of fleet vehicle accidents without extracting meaningful latent representations. We found that the CVAE marginally outperforms the MLP model, which suggests that a model able to learn latent features has added value over one that does not. We also compared the CVAE with another basic probabilistic model, the generalized linear model (GLM), as well as with classification models. We found that the CVAE and GLM using the Negative Binomial distribution tend to show better results. Moreover, we provide a feature engineering scheme that incorporates features related to the whole fleet in addition to individual features for each vehicle that translates into improved performances of all the models implemented in our work used to evaluate the probability of a fleet vehicle accident.

Résumé

Le risque est la possibilité d'un résultat négatif ou indésirable. Dans nos travaux, nous évaluons le risque d'accident d'un véhicule de flotte à partir des données de 1998 et 1999 fournies par la Société d'assurance automobiles du Québec (SAAQ), où chaque observation correspond à un camion transporteur de marchandises, et pour lequel le nombre d'accidents qu'il a eues l'année suivante est connue. Pour chaque véhicule, nous avons des informations telles que le nombre et le type d'infractions qu'il a eues, ainsi que certaines de ses caractéristiques comme la taille ou le nombre de cylindres. Avec notre objectif à l'esprit, nous proposons une nouvelle approche utilisant des auto-encodeurs variationnels conditionnels (CVAE) en considérant deux hypothèses de distribution, Binomiale Négative et Poisson, pour modéliser la distribution d'un accident de véhicule de flotte. Notre motivation principale pour l'utilisation d'un CVAE est de capturer la distribution conjointe entre le nombre d'accidents d'un véhicule de flotte et les variables prédictives de tels accidents, et d'extraire des caractéristiques latentes qui aident à reconstruire la distribution du nombre d'accidents de véhicules de flotte. Nous comparons ainsi la CVAE avec d'autres méthodes probabilistes, comme un modèle MLP qui apprend la distribution du nombre d'accidents de véhicules de flotte sans extraire de représentations latentes significatives. Nous avons constaté que le CVAE surpasse légèrement le modèle MLP, ce qui suggère qu'un modèle capable d'apprendre des caractéristiques latentes a une valeur ajoutée par rapport à un autre qui ne le fait pas. Nous avons également comparé le CVAE avec un autre modèle probabiliste de base, le modèle linéaire généralisé (GLM), ainsi qu'avec des modèles de classification. Nous avons constaté que le CVAE et le GLM utilisant la distribution binomiale négative ont tendance à montrer de meilleurs résultats. De plus, nous développons de nouvelles variables prédictives qui intègrent des caractéristiques liées à l'ensemble de la flotte en plus des caractéristiques individuelles pour chaque véhicule. L'utilisation de ces nouvelles variables préfictives se traduit par une amélioration des performances de tous les modèles mis en œuvre dans nos travaux utilisés pour évaluer la probabilité d'un accident de véhicule de flotte.

Contents

List of Tables

List of Figures

Acknowledgements

I give special thanks to my supervisors, Fabian Bastin, Guillaume Rabusseau and Georges Dionne, for their help, time and financial support. The completion of this project would never have been possible without them.

Introduction

Risk can be thought of as the possibility of a negative or undesired outcome. Therefore, understanding and managing risk is crucial for just about anyone: individuals, investors, companies, insurers, governments and the list goes on. They make decisions based on their conscious or unconscious predictions of the future, but they face various forms of uncertainty. From a business perspective, this is evidenced by the emergence of insurance products, of derivatives in finance and of the use of models to estimate default and credit risk to name a few. More generally, modelling the probability of an event occurrence can be applied in many areas ranging from healthcare and medicine to insurance, banking and finance.

In this work, we focus on an insurance problem using a data set of fleet vehicles owned by carrier companies and used to transport different kinds of goods from one place to another. The event occurrence we aim at modelling is a vehicle accident. Carriers need to insure each vehicle they own and they often own more than one. For that reason, insurance premiums are often offered in a bundle for an entire fleet. From an insurer's perspective, insuring a fleet represents an opportunity as it provides more information about the way the carrier manages its fleet.

In order to price adequately an insurance premium, insurers need to evaluate accurately the underlying risk of a fleet. Obviously, this risk is influenced by many factors, some of which can be directly observed in collected data like the number of violations, the types of goods carried, the size of the vehicles and so on. However, there are risk factors that are invisible to the insurer. For instance, the carrier company's management might neglect regular maintenance on its vehicles, it might overwork its drivers, or perhaps one of its drivers frequently engages in risky behaviors putting the entire fleet at risk. Of course, as an insurer, it is impossible to have a complete view on how a fleet is managed. Nevertheless, we aim at leveraging the information of the entire fleet to evaluate the risk of each vehicle, and perhaps unveil some invisible patterns of contagion within a fleet, whether positive or negative.

We propose a new probabilistic approach by adapting a conditional variational auto-encoder (CVAE) to our specific problem and by testing two distributional hypothesis. We

compare our approach to various benchmarks which can be split in two categories: classification approaches and other probabilistic methods. We demonstrate that our approach has important advantages over most of our benchmarks. Moreover, one of the biggest strength of our model is its capacity to scale to large amounts of data, which is often a hurdle with statistical methods.

To summarize, the objective of our work is to compare a new approach based on recent innovations in deep learning to other machine learning and statistical methods that can be used to estimate the probability of an accident given a set of information. Keeping that in mind, we make the following contributions:

- We provide a feature engineering scheme that improves performance by leveraging fleet information (Chapter 2 and 5).
- We define a new approach combining different ideas found in the literature. This approach offers scalability to high dimensional problems (Chapter 4).
- We test this new model's ability to improve the estimation of probability of accident occurrence (Chpater 5).
- We compare our model against several other approaches found in the literature (Chapter 5).

In this work, we first walk the reader through a review of the literature on similar insurance problems, on more general risk evaluation problems as well as machine learning and deep learning applications. We also review the literature on our new proposed approach using a conditional variational auto-encoder. Next, we give more insight on the data set used, the feature engineering scheme as well as a preliminary clustering analysis. We then present the classification and probabilistic methods implemented to tackle our problem. Lastly, we present and compare the results for each approach before concluding.

Chapter 1

Review of Literature

As highlighted previously, understanding and managing risk is crucial for insurers, and therefore being able to estimate risk is an essential part of their business. In this work, the insurance problem we tackle involves evaluating the risk of a vehicle pertaining to a fleet of having an accident. Of course, the goal of estimating the probability of an event occurrence, accidents in our case, can be tackled in many ways. In a later chapter, we propose a new probabilistic approach using a type of conditional variational auto-encoder. We compare this approach with several classification models as well as other probabilistic methods. From a literature standpoint, and given our problem and proposed solution, three main themes emerge: the insurance problem, risk evaluation in general and the use of machine learning and generative models. Therefore, in the rest of this chapter, we will review articles that relate to each of these themes and how our work compares and differentiates itself from them.

Our work stems from Dionne et al. [2005] in which they propose a parametric model to rate insurance for vehicles pertaining to a fleet. In fact, we use the same data set of carrier fleets. Moreover, in the article, the authors explain how they price premiums by taking into account the same predictor variables as we did, and present an extension of a bonus-malus[1] type automobile insurance model for single premiums, rather than bundled premiums. Another contribution they make is to model separately the fleet and vehicle effects. Our approach is different in that regard: we use engineered features from other vehicles in the same fleet to enhance the performance of our models. We make the hypothesis that if there is a fleet effect, it will translate into a better performance of the models if engineered fleet features are used. Along the same lines, Desjardins et al. [2001] provide a bonus-malus system for fleet vehicles using experience found in historical data, whereas, in our case, we only use one year of data. Furthermore, Angers et al. [2006] offers a parametric model that models event distribution for individuals in different firms. With a similar application to a different kind of fleet, taxicabs to be exact, Antonio et al. [2010] use multilevel models to

[1]Bonus-malus means *good-bad* in latin. In insurance jargon, it refers to the reward and penalty applied to premiums based on the evaluation of risk.

analyze data on claim counts. They have a similar concept to fleet effect which they call intercompany experience. Additionally, they use similarly a Poisson and a Negative Binomial distribution to model the distribution of the number of fleet vehicle accidents. Contrarily to us, they also use zero-inflated and hurdle Poisson distributions.

Moving away from fleet, but still in the insurance theme, Gabrielli [2020] estimate overdispersed Poisson models to predict claim counts. Interestingly, they combine this method with a neural network architecture used to embed features into a latent representation. This approach compares to our own as conditional variational auto-encoders first reduce the dimensionality of the input vector before estimating the distribution of the response variable.

In the category of classification tasks, Siordia et al. [2010] test machine learning models to classify automatic driving into different risk levels. They implement classification and regression trees, k-nearest neighbors, artificial neural networks as well as support vector machines with this objective. We also use tree-based and neural network models as benchmarks to the more sophisticated deep learning based probabilistic approach we propose. With regards to risk evaluation, Aleandri [2018] models lapse risk using logistic regression and classification trees. In the category of probabilistic methods, Antonio and Beirlant [2007] apply generalized linear models and provide a modification to incorporate repeated measurements, or longitudinal data, applied to actuarial problems. This article is worth mentioning as we also use generalized linear models as benchmarks in an actuarial context. However, as previously mentioned, we do not use longitudinal data. Finally, some benchmarks used in our experiments are mentioned in Antonio and Valdez [2012]. The authors provide a survey on statistical tools for risk classification in insurance.

In recent years, we have seen several articles of machine learning applications to credit risk. As an example, Chopde et al. [2012] apply decision tree techniques to credit risk analysis. The application is different from the fleet insurance problem we address, but it is similar in the sense that they estimate the risk of a rare event occurrence. The classification methods we employ as benchmarks include tree-based models, namely decision trees and random forests. Along the same lines, Lee [2012] use support vector machines combined with a dimensionality reduction technique to model enterprise credit risk. Furthermore, Addo et al. [2018] apply a classification approach, using tree-based models to predict loan defaults.

Although the research in deep learning has exploded in recent years, applications in insurance and risk evaluation in general are rare. Deep learning used in a probabilistic way to estimate the probability of an event is a field of research that has not been fully explored yet. A great strength of deep learning approaches is their ability to scale to large amounts of high-dimensional data. Despite the fact that neural networks have existed for a long time, it is only recently that we have been able to explore their potential with increasing computing power. The Multilayer Perceptrons have been present in the literature since 1961,

originally introduced in Rosenblatt [1961] for which only the output layer was trainable. It was then much improved in Rumelhart et al. [1985] several years later. Now, Goodfellow et al. [2016] is a standard reference in the field. Multilayer Perceptrons, or neural networks, are versatile models: they can be used for classification, for regression, for dimensionality reduction. They can also be used as generative models. As we present in our work, we can use an adaptation of conditional variational auto-encoders, a generative neural network based model, in a probabilistic way to estimate the distribution of a variable. More precisely, in our context, that variable is the number of accidents experienced by a given vehicle.

With that said, one important distinction between a vanilla multilayer perceptron and an auto-encoder is that the former falls into the supervised learning category whereas the latter falls into the unsupervised learning category. Auto-encoders try to reconstruct the initial input. It does so by learning the important properties of the data. Hinton and Zemel [1994] is the genesis of auto-encoders as we know them today. Other important work on auto-encoders include Vincent et al. [2010], who propose an auto-encoder to denoise noisy inputs, Hinton et al. [2011], who show that they can be used to learn features in computer vision, and Liou et al. [2014], who apply them in various applications in transportation.

Extending the idea of auto-encoders, Kingma and Welling [2013] proposed the variational auto-encoder (VAE), which is essentially an auto-encoder with a probabilistic twist. A VAE is a generative model that can be used to learn the distribution of a data set and generate new examples. A flaw of the VAE proposed by Kingma and Welling [2013] is that each of the input variables is assumed to follow the same distribution which is rarely the case in practice. Nazabal et al. [2018] present the HI-VAE that addresses this weakness by mixing different types of distribution, depending on the input variable.

With a similar purpose, Zhao et al. [2017] introduced the conditional variational auto-encoder. There motivation was to generate images by label. Taking this idea, Sohn et al. [2015] show that we can think of this problem inversely by modeling a distribution of the label given the image, or features. This article was the starting point to build our adaptation of a conditional variational auto-encoder: we want to model the distribution of a label, the number of accidents, given the information given about the vehicle. However, since our application is significantly different than the one used in Sohn et al. [2015], we had to adapt the model to our needs. We took inspiration from both Nazabal et al. [2018] and Zhao et al. [2018] who use different distributions in their implementations. More precisely, the former shows how to estimate a Poisson distribution when modelling count data and the latter shows how to use a Negative Binomial distribution for the same purpose.

Our proposed approach combines these different ideas and applies them to a fleet insurance problem while comparing other methods such as classification models, generalized linear models and multilayer perceptrons. In later chapters, we will describe in details each of those models and how they can be implemented. Moreover, we will discuss and contrast

their performances, their advantages and disadvantages while keeping in sight our insurer's perspective of evaluating as accurately as possible the probability of a vehicle of having an accident relatively to other vehicles it insures.

Chapter 2

Data

The primary goal of our work is to evaluate the risk of a vehicle accident pertaining to a fleet. From an insurer's perspective, the motivation behind is to price as accurately as possible the insurance premium. In our case, the insurer's client is a carrier company owning a fleet of vehicles. Therefore, the insurer cares to price accurately the entire fleet insurance premium. Moreover, the fact that the fleet is owned by the same company gives us additional information that can be used to evaluate the risk of each vehicle. With this in mind, in this chapter, we introduce the data set used to test all the approaches we have yet to introduce: classification models and probabilistic approaches. We also explain how we used information about other vehicles in a fleet to engineer new variables in the hope of improving the performance of our models. Moreover, we present a preliminary clustering analysis exhibiting interesting results.

2.1. Description

Our data is the same used by Dionne et al. [2005]. It comes from the files of the Société d'assurance automobiles du Québec (SAAQ) and includes 1998 and 1999 records. Each observation in the data set corresponds to a truck carrier of merchandise and there are 103,848 data points in total. Near 40%, 43,679 to be exact, are vehicles that are not part of a fleet. Since the goal of this work is to offer a machine learning perspective on the work of Dionne et al. [2005], we ignored the fleets of size 1. This leaves us with 73,328 vehicles that are part of a fleet of at least two vehicles. Overall, the data set contains 13,159 carriers. The data is aggregated from 31 December 1998 to 31 December 1999. The information that is gathered over this period of time are shown in Table 2.1, as well as the number of accidents for each vehicle. The explanatory variables, also called features, include the number of years the carrier has been in business as this might have some causality on the number of accidents. One explanation might be that a more experimented company is less at risk of accidents. It also includes the sector of activity of the carrier (some sectors might be riskier than others),

the size of the fleet as well as the number of days the vehicle is authorized to circulate over a year. Logically, the less often it circulates the less chance it has of having an accident. A proxy of risk exposure of a vehicle is the number violations it contracted over the period as it is indicative of not only the behavior of the managers of the fleet (violations for trucking) but also that of the drivers (driving violations). Moreover, we have some information about the vehicle: the type of use, the type of fuel, the number of cylinders and the number of axles.

Table 2.1. List of variables in the initial data set. Those variables are the same used in Dionne et al. [2005]. We have added the type of variable which indicates if the were treated as categorical or as numerical in our models.

Explanatory variable	Type
Number of years as carrier at 31 December 1998	Numerical
Sector of activity in 1998	Categorical Nominal
Other sector	
General public trucking	
Bulk public trucking	
Private trucking	
Short-term rental firm	
Size of fleet	Categorical Ordinal
2	
3	
4 to 5	
6 to 9	
10 to 20	
21 to 50	
More than 50	
Number of days authorized to circulate in 1997	Numerical
Number of violations of trucking standards in 1997	Numerical
For overload	
For excessive size	
For poorly secured cargo	
For failure to respect service hours	
For failure to pass mechanical inspection	
For other reasons	
Type of vehicle use	Categorical Nominal
Commercial use including transport of goods	
without C.T.Q. permit	
Transport of other than bulk goods	
Transport of bulk goods	
Type of fuel	Categorical Nominal
Diesel	
Gas	
Others	
Number of cylinders	Categorical Ordinal
1 to 5 cylinders	
6 to 7 cylinders	
8 or more than 10 cylinders	
Number of axles	Categorical Ordinal
2 axles (3,000 to 4,000 kg)	
2 axles (more than 4,000 kg)	

Explanatory variable	Type
3 axles	
4 axles	
5 axles	
6 axles or more	
Number of violations with demerit points in 1997	Numerical
For speeding	
For driving under suspension	
For running a red light	
For ignoring stop sign or traffic agent	
For not wearing a seat belt	
For other offenses	

2.2. Feature Engineering and Preprocessing

Feature engineering is a common step in machine learning modelling. In our case, one question we aim at answering is whether or not the risk of a vehicle is influenced by the risk of other fleet members. In its original state, each data point contains only the information about this specific vehicle (shown in Table 2.1). Keeping that in mind, we engineered some features based on the information we have about other vehicles in the fleet. More precisely, we created two types of new features: averages for numerical variables and proportions for categorical variables. For example, a given vehicle i may not have had any violations over the stated period, but perhaps another vehicle in the same fleet had 10 violations for instance. We assume that vehicle i is riskier than its otherwise identical counterpart vehicle j of a different fleet for which none of the other fleet vehicles had violations. We argue that contagion occurs among vehicles of the same fleet that may come from various risk factors: perhaps poor management or poor driving habits from one or more drivers employed by the carrier company.

Again, this is a way of integrating information about other vehicles in the fleet as predictors for a vehicle risk. Throughout this work, we refer to these additional features as fleet information. Hence, whenever we use the terms *with fleet information*, we refer to these additional engineered features in addition to the variables shown in Table 2.1, and whenever we use the terms *without fleet information*, we refer to the variables shown in Table 2.1 only.

Finally, in terms of preprocessing, numerical features were normalized to a mean of 0 and a standard deviation of 1 and categorical features were one-hot encoded (Harris and Harris [2015]).

Table 2.2. Silhouette score for different clustering algorithms. We fitted each clustering algorithms using 2,3,4,5 and 6 clusters and computed the silhouette score. The scores in bold are the highest and hence the number of clusters picked for each algorithm. KM stands for K-means, GMM for Gaussian mixture model and KP for K-prototype.

A. Silhouette score without fleet information			
Nb. of clusters	KM	GMM	KP
2	**0.7050**	0.6564	**0.5626**
3	0.7009	0.4902	0.5373
4	0.6866	**0.7595**	0.5318
5	0.5684	0.4520	0.2389
6	0.7003	0.6643	0.3059
B. Silhouette score with fleet information			
Nb. of clusters	KM	GMM	KP
2	0.2503	**0.5846**	0.2481
3	**0.2561**	0.2497	**0.2548**
4	0.1263	0.2438	0.1257
5	0.1305	0.1647	0.1356
6	0.1390	0.2619	0.1401

2.3. Preliminary Clustering Analysis

In the next chapters, we will be discussing various approaches to evaluate the probability of a vehicle of having an accident, namely classification and probabilistic approaches. But first, this section presents a preliminary clustering analysis to see if there is a relationship between the number of accidents and the explanatory variables that can be detected in an unsupervised manner. We hence tried three commonly used clustering algorithms: K-means (KM) originally tried by Lloyd [1982], Gaussian Mixture Model (GMM) found in McLachlan and Basford [1988] and K-prototype (KP) found in Liu et al. [2006]. Both K-means and Gaussian mixture model can only be estimated with numerical variables, whereas K-prototype is a recent generalization of the K-means algorithm that allows the inclusion of categorical variables. We also compare each clustering method without fleet information and with fleet information to see if the latter detects different patterns.

The first step in clustering is to determine the optimal number of clusters to use. We tested each algorithm with $n = [2,3,4,5,6]$ clusters and chose n that yield the highest silhouette score, keeping in mind that some scores might be close to one another and not necessarily significantly different. The silhouette score measures how similar an object is compared to its own cluster. Values can range from -1 to $+1$, with the former being the worst possible value and the latter being the best. Values around 0 tend to indicate that clusters overlap one another (Rousseeuw [1987]). Table 2.2 shows the results of this analysis. As shown in part A of Table 2.2, when fleet information is included, for both K-means and K-prototype, the optimal number of clusters is three ($n = 3$). For the Gaussian mixture model, the optimal number of clusters is two ($n = 2$). However, when fleet information is

not included, the optimal number of clusters for K-means and K-prototype is two ($n = 2$), whereas the optimal number of cluster for Gaussian mixture model is four ($n = 4$).

One might be surprised at the low number of clusters identified by the three algorithms. However, the fleet vehicle data set is quite sparse due to a high number of variables containing zeros. For instance, all the trucking and driving violations as well as the fleet-engineered violation features contain mostly zeros. Hence it is not surprising that we end up with a few large clusters. Depending on the algorithm used, there seems to be two to four groups of vehicles that resemble each other on some level.

As shown in Table 2.3, each algorithm seems to be detecting some patterns indicative of the risk of the vehicles. In part A, we first notice that both K-means and K-prototype yield exactly the same results. The categorical variables are not giving additional insight when fleet information is not included. Second, K-means and K-prototype are able to isolate high risk vehicles in the second cluster with average number of accidents as high as 0.2636, when the overall average number of accidents is 0.1420. The Gaussian mixture model is also able to isolate higher risk vehicles into three different clusters (clusters 2, 3 and 4). In part B, the results when fleet information is included contrast with the results shown in part A. First, we notice that both K-means and Gaussian mixture model seem to be isolating a low risk cluster (cluster 1) where the average number of accidents is as low as 0.0358. However, when we look deeper into which vehicles were in that cluster, we realized they all pertain to the same fleet. Hence, cluster 1 identified by both K-means and Gaussian mixture model represent the largest fleet in our data set that contains exactly 1,760 vehicles. We can explain this by the repetition of fleet information variables. For example, the average number of violations in that fleet will be repeated for 1,760 rows in the data set and the same goes for all fleet-related variables. Hence, this creates a bias in the clustering analysis. Finally, the K-prototype seems to be less affected by this bias introduced by fleet information. The first cluster contains most of the vehicles with a lower average number of accidents where as cluster 2 and 3 contain higher risk vehicles.

