VARIATIONAL AUTOENCODER FOR SYNTHETIC INSURANCE DATA

Charlotte Jamotton, Donatien Hainaut

LIDAM Discussion Paper ISBA
2023 / 25

UCLouvain

LIDAM
ISBA | Louvain Institute of Data Analysis and Modeling in economics and statistics

ISBA

Voie du Roman Pays 20 - L1.04.01

B-1348 Louvain-la-Neuve

Email : lidam-library@uclouvain.be

https://uclouvain.be/en/research-institutes/lidam/isba/publication.html

Variational autoencoder for synthetic insurance data

Charlotte Jamotton, *UCLouvain*
Donatien Hainaut, *UCLouvain*

29th June 2023

Abstract

This article explores the application of variational autoencoders (VAEs) to insurance data. Previous research has demonstrated the successful use of generative models, particularly VAEs, in various domains such as image recognition, text classification, and recommender systems. However, their application to insurance data, specifically heterogeneous insurance portfolios with mixed continuous and discrete attributes, remains unexplored. This study introduces novel insights into utilizing VAEs for unsupervised learning tasks in the actuarial field, including dimensionality reduction and synthetic data generation. We propose a VAE model with a quantile transformation of continuous data and a reconstruction loss that combines categorical cross-entropy and mean squared error, along with a KL divergence-based regularization term. The architecture of our VAE model eliminates the need for pre-training layers to fine-tune categorical features representations. We analyze our VAE's ability to reconstruct complex insurance data and generate synthetic insurance policies using a motor portfolio. Our experimental results and analysis highlight the potential of VAEs for addressing challenges related to privacy and anti-discriminatory regulations, bias correction, and data availability in the insurance industry.

Keywords: autoencoder, variational inference, synthetic data generation, heterogeneous insurance data, dimensionality reduction.

1 Introduction

The insurance industry heavily relies on data to accurately assess its risk exposure, which ultimately determines the success or failure of any business model built on risk management. Insurance data is, however, known to contain sensitive and private information, such as the policyholder's name, address, or gender, which are subject to regulatory requirements[1]. Merely deleting these identification factors does not guarantee complete anonymization of the data, and the more data we conceal, the less valuable information

[1]The EU's General Data Protection Regulation (GDPR) on data protection and privacy, and the European Court of Justice's ruling in the Test-Achats case (C-236/09) concerning sex discrimination respectively came into effect in 2018 and 2011.

is left for future analysis. This sparks interest in *synthetic* data and its associated deep learning-based generative model.

By definition, a generative model, also called synthesis, is trained to generate artificial data that share similar structure and characteristics with the original data. The authentic and synthetic data should then yield sensibly comparable results when subjected to the same statistical analysis. In actuarial sciences, such generative processes could provide access to unlimited data that mimics the risk profiles of actual policyholders. Compliance with privacy data regulations would no longer be a concern, as the computer-generated data is no longer personally identifiable information nor associated with a data subject. Synthetic data could serve as a valuable insurance tool, not only to sidestep privacy regulations that restrict access to real data, but also to handle situations where real data is not available (as is often the case when designing a new product) or to compensate for biased training data. For example, skewness towards racial or gender bias, as well as the lack of representation of other minority groups could be corrected through the generation of additional representative policies.

The Generative Adversarial Networks (GANs) introduced by I. J. Goodfellow et al. (2014), and the Variational AutoEncoders (VAEs) proposed by Kingma and Welling (2013) are the two most prominent approaches to generative modeling and have established themselves over the last few years for their ability to generate highly realistic samples of images, texts or sounds. In the medical domain, the state-of-the-art results obtained in image generation have notably been used by An and Cho (2015) and Baur et al. (2019) to detect abnormal lesions or pathologies (e.g., in brain MR images). The applications of VAEs further include text classification (W. Xu et al. 2017), image captions modeling (Pu et al. 2016), recommender systems (Li and She 2017), disentangled representation learning (Higgins et al. 2017), and discrete molecule data generation (Kusner, Paige and Hernández-Lobato 2017).

While GANs are often praised for their ability to generate high-quality visual samples, they are also singled out for their time-consuming iterative optimization. In contrast, VAEs are known for their relatively easy implementation and hyperparameters set up, as well as their explicit inference network and tractable likelihood, but tend to generate blurry or fuzzy samples (see e.g., Zhao, Song and Ermon (2017)). The bottleneck architecture, unique to autoencoders, led us to prioritize VAEs over the computationally expensive GANs in this research article. The appeal of a bottleneck layer stems from the manifold hypothesis, which suggests that, while the actual data might have hundreds of dimensions, its underlying structure can be sufficiently described using a lower-dimensional representation. This manifold learning approach is akin to our human perception, where we focus on key features rather than intricate details (e.g., we do not keep in mind every aspect of a dog's appearance to recognize it or distinguish it from a cat; we rather focus on key characteristics). This is the motivation behind dimensionality reduction techniques such as principal component analysis (Abdi and Williams 2010), t-distributed Stochastic Neighbor Embedding (Van der Maaten and Hinton 2008), or autoencoders (often defined as non-linear PCA). Ideally, when applied to insurance data, an autoencoder should learn

some compressed representation of the policies' descriptive attributes, and policyholders that depict similar risk profiles should have a very close representation in the latent space, thereby facilitating the clustering of policies or the generation of new meaningful policies.

Originally build on numeric data, VAEs perform poorly on real-world datasets containing various types of features (i.e., (positive) numerical, binary, multinomial, and ordinal) with a variety of marginal distributions (e.g., multimodal, long tail). In the literature, there are two foundational papers by L. Xu and Veeramachaneni (2018) and L. Xu, Skoularidou et al. (2019) that explore the generation of mixed-type tabular data like medical or educational records. To overcome imbalanced categorical variables, they propose a conditional generator and training-by-sampling. They further address the challenges of multi-modal non-Gaussian distributions by introducing a mode-specific normalization using Gaussian mixture models (Bishop and Nasrabadi 2006). However, this normalization scheme increases, for each continuous variable, the input dimensionality by the number of gaussians used to *approximate* the variable's distribution.

In the same spirit, Nazabal et al. (2020) and Ma et al. (2020) proposed similar two-stage processes that enable VAEs to handle mixed numerical and categorical likelihood models in supervised tasks. The bottom level in their hierarchical VAE accounts for heterogeneous likelihood using independent deep neural networks (DNNs). A shared DNN on the top level then captures statistical dependencies among the heterogeneous attributes. In the same vein, Suh and Choi (2016) employ Gaussian copula to model local dependencies.

In the actuarial literature, the use of Neural Networks (NNs) have so far been mainly, if not exclusively, limited to supervised learning tasks, with the exception of Hainaut (2019) and Denuit, Hainaut and Trufin (2019) who explored the predictive capacity of unsupervised self-organizing maps for non-life insurance data. Hainaut (2018), Noll, Salzmann and Wuthrich (2020) and Ferrario, Noll and Wuthrich (2020) were among the first to promote the use of neural networks to supervised actuarial applications. In such supervised approaches, the network learns a function $f : X \to Y$ where X is typically an insurance portfolio defined by n policies and p rating factors (e.g., driver age, years of driving experience, vehicle brand) and the target variable Y is either a continuous (claim frequency) or discrete (number of claims) function of X. The unsupervised learning approach used in this paper leads to a starkly different optimization problem where the aim is to learn and understand the structure of our inputs, rather than trying to map inputs to target(s).

The presence of mixed-type values, and more specifically categorical rating factors, has also already been addressed in the actuarial literature by Noll, Salzmann and Wuthrich (2020) and Ferrario, Noll and Wuthrich (2020) through the use of entity embedding (Guo and Berkhahn 2016) or joint embedding (Delong and Kozak 2021). However, both embedding approaches require a pre-training of layers to fine-tune the numerical representation of the categorical features. The time allotted to the model calibration further increases with the number of categorical attributes, which is not in line with our intention to train an all-in-one *un*supervised autoencoder model.

The aim of this research article is to find a low-dimensional representation of heterogeneous tabular insurance data (i.e., to reduce and 'blur' the rating factors space) and further

use the decoder network to generate synthetic insurance policies. This article contributes to the literature by introducing:

1. An application of a *variational* autoencoder, a Bayesian extension of the autoencoder (Kramer 1991), to insurance data;
2. A quantile transformation to pre-process multimodal non-Gaussian data;
3. An all-in-one training of a VAE for heterogeneous data (no pre-training of layers to fine-tune a numerical representation of the categorical features needed);
4. A loss function that combines a reconstruction loss, as a mixture of categorical cross entropy and mean squared error, and a regularization term defined by KL divergence;
5. The generation of high-quality synthetic insurance policies.

The article is organized as follows. Section 2 introduces the VAE model with a theoretical review of the classical (variational) autoencoder and motivates its use as a generative process. Section 3 introduces our main contributions and adjustments to the classical VAE. Section 4 explains our experiment settings and results. Subsection 4.1 introduces the heterogeneous insurance portfolio used to evaluate the quality of our VAE model. Subsection 4.2 reviews the setup of the model's hyperparameters. Subsection 4.3 presents the quality evaluation of the reconstructed portfolio and the structure of the latent space. Subsection 4.4 investigates whether our VAE can be utilized to generate high-quality synthetic insurance portfolios characterized by discrete and continuous variables. Section 5 gives our conclusion. To the best of our knowledge, the use of quantile transformations and variational autoencoders for such unsupervised learning tasks (dimension reduction) and synthetic data generation has so far not been investigated in the actuarial literature.

2 Variational autoencoder

2.1 Introduction

An autoencoder (Kramer 1991) is a pair of two connected feed-forward Neural Networks (NNs), the encoder and the decoder. The encoder network takes in an input (e.g., insurance policies) and maps it into a compressed representation, the latent encoding, that contains enough relevant information for the decoder to convert it back to the original input. Formally, the encoder network learns a non-linear transformation $e_\phi : \mathcal{X} \to \mathcal{Z}$ parametrized by ϕ that maps each input vector $\mathbf{x}$ of dimension p from the original high-dimensional input space $\mathcal{X}$ to a lower-dimensional latent space $\mathcal{Z}$. The latent encoding $\mathbf{z} = e_\phi(\mathbf{x})$ is a low-dimensional representation of the original input. Conversely, the decoder network learns a non-linear transformation $d_\theta : \mathcal{Z} \to \mathcal{X}$ parametrized by θ that projects the latent codes $\mathbf{z}$ of dimension $d < p$ back into the original high-dimensional input space $\mathcal{X}$. This transformation takes as input the latent vector $\mathbf{z}$ to reconstruct the original input data

$\hat{\mathbf{x}} = d_\theta(\mathbf{z})$. The autoencoder $f(\mathbf{x}) = d_\theta(e_\phi(\mathbf{x}))$, illustrated in Fig.1, is trained as a whole to minimize the difference between each input $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$.

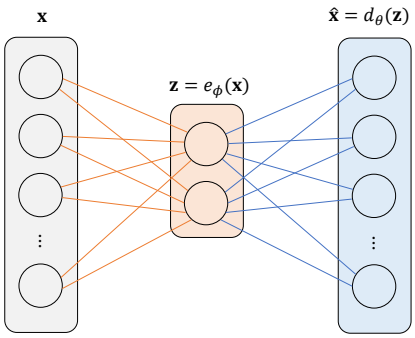

Figure 1: Architecture of a shallow autoencoder.

Because the traditional encoder network projects each input vector $\mathbf{x} \in \mathbf{X}$ (e.g., each insurance policy) to a *single* latent value z_j (for each encoding dimension $j = 1, \ldots, d$), each sample $\mathbf{x}$ is encoded independently of other samples, even if they are from the same class or distribution (e.g., have similar vehicle characteristics and risk profiles). This makes the latent space unlikely to be continuous and policies with similar risk profiles might be mapped to very different or distant latent encodings. This becomes problematic when the decoder network is used as a generative model. If the latent space has discontinuities or gaps and a variation is sampled from there (i.e., from outside the manifold of the latent codes), the decoder will not be able to generate meaningful and realistic outputs because it was never trained with encoded vectors coming from that region of the latent space so it has no idea how to deal with it. The regularity that is expected from the latent space of a generative model can be expressed through two properties:

1. continuity: the transformation $e_\phi : \mathcal{X} \to \mathcal{Z}$ should ensure that (dis)similar samples $\mathbf{x}$ have (dis)similar latent vectors $\mathbf{z}$. Similarly, the transformation $d_\theta : \mathcal{Z} \to \mathcal{X}$ should ensure that two close points in the latent space have similar contents once decoded. In actuarial terms, this ensures that policies that depict the same vehicle characteristics and risk profile have very close representations in the latent space and similar contents once decoded.

2. completeness: a point sampled from the latent space should give meaningful and realistic content once decoded.

The variational autoencoder (VAE), illustrated in Fig.2 and introduced by Kingma and Welling (2013) and D. J. Rezende, Mohamed and Wierstra (2014) as a Bayesian extension to the autoencoder (Kramer 1991), tries to solve this problem by forcing the NNs to learn a probabilistic (instead of deterministic) representation of the data points in the latent space, which should improve the decoder's robustness to small changes in $\mathbf{z}$.

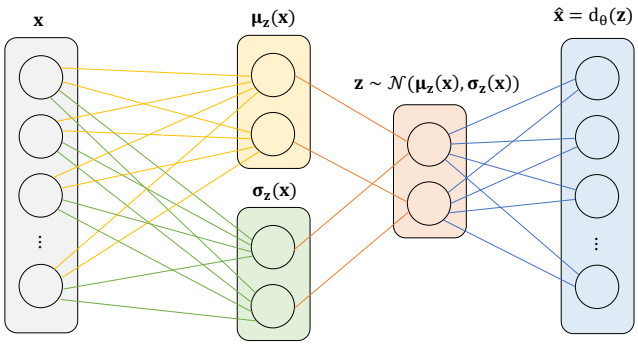

Figure 2: Architecture of a shallow variational autoencoder with a Gaussian prior.

Such a probabilistic approach comes nevertheless with its fair share of setbacks - and their solutions. Modeling the true probability distribution $p(\mathbf{x})$ over the limited set of insurance policies $\mathbf{X} = (\mathbf{x}_i)_{i=1,\dots,n}$ is a tedious and challenging task (which involves e.g., modelling complex correlations between the rating factors). Instead of learning $p_\theta(\mathbf{x})$ directly, we can introduce (unobserved) vectors $\mathbf{z}$ and define a conditional distribution, denoted $p_\theta(\mathbf{x}|\mathbf{z})$, from which we assume the data points $\mathbf{x}_i$'s are generated. θ is the true, but unknown, set of parameters.

We can further introduce a prior distribution over $\mathbf{z}$, denoted as $p_\theta(\mathbf{z})$. The latent codes $\mathbf{z}$ will be sampled from that distribution which is assumed to follow an isotropic standard multivariate normal distribution:

$$p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z} \,|\, 0, \boldsymbol{I}) \ . \tag{1}$$

where $\boldsymbol{I}$ is the identity matrix of dimension $d \times d$. We can only observe $\mathbf{x}$, but we would like to infer the characteristics of $\mathbf{z}$. In other words, we want to compute the (true) posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. Using Bayes theorem,

$$p_\theta(\mathbf{x}|\mathbf{z})\, p_\theta(\mathbf{z}) = p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})\, p_\theta(\mathbf{x}) \ , \tag{2}$$

we have

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})\, p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})} \ , \tag{3}$$

where $p_\theta(\mathbf{x}, \mathbf{z})$ denotes the joint distribution. To obtain the true data distribution $p_\theta(\mathbf{x})$, we need to marginalize over the latent codes $\mathbf{z}$. The marginal likelihood of $\mathbf{x}$ for a candidate set of parameters θ is defined as the integral over the generative model $p_\theta(\mathbf{x}|\mathbf{z})\, p_\theta(\mathbf{z})$ (Kingma and Welling 2013):

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) \ d\mathbf{z} = \int p_\theta(\mathbf{x}|\mathbf{z})\, p_\theta(\mathbf{z}) \ d\mathbf{z} \, . \tag{4}$$

In practise, the use of non-linear deep neural nets makes the integral of the marginal likelihood intractable (i.e., no analytical solution) and the true posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$

unknown. While generative adversarial networks avoid the integral in Eq.(4) altogether, variational autoencoders use an approximation of the true, but unknown, posterior with a recognition model and a maximization of the log-likelihood lower bound (found by variational inference).