In Figure 2.1, we show six different figures. Figures in column (a), on the left-hand side represent the average per cluster for different variables when fleet information is not included. Figures in column (b), on the right-hand side represent the average per cluster for different variables when fleet information is included. Let us first highlight the results from the left-hand side column, when fleet information is not included.

Figures 2.1(a) for K-means and K-prototype show a clear difference in the average per cluster for all variables: number of violations for not wearing the security belt (nb_dviol_belt), for speeding (nb_dviol_vit), for crossing a red light (nb_dviol_red), for missing a stop signal (nb_dviol_stop), for carrying excess weight (nb_tviol_over) and the total number of trucking infractions (nb_tviol). Similarly for Figure 2.1(a) with the Gaussian mixture model, there is a big difference between cluster 1 and the three other

Table 2.3. Average number of accidents per cluster for different clustering algorithms. We selected the optimal number of clusters based on the silhouette score shown in Table 2.2. KM stands for K-means, GMM for Gaussian mixture model and KP for K-prototype. For each method, the first column denotes the average number of accidents in each cluster and the second column, denotes the number of vehicles in each cluster. The overall average number of accidents is 0.1420.

A. Average number of accidents and cluster size without fleet information						
Clusters	KM		GMM		KP	
1	0.1254	64,505	0.1368	70,252	0.1254	64,505
2	0.2636	8,823	0.2053	375	0.2636	8,823
3	-	-	0.2660	2,338	-	-
4	-	-	0.2782	363	-	-
B. Average number of accidents and cluster size with fleet information						
Clusters	KM		GMM		KP	
1	0.0358	1,760	0.0358	1,760	0.1304	52,916
2	0.1441	66,204	0.1446	71,568	0.1633	7,042
3	0.1514	5,364	-	-	0.1768	13,370

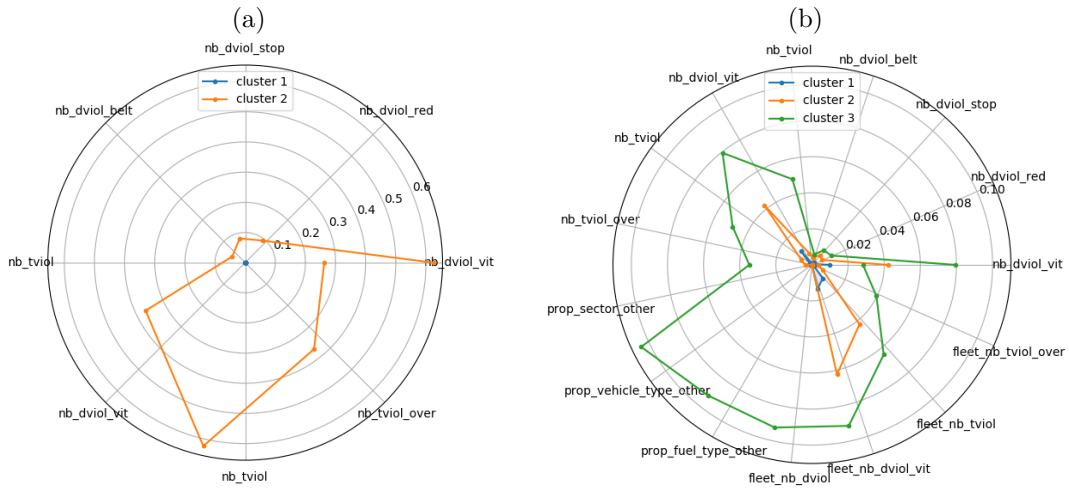
clusters. Cluster 4 is the cluster with the highest average number of accidents, the two variables that stand out the most are the number of violations for carrying excess weight and the number of infractions for speeding.

For Figures 2.1(b), as mentioned previously, there is a bias towards the largest fleet in the data set. We observe for the Gaussian mixture model, that cluster 1 has lower values across all variables. The fact that they all pertain to the same fleet does not allow us to interpret the low values as an indication of low risk. For Figures 2.1(b) for K-means and K-prototype, we note an important difference in the average number of violations in the fleet. This means that you may have a vehicle with a perfect profile (no violation), but simply because it pertains to a fleet where other vehicles have had violations, it increases its risk. This is an indication that fleet membership plays a role in a vehicle's level of risk.

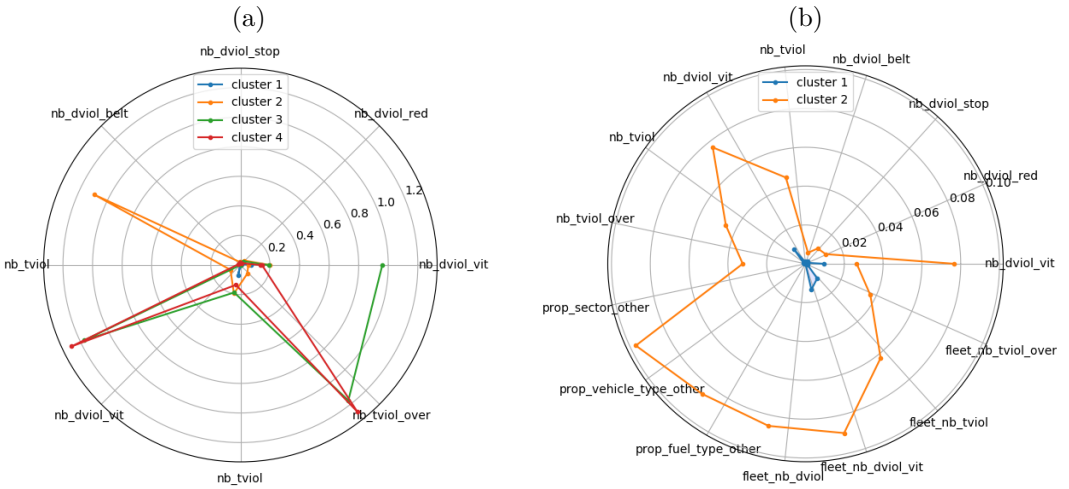
In short, we have introduced the data set that will be used to test classification models and probabilistic approaches which we will discuss in the next two chapters. We explained how we produced new features that give information about the other vehicles in the fleet. Thanks to a preliminary clustering analysis, we saw some first signs that the engineered fleet features add value in evaluating the risk level of a vehicle. More specifically, a vehicle that might seem at low risk of an accident, can be detected as higher risk due to information about the other vehicles in its fleet. In the next chapter, we will discuss in detail the classification models used as benchmarks to our adaptation of a conditional variational auto-encoder.

Figure 2.1. Explanatory variable averages per cluster. We selected the optimal number of clusters based on the silhouette score shown in Table 2.2. Figures on the right-hand side show variable averages without fleet information, whereas figures on the left-hand side show variable averages with fleet information.

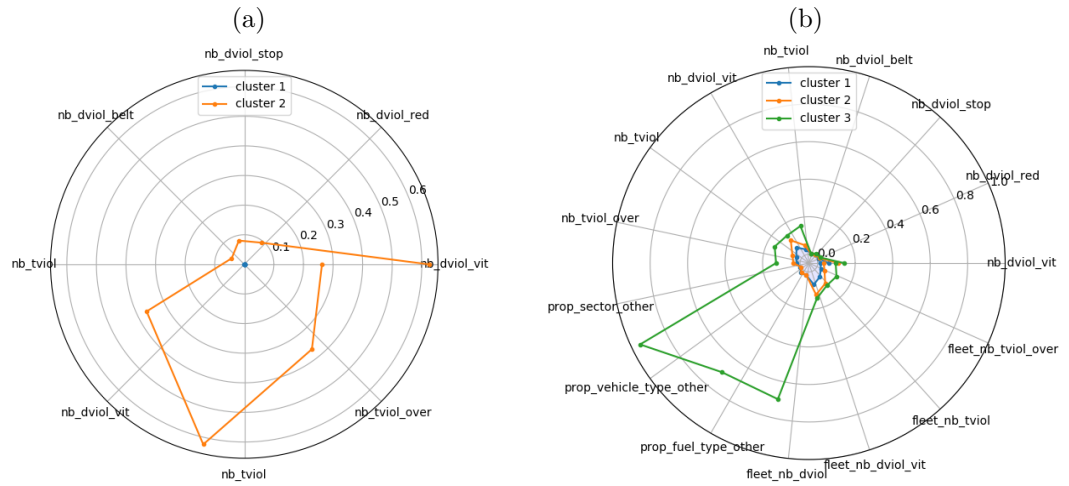
KM



GMM



KP



Chapter 3

Classification Approaches

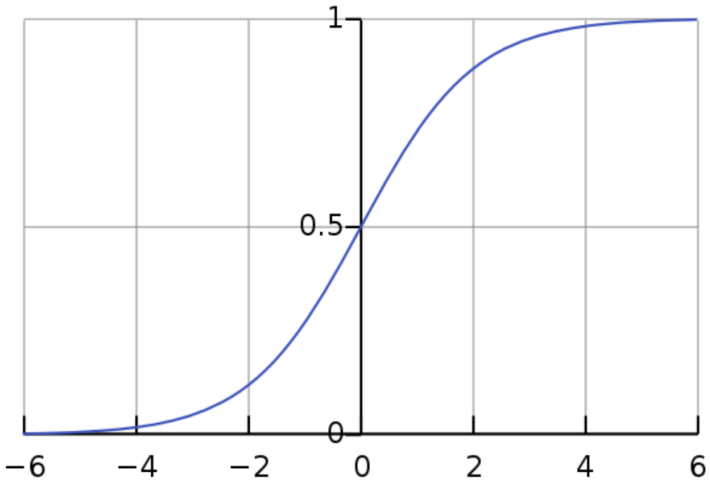
In Chapter 1, we discussed various approaches that have been used in the literature to estimate the risk of an event occurrence. Some articles discussed their problem as a classification task and used machine learning models to make predictions. In this chapter, we describe three machine learning models that we used to estimate the risk of a fleet vehicle accident. More precisely, we give an overview of the Logistic Regression, Decision Tree and Random Forest and show how they are fitted on a training set to render predictions about a new unseen observation. These models serve as benchmarks to the conditional variational auto-encoder we will present in the next chapter.

3.1. Logistic regression

In this section, we present how logistic regression can be used to model the distribution of a categorical random variable. For classification approaches, the dependent variable $\boldsymbol{y}$ was binarized. Hence, if vehicle i did not have any accident, $\boldsymbol{y}_i = 0$, however, if it had one or more accidents, $\boldsymbol{y}_i = 1$. The independent variables represented by $\boldsymbol{x}_i$ contain information about vehicle i, more precisely, the number of driving and trucking violations, properties of the vehicle and the rest of the fleet and so on. Logistic Regression has similarities to the Linear Regression: the parameters θ are estimated such that $\boldsymbol{y}_i = \sigma(\boldsymbol{x}_i^T\theta + \epsilon_i)$, where σ is the sigmoid function. Applied to binary classification, the sigmoid function illustrated in Figure 3.1 is used to compress any value between zero and one, which can be interpreted as a probability of pertaining to the positive class ($\boldsymbol{y}_i = 1$). The classification is done by setting a *threshold*, often 0.5: if the probability returned by the sigmoid function exceeds the *threshold*, then the predicted class will be positive, inversely, if it is below the *threshold*, the predicted class will be negative.

In order to understand how the parameters of a Logistic Regression model are estimated, we need to address a key concept called *log-odds*. The *log-odds* of an event occurrence, $\boldsymbol{y}_i = 1$

Figure 3.1. The sigmoid or logistic function Responsible for compressing any value between 0 and 1 and hence, enabling the classification task to be carried out. Taken from `https://en.wikipedia.org/wiki/Logistic_function`



is given by a linear combination of the independent variables $\boldsymbol{x}_i$ such that

$$\ell_i = \ln \frac{p_i}{1-p_i} = \theta_0 + \theta_1 \boldsymbol{x}_{i1} + ... + \theta_p \boldsymbol{x}_{ip}, \tag{3.1.1}$$

where the ratio $\frac{p_i}{1-p_i}$ is called the *odds*. From this, it is straightforward to isolate the probability p_i of an accident occurrence ($\boldsymbol{y}_i = 1$)

$$p_i = \frac{1}{1 + e^{-\boldsymbol{x}_i^T \theta}} \tag{3.1.2}$$

where $\theta = [\theta_0, \theta_1, ..., \theta_p]$ and hence, $\boldsymbol{x}_{i0} = 1$.

Logistic Regression can be seen as a version of Generalized Linear Model which we cover in the next chapter on probabilistic approaches. In the Logistic Regession, the conditional distribution $\boldsymbol{y}|\boldsymbol{x}$ is a Bernoulli distribution whereas, in a Linear Regression, it is assumed to be Gaussian. Re-writing Equation (3.1.2) as the conditional probability that $\boldsymbol{y}_i = 1$ given $\boldsymbol{x}_i$ as

$$p(\boldsymbol{y}_i = 1|\boldsymbol{x}_i,\theta) = \frac{1}{1 + e^{\theta^T \boldsymbol{x}_i}} \tag{3.1.3}$$

$$= f_\theta(\boldsymbol{x}_i), \tag{3.1.4}$$

can also be viewed as a function f of $\boldsymbol{x}_i$ with respect to the parameters θ. Similarly, the probability that $\boldsymbol{y}_i = 0$ given $\boldsymbol{x}_i$ can be written as

$$\begin{aligned} p(\boldsymbol{y}_i = 0|\boldsymbol{x}_i,\theta) &= 1 - p(\boldsymbol{y}_i = 1|\boldsymbol{x}_i,\theta) \\ &= 1 - f_\theta(\boldsymbol{x}_i). \end{aligned} \tag{3.1.5}$$

Using Equation (3.1.4) and (3.1.5), we can derive the likelihood function as the product of the probabilities for all observations in a data set, assuming all observations are independent,

$$\begin{aligned} L(\theta|\boldsymbol{x}) &= \prod_i p(\boldsymbol{y}_i|\boldsymbol{x}_i;\theta) \\ &= \prod_i f_\theta(\boldsymbol{x}_i)^{\boldsymbol{y}_i}(1-f_\theta(\boldsymbol{x}_i))^{1-\boldsymbol{y}_i}. \end{aligned} \tag{3.1.6}$$

Taking the logarithm of Equation (3.1.6) gives us

$$\log L(\theta|\boldsymbol{x}) = \sum_i^N \log p(\boldsymbol{y}_i|\boldsymbol{x}_i,\theta), \tag{3.1.7}$$

where N is the number of observations in the training set. The optimal parameters θ^* are then found by maximizing Equation (3.1.7)

$$\theta^* = \arg\max \log L(\theta|\boldsymbol{x}). \tag{3.1.8}$$

Once the parameters have been estimated, we can predict if a new vehicle k will have an accident with the following equation

$$\boldsymbol{y}_k = \begin{cases} 0 & \text{if } \dfrac{1}{1+e^{-x_k^T\theta^*}} \leq \rho, \\ 1 & \text{otherwise} \end{cases}$$

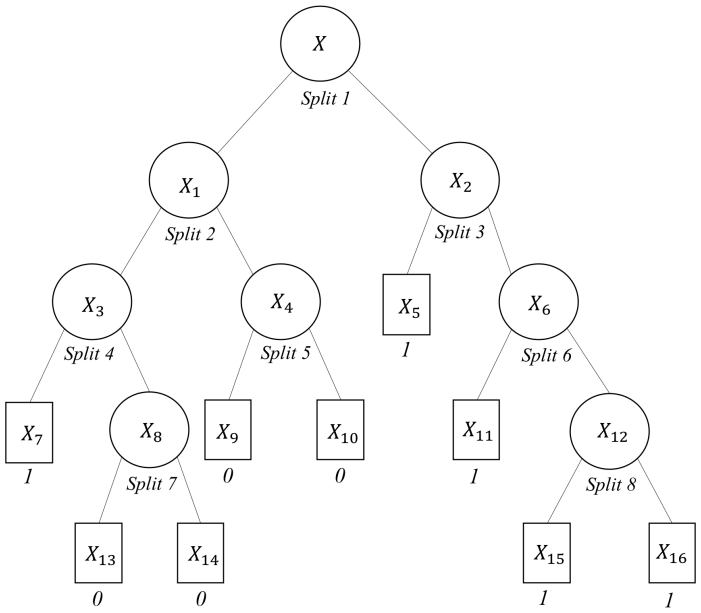
where $\rho \in [0,1]$ is a predefined *threshold*.

To recap, we just saw how a Logistic Regression model can be fitted on a training set and then used to not only predict an accident, but also provide us with a probability of having one. Next, we will discuss a tree-based approach for the same objective.

3.2. Decision Tree

Similarly to Logistic Regression, a Decision Tree can be used for regression as well as classification tasks. However, in this section, we will describe their use in the latter context. Decision Trees aim at finding the combinations of features, or independent variables, that make the response variable one class or another. Breiman et al. [1984] refers to Decision Trees as a kind of divide and conquer method. The analogy is well-suited since the approach iteratively splits the data into subsets, evaluates their purity and repeats the last two steps until the resulting subsets are sufficiently pure. To illustrate the concept of purity, let us assume that we split the fleet data set into two subsets: the first containing the vehicles that had no infraction and the second containing the vehicles that had one or more infractions. By doing so, let us imagine that all the vehicles in the resulting subsets pertain to the same class, then we would have achieved maximum purity. Therefore, the idea behind the construction of a Decision Tree is to find how to split the data set into subsets that maximize purity.

Figure 3.2. Tree Structured Classifier. X represents the data set and $X_1, ..., X_{16}$ represent subsets of X. This figure has been adapted from Breiman et al. [1984]. Circles represent non-terminal nodes where splits occur. Integers 0 and 1 below each terminal node represented by squares are the class or label associated with that node.



A Decision Tree, as its name suggests, has a tree-like structure to represent groups of hierarchical relationships between variables. The first split starts from the root node in the tree and at each split only one variable is used. The tree construction process will further split the data set into more branches. In non-techinical terms, we can think of a split as a question asked about a variable. For example, a question could be *How many infractions for speeding did the drivers of the vehicle have in the last year?* or *What type of fuel does the vehicle use?*. Such questions are answered to build the decision tree incrementally.

To support our explanations of tree construction, we follow closely Breiman et al. [1984] and use an adaptation of a figure found in their book, illustrated in Figure 3.2. It shows a hypothetical two-class tree where $X = X_1 \cup X_2$ represents the first split into two subsets and $X_2 = X_5 \cup X_6$ show a second split applied on X_2. The final subsets represented by squares are called leaves or terminal nodes, whereas circles represent non-terminal nodes. Each terminal node or leaf will have a class associated to it. In the example shown in Figure 3.2, all the data points contained in the subset X_{15} are labeled as class 1, all the data points contained in the subset $X_{13} \cup X_{10}$ are labeled as class 0 and so on.

As we mentioned earlier, each split is formed by asking a question which translates into a set of conditions on a given data point i represented by the vector $\boldsymbol{x}_i = [\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{im}]$ where m is the number of independent variables. For example, split 1 of X into X_1 and X_2

could be something like the following

$$X_1 = \{\boldsymbol{x}_i : \boldsymbol{x}_{i4} \leq 2 \quad \forall i\}, X_2 = \{\boldsymbol{x}_i : \boldsymbol{x}_{i4} > 2 \quad \forall i\},$$

This is equivalent to saying that, for a given data point $\boldsymbol{x}_i$, if the fourth variable $\boldsymbol{x}_{i4}$ is greater or equal to 2, then $\boldsymbol{x}_i$ should go in the first subset X_1, otherwise it should go in the second subset X_2.

The construction of a tree implies deciding which questions to ask, and as a result how to split the data set into subsets. The process of building a tree classifier revolves around two main elements: the split selection and the assignment of a label to each terminal node. First, a split will be selected based on the purity of the nodes. More specifically, a split has to result into subsets that are purer than its parent subset. The higher the purity, the higher the information gain.To summarize, the split selection process goes as follows:

(1) Compute the proportions $p(c|X)$ in the parent node where $c = \{0,1\}$ are the classes, and where $p(0|X) + p(1|X) = 1$.

(2) Compute a measure of impurity $I(X)$. This measure should be a nonnegative function ϕ of the proportions $p(0|X), p(1|X)$ such that $\phi(\frac{1}{2}, \frac{1}{2}) = I_{max}$ is the maximum impurity a split can get and such that $\phi(1,0) = 0$ is the minimum impurity a split can get.

(3) Choose a candidate set S of splits s, compute the impurity decrease for each candidate and choose the split s that yields the biggest impurity decrease compared to the parent node.

There are a few impurity measures found in the literature: the Gini index and the entropy measure are two common options. The Gini index is defined as

$$I_G(p) = \sum_{i=1}^{J} p_i(1 - p_i) \tag{3.2.1}$$

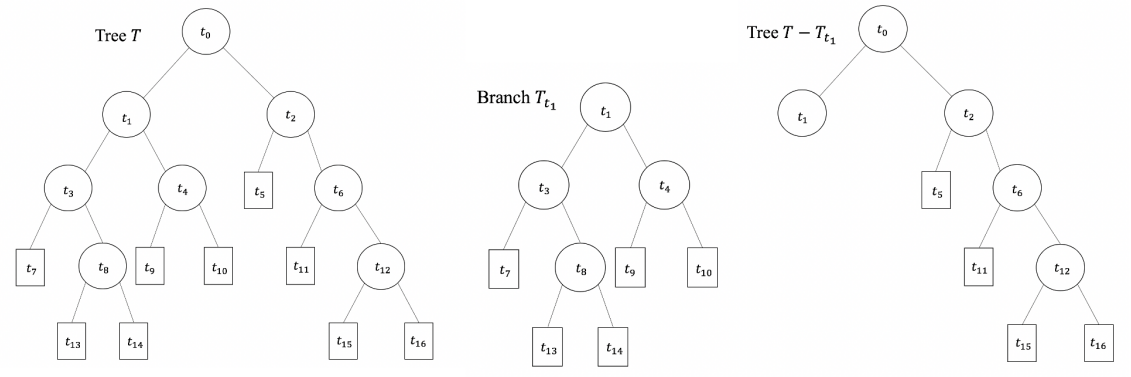
where J is the number of classes and p_i is the proportion of data points labeled as pertaining to class i in a given data set. The Gini index is, by default, used in the Python library Scikit-Learn (Pedregosa et al. [2011]) when constructing Tree-based algorithms. Another option is the entropy measure (**?**) for a tree T defined as

$$H(T) = \sum_{i=1}^{J} p_i \log_2 p_i \tag{3.2.2}$$

where identically, J represents the number of labels, and p_i represents the proportion of data points pertaining to class i in a given data set.

If we used the tree construction procedure we just explained, we could repeat the steps until we achieve 100% purity on the training set. The issue however is that it would not perform as well on unseen data. Growing the tree until 100% purity is reached is equivalent to overfitting the training set. This is why there is a second step in a Decision Tree algorithm

Figure 3.3. Pruning. The figure on the left-hand side shows the original tree, the figure in the middle shows a branch of the original tree and the the figure on the right-hand side shows the pruned tree.



called *pruning*. First, a large tree is grown, $T_{\max}$, and second, it is pruned into a smaller tree less prone to overfitting. Before we dig deeper into the details of pruning, we will introduce a few new notations. First, we denote a descendant or child node of node t as t' if there is path down the tree leading to t' starting from t. This also means that t is an ancestor of t'. We denote T_t as a branch, and all of its descendants, of a tree T with a root node $t \in T$. Pruning is hence defined as the removal of branches T_t from T. More precisely, pruning the branch T_t from T means removing all the descendants of t, except the node t from T. We refer the reader to Figure 3.3 for a visual explanation of branch removal and notation.

The resulting tree from the tree construction process contains a large number of subtrees and it would be inefficient to exhaustively go through all the possibilities in order to pick the best one. Hence, the pruning procedure needs to be computationally efficent. First, let us define the re-substitution error $R[T]$ which is the weighted sum of misclassification errors across all terminal nodes in the tree T:

$$R[T] = \sum_{t \in \text{Leaves}(T)} r(t)p(t) \tag{3.2.3}$$

$$= \sum_{t \in \text{Leaves}(T)} R[t], \tag{3.2.4}$$

where Leaves(T) denotes the terminal nodes in T, $r(t) = 1 - \max_k p(k|t)$ is the misclassification rate in node t, and lastly, $p(t)$ is the proportion of data points in node t. The re-substitution error of a parent node t is always greater or equal to the sum of the training error of its children nodes t_R and t_L

$$R[t] \geq R[t_L] + R[t_R]. \tag{3.2.5}$$

Put differently, this means that growing an always bigger tree leads to smaller errors, but as we said, it also leads to overfitting.

A common pruning method is called *minimum cost complexity pruning* and is used in R and Python libraries. This pruning algorithm is parametrized by a regularization parameter $\lambda \geq 0$ and is used in the definition of what is called the cost-complexity measure, $R_\lambda[T]$, for each internal node that defines a tree T. Essentially, this measure is the weighted sum of misclassification rates $R[T]$ define earlier in Equation (3.2.3) with the addition of a penalty based on the size of the tree:

$$R_\lambda[T] = R[T] + \lambda|\text{Leaves}(T)| \tag{3.2.6}$$

where $|\text{Leaves}(T)|$ is the number of terminal nodes, or leaves, in T. If λ is equal to zero, then the misclassification rate is smaller for the branch T_t than for its root node t

$$R_0[T_t] < R_0[t],$$

However, if we increase λ, since $R_\lambda[T_t]$ increases faster than $R_\lambda[t]$, we will reach a point where

$$R_\lambda[T_t] = R_\lambda[t]$$

for a given λ. From this point on, if we further increase λ, the inequality will reverse itself as follows

$$R_\lambda[T_t] > R_\lambda[t].$$

What we want is to solve the inequality for λ in Equation (3.2.6) such that

$$R_\lambda[T_t] < R_\lambda[t] \iff R[T_t] + \lambda|\text{Leaves}(T_t)| < R[t] + \lambda|\text{Leaves}(t)| \tag{3.2.7}$$

where $|\text{Leaves}(t)| = 1$. Hence, we have

$$g(t) = \frac{R[T_t] - R[t]}{|\text{Leaves}(t)| - 1} > \lambda, \tag{3.2.8}$$

The function $g(t)$ can be interpreted as a measure of the tradeoff between information gain and complexity: the ratio of the difference in misclassification rate and the complexity of the branch.

When pruning the tree, the node of the branch that should be removed next is called the *weakest link*. It is identified by finding the node t in the tree that has the smallest ratio defined in Equation (3.2.8). Removing the branch starting at the weakest link is equivalent to saying that the branch does not add significant value relative to the complexity it adds to the tree. The identification of the weakest link is done in two steps: first, by finding $\lambda = \min_{t \in T} g(t)$ and second by finding the $t_\lambda = \arg\min_t g(t)$.

Putting it all together, the pruning algorithm, starting with the large tree $T \leftarrow T_{max}$, can be summarized into the four following steps:

(1) Using Equation (3.2.8), find the optimal λ for a given tree T, and than with this λ, find the weakest link in the tree, node t_λ.

(2) Remove the branch starting at node t_λ from T. Now, the tree T becomes $T - T_{t_\lambda}$ ($T \leftarrow T - T_{t_\lambda}$).

(3) Repeat the last two steps until the root node t_0 is reached and keep a log of the λ values and of the resulting trees T at each iteration.

(4) The previous step will result in a sequence of parameters λ_h and a sequence of trees T_h where h denotes the iteration index. Using a validation set, the best tree T^* is the one that yields the smallest misclassification error.

Finally, given a new vehicle k, we can predict if it will have an accident or not by passing $\boldsymbol{x}_k$ through the tree until it lands in a leaf l, and

$$\boldsymbol{y}_k = \begin{cases} 0 & \text{if } \dfrac{n_0^l}{n_0^l + n_1^l} \leq \rho, \\ 1 & \text{otherwise} \end{cases}$$

where $\rho \in [0,1]$ is a predefined threshold, n_0^l and n_1^l are the numbers of training examples captured by leaf l pertaining to class 0 and 1 respectively.

To summarize, we saw that a Decision Tree is fitted to a training set by first constructing a large tree and then pruning it into a smaller one less prone to overfitting. In the next section, we will discuss a last classification model, Random Forest, in which multiple Decision Trees are aggregated.

3.3. Random Forest

Random Forests are part of the family of ensemble learning methods. Like other models presented so far, Random Forests can be used for either classification or regression, but we discuss their implementation in the former context. Previously, we talked about Decision Trees which are a key component of a Random Forest. As the name suggests, they address a task by building several Decision Trees, a forest of trees, and using the mode of the predicted class of the trees. The first Random Forest algorithm was proposed by Kam [1995]. He showed that gains in accuracy are possible if the forest is constructed on randomly selected and restricted features, thus avoiding overfitting problems. Cutler et al. [2012] later extended his idea by adding what is known as *bootstrap aggregating* to the random selection of features introduced by Kam [1995]. Nowadays, Random Forests are a trademark: packages and libraries are available in several programming languages to facilitate their implementation. In the rest of this section, we cover *bootstrap aggregating*, or *bagging* for short, and we will see how it is incorporated into a Random Forest algorithm shown in Algorithm 1.

From a statistical point of view, averaging a set of observations reduces variance. For example, if we are given a set of n independent observations $Z_1, Z_2, ..., Z_n$ with each a variance of σ^2, then the variance of the mean $\bar{Z}$ is given by σ^2/n. Applying this logic to classification with tree-based models leads to what is known as *aggregating*. The classification methods

we have so far talked about in this chapter, namely, Logistic Regression and Decision Tree, are statistical learning procedures that yield a single set of results. In other words, it maps X to y once, producing one equation or set of leafs that allow to produce a prediction. Now, we are moving towards aggregating procedures, which combine many sets of output to yield a result. That being said, X is mapped to y several times in a similar way as we saw before, but with slight variations that ensures every fit or model is different from one another. The benefits of ensemble methods is the possible reduction of overfitting due to the averaging over many fitted values. It will tend to ignore features that are less informative that would otherwise be captured by a single model. Hence, the aggregated result is more stable and robust. Because there is a high number of fitting attempts, the fitting functions will be more flexible. Overall, it improves the bias-variance tradeoff.

Aggregating produces a set of models, Decision Trees in the case of Random Forests: $\hat{f}^1(\boldsymbol{x}), \hat{f}^2(\boldsymbol{x}), ..., \hat{f}^B(\boldsymbol{x})$ using B separate training sets where no observations overlap. Then, the results are averaged, resulting in a lower variance model:

$$\hat{f}^{avg}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^b(\boldsymbol{x}) \tag{3.3.1}$$

One issue with *aggregating* is that it requires a huge amount of data to get sufficient not-overlapping training sets. A solution is to use *bootstrapping aggregating* or *bagging* for short. It is essentially the same, but it does not require separate training sets. From the initial dataset, smaller observations are sampled with replacement to create new training sets and the results of the fitted models are aggregated in the same way as in Equation (3.3.1):

$$\hat{f}^{bag}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(\boldsymbol{x}) \tag{3.3.2}$$

Random Forest goes a little bit further then *bagging*. The algorithm first takes a random sample with replacement of data of size n, as well as a random sample without replacement of the independent variables. Each split for the construction of each tree is then made using only the selected predictors, until the tree is large enough. Contrarily to a Decision Tree, a Random Forest does not prune its trees. A sequence of trees are grown until a defined stopping rule is reached: a maximum number of data points in terminal nodes or a maximum depth of the tree, for example. Lastly, a majority vote is taken on each terminal node to assign it a class. The reason why Random Forests are often more robust than other ensemble methods is because not only does it use a series of models to aggregate the results but those models are less correlated due to the randomly sampled features without replacement in the split selection process. The main effect of this is reduction in variance. Another way in which Random Forest adds value relative to a Decision Tree is that in a single tree, a few predictors might dominate the fitting process. Other perhaps less informative features, that could be useful as local features are rarely selected in a split.

In a Random Forest, the fact that only a few predictors are selected as candidates for each split gives them a high chance of playing a role several times in the tree construction, and this ultimately reduces bias.

Algorithm 1: Random Forest algorithm

Data: B hyperparameter of number of times to repeat the bagging, n hyperparameter of number of samples to select.

Result: B trees $T_1, T_2, ..., T_B$.

for $b = 1,...,B$ **do**

- Sample with replacement n training examples from X and y: $X_b, y_b \sim X, y$.
- Fit a classification tree T_b on X_b, y_b by sampling without replacement p predictor at each split.
- Assign a class to each leaf in the resulting tree by taking the majority class of the training examples that were captured by this leaf.

Lastly, given a new vehicle k, we can predict if it will have accident or not by passing $\boldsymbol{x}_k$ through each tree and

$$\boldsymbol{y}_k = \begin{cases} 0, & \text{if } \dfrac{\hat{n}_0}{\hat{n}_0 + \hat{n}_1} \leq \rho \\ 1, & \text{otherwise} \end{cases}$$

where $\rho \in [0,1]$ is a predefined threshold, $\hat{n}_0$ and $\hat{n}_1$ are the number of trees that predicted class 0 and class 1 respectively.

Like Logistic Regression and Decision Tree, Random Forest can produce a probability and predict the class a vehicle will fall into. As our goal is to estimate the risk of a fleet vehicle of having an accident, the probability produced can be estimated to differentiate high risk vehicles from low risk ones. In the next chapter, we will introduce probabilistic approaches. Instead of classifying a vehicle into two classes: class 0, will not have, and class 1, will have an accident, the methods we will describe estimate the distribution of the number of accidents. However, the goal of estimating the risk of a vehicle remains the same.

Chapter 4

Probabilistic Approaches

In Chapter 3, we described three machine learning models that we applied to estimate the risk of a fleet vehicle accident. The problem was framed as a binary classification and each model proposed renders a probability that can be used to compare and differentiate low-risk vehicles from high-risk ones. In this chapter, we describe three probabilistic approaches: a generalized linear model, an adaptation of a multilayer perceptron and of a conditional variational auto-encoder to model the probability of a vehicle accident. Contrary to the classification models described previously, this type of approach renders the parameters of a chosen distribution that are unique to each observation. This allows to estimate not only the probability of having one accident, but of having any number of accidents. In this chapter, there is a section dedicated to each approach, each subsequently divided in three subsections. We first give some background context, then we explain the theory behind the method and last we describe how it is implemented.

4.1. Generalized Linear Model

Machine learning and, in particular, deep learning models, have captured a lot of attention in recent years. They are often tried out first as a one-size-fits-all approach even before more simple standard models. Indeed, it is not rare to see sophisticated models applied even before simple models that have been designed specifically for that particular task at hand. One example of such simple but efficient approaches are Generalized Linear Models (GLM). They were originally introduced as a generalization of linear models used to approximate the probability distribution of a certain outcome under some hypotheses. In this section, we discuss very briefly GLMs and our exposition follows standard books on the subject such as Scott Long [1997]. Our discussion has been articulated around three parts. First, we give a high-level introduction to GLMs and how they are derived from classic linear regression models. Second, we discuss in a somewhat more rigorous manner the mathematical aspects of the theory behind GLMs. In particular, we discuss the optimization problem behind

the parameter estimation. Finally, we discuss how GLMs are of interest in the context of the particular problem at hand. As we discussed in the introduction, our interest lies in understanding and predicting the probability of observing a given number of accidents of a vehicle belonging to a particular fleet. In order to carry out this implementation we test and compare two assumptions: 1) the number of accidents of a fleet vehicle follows a Negative Binomial distribution and 2) the number of accidents of a fleet vehicle follows a Poisson distribution. In the insurance literature, the use of Negative Binomial and Poisson distributions to evaluate the probability of an accident is widespread.

4.1.1. Background

Linear models are applied to either identify the features $\boldsymbol{x}_i$ that can explain a response variable $\boldsymbol{y}_i$ or to make a prediction of the response variable. The classic linear model is of the form

$$\begin{aligned} \boldsymbol{y}_i &= \boldsymbol{x}_i^T \beta + \epsilon_i, \\ \epsilon_i &\sim \mathcal{N}(0, \sigma^2) \ . \end{aligned} \tag{4.1.1}$$

Here, $\boldsymbol{y}_i$ corresponds to the dependent variable for one data point i, e.g. the number of accidents of vehicle i. The feature vector $\boldsymbol{x}_i = [\boldsymbol{x}_{i0}, \boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, ..., \boldsymbol{x}_{im}]$ corresponds to the values for each variable [1,...,m] for one data point i, and $\boldsymbol{x}_{i0} = 1$ corresponds to the offset. Finally, $\beta = [\beta_0, \beta_1, \beta_2, ..., \beta_m]$ are the learnt coefficients of the regression. Equation (4.1.1) implies that we can explain a variable $\boldsymbol{y}_i$, as a weighted sum of predictor variables contained in the vector $\boldsymbol{x}_i$ and a normally distributed error term. Another way to express this relationship is through the conditional distribution $\boldsymbol{y}_i|\boldsymbol{x}_i$ for data point i

$$\boldsymbol{y}_i|\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{x}_i^T \beta, \sigma^2) \ . \tag{4.1.2}$$

Given the objective to model the probability of a vehicle of having an accident, if we used a linear model, we would implicitly be making the assumption that the number of accidents a vehicle will have over a predetermined period is normally distributed where the mean is a function of $\boldsymbol{x}_i$: $\mu(\boldsymbol{x}_i) = \boldsymbol{x}_i^T \beta$. Beyond being the standard distributions used in the literature for this type of application, another motivation for using either a Negative Binomial or a Poisson distribution is that they are suited to model count data, such as the number of vehicle accidents. Despite the fact that the linear model is not suited for our problem, we will look into some of the assumptions it makes and how they can be relaxed to achieve our goal of modelling the probability of a fleet vehicle accident.

4.1.2. Theory Behind GLMs

As mentioned, a good way to start is with the Linear Regression model, which is a simple version of GLM. We can think of a linear regression in GLM terms which can be summarized by the two following components:

(1) **The random component**: the response variable $\boldsymbol{y}_i|\boldsymbol{x}_i$ is continuous and normally distributed with mean $\boldsymbol{\mu} = \mu(\boldsymbol{x}_i) = \mathbf{E}[\boldsymbol{y}_i|\boldsymbol{x}_i]$

(2) **The link function**: it determines the relationship between the response variable and the covariates as $g(\mu(\boldsymbol{x}_i)) = \boldsymbol{x}_i^T\beta$, where $g(\mu(\boldsymbol{x}_i)) = \mu(\boldsymbol{x}_i)$.

Extending to generalized linear models by making the following modifications to the two aforementioned components:

(1) **The random component**: Instead of using a normal distribution, the response variable $\boldsymbol{y}_i|\boldsymbol{x}_i$ can follow any distribution from the exponential family.

(2) **The link function**: instead of explaining the mean as a linear regression, $\mu(\boldsymbol{x}_i) = \boldsymbol{x}_i^T\beta$, we can explain it as a function f applied to the linear regression, $\mu(\boldsymbol{x}_i) = f(\boldsymbol{x}_i^T\beta)$. Put differently, we can write $g(\mu(\boldsymbol{x}_i)) = \boldsymbol{x}_i^T\beta$ where $g = f^{-1}$. The function g depends on the choice of distribution.

The last point introduces the notion of link function which is an important concept behind GLMs. The link function g is responsible for making sure the two components are compatible. Note that, regardless of the chosen distribution and the link function, we always have $\boldsymbol{\mu}_i = E[\boldsymbol{y}_i|\boldsymbol{x}_i]$.

Given a data set, in order to fit the model, we need to estimate the parameters β. Generalized linear models are calibrated through likelihood maximization. As mentioned earlier, we assume that each component of the dependent variable $\boldsymbol{y}_i$ pertains to a distribution of the exponential family, which is of the form

$$p(\boldsymbol{y}_i; \theta_i, \phi) = \exp\{(\boldsymbol{y}_i\theta_i - b(\theta_i))/a(\phi) + c(\boldsymbol{y}_i, \phi)\}, \tag{4.1.3}$$

where the functions $a(\cdot)$, $b(\cdot)$ and $c(\cdot)$ vary depending on the chosen distribution. There are various ways in the literature to write 4.1.3; this form was taken from McCullagh and Nelder [1989]. Also note that θ_i is unique to each observation for reasons that will become apparent in the next subsection. We will see that θ_i depends on the parameters β that are not unique to each observation and that depends on the choice of distribution. Therefore, the log-likelihood should be written in terms of β as

$$\mathcal{L}(\boldsymbol{y}; \beta) = \sum_{i=1}^{N} \log p(\boldsymbol{y}_i; \theta_i, \phi), \tag{4.1.4}$$

The log-likelihood is the medium by which the parameters are estimated, but first, an assumption need to be made about the distribution of the independent variable, the number of accidents. In the next subsection, we explore two possibilities as mentioned earlier: a

Negative Binomial and Poisson distribution. We will refer to each model as NB-GLM and P-GLM for short.

4.1.3. Implementation: Evaluating the Probability of Accidents with GLM

Given the context of our problem, our objective is to design models based on the following two distributional assumptions:

(1) The number of accidents $\boldsymbol{y}_i$ can be explained by a Poisson distribution: $\boldsymbol{y}_i \sim Pois(\boldsymbol{y}_i; \boldsymbol{\lambda}_i)$.

(2) The number of accidents $\boldsymbol{y}_i$ can be explained by a Negative Binomial distribution: $\boldsymbol{y}_i \sim NB(\boldsymbol{y}_i; \alpha, \boldsymbol{\mu}_i)$.

Notice that the Poisson distribution has a parameter λ_i which varies for each data point. Likewise, the Negative Binomial above has a parameter α, which is constant for all data points, and a parameter μ_i which varies for each data point. We will examine these two models in more details. For the rest of this section, the objective is to formally define $p(\boldsymbol{y}_i; \theta_i, \phi)$ in Equation (4.1.4) so that the parameters of the models can be estimated and so that we can predict the probability of a new vehicle of having an accident.