2.2 Variational approximation of the posterior

The exact posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is approximated with a tractable distribution known as the variational or approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$. In machine learning, a probabilistic encoder network e_ϕ (also called recognition model) is used to learn the set of variational parameters ϕ such that the tractable $q_\phi(\mathbf{z}|\mathbf{x})$ is used to perform approximate inference of the intractable posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. The parameters ϕ of the recognition model e_ϕ are learned jointly with the parameters θ of the generative model d_θ. Since the prior $p_\theta(\mathbf{z})$ is assumed to follow an isotropic Gaussian distribution, the encoder network will output two parameters: the mean $\boldsymbol{\mu}_z$ and covariance matrix $\boldsymbol{\Sigma}_z$. Formally, the approximate posterior is assumed to follow an isotropic multivariate Gaussian distribution of dimension d:

$$q_\phi(\mathbf{z}|\mathbf{x}) \;=\; \mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu}_z(\mathbf{x}), \boldsymbol{\Sigma}_z(\mathbf{x})\right),$$

where $\boldsymbol{\mu}_z(\mathbf{x}) \in \mathbb{R}^d$ and $\boldsymbol{\Sigma}_z(\mathbf{x}) = \boldsymbol{\sigma}_z^2 I \in \mathbb{R}^{(d\times d)}$ are such that:

$$\boldsymbol{\mu}_z(\mathbf{x}) = \begin{pmatrix} \mu_{z,1}(\mathbf{x}) \\ \vdots \\ \mu_{z,d}(\mathbf{x}) \end{pmatrix}, \quad \boldsymbol{\Sigma}_z(\mathbf{x}) = \begin{pmatrix} \sigma^2_{z,1}(\mathbf{x}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma^2_{z,d}(\mathbf{x}) \end{pmatrix}.$$

The variational parameters $\boldsymbol{\mu}_z(\mathbf{x})$ and $\boldsymbol{\Sigma}_z(\mathbf{x})$ are the outputs of a single layer neural network:

$$\begin{aligned} \mathbf{h}_z &= U\left(\mathbf{W}_1\mathbf{x} + \boldsymbol{b}_1\right), \\ \boldsymbol{\mu}_z(\mathbf{x}) &= \mathbf{W}_2\mathbf{h}_z + \boldsymbol{b}_2, \\ \ln \boldsymbol{\Sigma}_z(\mathbf{x}) &= \operatorname{diag}\left(\mathbf{W}_3\mathbf{h}_z + \boldsymbol{b}_3\right), \end{aligned}$$

where U is an element-wise activation function (e.g., sigmoid, rectified linear unit, scaled exponential linear unit, etc.), $\mathbf{W}$ is a weight matrix, and $\boldsymbol{b}$ is a bias vector. To ensure the positivity of $\boldsymbol{\sigma}_z^2$, the encoder network is asked to output its logarithm (the network may otherwise learn (by backpropagation) negative values for $\boldsymbol{\sigma}_z^2$).

2.3 Variational lower bound

Recall that we would like to approximate some posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ with a tractable distribution $q_\phi(\mathbf{z}|\mathbf{x})$ and make this approximate inference model $q_\phi(\mathbf{z}|\mathbf{x})$ as close as possible to the true, but intractable, posterior $p_\theta(\mathbf{z}|\mathbf{x})$. Mathematically, how to find such an approximation? We can derive a variational lower bound that will make the likelihood function, in Eq.(4), tractable. Such scheme involves the Kullback-Leibler (KL) divergence.

This divergence is a measure of dissimilarity between two probability distributions. If $p(y)$ and $q(y)$ are two densities, their reverse KL divergence from the approximate q to the true p, written as $D_{KL}(q\,||\,p)$, is defined as:

$$D_{KL}(q\,||\,p) \;=\; \mathbb{E}_{Y\sim q}\left[\ln\frac{q(Y)}{p(Y)}\right].$$

Or

$$D_{KL}(q\,||\,p) \;=\; \int q(y)\ln\left(\frac{q(y)}{p(y)}\right)dy\,.$$

Proposition 1. *The marginal log-likelihood is equal to the following sum*

$$\ln\left(p_\theta(\mathbf{x})\right) = D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\;||\,p_\theta(\mathbf{z}|\mathbf{x})\right) + \mathcal{L}(\mathbf{x};\theta,\phi)\;, \tag{5}$$

where $\mathcal{L}$ is called the variational lower bound:

$$\mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi}\left[-\ln q_\phi(\mathbf{z}|\mathbf{x}) + \ln p_\theta(\mathbf{x},\mathbf{z})\right]\;. \tag{6}$$

The proof (Kingma and Welling 2013) is given in Appendix 5. The first term in Eq.(5) is the KL divergence of the approximate from the true posterior. It is intractable due to $p_\theta(\mathbf{z}|\mathbf{x})$ but, since the KL divergence is a measure of difference between two probability densities, it is always positive. Therefore, the marginal log-likelihood (or evidence) is lower bounded by the variational lower bound $\mathcal{L}$:

$$\ln p_\theta(\mathbf{x}) \geq \mathcal{L}(\mathbf{x};\theta,\phi) = \mathbb{E}_{q_\phi}\left[-\ln q_\phi(\mathbf{z}|\mathbf{x}) + \ln p_\theta(\mathbf{x},\mathbf{z})\right]\;.$$

Minimizing the KL divergence between the exact posterior $p_\theta(\mathbf{z}|\mathbf{x})$ and the variational $q_\phi(\mathbf{z}|\mathbf{x})$ is then equivalent to maximizing this variational lower bound $\mathcal{L}$ during the training process. The complex inference problem is reduced to a simpler optimization problem:

$$\widehat{\theta},\widehat{\phi} \;=\; \arg\max_{\theta,\phi}\sum_{\mathbf{x}\in\mathbf{X}}\mathcal{L}(\mathbf{x};\theta,\phi)\;.$$

This is often referred to as the Evidence Lower Bound Optimization $ELBO(\mathbf{x};\theta,\phi)$. The closer the evidence lower bound is to the marginal likelihood, the closer the variational approximation will be to the true posterior distribution. For optimization, it is convenient to rewrite this variational (tractable) lower bound $\mathcal{L}(\mathbf{x};\theta,\phi)$ in the following way:

$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= \mathbb{E}_{q_\phi}\left[-\ln q_\phi(\mathbf{z}|\mathbf{x}) + \ln p_\theta(\mathbf{z}) + \ln p_\theta(\mathbf{x},\mathbf{z}) - \ln p_\theta(\mathbf{z})\right]\\ &= -\mathbb{E}_{q_\phi}\left[\ln\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})}\right] + \mathbb{E}_{q_\phi}\left[\ln\frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z})}\right]\\ &= -D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\;||\,p_\theta(\mathbf{z})\right) + \mathbb{E}_{q_\phi}\left[\ln p_\theta(\mathbf{x}|\mathbf{z})\right]\;.\end{aligned} \tag{7}$$

The first term in Eq.(7) is the KL divergence of the approximate posterior (i.e., the distribution of the encoder outputs) from the prior distribution $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}\,|\,0,\boldsymbol{I})$. Both of these functions are assumed to follow an isotropic Gaussian distribution and admit a closed-form solution.

Proposition 2. *The KL divergence involved in the definition of the variational lower bound is equal to*

$$D_{KL}\left(q_\phi\left(\mathbf{z}|\mathbf{x}\right)\,||\,p_\theta\left(\mathbf{z}\right)\right) \;=\; \frac{1}{2}\sum_{k=1}^{d}\Big(\sigma_{z,k}\left(\mathbf{x}\right)^2+\mu_{z,k}\left(\mathbf{x}\right)^2-\left(1+\ln\sigma_{z,k}^2\left(\mathbf{x}\right)\right)\Big)\;,$$

where d is the dimension of the latent space.

The proof is given in Appendix 5. This KL divergence term may be interpreted as a regularizer or encoding error that penalizes the learned distribution $q_\phi\left(\mathbf{z}|\mathbf{x}\right)$ when it diverges too much from the true prior distribution $p_\theta\left(\mathbf{z}\right)$. This penalty term is particularly useful for our generative process. By mapping each data point onto a distribution, instead of a single point in latent space, the variational encoding covers a certain area whose center and size are controlled by the means and standard deviations. However, if the outputted means are significantly different *and* the standard deviation are too small, there may still be some gaps in the latent space, leaving the decodable latent space discontinuous - which is exactly what we want to avoid. Ideally, our encodings should be distinct while still remaining as close as possible to each other. By introducing the KL divergence as a regularizer term, we force the encoder to find latent vectors that approximately follow a standard Gaussian distribution and therefore encourages the latent vectors to occupy a more centralized and uniform location.

While the KL divergence should ensure that the latent distribution is similar to the prior distribution, we do not want to end up describing every observation using the same unit Gaussian - this would treat every observation as having the same characteristics and fail to describe the original data. The second term in Eq.(7) prevents that by encouraging the decoder's output $\hat{\mathbf{x}}$ to reconstruct the original input $\mathbf{x}$. Formally, it represents the cross entropy of $p_\theta(\mathbf{x}|\mathbf{z})$ w.r.t. q_ϕ:

$$\mathbb{E}_{q_\phi}\left[\ln p_\theta(\mathbf{x}|\mathbf{z})\right]=\int \ln p_\theta(\mathbf{x}|\mathbf{z})\,q_\phi(\mathbf{z}|\mathbf{x})\,d\mathbf{z}\,.$$

The cross entropy of a random variable is the level of uncertainty inherent in the variable possible outcome and may be interpreted as a reconstruction loss. Statistically speaking, the decoder should parameterize a likelihood distribution that places enough probability mass on the true data.

While the KL divergence in Eq.(7) admits an analytical expression (see Proposition 2), the second term in Eq.(7) is usually approximated by L simulations:

$$\mathbb{E}_{q_\phi}\left[\ln p_\theta(\mathbf{x}|\mathbf{z})\right] \;\approx\; \frac{1}{L}\sum_{l=1}^{L}\ln p_\theta\left(\mathbf{x}|\mathbf{z}^{(l)}\right)\;, \tag{8}$$

where $\mathbf{z}^{(l)}\sim q_\phi\left(\mathbf{z}|\mathbf{x}\right)=\mathcal{N}\left(\mathbf{z}\,|\,\boldsymbol{\mu}_z(\mathbf{x}),\boldsymbol{\Sigma}_z(\mathbf{x})\right)$. This random sampling of the latent codes $\mathbf{z}^{(l)}$, illustrated in Fig.3, requires some extra attention.

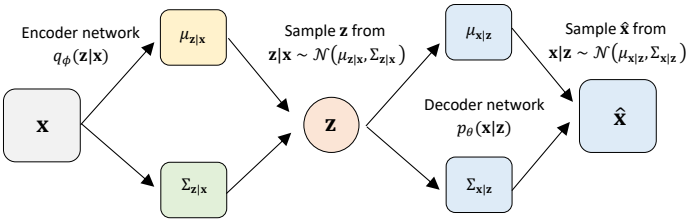

Figure 3: Illustration of the sampling process for a Gaussian encoder and decoder. The random node is represented by a circle.

2.4 Reparameterization trick

During the training process, each parameter of the network is evaluated with respect to the final output loss using a technique known as backpropagation. The random sampling of the latent code $\mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x})$ is making the lower-dimensional space stochastic, but NNs cannot backpropagate through stochastic nodes. Luckily, there exist a *reparameterization trick*, illustrated in Fig.4, which suggests that, instead of randomly sampling $\mathbf{z}$ from a Gaussian probability density, the latent codes can be sampled from an auxiliary random noise $\varepsilon \sim \mathcal{N}(0, \boldsymbol{I})$ shifted by the latent means vectors $\boldsymbol{\mu_z}(\mathbf{x})$ and scaled by the covariance matrix $\boldsymbol{\Sigma}_z(\mathbf{x})$:

$$\mathbf{z} = \boldsymbol{\mu_z}(\mathbf{x}) + \sqrt{\boldsymbol{\Sigma}_z(\mathbf{x})} \odot \varepsilon \,,\; \varepsilon \sim \mathcal{N}(0, \boldsymbol{I}) \,.$$

While the error term ε introduced is stochastic, the node $\mathbf{z}$ is now deterministic, enabling the backpropagation through it. Mathematically, we can now optimize $\mathcal{L}$ with respect to the parameters $\phi = \{\boldsymbol{\mu}, \Sigma\}$ of the variational posterior using the backpropagation technique. θ and ϕ are obtained by maximizing,

$$\sum_{\mathbf{x}\in\mathbf{X}} \tilde{\mathcal{L}}(\mathbf{x}; \theta, \phi) = \sum_{\mathbf{x}\in\mathbf{X}} \left(-D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \,||\, p_\theta(\mathbf{z})\right) + \frac{1}{L}\sum_{l=1}^{L} \ln p_\theta\left(\mathbf{x}|\mathbf{z}^{(l)}\right) \right) , \tag{9}$$

where $\mathbf{z}^{(l)} = \boldsymbol{\mu_z}(\mathbf{x}) + \sqrt{\boldsymbol{\Sigma}_z(\mathbf{x})} \odot \varepsilon^{(l)}$ and $\varepsilon^{(l)} \sim \mathcal{N}(0, \boldsymbol{I})$.

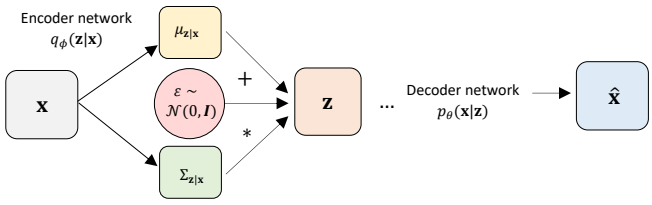

Figure 4: Illustration of the reparameterization trick. The random node is represented by a circle.

This random generation of $\mathbf{z}$ with $\varepsilon \sim \mathcal{N}(0, \boldsymbol{I})$ means the actual encoding will somewhat vary, for the same input $\mathbf{x}$, on every single pass through the sampling layer - even if the

means and standard deviations remain the same. This allows the decoder to be exposed to a range of variations of the encoding of the same input during training. The decoder thence learns that all nearby points in latent space should refer to a sample of the same class. By doing so, we are essentially enforcing a continuously meaningful latent space representation. Every point that is close to the encoded location of one of the input policies will be decoded to something similar to that original policy, making the VAE highly suitable for the generation of new samples. The number of simulations L in the gradient estimator in Eq.(8) and Eq.(9) can also be set to 1 - as the decoder is already exposed to variations (and the minibatch size m will be set large enough).

3 Heterogeneous data

Generative models, like the VAE we have just defined, perform poorly on real-world datasets (e.g., insurance portfolios) containing various types of features (i.e., (positive) numerical, binary, multinomial, and ordinal) with a variety of marginal distributions (e.g., multimodal, long tail). To deal with such heterogeneous datasets, we propose:

1. a one-hot encoding of categorical variables and softmax activation functions in the output layer of the decoder network.