Negative Binomial GLM

The key to fitting the NB-GLM is to write the probability mass function as a Poisson-Gamma mixture (Hilbe [2011]). This will allow us to derive a maximum likelihood estimator and provides a straightforward way of optimizing the parameters. Hence,

$$p(\boldsymbol{y} = \boldsymbol{y}_i | \mu_i, \alpha) = \frac{\Gamma(\boldsymbol{y}_i + \alpha^{-1})}{\Gamma(\boldsymbol{y}_i + 1)\Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + \mu_i} \right)^{\alpha^{-1}} \left(\frac{\mu_i}{\alpha^{-1} + \mu_i} \right)^{\boldsymbol{y}_i}, \tag{4.1.5}$$

is the conditional likelihood we use to fit the NB-GLM, where α is the heterogeneity parameter, Γ is the gamma function and μ_i is defined by the link function and the linear regression as such

$$\boldsymbol{\mu}_i = e^{\boldsymbol{x}_i^T \beta}. \tag{4.1.6}$$

Next, incorporating Equation (4.1.6) into Equation (4.1.5), we get

$$p(\boldsymbol{y} = \boldsymbol{y}_i | \boldsymbol{x}_i, \alpha) = \frac{\Gamma(\boldsymbol{y}_i + \alpha^{-1})}{\Gamma(\boldsymbol{y}_i + 1)\Gamma(\alpha^{-1})} \left(\frac{\alpha^{-1}}{\alpha^{-1} + e^{\boldsymbol{x}_i^T \beta}} \right)^{\alpha^{-1}} \left(\frac{e^{\boldsymbol{x}_i^T \beta}}{\alpha^{-1} + e^{\boldsymbol{x}_i^T \beta}} \right)^{\boldsymbol{y}_i}, \tag{4.1.7}$$

and taking the logarithm of Equation (4.1.7), we have

$$\log p(\boldsymbol{y} = \boldsymbol{y}_i|\boldsymbol{x}_i, \alpha) = \boldsymbol{y}_i \log(\alpha e^{\boldsymbol{x}_i^T \beta}) + \log \Gamma\left(\boldsymbol{y}_i + \frac{1}{\alpha}\right) - \frac{1}{\alpha}\log(1 + \alpha e^{\boldsymbol{x}_i^T \beta}) \tag{4.1.8}$$
$$- \log \Gamma(\boldsymbol{y}_i + 1) - \log \Gamma\left(\frac{1}{\alpha}\right) - \boldsymbol{y}_i \log(1 + \alpha e^{\boldsymbol{x}_i^T \beta}).$$

The optimal parameters β^* are obtained by maximizing the likelihood:

$$\beta^* \in \arg\max_{\beta} \sum_{i=1}^{N} \log p(\boldsymbol{y}_i|\boldsymbol{x}_i, \alpha). \tag{4.1.9}$$

Now, let us derive the maximum likelihood estimator when the Poisson assumption is made about the independent variable $\boldsymbol{y}_i$.

Poisson GLM

The probability mass function can be written as

$$p(\boldsymbol{y}_i|\lambda_i) = \frac{\lambda_i^{\boldsymbol{y}_i} e^{-\lambda_i}}{\boldsymbol{y}_i!} \tag{4.1.10}$$
$$= \exp\{\boldsymbol{y}_i \log(\lambda_i) - \lambda_i - \log(\boldsymbol{y}_i!)\} \tag{4.1.11}$$

Referring to Equation 4.1.3, this means that

$$\theta_i = \log \lambda_i, \quad b(\theta_i) = e^{\theta_i}, \quad \phi = 1,$$
$$a(\phi) = 1 \quad \text{and} \quad c(\boldsymbol{y}_i, \phi) = -\log \boldsymbol{y}_i.$$

More specifically, in the case of the P-GLM, the mean $\boldsymbol{\mu}_i$ of the dependent variable $\boldsymbol{y}_i$ is determined by the following linear regression,

$$\log \lambda_i = \boldsymbol{x}_i^T \beta \ . \tag{4.1.12}$$

The probability mass function in Equation (4.1.10) becomes

$$p(\boldsymbol{y}_i|\beta, \boldsymbol{x}_i) = \frac{e^{(\boldsymbol{x}_i^T \beta)\boldsymbol{y}_i} e^{-e^{(\boldsymbol{x}_i^T \beta)}}}{\boldsymbol{y}_i!}$$
$$= \exp[\boldsymbol{y}_i(\boldsymbol{x}_i^T \beta) - e^{\boldsymbol{x}_i^T \beta} - \log(\boldsymbol{y}_i!)] \ . \tag{4.1.13}$$

Combining Equation (4.1.13) and (4.1.4), the optimal parameters β^* can be obtained by maximizing the log-likelihood:

$$\beta^* \in \arg\max_{\beta} \sum_{i=1}^{N} \log p(\boldsymbol{y}_i|\boldsymbol{x}_i, \beta) \ . \tag{4.1.14}$$

To recap, once the parameters β^* have been optimized, we can predict the probability of a new unseen fleet vehicle with features $\boldsymbol{x}_k$ by computing the probability of having j

accidents as

$$p(\boldsymbol{y}_k = j|\boldsymbol{x}_k) = \frac{\Gamma(j+\alpha^{-1})}{\Gamma(j+1)\Gamma(\alpha^{-1})}\left(\frac{\alpha^{-1}}{\alpha^{-1}+e^{\boldsymbol{x}_k^T\beta^*}}\right)^{\alpha^{-1}}\left(\frac{e^{\boldsymbol{x}_k^T\beta^*}}{\alpha^{-1}+e^{\boldsymbol{x}_k^T\beta^*}}\right)^{j}, \text{ or} \tag{4.1.15}$$

$$p(\boldsymbol{y}_k = j|\boldsymbol{x}_k) = \exp\{j(\boldsymbol{x}_k^T\beta) - e^{\boldsymbol{x}_k^T\beta} - \log(j!)\} \tag{4.1.16}$$

for a Negative Binomial and Poisson distribution respectively. The two versions of GLM presented in this section are used as benchmarks for comparison with our novel deep leaning based approach, an adaptation of conditional variational auto-encoders. Before we introduce this approach, we will first discuss neural networks in general and how we adapted a multilayer perceptron to estimate the probability of a vehicle accident.

4.2. Multilayer Perceptron

In this section, we will discuss a family of models referred to as Artificial Neural Networks. Indeed, the main contribution of this masters thesis makes use of different neural networks architectures within a specific design for the problem at hand, namely estimating the distribution of accidents of a fleet vehicle as a non-parametric function of a given set of features.

In particular, we will start our exposition by discussing the so-called Multilayer Perceptron (MLP) or feedforward neural network, which can be thought of as a vanilla architecture of artificial neural networks. These models have been around for several decades now, but it is only in the early 2010's that their full potential could be expressed through the ever-increasing computer power as it reached new thresholds during the past ten years. Moreover, the discovery of new optimization tricks allowed training deeper neural networks. A thorough account on MLPs and a theoretical discussion can be found in Goodfellow et al. [2016]. We will closely follow this reference in our exposition.

This introductory discussion will allow us to introduce the more complex architectures that are at the heart of our contribution. With this objective in mind, we structure this section in three parts. First, we give a high level introduction to key concepts behind MLPs in the context of a supervised task. Second, we discuss, from a somewhat more theoretical perspective, the mathematics behind these models and how they generalize well-known classification and regression models. Finally, we will discuss how we use MLPs in our work to tackle the problem at hand: estimating the probability of a fleet vehicle accident.

4.2.1. Background

In the previous section, we discussed how GLMs can be used to estimate the probability distribution of a certain dependent variable $\boldsymbol{y}_i$ given a set of features or independent variables $\boldsymbol{x}_i$. This amounts to approximating or *learning* the relationship linking the input features to

the distribution of the dependent variable, i.e. we are learning the relationship $p(\boldsymbol{y}_i|\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$ where $\boldsymbol{x}_i$ is a given set of features. As mentioned earlier, this falls in the category of supervised tasks since the procedure implicitly requires a label data set where for each observed vector of features $\boldsymbol{x}_i$ we have an observation of the label $\boldsymbol{y}_i$. In a GLM framework we want to learn the relationship between one component of the distribution of the variable $\boldsymbol{y}_i$, namely the expectation $\mathbf{E}[\boldsymbol{y}_i]$, and a set of features $\boldsymbol{x}_i$. The object of interest in a GLM framework is the relationship

$$\mathbf{E}[\boldsymbol{y}_i|\boldsymbol{x}_i] = f(\boldsymbol{x}_i^T\beta) \ ,$$

where $\boldsymbol{x}_i^T\beta$ is a linear transformation of our original input features and f is the link function. In this section, we will discuss MLPs as generalization of this notion where we would like to allow for a compounding or stacking of layer after layer of non-linear transformations of our original inputs. This compounding or stacking can be described mathematically as composition of several functions g_j leading to the ultimate link function f.

With a slight abuse of notation, in an MLP framework the object of our attention is the relationship

$$p(\boldsymbol{y}_i|\boldsymbol{x}_i) = f(\boldsymbol{x}_i) \ ,$$

where we set f to be a composition of a number of nested link functions $g^{(j)}$ sequentially applied to a sequence of non-linear transformation of the original linearly transformed input features. This is, if we only consider one such link function, we can write

$$f(\boldsymbol{x}_i) = g(\boldsymbol{x}_i^T\mathbf{w}) \ ,$$

where $\boldsymbol{x}_i^T\mathbf{w}$ is a linear transformation of our original input features, $\mathbf{w}$ is a vector of weights and the function g is a non-linear function referred to as activation function. There are similarities with GLM, namely the linear transformation of input as well as the application of a non-linear function. Note that if we add more layers, the linear transformation becomes $\boldsymbol{x}_i^T\mathbf{W}$, where $\mathbf{W}$ is a matrix.

As an example, in the case of two stacked layers or nested link functions, we would write

$$f(\boldsymbol{x}_i) = g^{(2)}(h^{(1)}(\boldsymbol{x}_i)^T\mathbf{W}^{(2)}) \ ,$$

where $h^{(1)}(\boldsymbol{x}_i)^T\mathbf{W}^{(2)}$ is a linear transformation (through a matrix of weights $\mathbf{W}^{(2)}$) of the function h which in turn is given by

$$h^{(1)}(\boldsymbol{x}_i) = g^{(1)}(\boldsymbol{x}_i^T\mathbf{W}^{(1)}) \ ,$$

which is itself a linear transformation of the original input $\boldsymbol{x}_i$. We also remind the reader that $g^{(j)}$ is usually a non-linear function. In this recurrent fashion, one can define the function f in terms of as many as required compositions of nested activation functions applied iteratively to a series of linear transformations of the original input features. Each iteration is called a

layer of the MLP for reasons that will become apparent in the next section when we discuss in a somewhat more rigorous manner the mathematical formulation of MLPs.

In the following section, we will give a more formal definition of MLP as well as discuss how the parameters of such models can be estimated or trained. This implies defining a related optimization problem that, as it turns out, can be efficiently tackled in practice despite the nested structure of MLPs. This is the cornerstone and basis of the success of such structures in applied problems where the relationship to be learnt can be high-dimensional and complex.

4.2.2. Theory Behind MLPs

Let us define an MLP with L layers. Let us denote by $\boldsymbol{x}_i = [\boldsymbol{x}_{1i}, \boldsymbol{x}_{2i} \ldots, \boldsymbol{x}_{mi}]$ one example of an input vector consisting of m variables or features and by $\boldsymbol{y}_i = [\boldsymbol{y}_{1i}, \boldsymbol{y}_{2i} \ldots, \boldsymbol{y}_{pi}]$ one example of an output vector consisting of p responses. A data set will consist of a number of examples of such input and output vectors. But keep in mind that in the sections discussing implementation we will consider subscripts to identify examples within our data set.

For each layer $k \in 1,2,\ldots,L$, we can now define the notion of input layer activation. First notice that, using a slight abuse of notation, each layer takes in a vector of inputs $h^{(k-1)}(\boldsymbol{x}_i)$ and it outputs a vector $h^{(k)}(\boldsymbol{x}_i)$.

Definition 4.2.1. *For layer $k \in 1,2,\ldots,L$, we define the input layer pre-activation as follows*

$$a^{(k)}(\boldsymbol{x}_i) = \mathbf{b}^{(k)} + \mathbf{W}^{(k-1)} h^{(k-1)}(\boldsymbol{x}_i) \ ,$$

with $h^{(0)}(\boldsymbol{x}_i) = \boldsymbol{x}_i$. Here, $b^{(k)} = (b_1^{(k)}, b_2^{(k)}, \ldots, b_{d_{k-1}}^{(k)})$ is a vector often referred to as the bias and $\mathbf{W}^{(k)}$ is a $d_{k-1} \times d_k$ matrix containing the weights for each connection in that layer. d_k denotes the number of neurons in layer k.

Definition 4.2.2. *For layer $k \in 1,2,\ldots,L$, we define the hidden layer activation as follows*

$$h^{(k)}(\boldsymbol{x}_i) = g(a^{(k)}(\boldsymbol{x}_i)) \ ,$$

where g is a function often referred to as the activation function of that layer. That function adds non-linearity to each layer. For convenience, we suppose that it is the same function g for all layers but this can be generalized to a network with different activation functions per layer.

Notice that if we apply this recurrent definition we have a final output layer for $k = L$ of the form

$$h^{(L)}(\boldsymbol{x}_i) = o(a^{(L)}(\boldsymbol{x}_i)) = f(\boldsymbol{x}_i) \ ,$$

where o is a function often referred to as the output activation function of the whole network. The function f is the relationship that we seek to estimate, i.e. $\boldsymbol{y}_i = f(\boldsymbol{x}_i)$. This defines in a recurrent fashion a MLP with L layers where the initial data point $\boldsymbol{x}_i$ is inputted and sent

through several layers of non-linear transformations in order to produce a final response that would approximate the original response vector $\boldsymbol{y}_i$. One remark that is important to make is about the number of parameters of the neural network. Each layer has d_{k-1} bias parameters $(b_1^{(k)}, b_2^{(k)}, \ldots, b_{d_{k-1}}^{(k)})$ plus $d_{k-1} \times d_k$ weights parameters $\mathbf{W^{(k)}}$ to be estimated. That times L, the number of layers, would be the number of parameters to be estimated from a data set containing examples of $\boldsymbol{x}_i$ and $\boldsymbol{y}_i$. Now those parameters must be estimated or learnt such that the output of the network minimizes a suitable loss function. We will now discuss this in some depth.

These parameters are optimized to minimize a so-called loss function that targets a particular task. That is, we would like those parameters to be such that certain properties of the function linking input and output get replicated. For instance, in the context where the goal is to predict the response $\boldsymbol{y}_i$ given input $\boldsymbol{x}_i$, the approach would be to optimize a function that measures the error between the prediction and the observed response.

In more rigorous terms, if we let θ be the vector of parameters of the model, we would like to find the values of θ given by

$$\theta^* \in \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} l(f(\boldsymbol{x}_i; \theta), \mathbf{y_i}) + \lambda \boldsymbol{\Omega}(\theta) \ ,$$

where $l(f(\boldsymbol{x}_i; \theta))$ is a function of the estimated response relationship f. N is the number of observations in the training set. The function Ω is a suitably designed regularization function to prevent overfitting and penalizes certain values of θ in order to ensure an efficient convergence to the solution at hand. λ is a hyper-parameter which should be selected using a validation set. Notice that the loss function is a surrogate for what we truly should optimize.

But before, we will discuss the standard mechanism through which this optimization can be carried out. We will focus our discussion of the optimization problem around the method known as *stochastic gradient descent* or **SGD**. The method to find a set of values θ that minimizes the loss function can be described in algorithm 2, also found in Robbins and Monro [1951].

Algorithm 2: Stochatic Gradient Descent algorithm

Data: A data set of N training examples: $(\boldsymbol{x}_i,\boldsymbol{y}_i)$ for $i = 1,...,N$, the size of the gradient descent steps, also referred to as the learning rate α.

Result: The optimized network parameters θ^*

Generate a random initial vector $\theta = \{\mathbf{w}^{(1)}, \mathbf{b}^{(1)}, \ldots, \mathbf{w}^{(L)}, \mathbf{b}^{(L)}\}$ where L is the number of layers in the network.

for $t = 1,...,T$ **do**

- for each training example $(\boldsymbol{x}_i,\boldsymbol{y}_i)$ in that iteration, compute

$$\Delta = -\nabla_\theta l\Big((f(\boldsymbol{x}_i;\theta), \boldsymbol{y}_i)\Big) - \lambda\Omega(\theta) \ ,$$

$$\theta \longleftarrow \theta + \alpha\Delta \ .$$

This is a very general algorithm that in order to be applied in the context of a MLP would require the following

- an expression for the loss function l,
- a procedure to compute the parameter gradients $\nabla_\theta l$,
- the regularizer function Ω as well as the gradient $\nabla_\theta\Omega$,
- an initialization method.
- a learning rate α which refers to the size of the steps in the SGD algorithm.

The choice of function l and Ω depends on the task and context at hand. We will discuss this in more detail in a following section. We will also discuss the initialization method in the implementation section. What is of more particular interest in the context of MLP is the computation of the gradients. We will not develop the whole details here since this is standard literature (Goodfellow et al. [2016]). We will only point out that the function f is a nested composition of hidden activation layers. In order to compute the derivatives one has to use the chain rule a number of times in order to write out the gradient. In this case, the choice of activation function for each layer is key. Most commonly used activation functions are tractable and computations are standard. Currently available software libraries can perform these computations and automate the SGD procedure. We will not discuss these computations further. We refer the interested reader to standard books on the subject such as Goodfellow et al. [2016]. Now that we have introduced the basic concepts behind MLPs, we will further discuss how to adapt them to our problem: estimating the probability of fleet accident.

4.2.3. Implementation: Evaluating the Probability of Accidents

In the rest of the section, we discuss the implementation details necessary to evaluate the probability of accidents of our data set of fleet vehicles with a MLP. The purpose of this subsection is to formally derive the loss function needed to estimate the parameters of the networks.

Assumptions

Just like we did in the previous section using GLMs, we aim to test the two following assumptions:

(1) The number of accidents can be explained by a Negative Binomial distribution: $\boldsymbol{y}_i \sim NB(\boldsymbol{y}_i; \boldsymbol{r}_i, \boldsymbol{p}_i)$.

(2) The number of accidents can be explained by a Poisson distribution: $\boldsymbol{y}_i \sim Pois(\boldsymbol{y}_i; \boldsymbol{\lambda}_i)$.

Hence, we designed two MLP architectures that were inspired by the Conditional Variational Auto-encoder (CVAE) which we will cover in the next section. We will call the two models Poisson(P)-MLP and Negative-Binomial(NB)-MLP for short. Although implemented on CVAEs, the output functions used by Zhao et al. [2017] are still valid for MLPs. Hence, we implemented them in both models.

More specifically, the output of the model differs depending on the the assumption made about the dependent variable $\boldsymbol{y}_i$. For the NB-MLP, the number of accidents is inferred through

$$\boldsymbol{y}_i \sim NB(\boldsymbol{r}_i, \boldsymbol{p}_i) \quad \text{with} \quad \boldsymbol{r}_i = \exp\{f_{\theta^r}(\boldsymbol{x}_i)\}, \qquad \boldsymbol{p}_i = \frac{1}{1 + \exp\{f_{\theta^p}(\boldsymbol{x}_i)\}}, \tag{4.2.1}$$

where f_{θ^r} and f_{θ^p} are explicitly computed by a MLP structure. Whereas for the P-MLP, the number of accidents $\boldsymbol{y}_i$ is inferred through

$$\boldsymbol{y}_i \sim Pois(\boldsymbol{\lambda}_i) \quad \text{with} \quad \boldsymbol{\lambda}_i = \exp\{f_{\theta^\lambda}(\boldsymbol{x}_i)\}. \tag{4.2.2}$$

where, similarly, f_{θ^λ} is explicitly computed by a MLP structure. The output functions used for $\boldsymbol{r}_i$, $\boldsymbol{p}_i$ and $\boldsymbol{\lambda}_i$ are taken directly from Zhao et al. [2018]. They implement a variational auto-encoder with Negative Binomial and Poisson distributions.

Loss Function

We will now see how the assumptions made about the dependent variable $\boldsymbol{y}_i$ affect the loss function. In our case, since we are modelling a distribution, we will define the likelihood

function in the same way we did for the GLM implementation:

$$\mathcal{L}(\boldsymbol{y}, \boldsymbol{x}; \theta) = \mathbf{E}\Big[\log p_\theta(\boldsymbol{y}|\boldsymbol{x})\Big] \tag{4.2.3}$$

$$= \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i), \tag{4.2.4}$$

where θ are the parameters of the network and N is the number of training examples. If the Negative Binomial assumption is made, the conditional probability for a given data point with $\boldsymbol{y}_i$ accidents is

$$p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i) = NB(\boldsymbol{y}_i; r_i, p_i) = \binom{\boldsymbol{y}_i - \boldsymbol{r}_i - 1}{\boldsymbol{y}_i} \cdot (1 - \boldsymbol{p}_i)^{\boldsymbol{r}_i} \boldsymbol{p}_i^{\boldsymbol{y}_i} \tag{4.2.5}$$

Taking the logarithm of Equation (4.2.6), we get

$$\log p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i) = \log \binom{\boldsymbol{y}_i - \boldsymbol{r}_i - 1}{\boldsymbol{y}_i} + \boldsymbol{r}_i \log(1 - \boldsymbol{p}_i) + \boldsymbol{y}_i \log \boldsymbol{p}_i. \tag{4.2.6}$$

Notice that we are not using the same probability mass function as we did for the GLM. Indeed, there is more than one way to write the probability mass function of a Negative Binomial. Here, we used the one from Zhao et al. [2018], because they implement a similar deep learning model using a Negative Binomial distribution. This means, for a given number of accidents $\boldsymbol{y}_i$, we will obtain $\boldsymbol{r}_i$ and $\boldsymbol{p}_i$ by passing the input vector $\boldsymbol{x}_i$ through the MLP architecture and the conditional log-likelihood in Equation (4.2.6) can be computed. If the Poisson assumption is made, the conditional probability for a given data point with $\boldsymbol{y}_i$ accidents is

$$p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i) = Pois(\boldsymbol{y}_i; \lambda_i) = \frac{e^{-\lambda_i}\lambda_i^{\boldsymbol{y}_i}}{\boldsymbol{y}_i!}. \tag{4.2.7}$$

Taking the log of Equation (4.2.7), we get

$$\log p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i) = \boldsymbol{y}_i\lambda_i - \lambda_i - \log \boldsymbol{y}_i!. \tag{4.2.8}$$

Similarly to what we had with the Negative Binomial assumption, for a given number of accidents $\boldsymbol{y}_i$, the rate λ_i is obtained by passing the input $\boldsymbol{x}_i$ through the network and the conditional log-likelihood can be computed.