2. a quantile (non-parametric) transformation to force the distribution of continuous variables to follow an unimodal normal distribution, and a min-max scaling to the range (-1,1) to ensure the normally transformed variables conform with the range of the tanh activation function used in the output layer of the decoder network.

3. a weighted combination of mean squared error and categorical cross-entropy for the reconstruction loss (that is minimized during the training process) to account for the different distributions of the continuous and categorical variables (i.e., normal and multinomial distributions).

3.1 One-hot encoding and softmax activations

Deep learning models require all inputs to be numeric. Insurance policies are, however, known to also contain information in categorical forms (e.g., geographic location, vehicle power, brand or color). Whether nominal (e.g., color) or ordinal (e.g., vehicle power), categorical variables must be encoded to numbers to train any deep learning model in Keras. Methods that map a set of alternative values are very few. Dummy (one-hot) encoding is one of the most common and simplest approach to represent categorical variables numerically. It maps each label to a binary vector as illustrated in the following example.

Example. For each unique modality in the categorical column *Zone*, a new dummy column is created, such that:

Zone		Zone_1	Zone_2	Zone_3	Zone_4	Zone_5	Zone_6	Zone_7
3		0	0	1	0	0	0	0
7		0	0	0	0	0	0	1
1	→	1	0	0	0	0	0	0
4		0	0	0	1	0	0	0
⋮		⋮	⋮	⋮	⋮	⋮	⋮	⋮
6		0	0	0	0	0	1	0

Table 1: Dummy encoding, example for nominal variable *Zone* with 7 modalities.

Entity embeddings (Guo and Berkhahn 2016) has been gaining momentum in the data field for addressing the weaknesses of dummy encoding. By learning a distributed representation of the categories, entity embedding offers the benefits of a uniform mapping and a smaller vector space than one-hot encoding. Yet, to produce the best numerical representation of a categorical variable, entity embeddings is computationally intensive and the fine-tuning of the NNs hyperparameters can quickly become time consuming - compared to the ease of implementation of dummy encoding. And even entity embedding takes the detour of an ordinal or one-hot encoder before passing the so-encoded input into a set of layers. For more details, we refer to Delong and Kozak (2021) who recently proposed a joint embedding to learn the numerical representation of categorical rating factors and fed it, together with the original numerical features, as inputs of a neural network for a *supervised* learning task on insurance data. Our intention to develop a straightforward and all-in-one *un*supervised autoencoder lead us to select dummy encoding for mapping our categorical variables to numeric vectors.

Example 3. Because our VAE architecture jointly models the categorical variables contained in the input data, the decoder's output layer will be sliced to extract the values corresponding to each categorical variable. Each slice/portion will then be individually passed through a softmax activation function.

3.2 Quantile transformation, min-max scaling, and tanh activation

In addition to the categorical variables pre-processing, scaling continuous features has been shown to benefit NNs optimization by increasing the speed of convergence of the gradient descent and preventing the optimization from getting stuck in a local optimum (see e.g., LeCun et al. (2002) and I. Goodfellow, Bengio and Courville (2016)). Standardization and normalization are the two most common approaches to scale continuous variables. However, in real-world datasets, continuous variables come in many forms (i.e., skewed, long-tailed, multimodal and/or non-normal distributions). The challenges of training NNs on multimodal distributions has already been highlighted in the literature. Among others, Pascanu, Mikolov and Bengio (2013) discussed how the vanishing gradient problem is exacerbated by multimodal distributions, leading to gradient saturation and slow convergence. Non-normal, skewed, and long-tail distributions can further affect the distributional

assumptions of our model and the balance of our training samples, which can eventually impact the model's performance and generalization. In short, if the input distributions are multimodal and/or non-Gaussian, the learned latent space might fail to effectively capture all the modes or the complexities of the data, resulting in incomplete, blurred and/or biased reconstructions of the original input.

Moreover, if at least one of the latent encoding dimensions were to capture and reproduce all the modes present in the data distribution, another issue related to the use of the KL divergence would arise. If the true posterior distribution of the latent variables $p(\mathbf{z}|\mathbf{x})$ given the input data is also multimodal (i.e., it has multiple distinct peaks or modes), minimizing the reverse KL divergence (during the training process) can lead to the collapse of the approximate posterior distribution towards a single mode. This will result in the generated samples being biased towards a specific mode, neglecting the other modes of the true data distribution. This mode-seeking behavior of the KL divergence can also impact the generation of new data points. Researchers have proposed various techniques to address this issue and encourage the model to capture and generate samples from multiple modes. Some of these techniques include using alternative divergence measures like the Jensen-Shannon divergence (Deasy, Simidjievski and Liò 2020), incorporating adversarial training (Mescheder, Nowozin and Geiger 2017), or employing more flexible generative models like Normalizing Flows (D. Rezende and Mohamed 2015).

To address these challenges, we propose a quantile transformation to compel continuous variables to follow an unimodal standard normal distribution. The quantile transformer formula $\Phi^{-1}(F(X))$ is built around the probability integral transform. It states that if X is a random variable with a continuous cumulative distribution function F then $U = F(X)$ is uniformly distributed on $[0, 1]$. If U is a random variable with uniform distribution on $[0, 1]$ then the quantile function $\Phi^{-1}(U) = \Phi^{-1}(F(X))$ has distribution Φ. In practice, this non-linear transformation is applied on each feature independently. An estimate of the cumulative distribution function (CDF) $F(X) = P(X \leq x)$ of a feature X is first estimated using a reference set of quantiles and an estimation of the data percentiles and CDF. The estimated percentiles are then used to map the feature values to a uniform distribution $[0, 1]$ by applying the inverse of the estimated CDF, i.e., $F^{-1}(X)$. The uniformly distributed values can further be mapped to the desired output distribution Φ, typically a standard normal distribution $\mathcal{N}(0, 1)$, using the associated quantile function Φ^{-1}. Note that new values that fall below or above the fitted range will be mapped to the bounds of the output distribution. The quantile transformer to a Normal distribution is detailed in Algorithm 2 in Appendix.

Note that L. Xu and Veeramachaneni (2018) proposed a mode-specific normalization to specifically address the issue related to multi-modal distributions in VAEs and GANs. The representation of a continuous column C_i after applying this mode-specific normalization becomes the concatenation of a normalized vector $\alpha_{i,j} = \frac{c_{i,j} - \eta_k}{4\phi_k}$ and m binary vectors indicating which of the k Gaussian $\mathcal{N}(\eta_k, \phi_k)$ each value $c_{i,j}$ belongs to. Such a normalization further increases the (already significant) input dimensionality by the number of gaussians used to *approximate* the multi-modal distributions. Whereas our non-parametric

quantile transformer maintains the numeric input dimension, tends to spread out the most frequent values, smooth out unusual distributions, and is less influenced by outliers than 'traditional' scaling methods.

We further apply a min-max transformation to scale the values x of the normally transformed variable X to the range $(-1, 1)$:

$$x_{scaled} = \frac{x - \min(x)}{\max(x) - \min(x)} \times (b_s - b_i) + b_i \qquad \text{for } b_i = -1, b_s = 1\,.$$

The range was chosen based on the activation function used in the output layer of our decoder network. While a min-max scaling to the range $(0, 1)$ could be considered together with a sigmoid activation function, the hyperbolic tangent ($tanh$), whose range is from -1 to 1, is often recommended over the sigmoid function for having stronger (and non-biased) gradients (see e.g., LeCun et al. (2002)).

Because our VAE architecture jointly models the numerical and categorical variables contained in the input data, the decoder's output layer will be sliced to extract the values corresponding to the numerical variables before passing them through a tanh activation function.