To summarize, once the parameters θ^* have been calibrated, we can predict the probability of a new unseen fleet vehicle with features $\boldsymbol{x}_k$ by computing the probability of having j accidents as

$$p(\boldsymbol{y}_k = j|\boldsymbol{x}_k) = \binom{j - \boldsymbol{r}_k - 1}{j} \cdot (1 - \boldsymbol{p}_i)^{\boldsymbol{r}_i} \boldsymbol{p}_i^{j}, \text{ or} \tag{4.2.9}$$

$$p(\boldsymbol{y}_k = j|\boldsymbol{x}_k) = \frac{e^{-\lambda_i}\lambda_i^{j}}{j!} \tag{4.2.10}$$

for a Negative Binomial and Poisson distribution respectively and where $\mathbf{r}_i$, $\mathbf{p}_i$ and λ_i are functions of $\boldsymbol{x}_k$ given by Equations (4.2.1) and (4.2.2). The benchmark models defined in this section will be compared to an adaptation of conditional variational auto-encoders which we present in the next section.

4.3. Conditional Variational Autoencoder

The main goal of our work is to evaluate the probability of a fleet vehicle accident given some characteristics about that vehicle. We have so far introduced two models, the Generalized Linear Model and the Multilayer Perceptron model. In this section, we introduce a type of variational auto-encoder and show how this kind of neural network can be used for the same purpose. We are mostly interested in comparing the results of the MLP and the CVAE models in order to answer the following question: can learning latent representations capturing the conditional distribution of the target given the input features be beneficial to the task at hand? With this in mind, this section is divided into three parts similar to previous sections. First, we give a high-level introduction of conditional variational auto-encoders (CVAE) and how they are derived from standard auto-encoders. Second, we lay out the mathematical details behind CVAEs, which are necessary to optimize the parameters of the network. Lastly, we will discuss how to implement a CVAE given our particular problem.

4.3.1. Background

In the previous section, we saw how MLPs are used to address supervised tasks by learning a function that links some dependent variable $\boldsymbol{y}$ to some input variables $\boldsymbol{x}$. With that said, one important distinction between an MLP and an auto-encoder is that the former falls into the supervised learning category whereas the latter falls into the unsupervised learning category. More precisely, an MLP maps input variables $\boldsymbol{x}$ to output variable $\boldsymbol{y}$. The auto-encoder however maps input variables $\boldsymbol{x}$ to itself. In other words, it tries to reconstruct the initial input $\boldsymbol{x}$ into a replica $\hat{\boldsymbol{x}}$. One might wonder what is the use for the output of this network. The purpose of the auto-encoder is to learn a meaningful low-dimensional representation of the inputs. In practice, we do not so much care about the output of the auto-encoder since it is just a copy of the original input. We care about the output only to the extent that it allows us to see how well the auto-encoder can reconstruct the input from the low-dimensional representation. This is an indication of how well the low-dimensional representation summarizes the input $\boldsymbol{x}$.

An important property of vanilla autoencoders is their bottleneck shape architecture composed of two sequential parts. The bottleneck is what enforces the network to learn a useful low-dimensional representation. The first part is called the encoder and compresses gradually the input into a latent representation $\boldsymbol{z}$. That is, each layer has a higher dimension

than the one that follows. The second part is called the decoder and expands gradually the latent representation $\boldsymbol{z}$ back to the original input dimension. In other words, it mirrors the encoder with each layer having a smaller dimension than the one that follows. This type of architecture is the key to learning the important properties of the data: those that will be most useful for reconstruction. If vanilla auto-encoders did not have a bottleneck shape architecture, the network would learn to copy over the input from one layer to the next until it reaches the output layer. It would not be forced to ignore the noise and focus on what is most important about the data. Auto-encoders are often compared to Principal Component Analysis (PCA). They are both dimensionality reduction methods, but what makes the auto-encoder a powerful tool is its ability to exploit non-linear relationships in the data, whereas PCA only exploits linear relationships. Auto-encoders implicitly model the distribution of the data.

Extending this idea, Kingma and Welling [2013] proposed the variational auto-encoder (VAE), which is essentially an auto-encoder with a probabilitic twist. Contrary to the auto-encoder however, VAEs explicitly model the distribution of the data which requires two main implementation choices. The first one is regarding the latent representations $\boldsymbol{z}$. Instead of compressing the input into a smaller dimension, the network maps the input into a parameter vector $\boldsymbol{\delta}_z$ of a multivariate distribution of choice D_z from which the latent representations are sampled: $\boldsymbol{z} \sim D_z(\boldsymbol{\delta_z})$. Across the literature, a Gaussian distribution is used for D_z with independent marginals and the parameter vector $\boldsymbol{\delta_z} = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. As a result, the latent representations $\boldsymbol{z}$ are forced to be independently normally distributed. Perhaps, better non-Gaussian latent representations could be found, but in order to yield a loss function that is sufficiently easy to optimize during training, this simplifying assumption is most often made. Also, although the latent representations are sampled independently from one another, they are indirectly related since the parameters $\boldsymbol{\delta}_z = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ are functions of the same input $\boldsymbol{x}$. As we just described, the first implementation choice is the distribution of the latent representation, chosen to be a Gaussian for convenience reasons. For the second implementation choice, we have more freedom over the distribution to choose. Ideally, we want to select something that resembles the true distribution of our data $\boldsymbol{x}$. An assumption of the standard VAE is that each of the variables in $\boldsymbol{x}$ are assumed to follow the same distribution which is the case for pixels in images, on which VAEs were first deployed. Nazabal et al. [2018] propose the HI-VAE that addresses this assumption by mixing different types of distribution depending on the input variable contained in $\boldsymbol{x}$.

Trained on a set of normal observations, auto-encoders can be used to detect outliers since the reconstruction loss will be higher in the case of an anomaly. Just like auto-encoders, VAEs can be used to detect outliers. Furthermore, and unlike auto-encoders, they can be used to generate new data points from the same distribution as the original dataset. With a similar purpose, Zhao et al. [2017] introduced conditional variational auto-encoders. Their

motivation was to generate images by label. For example, training a CVAE on the MNIST dataset would allow one to specifically generate a hand-written digit by specifying the label to the model. In other words, the model learns a conditional distribution $p(\boldsymbol{x}|\boldsymbol{y})$ where $\boldsymbol{y}$ is the label and $\boldsymbol{x}$ represents the vectorized pixels. Following this idea, Sohn et al. [2015] show that one can think of this problem inversely by modelling the distribution $p(\boldsymbol{y}|\boldsymbol{x})$. In the context of the MNIST data set, this means that the model would learn the most likely label given the pixels contained in $\boldsymbol{x}$.

In our case, we want to achieve a similar goal: estimate the probability that a vehicle will have $\boldsymbol{y}_i$ accidents given its characteristics $\boldsymbol{x}_i$. In some way, this particular CVAE is easier to implement than a standard VAE since we only need to make an implementation choice for the dependent variable $\boldsymbol{y} \sim D_{\boldsymbol{y}}(\boldsymbol{\delta}_{\boldsymbol{y}})$. We will further discuss implementation choices later in this chapter. The goal of the next section is to layout the theory behind CVAEs in order to yield a loss function we can use to optimize the network parameters.

Before we get into those details, we want to remind the underlying motivation for implementing both an MLP, as seen in the previous section, as well as a CVAE architecture. One of the key differences between those models is that the latter learns the latent representations of both $\boldsymbol{x}$ and $\boldsymbol{y}$. In other words, it implicitly learns the conditional distribution of the target given the input features, captured by the latent vector $\boldsymbol{z}$. Although the distribution of the number of fleet vehicle accidents is simple and univariate, the conditional distribution of the number of fleet vehicle accidents given the characteristics of the vehicle is more complex. Hence, one important question we aim at answering by implementing both architectures is whether or not capturing the conditional distribution of $\boldsymbol{y}$ and $\boldsymbol{x}$ through the latent representation $\boldsymbol{z}$ helps to better evaluate the probability of a vehicle of having an accident. If indeed it does, we should observe improvements in the results of the CVAE relative to those of the MLP.

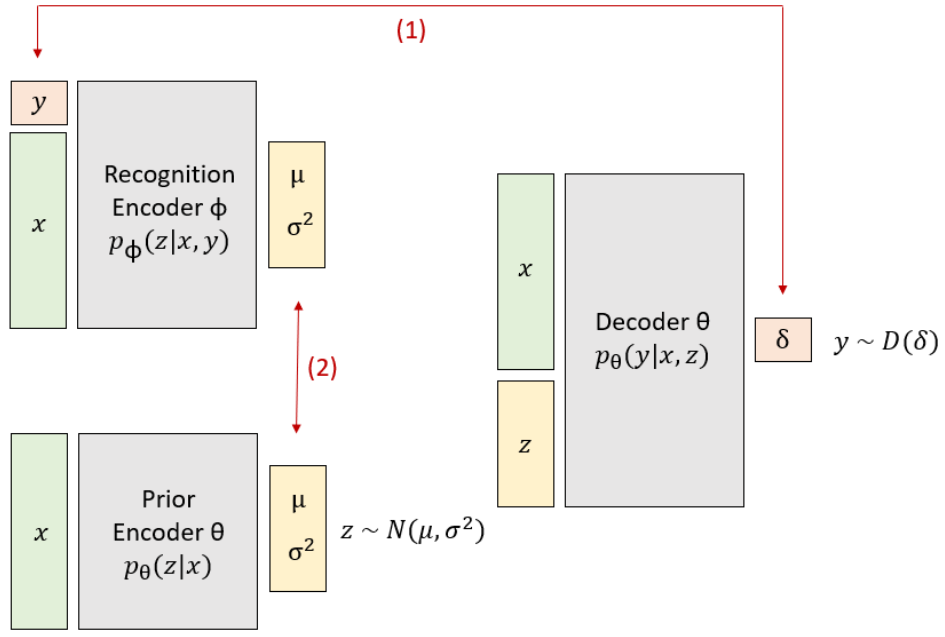
4.3.2. Theory Behind Conditional CVAE

Let us define three variables: the input variable $\boldsymbol{x}_i$, corresponding to a vector of features of vehicle i at the beginning of the year, the output variable $\boldsymbol{y}_i$, corresponding to the number of accidents vehicle i had the following year, and the latent representation $\boldsymbol{z}_i$, corresponding to the encoded features contained in $\boldsymbol{x}_i$. The conditional process of generating the dependent variable $\boldsymbol{y}_i$ is done in two sequential steps:

(1) Given the input variables $\boldsymbol{x}_i$, the latent representation $\boldsymbol{z}_i$ is first sampled from a prior conditional distribution $p_\theta(\boldsymbol{z}_i|\boldsymbol{x}_i)$.

(2) Then, given the input variables $\boldsymbol{x}_i$ and the latent representation $\boldsymbol{z}_i$, the output variable $\boldsymbol{y}_i$ is generated from the distribution $p_\theta(\boldsymbol{y}_i|\boldsymbol{z}_i, \boldsymbol{x}_i)$.

Figure 4.1. Conditional Variational Auto-encoder Architecture. Red arrows represent the two components of the loss function: (1) the log $p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z})$ term, responsible for reconstructing the output variable $\boldsymbol{y}$ as well as possible, (2) the KL divergence, acting as a regularizer by forcing the prior encoder to mimic the recognition encoder as well as possible.



The CVAE architecture is the composition of three MLPs: the recognition encoder network $q_\phi(\boldsymbol{z}_i|\boldsymbol{y}_i, \boldsymbol{x}_i)$, the prior encoder network $p_\theta(\boldsymbol{z}_i|\boldsymbol{x}_i)$ and the decoder network $p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{z}_i)$. In Sohn et al. [2015], they build the three MLPs just mentioned on top of a baseline classifier, such that an initial prediction of the output variable $\hat{\boldsymbol{y}}_i$ serves as input to the prior encoder network. In our case, we omitted this part of the model. We invite the reader to view Figure 4.1 for visual support.

The conditional variational autoencoder is trained by maximizing a lower bound of the conditional log-likelihood, which is derived as follows:

$$\log p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i) = \mathbf{E}_{\boldsymbol{z}_i\sim q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)}\Big[\log p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i)\Big] \tag{4.3.1}$$

$$= \mathbf{E}_{\boldsymbol{z}_i}\left[\log \frac{p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i,\boldsymbol{z}_i)p_\theta(\boldsymbol{z}_i|\boldsymbol{x}_i)}{p_\theta(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)}\frac{q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)}{q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)}\right] \tag{4.3.2}$$

$$= \mathbf{E}_{\boldsymbol{z}}\Big[\log p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i,\boldsymbol{z}_i)\Big] - \mathbf{E}_{\boldsymbol{z}_i}\left[\log \frac{q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)}{p_\theta(\boldsymbol{z}_i|\boldsymbol{x}_i)}\right] + \mathbf{E}_{\boldsymbol{z}_i}\left[\log \frac{q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)}{p_\theta(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)}\right] \tag{4.3.3}$$

$$= \mathbf{E}_{\boldsymbol{z}_i}\Big[\log p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i,\boldsymbol{z}_i)\Big] - \mathbf{KL}[q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)||p_\theta(\boldsymbol{z}_i|\boldsymbol{x}_i)] + \mathbf{KL}[q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)||p_\theta(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)] \tag{4.3.4}$$

$$= L(\boldsymbol{x}_i,\boldsymbol{y}_i;\theta,\phi) + \mathbf{KL}[q_\phi(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)||p_\theta(\boldsymbol{z}_i|\boldsymbol{x}_i,\boldsymbol{y}_i)]; \tag{4.3.5}$$

$$\geq L(\boldsymbol{x}_i,\boldsymbol{y}_i;\theta,\phi) \tag{4.3.6}$$

In Equation (4.3.1), we can re-write the log-likelihood as an expectation with respect to $\boldsymbol{z}_i$, an approximation is obtained by sampling $\boldsymbol{z}_i$ many times to get an average. In Equation (4.3.2), we re-write $p_\theta(\boldsymbol{y}_i|\boldsymbol{x}_i)$ using Bayes' rule and multiply by 1 to introduce the recognition encoder into the objective function. We can simply use logarithmic rules and the linearity of expectations to rearrange the terms into Equation (4.3.3). The last two terms of Equation (4.3.4) are Kullback-Leibler (KL) divergences, a measure of the distance between two distributions [1]. Lastly, the lower bound is simply the sum of the two first terms in Equation (4.3.4). Indeed, the third term, the KL divergence, is always bigger or equal to zero by definition. We derived the lower bound for a single data point i. The lower bound for a data set or mini batch of N data points is hence given by

$$L(\boldsymbol{x},\boldsymbol{y};\theta,\phi) = \frac{1}{N}\sum_{i=1}^{N} L(\boldsymbol{x}_i,\boldsymbol{y}_i;\theta,\phi). \tag{4.3.7}$$

The optimization problem, therefore becomes

$$\phi^*,\theta^* \in \arg\max L(\boldsymbol{x},\boldsymbol{y};\theta,\phi) \tag{4.3.8}$$

$$\in \arg\max \mathbf{E}_{\boldsymbol{z}}\Big[\log p_\theta(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z})\Big] - \mathbf{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})||p_\theta(\boldsymbol{z}|\boldsymbol{x})] \tag{4.3.9}$$

The first term of the objective function trains the model to reconstruct the variable $\boldsymbol{y}_i$ as well as possible. The decoder can compute an estimate of this term through sampling, which remains differentiable with the reparametrization trick discussed in Kingma and Welling

[1] For distributions P and Q of a continuous random variable, the Kullback-Leibler divergence is defined as $D_{KL}(P||Q) = \int p(x)\log\frac{p(x)}{q(x)}dx$ (see `https://en.wikipedia.org/wiki/Kullback-Leibler_divergence`).

[2013]. Second, the KL-divergence term acts as a regularizer by forcing the prior encoder to mimic the recognition encoder as well as possible.

We now have a loss function to optimize the parameters of the CVAE which will allow us to estimate the probability of fleet accidents. Next, we need to formally define the loss function according to both distributional assumptions.

4.3.3. Implementation: Evaluating the Probability of Accidents

In the rest of the section, we discuss the implementation details necessary to evaluate the probability of accidents of the fleet data set with a conditional variational auto-encoder.

Assumptions

Just like we did in the previous section using MLPs, we aim to test two assumptions:

(1) The number of accidents can be explained by a Negative Binomial distribution: $\boldsymbol{y}_i \sim NB(\boldsymbol{y}_i; \boldsymbol{r}_i, \boldsymbol{p}_i)$.

(2) The number of accidents can be explained by a Poisson distribution: $\boldsymbol{y}_i \sim Pois(\boldsymbol{y}_i; \boldsymbol{\lambda}_i)$.

Hence, we implemented two versions of the CVAE. We call the two models Poisson(P)-CVAE and Negative-Binomial(NB)-CVAE for short. First, we used the general framework from Sohn et al. [2015] to derive the loss function as we saw in the previous section. The authors use a CVAE for image recognition and therefore, convolutional layers are used in the encoder and decoder. In our case, we used standard fully-connected hidden layers. Second, we applied the output activation functions provided by Zhao et al. [2018]. More precisely, we define the recognition process for both the P-CVAE and NB-CVAE as follows:

$$\boldsymbol{z}_i \sim N(\boldsymbol{\mu}_i^P, (\boldsymbol{\sigma}_i^P)^2), \qquad \boldsymbol{\mu}_i^P = f_{\phi^{\mu^P}}(\boldsymbol{x}_i), \quad \text{with} \quad \boldsymbol{\sigma}_i^P = f_{\phi^{\sigma^P}}(\boldsymbol{x}_i), \tag{4.3.10}$$

where $\boldsymbol{\mu}_i^P$ and $\boldsymbol{\sigma}_i^P$ denote the output of the prior encoder. Similarly, $f_{\phi^{\mu^P}}(\boldsymbol{x}_i)$ and $f_{\phi^{\sigma^P}}(\boldsymbol{x}_i)$ denotes the function given by the prior encoder with parameters ϕ^P which we seek to optimize via training. Although we do not need it for the generative process just described, note that $\boldsymbol{\mu}_i^R = f_{\phi^{\mu^R}}(\boldsymbol{x}_i)$ and $\boldsymbol{\sigma}_i^R = f_{\phi^{\sigma^R}}(\boldsymbol{x}_i)$ denote the output of the recognition encoder which we will need later on.

Then, the inference process differs depending on the the assumption made about the dependent variable $\boldsymbol{y}_i$. For the NB-CVAE, the number of accidents is inferred through

$$\begin{aligned} \boldsymbol{y}_i \sim NB(\boldsymbol{r}_i, \boldsymbol{p}_i) \quad \text{with} \quad \boldsymbol{r}_i &= \exp\{f_{\theta^r}(\boldsymbol{z}_i, \boldsymbol{x}_i)\}, \\ \boldsymbol{p}_i &= \frac{1}{1 + \exp\{f_{\theta^p}(\boldsymbol{z}_i, \boldsymbol{x}_i)\}}, \end{aligned} \tag{4.3.11}$$

Whereas for the P-CVAE, the number of accidents $\boldsymbol{y}_i$ is inferred through

$$\boldsymbol{y}_i \sim Pois(\boldsymbol{\lambda}_i) \quad \text{with} \quad \boldsymbol{\lambda}_i = \exp[f_{\theta^\lambda}(\boldsymbol{z}_i, \boldsymbol{x}_i)]. \tag{4.3.12}$$

Again, the output functions for $\boldsymbol{p}_i$, $\boldsymbol{r}_i$ and $\boldsymbol{\lambda}_i$ were taken directly from Zhao et al. [2018] where they implement a variational auto-encoder with Negative Binomial and Poisson distributions.

Loss Function

We will now see how the assumptions made about the dependent variable $\boldsymbol{y}_i$ affects the loss function. Recall that the lower bound derived earlier is given by:

$$L(\boldsymbol{x};\theta,\phi) = \mathbf{E}_{\boldsymbol{z}}\Big[\log p_\theta(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z})\Big] - \mathbf{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})||p_\theta(\boldsymbol{z}|\boldsymbol{x})]. \tag{4.3.13}$$

First, let's work out the first term of the lower bound, the reconstruction loss, which is responsible for training the network to reconstruct the dependent variable $\boldsymbol{y}_i$ as well as possible. If the Negative Binomial assumption is made, the conditional prior for a given data point with $\boldsymbol{y}_i$ accidents is

$$p_\theta(\boldsymbol{y}_i|\boldsymbol{z}_i,\boldsymbol{x}_i) = NB(\boldsymbol{y}_i; r_i, p_i) = \binom{\boldsymbol{y}_i - \boldsymbol{r}_i - 1}{\boldsymbol{y}_i} \cdot (1-\boldsymbol{p}_i)^{r_i} \boldsymbol{p}_i^{\boldsymbol{y}_i} \tag{4.3.14}$$

Taking the log of Equation (4.3.14), we get

$$\log p_\theta(\boldsymbol{y}_i|\boldsymbol{z}_i,\boldsymbol{x}_i) = \log \binom{\boldsymbol{y}_i - \boldsymbol{r}_i - 1}{\boldsymbol{y}_i} + \boldsymbol{r}_i \log(1-\boldsymbol{p}_i) + \boldsymbol{y}_i \log \boldsymbol{p}_i. \tag{4.3.15}$$

This means, for a given number of accidents $\boldsymbol{y}_i$, we will obtain $\boldsymbol{r}_i$ and $\boldsymbol{p}_i$ by passing the sampled latent representation $\boldsymbol{z}_i$ through the decoder and the conditional log-likelihood above can be computed.