3.3 Heterogeneous reconstruction loss

Recall that the parameters of the VAE are trained using a negative log-likelihood (reconstruction loss) with a regularizer (KL divergence):

$$\sum_{\mathbf{x}\in\mathbf{X}} \tilde{\mathcal{L}}(\mathbf{x};\theta,\phi) = \sum_{\mathbf{x}\in\mathbf{X}} \left(-D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x}) \,||\, p_\theta(\mathbf{z})\right) + \frac{1}{L}\sum_{l=1}^{L} \ln p_\theta\left(\mathbf{x}|\mathbf{z}^{(l)}\right) \right).$$

The reconstruction loss $\ln p_\theta\left(\mathbf{x}|\mathbf{z}^{(l)}\right)$ can easily be transformed depending on the distribution assumption of the decoder outputs, and the choice of the decoder depends upon the nature of our input $\mathbf{X}$. Because we assume $\mathbf{X}$ to be a real-world dataset containing various types of features with a variety of marginal distributions, the decoder output - and the associated reconstruction loss - should reflect its heterogeneity. Assume the first p_c dimensions of each input sample $\mathbf{x} \in \mathbf{X}$ contain the observed values of the scaled numeric variables $(X_j)_{j=1,\dots,p_c}$. The following $p_b = p - p_c$ dimensions of each input sample $\mathbf{x} \in \mathbf{X}$ contain the observed values of the b dummy encoded categorical variables $(X_k)_{k=1,\dots,b}$ such that the dataset $\mathbf{X} = (\mathbf{x}_i)_{i=1,\dots,n} = \left(\{x_{i,j}\}_{j=1,\dots,p_c}, \{x_{i,k}\}_{j=p_c+1,\dots,p_b}\right)_{i=1,\dots,n}$ where $p_b = \sum_{k=1}^{b} C_k^*$ and C_k^* is the number of modalities in the k^{th} categorical variable.

The *scaled* numeric variables are assumed to follow a standard Normal distribution. Maximizing the likelihood of a model whose predictions are assumed to follow a Gaussian distribution is equivalent to minimizing mean squared error. In addition, each one of the dummy encoded categorical variables are treated as a multi-class classification problem. Maximizing the likelihood of a model whose predictions are assumed to follow a Multinomial distribution is equivalent to minimizing categorical cross-entropy (also called softmax

loss). To this end, we propose a reconstruction loss that combines the mean squared error and the categorical cross-entropy.

Mathematically, if $\mathbf{x}$ is a vector in $\mathbb{R}^{p_c}$, a natural choice for the decoder is the multivariate Gaussian distribution with e.g., a diagonal covariance structure (to reduce the number of parameters to estimate)

$$p_\theta(\mathbf{x}|\mathbf{z}) \quad = \quad N\left(\mathbf{z} \,|\, \boldsymbol{\mu}_x(\mathbf{z}), \boldsymbol{\Sigma}_x(\mathbf{z})\right) \,,$$

where $\boldsymbol{\mu}_x\left(\mathbf{z}\right) \in \mathbb{R}^{p_c}$ and $\boldsymbol{\Sigma}_x\left(\mathbf{z}\right) = \boldsymbol{\sigma}_x^2\left(\mathbf{z}\right) \boldsymbol{I} \in \mathbb{R}^{p_c \times p_c}$ are the outputs of a single layer neural network with weights $\mathbf{W}$ and biases $\mathbf{b}$:

$$\begin{aligned}
\mathbf{h}_x &= \mathrm{SELU}\left(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1\right) \,, \\
\boldsymbol{\mu}_x\left(\mathbf{z}\right) &= \tanh\left(\mathbf{W}_2\mathbf{h}_x + \mathbf{b}_2\right) , \\
\ln \boldsymbol{\Sigma}_x\left(\mathbf{z}\right) &= \mathrm{diag}\left(\tanh\left(\mathbf{W}_3\mathbf{h}_x + \mathbf{b}_3\right)\right) \,,
\end{aligned}$$

where

$$\boldsymbol{\mu}_x\left(\mathbf{z}\right) = \begin{pmatrix} \mu_{x,1}(\mathbf{z}) \\ \vdots \\ \mu_{x,p_c}(\mathbf{z}) \end{pmatrix} \,, \quad \boldsymbol{\Sigma}_x\left(\mathbf{z}\right) = \begin{pmatrix} \sigma^2_{x,1}(\mathbf{z}) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma^2_{x,p_c}(\mathbf{z}) \end{pmatrix} \,,$$

and

$$\begin{cases} \mathrm{SELU}(x) = \lambda x & \text{if } x > 0 \quad \text{with } \lambda \approx 1.05 \\ \mathrm{SELU}(x) = \lambda\alpha\left(e^x - 1\right) & \text{if } x \leq 0 \quad \text{with } \alpha \approx 1.67 \end{cases} \,, \qquad \tanh\left(x\right) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \,.$$

In this case, the probabilistic decoder has density:

$$p_\theta\left(\mathbf{x}|\mathbf{z}\right) = \frac{\exp\left(-\frac{1}{2}\sum_{k=1}^{p_c} \frac{\left(x_k - \mu_{x,k}(\mathbf{z})\right)^2}{\sigma_{x,k}(\mathbf{z})^2}\right)}{\left(2\pi\right)^{\frac{1}{2}p_c} \sqrt{\prod_{k=1}^{p_c} \sigma^2_{x,k}\left(\mathbf{z}\right)}} \,,$$

with a log-likelihood equal to

$$\ln p_\theta(\mathbf{x}|\mathbf{z}) = -\frac{p_c}{2}\ln\left(2\pi\right) - \frac{1}{2}\sum_{k=1}^{p_c} \ln \sigma^2_{x,k}(\mathbf{z}) - \frac{1}{2}\sum_{k=1}^{p_c} \frac{\left(x_k - \mu_{x,k}(\mathbf{z})\right)^2}{\sigma_{x,k}(\mathbf{z})^2}$$

If we both consider a Gaussian encoder and decoder, network parameters are obtained by maximizing

$$\begin{aligned}
\sum_{\mathbf{x}\in\mathbf{X}} \tilde{\mathcal{L}}\left(\mathbf{x};\theta,\phi\right) &= \sum_{\mathbf{x}\in\mathbf{X}} \left(-D_{KL}\left(q_\phi\left(\mathbf{z}|\mathbf{x}\right) \,||\, p_\theta\left(\mathbf{z}\right)\right) + \frac{1}{L}\sum_{l=1}^{L} \ln p_\theta\left(\mathbf{x}|\mathbf{z}^{(l)}\right)\right) \\
&= \sum_{\mathbf{x}\in\mathbf{X}} \frac{1}{2}\sum_{k=1}^{d} \left(1 + \ln \sigma^2_{z,k}(\mathbf{x}) - \sigma_{z,k}\left(\mathbf{x}\right)^2 - \mu_{z,k}\left(\mathbf{x}\right)^2\right) \\
&+ \sum_{\mathbf{x}\in\mathbf{X}} \frac{1}{L}\sum_{l=1}^{L} \left(-\frac{p_c}{2}\ln\left(2\pi\right) - \frac{1}{2}\sum_{k=1}^{p_c} \ln \sigma^2_{x,k}\left(\mathbf{z}^{(l)}\right) - \frac{1}{2}\sum_{k=1}^{p_c} \frac{\left(x_k - \mu_{x,k}\left(\mathbf{z}^{(l)}\right)\right)^2}{\sigma_{x,k}\left(\mathbf{z}^{(l)}\right)^2}\right) \,,
\end{aligned}$$

where

$$\mathbf{z}^{(l)} = \boldsymbol{\mu}_{\boldsymbol{z}}(\mathbf{x}) + \sqrt{\boldsymbol{\Sigma}_z\left(\mathbf{x}\right)}\boldsymbol{\epsilon}^{(l)} \, .$$