If the Poisson assumption is made, the conditional prior for a given data point with $\boldsymbol{y}_i$ accidents is

$$p_\theta(\boldsymbol{y}_i|\boldsymbol{z}_i,\boldsymbol{x}_i) = Pois(\boldsymbol{y}_i;\lambda_i) = \frac{e^{\lambda_i}\lambda_i^{\boldsymbol{y}_i}}{\boldsymbol{y}_i!}. \tag{4.3.16}$$

Taking the log of Equation (4.3.16), we get

$$\log p_\theta(\boldsymbol{y}_i|\boldsymbol{z}_i,\boldsymbol{x}_i) = \boldsymbol{y}_i\lambda_i - \lambda_i - \log \boldsymbol{y}_i!. \tag{4.3.17}$$

Similarly to what we had with the Negative Binomial assumption, for a given number of accidents $\boldsymbol{y}_i$, the rate λ_i is obtained by passing the sampled latent representation $\boldsymbol{z}_i$ through the decoder and the conditional log-likelihood can be computed. Thus, the reconstruction loss for a training iteration is given by

$$\mathbf{E}_{\boldsymbol{z}}\Big[\log p_\theta(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z})\Big] \approx \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\boldsymbol{y}_i|\boldsymbol{z}_i,\boldsymbol{x}_i), \tag{4.3.18}$$

where N is the number of data points in a mini-batch.

Let us now conclude with the regularizer term of the lower bound, the Kullback-Leibler divergence between two Gaussians that has the following closed-form solution:

$$\mathbf{KL}[q_\phi(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})||p_\theta(\boldsymbol{z}|\boldsymbol{x})] = \frac{1}{N}\sum_{i=1}^{N} \log \frac{\boldsymbol{\sigma}_i^P}{\boldsymbol{\sigma}_i^R} + \frac{\boldsymbol{\sigma}_i^R + (\boldsymbol{\mu}_i^R - \boldsymbol{\mu}_i^P)^2}{(\boldsymbol{\sigma}_i^P)^2} - \frac{1}{2}. \tag{4.3.19}$$

We remind the reader that $\boldsymbol{\mu}_i^P = f_{\phi^{\mu^P}}(\boldsymbol{x}_i)$ and $\boldsymbol{\sigma}_i^P = f_{\phi^{\sigma^P}}(\boldsymbol{x}_i)$ are the outputs of the prior encoder. Likewise, $\boldsymbol{\mu}_i^R = f_{\phi^{\mu^R}}(\boldsymbol{x}_i)$ and $\boldsymbol{\sigma}_i^R = f_{\phi^{\sigma^R}}(\boldsymbol{x}_i)$ are the outputs of the recognition encoder.

In summary, once the parameters ϕ^{μ^R}, ϕ^{σ^R} and θ have been optimized, we can predict the probability of a new unseen fleet vehicle with features $\boldsymbol{x}_k$ of having j accidents as

$$p(\boldsymbol{y}_k = j|\boldsymbol{x}_k) = \binom{j - \boldsymbol{r}_k - 1}{j} \cdot (1 - \boldsymbol{p}_i)^{\boldsymbol{r}_i} \boldsymbol{p}_i^j, \text{ or} \tag{4.3.20}$$

$$p(\boldsymbol{y}_k = j|\boldsymbol{x}_k) = \frac{e^{-\lambda_i} \lambda_i^j}{j!} \tag{4.3.21}$$

for a Negative Binomial and Poisson distribution respectively and where $\mathbf{r}_i$, $\mathbf{p}_i$ and λ_i are functions of $\boldsymbol{x}_k$ given by Equations (4.3.11) and (4.3.12).

In conclusion, as our goal is to estimate the risk of a fleet vehicle accident, we can compute the probabilities of having j accidents for each new observation, and differentiate low-risk vehicles for high-risk ones. In the next chapter, we will present and discuss the results of this novel approach, and compare it to the various benchmarks previously presented from classification methods (Logistic Regression, Decision Tree, Random Forest) to other probabilistic approaches (Generalized Linear Model and Multilayer Perceptron). There is a common denominator to the probabilistic approaches we have discussed in the chapter. Indeed, the parameters of a chosen distribution are estimated by some function, a linear transformation or a neural network. Then, unique parameters can be computed for an new unseen vehicle from which the probability of having an accident is determined in exactly the same way: using either a Negative Binomial or a Poisson distribution.

Chapter 5

Results

The main objective behind our work is to evaluate the risk of a vehicle accident and more specifically, the risk of a carrier vehicle pertaining to a fleet. Furthermore, we aim at validating the hypothetical relationship between a vehicle and the rest of its fleet. In other words, we make the hypothesis that there is a form of contagion, whether positive or negative, among the vehicles of the same fleet. For instance, a vehicle might appear at low risk of having an accident by looking at its characteristics, but by looking at the characteristics of other vehicles in its fleet, we might see a different picture. There could be various unobservable factors causing fleet contagion: for example, the management might neglect to do regular maintenance on the fleet vehicles, perhaps it does not respect certain safety measures or maybe one of the drivers employed by a carrier engages in risky driving behaviors affecting the entire fleet. In short, there might be factors of risk we cannot observe by simply looking at the features of one vehicle, but that can be unveiled by looking at the fleet as a whole.

As we have seen in Chapter 3 and Chapter 4, there are many ways by which we can tackle this problem, and the ones we presented are by no means an exhaustive list of all the methods we could test. The core contribution of our work lies in a deep learning approach previously presented. We aim at applying a recent innovation in deep learning, namely the Conditional Variational Autoencoder (CVAE) with a Negative Binomial or with a Poisson distribution applied to accident risk evaluation. CVAEs with either of these distributions are not present in the literature yet, far less applied to this context. All other methods we implemented and presented serve as benchmarks to the CVAE models. In the classification category, we implemented a Logistic Regression, a Decision Tree and a Random Forest model. In the probabilistic category, we implemented a Generalized Linear Model (GLM) and a Multilayer Perceptron (MLP) with both distributions.

In this chapter, we present the results in three steps. First, we will compare the results obtained with probabilistic models (Section 5.1). Second, we will do the same for the classification models (Section 5.2). Third, we compare the two category of approaches among each

other (Section 5.3). To compare the probabilistic models, we will look at the likelihood and the average predicted probabilities per true number of accidents. As for the comparison of classification models, we used the common evaluation metrics (e.g. accuracy, F1 scores), as well as the average predicted probabilities of pertaining to the negative class and aggregated by the true number of accidents. It is however impossible to compare both approaches, probabilistic and classification, using the likelihood. Hence, we converted the probabilistic results into a binary classification framework that allows us to compare all the models together. We will present those results in the last section of this chapter.

5.1. Comparing Probabilistic Models

As described in Chapter 4, we implemented three different probabilistic models with two distributional hypothesis: the response variable $\boldsymbol{y}$ follows a Poisson distribution or a Negative Binomial distribution. There are hence two versions of each model we tested: Generalized Linear Models (NB-GLM and P-GLM), Multilayer Perceptron (NB-MLP and P-MLP) and Conditional Variational Autoencoder (NB-CVAE and P-CVAE). For this category of model, the response variable $\boldsymbol{y}$ was not binarized as we did for classification models. Here, we model the number of accidents rather than whether or not there will be an accident. In this section, we present the results of our experiments designed to answer the following questions:

(1) Which distribution is most appropriate: Negative Binomial or Poisson?
(2) Which model performs the best?
(3) Does fleet information improve the accident risk evaluation?
(4) How important violations are in evaluating the accident risk of a given vehicle?

Throughout this section, the results were obtained on a test set, representing 15% of the entire data set (10,999 vehicles). Moreover, we refer to various experiments which we broke down as follows:

- Experiment A: models are implemented without fleet information, more precisely, using the variables found in Table 2.1 excluding engineered fleet features described in Chapter 2,
- Experiment B: models are implemented with fleet information, in combination with variables found in Table 2.1 from Chapter 2,
- Experiment C, models are implemented using a subset of variables, namely the driving violations for a given vehicle,
- Experiment D, models are implemented using a subset of variables, namely the trucking violations for a given vehicle, and lastly,
- Experiment E, models are implemented using a subset of variables, namely both driving violations and trucking violations for a given vehicle.

Before we present and discuss the results, let us mention a few implementation details with regards to the models. As described in Chapter 2, numerical variables were normalized and categorical variables were one-hot encoded. For the GLM, there is no hyperparameter tuning to be done and the Python library Statsmodels (Seabold and Perktold [2010]) was used for the implementation. For the MLP, we chose the architecture of the model to have approximately the same number of parameters as the CVAE, and the other hyperparameters were tuned using a randomized search approach. The learning rate used is 1e-5, the batch size is 32, and the architecture has 5 hidden layers of 60, 60, 30, 30 and 10 neurons per layer respectively[1]. We also used a dropout rate of 0.3 and a weight decay of 0.01. We used Adam (Kingma and Ba) optimizer and the implementation was done in Pytorch. For the CVAE, we used the architecture proposed in Sohn et al. [2015] where the encoder and decoder have 2 layers each of 60 and 30 neurons respectively, and the dimension of the latent vector is 10. The prior MLP Classifier had 2 layers of a 100 neurons each. The hyperparameters were also tuned using a randomized search approach that resulted in using a batch size of 256 and a learning rate of 1e-3. We also used a dropout rate of 0.3 and weight decay of 0.001. The optimizer used was also Adam and the implementation was done in Pytorch (Paszke et al. [2019]).

The first question we aim at answering relates to which distribution, Poisson or Negative Binomial, is a better choice given the data set of fleet vehicles. In Table 5.1, we present the likelihood obtained for each model. To facilitate the comparisons, the models are ranked from the highest likelihood to the lowest for each experiment in part 1, and for each type of model in part 2. From part 1 of Table 5.1, we note that for each experiment (A to E), the Negative Binomial models are always the first listed; the NB-CVAE and the NB-GLM are among the highest. It suggests that the Negative Binomial distribution is a better choice for probabilistic models.

The second question we aim at answering relates to which model performs best. We have already mentioned from answering the first question that the NB-CVAE and the NB-GLM appear to be the top performers regardless of the experiment. More specifically, from part 1 of Table 5.1, when fleet information is not used (Experiment A), the NB-CVAE has a higher likelihood than the NB-GLM. However, when fleet information is included (Experiment B), it is the other way around. Moreover, the NB-GLM yields a higher likelihood for Experiment C, D, when only a subset of features such as driving violations, trucking violation. However, the NB-CVAE renders a higher likelihood when both types of violations are used (Experiment E). To summarize, there is not a single model that significantly dominates the others. They all more or less perform the same with a seemingly marginal superiority of the NB-GLM

[1]Preliminary experiments suggested that hyperparameters relative to the architecture of the models had little impact on the validation error. Hence, the same architecture as proposed in Sohn et al. [2015] was used for the CVAE model, and an architecture with similar number of parameters was chosen for the MLP model

and the NB-CVAE. Finally, the results shown in Table 5.1 are the first indication that the CVAE adds value over the MLP, which suggests that the ability of the CVAE to learn the joint distribution between the target variable and the predictor variables results in better overall results.

Table 5.1. Likelihood Comparison for Models and Experiments. Likelihoods are multiplied by 100. Part 1 compares the models for each experiment (A. without fleet information, B. with fleet information, C. Driving violations only, D. Trucking violations only and E. All violations only). Part 2 compares the experiments for each model.

1. Model Comparison			2. Experiment Comparison		
Model	likelihood	Exp	Model	likelihood	Exp
NB-CVAE	65.51	A	NB-GLM	65.68	B
NB-GLM	64.31	A	NB-GLM	64.60	D
P-MLP	64.29	A	NB-GLM	64.60	C
P-GLM	63.90	A	NB-GLM	64.32	E
P-CVAE	63.32	A	NB-GLM	64.31	A
NB-MLP	63.44	A	P-GLM	65.51	B
NB-GLM	65.68	B	P-GLM	63.91	E
NB-CVAE	65.52	B	P-GLM	63.91	C
P-GLM	65.51	B	P-GLM	63.90	D
P-MLP	64.46	B	P-GLM	63.90	A
NB-MLP	64.38	B	NB-CVAE	65.52	B
P-CVAE	63.55	B	NB-CVAE	65.51	A
NB-GLM	64.60	C	NB-CVAE	64.59	E
NB-CVAE	64.46	C	NB-CVAE	64.49	C
NB-MLP	63.95	C	NB-CVAE	64.41	D
P-GLM	63.91	C	P-MLP	64.46	B
P-CVAE	63.50	C	P-MLP	64.29	A
P-MLP	63.51	C	P-MLP	63.83	E
NB-GLM	64.60	D	P-MLP	63.56	D
NB-CVAE	64.41	D	P-MLP	63.51	C
P-GLM	63.90	D	NB-MLP	64.38	B
NB-MLP	63.89	D	NB-MLP	64.22	E
P-MLP	63.56	D	NB-MLP	63.95	C
P-CVAE	63.36	D	NB-MLP	63.89	D
NB-CVAE	64.59	E	NB-MLP	63.44	A
NB-GLM	64.32	E	P-CVAE	63.32	B
NB-MLP	64.22	E	P-CVAE	63.25	A
P-GLM	63.91	E	P-CVAE	63.21	E
P-MLP	63.83	E	P-CVAE	63.20	D
P-CVAE	63.74	E	P-CVAE	63.19	C

For the third question, *Does fleet information improve performance of the models?*, by looking at part 2 of Table 5.1, we notice that Experiment B is always the top listed across all models. Hence, including the fleet engineered features always leads to higher likelihood regardless of the model used. Although the likelihoods are higher with fleet information, the differences are overall marginal. It suggests that a fleet effect is present, but that other variables are more important in evaluating the risk level. This is also revealed with experiment E, where only the violations are used as predictor variables, and yield lower but similar

likelihoods as when all the features are included. As for the fourth question, we wondered if the driving violations, indicative of the drivers behavior, is a better predictor of future accidents than the trucking violations, indicative of the fleet's management. The purpose of Experiment C, D and E is to answer that question. The results are quite inconclusive. Indeed, we see that the likelihood for Experiment C, D and E are rarely significantly different. Additionally, we notice the exception of the NB-MLP and the P-CVAE, for which including all violations yields a greater likelihood, followed by including only the driving violations and last, by including only the trucking violations. It seems like there is a stronger relationship between accidents and driving violations. We will later compare the probabilities obtained with each model and for each experiment.

In Table 5.2, we compare the estimated number of accidents for each model with the empirical average of 14.71% for the test set. More precisely, for the 10,999 vehicle in the test set, 87.41% of the vehicles had 0 accident, 10.76% had 1 accident, 1.58% had 2 accidents, 0.23% had 3 accidents and 0.03% had 4 accidents. The estimated average number of accidents for each model is given by

$$E[\boldsymbol{y}] = \frac{1}{N}\sum_{i=1}^{N} E[\boldsymbol{y}_i], \tag{5.1.1}$$

where $E[\boldsymbol{y}_i] = \sum_{k=1}^{4} p(\boldsymbol{y}_i = k|\boldsymbol{x}_i) \cdot k$.

The models that render an estimated average number of accidents that is closest to the empirical average are the NB-CVAE, P-GLM and NB-GLM regardless of the features used. These models underestimates slightly the empirical average. The other models overestimated the empirical average starting with the NB-MLP, followed by the P-MLP, and ending with the P-CVAE that shows the largest difference. These results are consistent with the results previously highlighted. The NB-CVAE appears better at modelling the true distribution, followed closely by the GLMs. It is however more apparent which models do worst, namely the P-CVAE and P-MLP. By comparing experiments for each type of model, we observe that for the NB-GLM, the estimated average is constant regardless of the features used. In contrast, for the NB-CVAE, we observe an improvement when fleet information is included (Experiment B) and when trucking violations only are used (Experiment C). Overall, Table 5.2 does not allow us to conclude which model is better, but it gives us some insight on their ability of modelling the empirical distribution.

We will now compare the probabilistic models in a different way, using the average predicted probabilities of having $\boldsymbol{y}$ accidents given X per class found in Tables 5.3 and 5.4. These average probabilities are obtained on the test set after the models have been trained. The probabilities for each data point are obtained from each model and then gathered and averaged by the true number of accidents. From a good model, we should expect to see a higher probability of having 0 accident in the 0-accident group than in the 1-accident,

Table 5.2. Estimated Average Number of Accidents compared to the Empirical Average. For each model, the estimated average number of accidents is given by Equation (5.1.1). Experiments A to E use different subsets of feature: A. without fleet information, B. with fleet information, C. Driving violations only, D. Trucking violations only and E. All violations only. Results are multiplied by 100.

	Experiment				
	A	B	C	D	E
NB-GLM	14.12	13.79	14.13	14.13	14.12
NB-CVAE	14.11	14.53	14.46	14.18	14.22
NB-MLP	17.00	16.22	16.34	16.79	16.02
P-GLM	14.23	13.90	14.10	14.03	13.90
P-CVAE	15.96	16.10	28.13	17.20	15.56
P-MLP	17.35	17.16	17.42	17.37	16.06
Empirical			**14.71**		

2-accident and so on. In contrast, we should expect a lower probability of having 1 accident in the 0-accident group than in other groups.

Table 5.3. Average probabilities of having y accidents given X per class for Poisson Models. For both GLM and MLP, the probabilities are inferred by passing each observation through the model that returns the parameters of the distribution. For the CVAE, each observation is passed 100 times through the model which returns the parameters of the distribution. Probabilities are then aggregated (averaged) by the true number of accidents.

1. Poisson GLM					
A. Probabilities without fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	86.82	12.26	0.87	0.04	0.00
1	86.83	12.25	0.87	0.04	0.00
2	86.75	12.32	0.88	0.04	0.00
3 or more	86.52	12.53	0.91	0.04	0.00
B. Probabilities with fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	87.82	11.18	0.93	0.06	0.00
1	83.94	14.21	1.60	0.17	0.02
2	81.76	15.70	2.07	0.33	0.08
3 or more	80.89	16.50	2.24	0.27	0.03
2. Poisson MLP					
A. Probabilities without fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	84.49	14.05	1.33	0.10	0.00
1	82.79	15.36	1.67	0.15	0.01
2	82.59	15.53	1.71	0.16	0.01
3 or more	81.64	16.41	1.79	0.14	0.00
B. Probabilities with fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	84.73	13.86	1.30	0.10	0.00
1	83.05	15.14	1.63	0.16	0.01
2	81.72	16.16	1.90	0.19	0.02
3 or more	80.66	16.83	2.22	0.26	0.03
3. Poisson CVAE					
A. Probabilities without fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	85.55	13.13	1.19	0.09	0.00
1	85.50	13.12	1.24	0.10	0.00
2	85.46	13.16	1.25	0.10	0.00
3 or more	85.18	13.34	1.32	0.12	0.00
B. Probabilities with fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	85.58	13.16	1.16	0.08	0.00
1	85.53	13.18	1.19	0.08	0.00
2	85.58	13.14	1.18	0.08	0.00
3 or more	85.64	13.11	1.15	0.08	0.00

Table 5.4. Average probabilities of having y accidents given X per class for Negative Binomial Models. For both GLM and MLP, the probabilities are inferred by passing each observation through the model that returns the parameters of the distribution. For the CVAE, each observation is passed 100 times through the model which returns the parameters of the distribution. Probabilities are then aggregated (averaged) by the true number of accidents.

1. Negative Binomial GLM					
A. Probabilities without fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	87.62	10.84	1.35	0.16	0.02
1	87.62	10.84	1.35	0.17	0.02
2	87.55	10.89	1.36	0.17	0.02
3	87.34	11.05	1.40	0.18	0.02
B. Probabilities with fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	88.64	9.76	1.33	0.21	0.04
1	85.26	12.06	2.07	0.43	0.11
2	83.40	13.15	2.49	0.59	0.18
3	82.65	13.71	2.71	0.64	0.18
2. Negative Binomial MLP					
A. Probabilities without fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	86.27	9.96	2.28	0.68	0.27
1	84.60	11.45	2.58	0.74	0.27
2	85.09	11.15	2.47	0.70	0.25
3	84.66	11.88	2.48	0.62	0.19
B. Probabilities with fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	87.61	9.04	2.28	0.69	0.22
1	85.24	10.82	2.74	0.80	026
2	84.79	11.22	2.82	0.80	0.24
3	85.46	10.70	2.68	0.78	0.25
3. Negative Binomial CVAE					
A. Probabilities without fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	88.48	9.78	1.45	0.23	0.04
1	85.13	12.34	2.06	0.36	0.06
2	84.15	12.98	2.31	0.44	0.09
3	84.28	12.98	2.25	0.39	0.07
B. Probabilities with fleet information					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	88.44	9.67	1.54	0.54	0.14
1	84.23	12.84	2.28	0.90	0.27
2	82.05	14.32	2.42	1.14	0.41
3	81.62	12.75	2.39	1.39	0.59

First, the Poisson GLM's results found in Table 5.3 are noticeably poor when fleet information is not used compared to when it is used. This is evidenced by the average probability per group of accidents that do not show a significant difference from one group to the next. From the 0-accident group to the 3-accident group, there is not a big difference in the probability of having zero accidents $P(y = 0|X)$. Indeed, we would expect a decrease in that probability. Similarly for the other probabilities, $P(y > 0|X)$, we would expect a significant increase as we move down the table. In contrast, when fleet information is incorporated as predictor variables, there is a pronounced difference in the average predicted probabilities from one group to the other. For instance, the 0-accident group's average is 87.82% whereas the 1-accident group's average is 83.94%, indicating that the model is somewhat able to differentiate the risk levels of these two groups. For the same reasons, the Poisson MLP seems better than the Poisson GLM when fleet information is not included. Moreover, it also seems like there is a small improvement in the performance of these models when fleet information is included. This is evidenced by a higher probability of having no accident for the 0-accident group and a lower probability of having no accident for the 3-accidents group when fleet information is included. In light of the results shown in Table 5.3, it seems fleet information has a significant impact on the model's ability to differentiate levels of risk. However, the Poisson distribution does not seem an adequate choice as evidenced by the CVAE results.