Maximizing the log-likelihood $\ln p_\theta(\mathbf{x}|\mathbf{z})$ of a model whose predictions are assumed to follow a Gaussian distribution is equivalent to minimizing the mean squared error:

$$L_{mse} = \sum_{i=1}^{n}\sum_{j=1}^{p_c} \|x_{i,j} - \hat{x_{i,j}}\|^2 \, .$$

In parallel, if $\mathbf{x}$ is a vector distributed according to a multinomial law, a natural choice for the decoder is:

$$p(\mathbf{x}|\mathbf{z}) = \prod_{j=1}^{C^*} y_j^{x_j}\left(\mathbf{z}\right),$$

where $y(\mathbf{z}) = \left(y_j(\mathbf{z})\right)_{j=1,\ldots,C_k^*}$ are the outputs of a single layer neural network:

$$\begin{aligned} \mathbf{h}_x &= \text{SELU}\left(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1\right), \\ y(\mathbf{z}) &= \text{softmax}\left(\mathbf{W}_2\mathbf{h}_x + \mathbf{b}_2\right) \, , \end{aligned}$$

where

$$\text{softmax}\left(x\right) = \left(\frac{e^{x_i}}{\Sigma_j^{C_k^*} e^{x_j}}\right)_{i=1,\ldots,C_k^*}, \tag{10}$$

is applied to the output scores inferred by the NN for each class $i = 1, \ldots, C_k^*$ in the multi-class variable X_k. The likelihood of one observation is in this particular case equal to the cross-entropy:

$$\ln p\left(\mathbf{x}|\mathbf{z}\right) = \Sigma_{j=1}^{C_k^*} x_j \ln\left(y_j(\mathbf{z})\right) \, . \tag{11}$$

Since the categorical variables $\left(X_k\right)_{k=1,\ldots,b}$ are one-hot encoded, there is only one element, denoted $x^{(p)}$, in the target vector $\mathbf{x} = \left\{x_1, \ldots, x_{C_k^*}\right\}$ which is not zero. We can rewrite Eq.(11) as follow:

$$\ln p\left(\mathbf{x}|\mathbf{z}\right) = \ln\left(\frac{e^{x^{(p)}}}{\sum_{j=1}^{C_k^*} e^{x_j}}\right)$$

Maximizing the likelihood of a model whose predictions are assumed to follow a Multinomial distribution is equivalent to minimizing this categorical cross-entropy.

During the training, the error calculated by the loss function is propagated backward through the network to update the network and variational parameters and eventually find the optimal values (i.e., the values that minimise the error). The goal of our VAE is not only to learn to reconstruct the heterogeneous insurance portfolio given its latent representation, but also to generate new meaningful policies. Ensuring the VAE learns a continuously meaningful latent space is the goal of the KL divergence regularizer term. Optimizing the reconstruction log-likelihood and regularizer simultaneously encourages the

latent state to have distributions close to the prior $\mathcal{N}(0,1)$ but deviating when necessary to describe salient features of the input portfolio. Higgins et al. (2017) proposed the β-VAE model to give higher weight to the KL divergence term with a parameter $\beta > 1$ in case the latent distributions appeared very tight. Our combined mean squared error, categorical cross-entropy, and KL divergence seem to achieve a very well balance. During training, one loss may dominate but when it (relatively) becomes low enough the other losses are factored in more.

The architecture of our variational autoencoder is detailed in Algorithm 1. For the probabilistic encoder, we use a single hidden layer with Gaussian output. For the probabilistic decoder, we propose a combination of Gaussian and Multinomial outputs, depending on the type of data (continuous or discrete). The algorithm starts drawing batches by taking the first m instances (first batch) from the randomly shuffled dataset, calculates the average error and updates the parameters. This completes one training step (i.e., one iteration or gradient update). When all batches have been drawn from the training dataset (i.e., the model has been trained on the entire dataset), that concludes one epoch.

Algorithm 1 Variational autoencoder for heterogeneous insurance portfolio.

Input: insurance portfolio $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}_p = \left\{\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}_{p_c}, \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}_b\right\}_p$ of dimension $n \times p$ (i.e., n insurance policies characterised by p_c continuous rating factors and b discrete rating factors).
Preprocessing step:
$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}_{p_c} \leftarrow$ Normal quantile transformation and min-max scaling to $(-1, 1)$.
$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}_{p_b} \leftarrow$ One-hot encoding (which increases the dimension from b to p_b).
Initialization:
$\alpha, m, N \leftarrow$ Set up hyperparameters (learning rate α, batch size m, and number of epochs N).
$\theta, \phi \leftarrow$ Initialize parameters.
repeat
  $\mathbf{X}^m \leftarrow$ Draw random minibatch $\mathbf{X}^m = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ of $m \leq n$ observations/policies/data points.
  for each $\mathbf{x} \in \mathbf{X}^m$ **do**
    $\boldsymbol{\epsilon} \leftarrow$ Generate random noise $\boldsymbol{\epsilon} = \{\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_m\}$ from the auxiliary noise distribution $p(\boldsymbol{\epsilon}) \sim \mathcal{N}(0, \boldsymbol{I})$.
    $\boldsymbol{\mu}, \log\left(\boldsymbol{\sigma}^2\right) \leftarrow e_\phi(\mathbf{x})$.
    for $l \leftarrow 1$ to m **do**
      $\mathbf{z}^{(l)} \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}^{(l)}$.
    end for
    $\mathcal{L}_{rec}^{(continuous)}(\mathbf{x}) \leftarrow \frac{1}{2m} \sum_{l=1}^{m} \| \{\mathbf{x}\}_{p_c} - \{\hat{\mathbf{x}}\}_{p_c} \|^2$ with $\{\hat{\mathbf{x}}\}_{p_c} = f\left(\left\{d_\theta\left(\mathbf{z}^{(l)}\right)\right\}_{p_c}\right)$ and $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.
    $\mathcal{L}_{rec}^{(discrete)}(\mathbf{x}) \leftarrow \frac{1}{m} \sum_{l=1}^{m} \left(\sum_{k=1}^{b} \left(\sum_{j=1}^{C_k^*} \{x_j\}_{C_k^*} \ln\left(\{\hat{x_j}\}_{C_k^*}\right)\right)\right)$ with $\{\hat{x_j}\}_{C_k^*} = g\left(\left\{d_\theta\left(\mathbf{z}^{(l)}\right)\right\}_{C_k^*}\right)$ and $g(x) = \frac{e^x}{\sum_j^{C_k^*} e^{x_j}}$.
    $\mathcal{L}_{KL}(\mathbf{x}) \leftarrow -\frac{1}{2m} \sum_{l=1}^{m} \sum_{j=1}^{d} \left(1 + \ln \sigma_{z,j}^2(\mathbf{x}) - \sigma_{z,j}(\mathbf{x})^2 - \mu_{z,j}(\mathbf{x})^2\right)$.
  end for
  $\mathbf{g} \leftarrow \nabla_{\theta,\phi} \widetilde{\mathcal{L}}^m(\mathbf{X}^m, \epsilon; \theta, \phi)$ Compute the gradients of minibatch estimator w.r.t. θ and ϕ.
  $\theta, \phi \leftarrow$ Update parameters using gradients $\mathbf{g}$
    $\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{\mathbf{x} \in \mathbf{X}^m} \mathcal{L}_{rec}(\mathbf{x})$ with $\mathcal{L}_{rec}(\mathbf{x}) = \mathcal{L}_{rec}^{(continuous)}(\mathbf{x}) + 2 \times \mathcal{L}_{rec}^{(discrete)}(\mathbf{x})$
    $\phi \leftarrow \phi - \alpha \nabla_\phi \sum_{\mathbf{x} \in \mathbf{X}^m} \left(\mathcal{L}_{rec}(\mathbf{x}) + \beta \mathcal{L}_{KL}(\mathbf{x})\right)$
until convergence of parameters (θ, ϕ)
return θ, ϕ

4 Model training

4.1 Motor insurance portfolio

The insurance portfolio X contains n policies characterised by p_c continuous variables and b categorical variables. In our application, each policy in the Swedish insurance company database[2] is described by two continuous variables, the *OwnersAge* and *VehicleAge*, and by three categorical variables, the *Gender* (with 2 modalities), *Zone* (with 7 modalities), and *Class* (with 7 modalities), detailed in Table 2. The portfolio further contains, for each policy $i = 1, \ldots, n$, an ordinal variable reporting the number of claims $N_i \in (0, 1, 2)$, and a continuous variable ν_i indicating the contract duration (exposure). After selecting contracts with a duration $\nu_i \in]0, 12]$ months, the dataset counts $n = 62\,289$ policies. As expected for a motor portfolio it is highly imbalanced, with 0 claims accounting for 98.95% of the policies, 1 claim for 1%, and 2 claims for 0.05%.