Now, looking at Table 5.4 that displays the results using a Negative Binomial distribution, we can identify similarities and differences compared to the Poisson results previously highlighted. First, the Negative Binomial GLM's results are noticeably poor when fleet information is not used compared to when it is used. This is again evidenced by the average probability per group of accidents that do not show a significant difference from one group to the next. Similarly, from the 0-accident group to the 4-accident group, there is not a big difference in the probability of having zero accidents $P(y = 0|X)$. In contrast, when fleet information is incorporated as predictor variables, there is a pronounced difference in the average predicted probabilities from one group to the other. For instance, the 0-accident group's average is 88.64% whereas the 1-accident group's average is 85.26%, indicating that the model is somewhat able to differentiate the risk levels of these two groups. The Negative Binomial MLP seems better than the Negative Binomial GLM when fleet information is not included. Moreover, it also seems like there is a small improvement when fleet information is included. Although it is not possible to draw any formal conclusion from this table, it appears like the NB-GLM is doing better than the NB-MLP for experiment B, as evidenced by more pronounced differences between groups. In part 3, the NB-CVAE's results appear better than the P-CVAE's results shown earlier. For experiment A, NB-CVAE seems better than NB-GLM, but worse than NB-MLP. For experiment B, there is a noticeable improvement for the NB-CVAE. In light of the results shown in Table 5.4, it seems fleet information

has a significant impact on the model's ability to differentiate levels of risk, which further validates the likelihood results previously discussed. It also confirms that Negative Binomial is more adequate than Poisson to predict the probability of a fleet vehicle accident.

To summarize the results of this section, we have seen that fleet information has value in the prediction of the probability of a fleet vehicle accident. Moreover, the results revealed that Negative Binomial is a better choice of distribution to model the number of accidents than Poisson. In addition, although the NB-CVAE appears slightly better than the NB-GLM, the results remain comparable. However, in terms of complexity, the NB-GLM is a clear winner. Indeed, using Python Librairies, implementing a GLM can be done in just a few lines of code. For the CVAE however, designing the architecture, along with the loss function to train the model is tedious and long. Several tests and hyperparameter searches also makes this model less attractive. Moreover, a GLM is also more intuitive to understand and explain, whereas the CVAE requires a solid background in computer science or related field to understand, implement it and explain it. In our context, the number of features used in the prediction is relatively small, which does not justify the use of a complex model like the NB-CVAE. However, perhaps in a context where the dimension of the feature space is high, the use of the NB-CVAE would be justified. In the following section, we will discuss the results of classification models.

5.2. Comparing Classification Models

As described in Chapter 3, we implemented three classification models: Logistic Regression (LR), Decision Tree (DT) and Random Forest (RF). For this category of models, we first binarized the response variable (the number of accidents) by setting the value of the variable $\boldsymbol{y}$ to 1, if the vehicle had one or more accidents, and to 0 if it did not have any accident. As described in Chapter 2, numerical variables were normalized and categorical variables were one-hot encoded. Hyperparameters were tuned for each model using a randomized search with cross validation using 5 folds. The Python library Scikit-learn (Pedregosa et al. [2011]) was used for the implementation and the evaluation of these models.
The results presented in this section aim at answering the following questions:

(1) How well can classification models predict an accident for a given vehicle?
(2) Which classification model performs best?
(3) Does fleet information improve accident prediction?

To answer these questions, we present a series of results shown in Table 5.5, Table 5.6, Figure 5.1 and Figure 5.3. Throughout this section, we refer to Experiment A, models implemented without fleet information, and Experiment B, models implemented with fleet information.

First, the standard classification evaluation metrics obtained on a test set, representing 15% of the entire data set (10,999 vehicles), are presented in Table 5.5. We remind the reader

that test set used is the same across all models and more details were given in the previous section. Moreover, for more details on the metrics used in this section, we refer the reader to Hossin and Sulaiman [2015]. In an ideal case, the accuracy, F1, recall, precision and ROC scores would be equal to 1.0, and the confusion matrix would have non-zero values only on the diagonal. In part A of Table 5.5, the evaluation metrics presented were obtained without fleet information. We remind the reader that we refer to fleet information as all the variables engineered from the other vehicles of a fleet presented in Chapter 2. Inversely, in part B, the evaluation metrics presented were obtained with fleet information. We determined the classification threshold for each model and for each experiment based on the F1-score. More specifically, we tested several potential thresholds and the one that returned the highest F1-score on the validation set was selected.

Overall, the classification results are poor, regardless of the model and regardless of the variables included as predictors. The null accuracy scores range from 64.6% to 71.4%, when the number of binarized accidents in the data set sums to 13.8%. Therefore, by systematically predicting the negative class all the time, we would have a better accuracy. However, it is often argued that accuracy is not the best metric for evaluating a classification model, especially with imbalanced classes.

The F1-score is the harmonic average of the recall and precision scores and is a more neutral metric. As a reminder, a recall of 55.7% means that out of all the positives (there will be an accident) the model should have predicted, 55.7% were indeed predicted as positives. A precision of 19.4% means that out of all the positive predictions (there will be an accident), 19.4% of these predictions were accurate. The F1-score as shown in Table 5.5 is low across all models. This is particularly due to the precision that is even lower. It means that the models tend to predict a vehicle will have an accident when it will not have an accident. This is further validated by the confusion matrices. Indeed, we observe between 21.6% to 29.3% of falsely predicted accidents by the different models.

If we compare the models based on the F1-score, they all yield more or less similar results with the Random Forest doing slightly better in both scenarios when fleet information is included and when it is not included. Next in line is the Logistic Regression that does better in both scenarios. As for the value-added of fleet information, there is a slight improvement in the F1-score for the Logistic Regression and the Random Forest, but there is a slight degradation for the Decision Tree model. Moreover, the bold metrics highlighted in Table 5.5 reveal that when fleet information, the Random Forest model is superior with respect to all metrics, whereas when fleet information is excluded, it is less obvious which model is best.

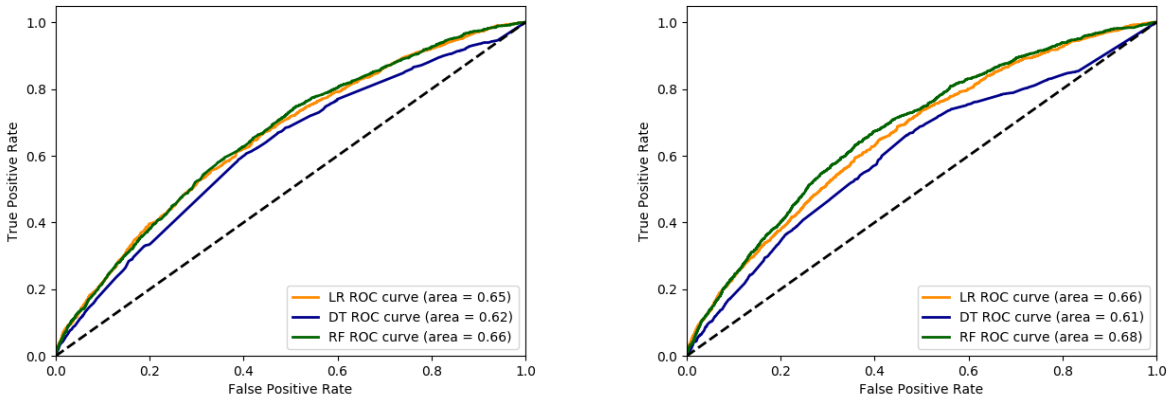
In Figure 5.1, the ROC curves are presented for each model. An ideal curve is one for which the area under the curve covers the entire surface and sums to 1.0. For both Figure 5.1 (a) and Figure 5.1 (b), the Random Forest model does better, followed by the Logistic

Table 5.5. Evaluation Metrics for different classification models. LR denotes Logistic Regression, DT denotes Decision Tree and RF denotes Random Forest. The results are obtained on binarized response variable, numerical variables are normalized and categorical variables are one-hot encoded. The Python library Scikit-learn was used for this implementation and evaluation. The confusion matrix results are in percentage and are normalized to sum to 100%.

A. Classification Metrics without fleet information			
	LR	DT	RF
Accuracy	65.2	**71.4**	66.4
ROC-score	65.2	62.4	**66.3**
F1-score	28.8	28.3	**29.1**
Recall	**55.7**	44.8	54.8
Precision	19.4	**20.7**	19.8
Confusion matrix	$\begin{bmatrix} 58.2 & 29.2 \\ 5.6 & 7.0 \end{bmatrix}$	$\begin{bmatrix} 65.8 & 21.6 \\ 6.9 & 5.6 \end{bmatrix}$	$\begin{bmatrix} 59.5 & 27.2 \\ 5.7 & 6.9 \end{bmatrix}$
B. Classification Metrics with fleet information			
	LR	DT	RF
Accuracy	65.1	64.6	**68.3**
ROC-score	66.2	61.1	**68.1**
F1-score	28.9	27.6	**31.0**
Recall	56.1	53.7	**56.5**
Precision	19.4	18.6	**21.4**
Confusion matrix	$\begin{bmatrix} 58.1 & 29.3 \\ 5.5 & 7.1 \end{bmatrix}$	$\begin{bmatrix} 57.8 & 29.6 \\ 5.8 & 6.8 \end{bmatrix}$	$\begin{bmatrix} 61.5 & 26.2 \\ 5.5 & 7.1 \end{bmatrix}$

Figure 5.1. Receiver operating characteristics for different classification models. LR denotes Logistic Regression, DT denotes Decision Tree and RF denotes Random Forest. The results are obtained on binarized response variable, numerical variables are normalized and categorical variables are one-hot encoded. The Python library Scikit-learn was used for this implementation and evaluation. The left-hand side denoted by (a) represents the ROC curve obtained without fleet information, whereas the right-hand-side denoted by (b) represents the ROC curve obtained with fleet information.

(a) (b)



Regression and lastly the Decision Tree. These results shown by the ROC curves are coherent with the evaluation metrics highlight before.

In light of the results shown so far, it seems classification models are not appropriate for addressing the task of accident risk evaluation. When we think about it, we do not so much care about predicting if the vehicle will have an accident or not. What we do care about its relative risk compared to other vehicles. From an insurer's perspective, a probability is more useful to determine the adequate insurance premium than prediction on whether or not the vehicle will have an accident. In other words, there is more information contained in a probability than in a binary prediction.

With that in mind, let us look instead at the average predicted class probabilities returned by each classification model. Here, we refer to the output of the classification model, a value between 0 and 1. In Table 5.6, we show the average class probabilities aggregated by the true number of accidents. Hence, what we expect is that the average class probability for the group that had 0 accident, should be the lowest, and the group that had 3 or 4 accidents should be the highest. The first thing we observe is that, as expected, the average class probability increases as the number of accidents increases, indicating that the class probability is perhaps more informative than the binary prediction as shown in previous results. This founding is especially interesting since the models were not trained using the number of accidents, but instead a binarized variable. Despite that, the models are still able to distinguish the different risk levels between 1-accident, 2-accidents and 3-accidents groups. In part A of Table 5.6, the results obtained without fleet information, we notice a clear difference between the average for the 0-accident group and the other groups: between 6% and 8% difference. In part B, the results obtained with fleet information, the differences in the averages are even more pronounced. This is another indication that the fleet information brings additional insight about the risk level of a vehicle.

Table 5.6. Average negative class probabilities predicted with classification models. LR denotes Logistic Regression, DT denotes Decision Tree and RF denotes Random Forest. The Python library Scikit-learn was used for this implementation and evaluation. Part A presents the results without fleet information, whereas part B presents the results with fleet information.

A. Average Negative Class Probability Predicted without fleet Information			
nb. accidents	LR	DT	RF
0	45.97	42.73	46.19
1	53.13	50.91	51.93
2	55.13	54.41	52.97
3 or more	55.26	52.68	53.02
B. Average Negative Class Probability Predicted with fleet Information			
nb. accidents	LR	DT	RF
0	45.27	39.74	45.14
1	53.35	48.55	51.39
2	56.62	52.16	54.28
3 or more	58.30	58.09	54.52

To summarize the classification results, we attempted to answer three questions: (1) How well can classification models predict an accident for a given vehicle?, (2) Which classification

model performs best? and (3) Does fleet information improve accident prediction? As for the first question, we saw that, overall, classification models exhibit poor performance at this task. We also discussed the fact that the approach itself is ill-suited as the goal should not be to predict an accident, but rather, to understand the relative risk associated to a vehicle. Indeed, information can be lost when using a binary target variable. For this reason, we looked at average negative class probabilities predicted by the classification models and we saw that insight can be found in using class probabilities rather than the binary predictions. For the second question, our results hint at a slight superiority of the Random Forest model especially with the inclusion of fleet information as evidenced by the evaluation metrics. Lastly, we also saw that the fleet information adds additional value as evidenced by slightly higher evaluation metrics and more pronounced differences in average negative class probabilities predicted between different groups.

In the next and final section of this chapter, we compare the probabilistic models to the classification models. To do so, we will look at the probabilistic models results from a classification perspective.

5.3. Comparing Probabilistic Models from a Classification Perspective

So far, we have compared probabilistic approaches and we were able to draw several conclusions. Namely, we were able to determine which models perform best at the task of evaluating the risk of a fleet vehicle accident, which distribution is best suited for this purpose and what is the effect of fleet information. We also compared classification approaches and we were able to confirm that fleet information adds value to the prediction. Moreover, we brought up the fact that information is lost when using a binary prediction in this context and we showed the value of using average predicted class probabilities.

In this section, we adapted the probabilistic models' results to a classification framework. We did this because it is impossible to compute the likelihood for all the classification approaches and thus difficult to compare them with the probabilistic ones. However, we still need a common ground to derive some final conclusions. With this purpose in mind, we used the probabilities of having one accident or more for each example in the test set to compute the classification metrics shown in Table 5.7. To split the data points into one of the two classes, we determined a threshold for each model and for each experiment based on the F1-score. In other words, we tested several potential thresholds and the one that returned the highest F1-score on the validation set was selected. These results can be compared to the ones found in Table 5.5 shown in the previous section. First and foremost, the classification metrics results are poor across all models. The first thing we notice is that ROC-scores are especially low for the P-GLM and the P-CVAE, 50.3 and 49.3 respectively. Second, for the same models we do see a some improvement in the ROC-score when fleet information is added, namely for the P-GLM that jumps to 66.1 from 50.2. Moreover, for the Negative Binomial models, we also see an improvement from the Poisson models as well as from adding the fleet information. This simply confirms our previous conclusions.

Now, let us compare Table 5.7 to Table 5.5. The highest ROC-score is 66.3 for classification models without fleet information and improves to 68.1 with fleet information. Although, the ROC-scores improve significantly when fleet information is included, relative to the classification models, they remain lower: the highest of all is 66.1 for the NB-GLM. F1-scores are also much lower for probabilistic models than for classification models. The NB-CVAE and the NB-GLM have comparable F1-scores to classification models, 28.9 and 27.4 respectively when fleet information is used, whereas it ranges from 27.6 to 31.0 for classification models. From bold metrics, it is easier to see models that yield better performances. The P-MLP is the top performer when fleet information is not used, whereas the P-GLM leads the way when fleet features are included. Additionally, the NB-CVAE performs best when fleet informtion is excluded, whereas when it is included, it is not clear which model performs best between the NB-GLM and NB-CVAE.

Finally, the ROC-curves illustrated in Figure 5.2 recap the ROC-curve results we have just highlighted and the bar plots in Figure 5.3 recap the classification metrics for both probabilistic and classification methods. In part (a) of the latter Figure, we can see that the NB-GLM, the NB-CVAE and the Decision tree have the highest accuracies. In terms of F1 score, ROC score and precision, the NB-CVAE and the classification models dominate. In terms of recall, the classification models do better. In part (b), Accuracies are highest for the P-CVAE, the NB-MLP and the NB-CVAE. In terms of recall, the NB-GLM and classification models dominate. Precisions, F1 score and ROC scores are comparable across most models, except for the P-MLP, the P-CVAE and the NB-MLP where it is lower.

Lastly, we look at the average predicted class probabilities in the same way we did for the classification models. That is, we aggregated the probabilities of having one accident $P(\boldsymbol{y} = 1|\boldsymbol{x})$ for each model. These results are actually derived from Table 5.4 and 5.3 and framed in an identical manner to make them comparable to the classification probabilities found in Table 5.6. The most obvious difference is the level of the average probabilities found in Table 5.8. Indeed, probabilities of having one accident average around 9 to 15% for most models, which is close to the empirical average of 13.78%. For the classification models however, it approximately ranges form 45 to 58% which is far off from the empirical truth.

In summary, in this chapter, we first compared the probabilistic models among each other. We found that the GLM and the CVAE with Negative Binomial distribution seem to marginally dominate other models. Going back to the insurer's perspective, his or her goal is to estimate the probability of a fleet vehicle of having an accident at minimal cost. Keeping this in mind, a GLM does the job. Indeed, it is easy to understand, simple to implement and yields strong results. The CVAE showed comparable results, but the addition of complexity would need to be justified before selecting this model. For instance, if the feature space was very large, perhaps it could be considered as a candidate. Furthermore, we also compared classification models and observed that valuable information is lost when using a binary target variable. Again from an insurers perspective, it does not make much sense to predict whether or not a vehicle will have an accident. What is important is to understand the underlying risk of each fleet vehicle relative to all other vehicles so that insurance premiums can be priced accurately. In that regard, there would be room for further research in revenue management, an unexplored field of application. Hence, on top of the poor performance of these models, they are not suited for the task of evaluating the probability of a vehicle accident. Lastly, we looked at the performance of the probabilistic models from a classification perspective so that we could compare them with the classification models. The use of probabilistic models was further justified by our analysis. Indeed, classification metrics for such models were comparable if not better than those obtained with the classification models.

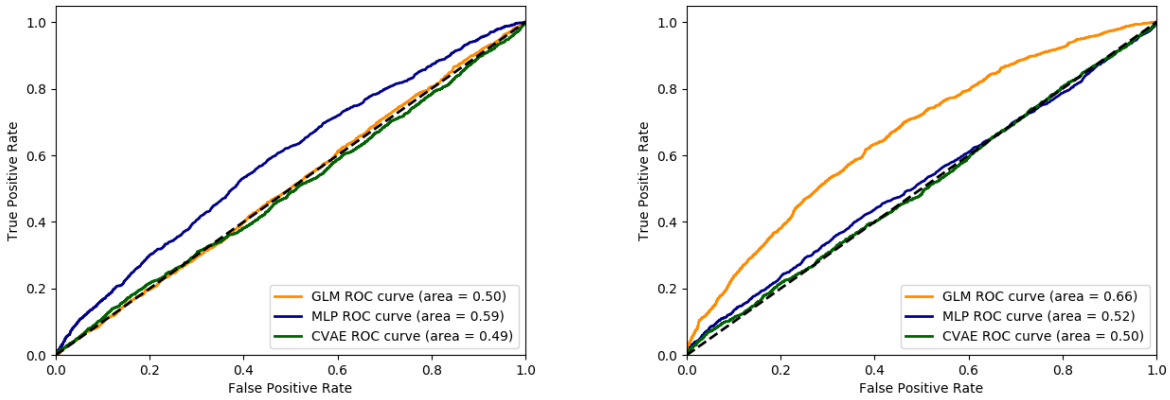
Table 5.7. Evaluation Metrics for probabilistic models. GLM denotes Generalized Linear Model, MLP denotes Multilayer Perceptron and CVAE denotes Conditional Variational Auto-encoder. The results are obtained using the probability $P(y = 1|x)$ computed from each model. The Python library Scikit-learn was used for this implementation and evaluation. The confusion matrix results are in percentage and are normalized to sum to 100%.

1. Poisson Models			
A. Classification Metrics without fleet information			
	GLM	MLP	CVAE
Accuracy	59.6	**63.4**	59.5
ROC-score	50.3	**58.7**	49.3
F1-score	20.1	**23.7**	18.4
Recall	39.0	**45.1**	36.2
Precision	12.7	**16.0**	12.3
Confusion matrix	$\begin{bmatrix} 54.2 & 33.3 \\ 7.5 & 5.1 \end{bmatrix}$	$\begin{bmatrix} 57.7 & 29.7 \\ 6.9 & 5.7 \end{bmatrix}$	$\begin{bmatrix} 54.9 & 32.5 \\ 8.0 & 4.6 \end{bmatrix}$
B. Classification Metrics with fleet information			
	GLM	MLP	CVAE
Accuracy	**73.0**	66.2	53.4
ROC-score	**66.1**	59.5	50.2
F1-score	**28.3**	23.8	19.5
Recall	42.2	42.0	**44.8**
Precision	**21.2**	16.6	12.5
Confusion matrix	$\begin{bmatrix} 67.7 & 19.7 \\ 7.3 & 5.3 \end{bmatrix}$	$\begin{bmatrix} 60.9 & 26.5 \\ 7.3 & 5.3 \end{bmatrix}$	$\begin{bmatrix} 47.8 & 39.6 \\ 6.9 & 5.6 \end{bmatrix}$
2. Negative Binomial Models			
A. Classification Metrics without fleet information			
	GLM	MLP	CVAE
Accuracy	69.3	66.3	**75.1**
ROC-score	50.3	58.7	**65.6**
F1-score	16.4	22.8	**27.8**
Recall	23.8	39.4	**38.1**
Precision	12.4	16.0	**22.0**
Confusion matrix	$\begin{bmatrix} 66.2 & 21.1 \\ 9.6 & 3.0 \end{bmatrix}$	$\begin{bmatrix} 61.4 & 26.1 \\ 7.6 & 5.0 \end{bmatrix}$	$\begin{bmatrix} 70.3 & 17.1 \\ 7.8 & 4.8 \end{bmatrix}$
B. Classification Metrics with fleet information			
	GLM	MLP	CVAE
Accuracy	63.6	72.9	**73.4**
ROC-score	66.1	61.6	**67.0**
F1-score	28.8	25.4	**28.9**
Recall	**58.8**	36.5	42.8
Precision	19.1	19.4	**21.8**
Confusion matrix	$\begin{bmatrix} 56.2 & 31.2 \\ 5.3 & 7.4 \end{bmatrix}$	$\begin{bmatrix} 68.3 & 19.1 \\ 8.0 & 4.6 \end{bmatrix}$	$\begin{bmatrix} 68.0 & 19.4 \\ 7.2 & 5.4 \end{bmatrix}$

Figure 5.2. Receiver operating characteristics for probabilistic models. GLM denotes Generalized Linear Model, MLP denotes Multilayer Perceptron and CVAE denotes Conditional Variational Auto-encoder. The results are obtained using the probability $P(y = 1|x)$ computed from each model. The Python library Scikit-learn was used for this implementation and evaluation. The left-hand side denoted by (a) represents the ROC curve obtained without fleet information, whereas the right-hand-side denoted by (b) represents the ROC curve obtained with fleet information.

Poisson Models

(a) (b)



Negative Binomial Models

(a) (b)

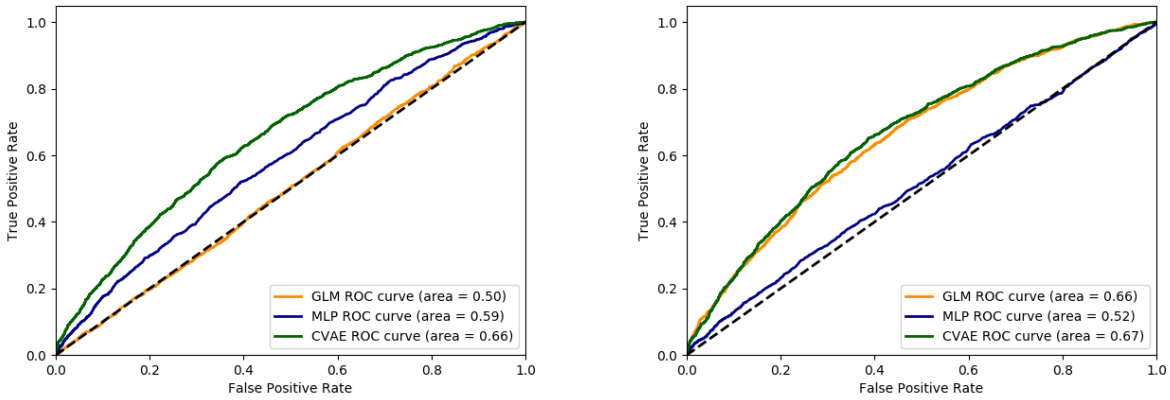


Figure 5.3. Summary of classification metrics across all models. LR denotes Logistic Regression, DT denotes Decision Tree and RF denotes Random Forest. P-GLM, P-MLP and P-CVAE denote the probabilistic models with a Poisson distribution. NB-GLM, NB-MLP and NB-CVAE denote the probabilistic models with a Negative Binomial distribution. The left-hand side denoted by (a) shows the metrics without fleet information, whereas the right-hand side denoted by (b) shows the metrics with fleet information.

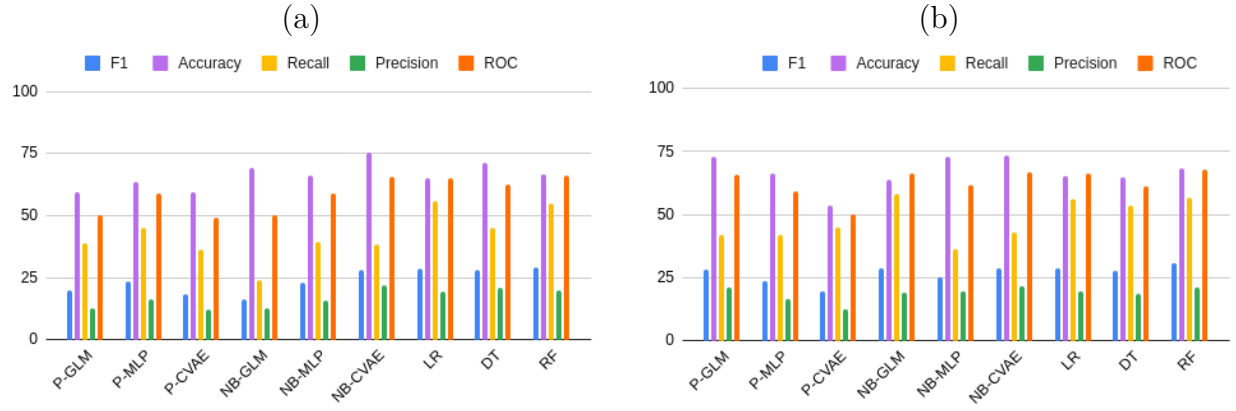


Table 5.8. Average predicted probabilities for probabilistic models. GLM denotes Generalized Linear Model, MLP denotes Multilayer Perceptron and CVAE denotes Conditional Variational Auto-encoder. The results are obtained using the probability $P(y = 1|x)$ computed from each model. The Python library Scikit-learn was used for this implementation and evaluation. Part A presents the results without fleet information, whereas part B presents the results with fleet information.

1. Poisson Models			
A. Average Probability Predicted without fleet Information			
nb. accidents	GLM	MLP	CVAE
0	12.25	14.05	13.15
1	12.25	15.36	13.19
2	12.32	15.53	13.14
3 or more	12.52	16.41	13.35
B. Average Probability Predicted with fleet Information			
nb. accidents	GLM	MLP	CVAE
0	11.18	13.86	13.16
1	14.20	15.13	13.19
2	15.70	16.16	13.14
3 or more	16.55	16.82	13.05
2. Negative Binomial Models			
A. Average Probability Predicted without fleet Information			
nb. accidents	GLM	MLP	CVAE
0	10.84	10.27	9.78
1	10.84	11.30	12.35
2	10.89	11.40	12.98
3 or more	11.05	11.24	12.98
B. Average Probability Predicted with fleet Information			
nb. accidents	GLM	MLP	CVAE
0	9.76	9.04	9.68
1	12.05	10.82	12.84
2	13.15	11.22	14.33
3 or more	10.68	10.70	14.71

Conclusion

In our work, we carried out a comparison of several approaches to evaluate the risk of a fleet vehicle accident. We tackled a typical insurance problem using a data set of fleet vehicles owned by carrier companies. From the perspective of these companies, they need to insure the vehicles they own against damage which could translate into financial or operational repercussions on their business. From the insurer's perspective, it is necessary to evaluate as accurately as possible the risk associated with the vehicles being insured so that they can charge an adequate premium. Of course, the method used to evaluate the risk plays a big role, but the data being consumed by the models are also key to achieve this goal. The data being captured by the insurer is what contains the information that reveals the risk level of a vehicle. In our specific use case, there are many factors influencing the risk level: the behavior of the drivers, the quality of the management of the fleet, the maintenance done on the vehicles, the roads it drives on, the weather conditions it faces to name a few. The information we had at our disposition did not capture every factor that might influence the risk of a vehicle, but we did have access to information that give insight about the driver's and the management's behavior, as well as some properties of the vehicle. We also performed feature engineering on the data set. More specifically, since we know which vehicle are part of what fleet, we could design features that reveal the risk level of other vehicles in the fleet and use those as additional predictors for the risk level of a given vehicle. The rationale behind this is that there might exist a form of contagion amongst vehicle of a same fleet. Overall, engineered-fleet features have shown to be helpful at predicting the probability of an accident regardless of the model used.

On the modelling side, we proposed a novel approach using Conditional Variational Auto-Encoders with Poisson or Negative Binomial distribution. This model was compared to two other probabilistic methods: a Generalized Linear Model and a Multilayer Perceptron with the same distributional assumptions. Moreover, we compared probabilistic approaches to classification models: Logistic Regression, Decision Tree and Random Forest. We observed that the classification models showed poor results and appeared not to be adequate models to evaluate such risk. It makes more sense to evaluate such risk as a probability rather that a binary variable. As for probabilistic models, we saw that the Negative Binomial distribution

is a better assumption to model the number of accidents given a set of features. We also saw that the CVAE performs slightly better than the GLM. However, from a complexity standpoint, the latter is a better choice since it does not require as much effort and skill to implement it. The strength of the CVAE is its ability to handle larger dimensions. Nevertheless, in our application, this was not the case.

We applied two deep learning approaches, and another more classic probabilistic method, as well as classification models to evaluate the risk of a vehicle accident. Risk, defined as the probability of an undesired outcome, is part of anyone's life. Therefore, the models we proposed could be applied to any situation where we might want to evaluate the probability of some event occurrence: economic crisis, credit defaults, weather disasters and so on.

References

Peter Martey Addo, Dominique Guegan, and Bertrand Hassani. Credit risk analysis using machine and deep learning models. Risks, 6(2):38, 2018.

Marco Aleandri. Modeling dynamic policyholder behavior through machine learning techniques. Università La Sapienza. Rapporto Tecnico, 2018.

JF Angers, D Desjardins, G Dionne, and F Guertin. Individual and firms random effects in the estimation of event distributions. Technical report, Working Paper, Canada Research Chair in Risk Management, HEC Montréal, 2006.

Katrien Antonio and Jan Beirlant. Actuarial statistics with generalized linear mixed models. Insurance: Mathematics and Economics, 40(1):58–76, 2007.

Katrien Antonio and Emiliano A Valdez. Statistical concepts of a priori and a posteriori risk classification in insurance. AStA Advances in Statistical Analysis, 96(2):187–224, 2012.

Katrien Antonio, Edward W Frees, and Emiliano A Valdez. A multilevel analysis of intercompany claim counts. ASTIN Bulletin: The Journal of the IAA, 40(1):151–177, 2010.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. Classification and regression trees. CRC press, 1984.

Ketaki Chopde, Pratik Gosar, Paras Kapadia, Niharika Maheshwari, and Pramila M Chawan. A study of classification based credit risk analysis algorithm. Int J Eng Adv Technol, 1:142–144, 2012.

Adele Cutler, D Richard Cutler, and John R Stevens. Random forests. In Ensemble machine learning, pages 157–175. Springer, 2012.

Denise Desjardins, Georges Dionne, and Jean Pinquet. Experience rating schemes for fleets of vehicles. ASTIN Bulletin: The Journal of the IAA, 31(1):81–105, 2001.

Georges Dionne, Jean-François Angers, Denise Desjardins, and François Guertin. Vehicle and fleet random effects in a model of insurance rating for fleets of vehicles. Available at SSRN 601801, 2005.

Andrea Gabrielli. A neural network boosted double overdispersed poisson claims reserving model. ASTIN Bulletin: The Journal of the IAA, 50(1):25–60, 2020.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

Sarah Harris and David Harris. Digital design and computer architecture: arm edition. Morgan Kaufmann, 2015.

Joseph M Hilbe. Negative binomial regression. Cambridge University Press, 2011.

Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length and helmholtz free energy. In Advances in neural information processing systems, pages 3–10, 1994.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In International conference on artificial neural networks, pages 44–51. Springer, 2011.

Mohammad Hossin and MN Sulaiman. A review on evaluation metrics for data classification evaluations. International Journal of Data Mining & Knowledge Management Process, 5 (2):1, 2015.

Ho Tin Kam. Random decision forest. In Proceedings of the 3rd International Conference on Document Analysis and Recognition, volume 1416, page 278282. Montreal, Canada, August, 1995.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.

DP Kingma and J Ba. Adam. a method for stochastic optimization. arxiv 2017. arXiv preprint arXiv:1412.6980, 106.

Ming-Chang Lee. Enterprise credit risk evaluation models: A review of current research trends. International Journal of Computer Applications, 44(11):37–44, 2012.

Cheng-Yuan Liou, Wei-Chen Cheng, Jiun-Wei Liou, and Daw-Ran Liou. Autoencoder for words. Neurocomputing, 139:84–96, 2014.

Hui Liu, BO Dai, Hui He, and Yang Yan. The k-prototype algorithm of clustering high dimensional and large scale mixed data. In Wavelet Active Media Technology And Information Processing: (In 2 Volumes), pages 738–743. World Scientific, 2006.

Stuart Lloyd. Least squares quantization in pcm. IEEE transactions on information theory, 28(2):129–137, 1982.

Peter McCullagh and John A Nelder. Generalized linear models 2nd edition chapman and hall. London, UK, 1989.

Geoffrey J McLachlan and Kaye E Basford. Mixture models: Inference and applications to clustering, volume 38. M. Dekker New York, 1988.

Alfredo Nazabal, Pablo M Olmos, Zoubin Ghahramani, and Isabel Valera. Handling incomplete heterogeneous data using vaes. arXiv preprint arXiv:1807.03653, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In

H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. Journal of machine learning research, 12(Oct): 2825–2830, 2011.

Herbert Robbins and Sutton Monro. A stochastic approximation method. The annals of mathematical statistics, pages 400–407, 1951.

Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, 20:53–65, 1987.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

John Scott Long. Regression models for categorical and limited dependent variables. Advanced quantitative techniques in the social sciences, 7, 1997.

Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In 9th Python in Science Conference, 2010.

Oscar S Siordia, Isaac Martín de Diego, Cristina Conde, Gerardo Reyes, and Enrique Cabello. Driving risk classification based on experts evaluation. In 2010 IEEE Intelligent Vehicles Symposium, pages 1098–1103. IEEE, 2010.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In Advances in neural information processing systems, pages 3483–3491, 2015.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research, 11(Dec):3371–3408, 2010.

He Zhao, Piyush Rai, Lan Du, Wray Buntine, Dinh Phung, and Mingyuan Zhou. Deep generative models of sparse and overdispersed discrete data. 2018.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. arXiv preprint arXiv:1703.10960, 2017.

Appendix A

Supplementary Results

Table A.1. Average probabilities of having y accidents given X per class for GLM Models for experiment C., D. and E. Results are obtained using a restricted subset of features. The probabilities are obtained using the fitted model which returns the parameters of the distribution for each observation. Probabilities are inferred and aggregated by groups of number of accidents.

1. Poisson GLM					
C. Probabilities with driving violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	86.81	12.27	0.87	0.04	0.00
1	86.81	12.28	0.86	0.04	0.00
2	86.81	12.28	0.86	0.04	0.00
3	86.79	12.29	0.87	0.04	0.00
4	86.62	12.43	0.89	0.04	0.00
D. Probabilities with trucking violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	86.81	12.27	0.87	0.04	0.00
1	86.82	12.27	0.87	0.04	0.00
2	86.76	12.31	0.88	0.04	0.00
3	86.80	12.28	0.87	0.04	0.00
4	86.80	12.28	0.87	0.04	0.00
E. Probabilities with all violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	86.81	12.27	0.87	0.04	0.00
1	86.82	12.27	0.87	0.04	0.00
2	86.76	12.32	0.88	0.04	0.00
3	86.78	12.30	0.87	0.04	0.00
4	86.62	12.43	0.89	0.04	0.00
2. Negative Binomial GLM					
C. Probabilities with driving violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	88.06	10.51	1.26	0.15	0.02
1	87.69	10.75	1.35	0.17	0.02
2	87.72	10.72	1.34	0.17	0.02
3	87.47	10.90	1.40	0.19	0.03
4	87.28	11.08	1.42	0.19	0.03
D. Probabilities with trucking violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	87.61	10.85	1.34	0.17	0.02
1	87.62	10.85	1.34	0.17	0.02
2	87.57	10.88	1.36	0.17	0.02
3	87.60	10.86	1.36	0.17	0.02
4	87.60	10.86	1.34	0.17	0.02
E. Probabilities with all violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	87.61	10.85	1.35	0.17	0.02
1	86.62	10.85	1.34	0.17	0.02
2	87.56	10.88	1.36	0.17	0.02
3	86.58	10.87	1.35	0.17	0.02
4	86.44	10.98	1.38	0.17	0.02

Table A.2. Average probabilities of having y accidents given X per class for MLP Models for experiment C., D. and E. Results are obtained using a restricted subset of features. The probabilities are obtained by passing each observation in the test set through the MLP and aggregated by groups of number of accidents.

1. Poisson MLP					
C. Probabilities with driving violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	84.15	14.40	1.34	0.09	0.00
1	83.94	14.52	1.42	0.10	0.00
2	84.24	14.27	1.36	0.11	0.00
3	85.74	13.10	1.08	0.06	0.00
4	82.72	15.53	1.62	0.12	0.00
D. Probabilities with trucking violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	84.16	14.40	1.33	0.09	0.00
1	84.02	14.46	1.38	0.11	0.00
2	84.39	14.23	1.28	0.08	0.00
3	84.88	13.78	1.24	0.08	0.00
4	84.91	13.88	1.14	0.06	0.00
E. Probabilities with all violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	85.26	13.52	1.14	0.07	0.00
1	84.89	13.79	1.22	0.08	0.00
2	85.72	13.15	1.06	0.06	0.00
3	84.89	13.79	1.22	0.08	0.00
4	86.34	12.66	0.94	0.04	0.00
2. Negative Binomial MLP					
C. Probabilities with driving violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	87.36	9.71	2.13	0.56	0.16
1	87.19	9.86	2.14	0.56	0.16
2	87.05	9.93	2.17	0.57	0.17
3	87.27	9.81	2.13	0.56	0.16
4	86.81	10.34	2.09	0.53	0.15
D. Probabilities with trucking violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	87.05	9.88	2.19	0.59	0.18
1	86.97	9.96	2.21	0.59	0.18
2	87.31	9.77	2.11	0.55	0.16
3	86.58	10.44	2.20	0.55	0.15
4	86.39	9.92	2.51	0.75	0.24
E. Probabilities with all violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	87.31	10.03	2.02	0.47	0.11
1	86.88	10.25	2.13	0.52	0.14
2	87.20	10.06	2.06	0.49	0.13
3	86.28	10.68	2.27	0.56	0.15
4	89.75	8.47	1.43	0.27	0.05

Table A.3. Average probabilities of having y accidents given X per class for CVAE Models for experiments C., D. and E. Results are obtained using a restricted subset of features. The probabilities are obtained by passing each observation in the test set through the CVAE 100 times and by taking the average. These averages are aggregated by groups of number of accidents. In parentheses are the standard deviations.

1. Poisson CVAE					
C. Probabilities with driving violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	85.50	13.15	1.22	0.09	0.00
1	85.50	13.12	1.25	0.10	0.00
2	85.50	13.12	1.26	0.10	0.00
3	85.35	13.24	1.28	0.10	0.00
4	85.77	12.89	1.22	0.10	0.00
D. Probabilities with trucking violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	86.21	12.80	0.95	0.05	0.00
1	86.20	12.82	0.95	0.05	0.00
2	86.22	12.83	0.94	0.05	0.00
3	86.21	12.80	0.95	0.05	0.00
4	86.19	12.80	0.93	0.05	0.00
E. Probabilities with all violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	85.55	13.26	1.02	0.05	0.00
1	85.49	13.25	1.09	0.07	0.00
2	85.48	13.29	1.15	0.07	0.00
3	85.60	13.14	1.08	0.07	0.00
4	85.80	13.10	1.14	0.07	0.00
2. Negative Binomial CVAE					
C. Probabilities with driving violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	88.22	9.70	1.66	0.32	0.06
1	87.16	10.42	1.90	0.39	0.08
2	87.18	10.42	1.89	0.38	0.08
3	87.03	10.60	1.88	0.37	0.07
4	86.29	11.27	1.97	0.37	0.07
D. Probabilities with trucking violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	88.10	10.00	1.56	0.26	0.04
1	87.78	10.23	1.62	0.28	0.05
2	87.69	10.22	1.67	0.31	0.06
3	87.71	10.28	1.64	0.29	0.05
4	88.72	9.56	1.43	0.23	0.04
E. Probabilities with all violations only					
Nb. accidents	$P(y=0\|X)$	$P(y=1\|X)$	$P(y=2\|X)$	$P(y=3\|X)$	$P(y=4\|X)$
0	88.34	9.69	1.59	0.29	0.06
1	87.03	10.59	1.88	0.37	0.08
2	87.05	10.53	1.91	0.39	0.09
3	87.08	10.69	1.81	0.33	0.09
4	86.70	11.06	1.84	0.32	0.06