Rating factors	**Class**	**Class description**
Gender	M	Male (ma)
	K	Female (kvinnor)
Geographic area	1	Central and semi-central parts of Sweden's three largest cities
	2	Suburbs plus middle-sized cities
	3	Lesser towns, except those in 5 or 7
	4	Small towns and countryside
	5	Northern towns
	6	Northern countryside
	7	Gotland (Sweden's largest island)
Vehicle class	1	EV ratio -5
	2	EV ratio 6-8
	3	EV ratio 9-12
	4	EV ratio 13-15
	5	EV ratio 16-19
	6	EV ratio 20-24
	7	EV ratio 25-

Table 2: Description of the modalities of the categorical variables *Zone*, *Class*, and *Gender*.

The *OwnersAge*, *VehicleAge* and *Duration* variables all follow a multi-modal distribution (see Fig.5). We use the quantile (non-parametric) transformation, detailed in Algorithm 2, to map the values of each continuous variable to an unimodal standard normal distribution. We then apply a min-max transformation to scale the normally distributed data to the range $(-1, 1)$. The distributions before and after scaling are showed in Fig.5.

[2] Available on the companion website of the book by Ohlsson and Johansson (2010).

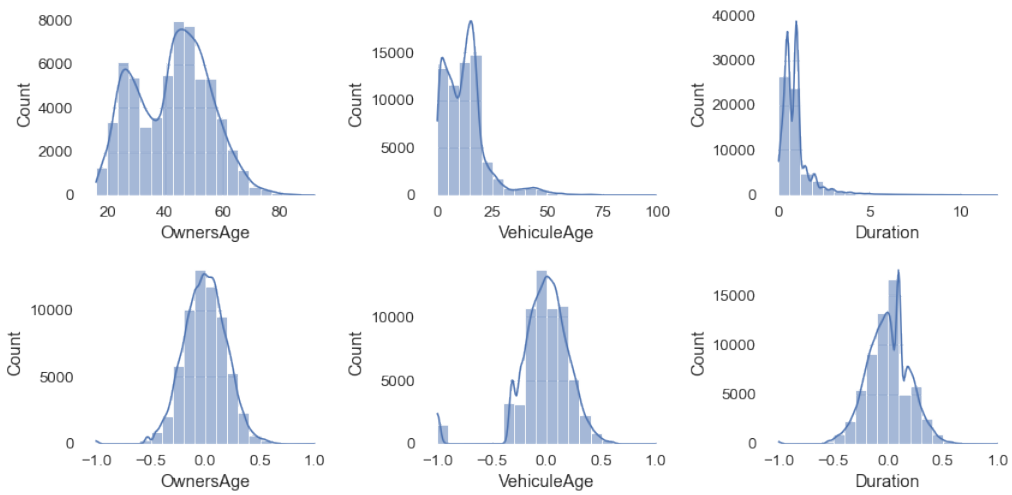

Figure 5: Distributions of *OwnersAge*, *VehicleAge*, and *Duration* before (upper plots) and after (lower plots) applying a normal quantile transformation and a min-max scaling to the range $(-1, 1)$.

The categorical variables *Zone*, *Class*, *Gender* and *NumberClaims* are one-hot encoded with, respectively, 7, 7, 2, and 3 dummy columns. A quick analysis on our insurance tabular data reveals (see Fig.6) a highly imbalanced dataset in which the number of policies in one modality (e.g. male policyholders) greatly outnumbers the policies in another (e.g. female policyholders). Vehicles with the highest power ratio (i.e., Class 7), as well as Northern geographic location (Zone 5, 6, and 7) are also strongly under-represented.

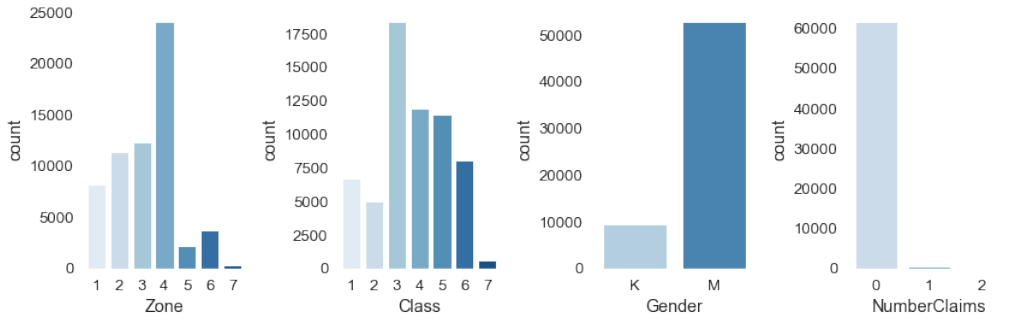

Figure 6: Imbalanced classes in variables *Zone*, *Class*, *Gender* and *NumberClaims*.

Imbalanced data can lead to difficulties in learning a meaningful latent space. If certain classes are underrepresented in the training data, the latent space representation may not be well-distributed or separated for the different classes, leading to challenges in generating diverse and high-quality samples for the minority classes. We recommend the use of smaller batch sizes during the training of our VAE as larger batch sizes might inadvertently amplify the influence of the majority class(es) and lead to poor performance on

the minority class(es). In the following Sections, Fig.16 and Fig.10 show that our VAE was able to handle imbalanced classes and generate new policies representing all classes, including those with fewer observations.

4.2 Model architecture and hyperparameters set up

We train our variational autoencoder model (see Algorithm 1) on the pre-processed insurance portfolio with $n = 62\,289$ rows, $p_c = 3$ quantile and min-max transformed columns, and $p_b = 19$ dummy columns. The key hyperparameters involved in the training of our VAE are the latent space dimension, the learning rate, the batch size, the number of epochs, the loss function and regularization term(s), the network initialization, the encoder and decoder architectures and their respective activation functions. Experimentation and tuning are often necessary to find optimal values for these hyperparameters depending on the input data and its complexity, and the desired properties of the learned latent space.

The architecture of our encoder and decoder networks (see Fig.12 in Appendix) refer to the structure and configuration of their hidden layer(s) and nonlinear activation functions. Since our VAE is trained to learn to reconstruct the input, the input and output space are both of dimension $p = 22$ (i.e., the first layer of the encoder and the last layer of the decoder both have 22 neurons). Our intention to build a simple and straightforward autoencoder led us to choose a shallow architecture with a single hidden layer in the encoder NN. The mapping of the original input to the latent space is done through the use of a Scaled Exponential Linear Unit (SELU) activation function. By definition, an activation function determines whether a neuron should be activated or not by combining the sum of the weighted inputs and a bias term. SELUs are activation functions that induce self-normalization (i.e., automatically converge to a zero mean and unit variance) and are a common choice for NNs, not only because they help alleviate the vanishing gradient problem (like ReLUs), but also because they cannot die (contrary to ReLUs). SELUs can also get below 0 allowing the system to have a zero average output which can improve the speed of convergence (Glorot, Bordes and Bengio 2011).

The input portfolio is thence mapped to its latent representation through a single hidden layer with a SELU activation function and a Lecun normal initializer. The variational parameters are also initialized with zeros (for the means), and a small constant value set to 0.1 (for the log variances). We choose a latent space of dimension 15, thereby reducing the dimension of about 1/3. The latent (hidden) representation space is typically referred to as a bottleneck layer (Tishby, Pereira and Bialek 2000) - as it contains fewer neurons than the layer below or above it. This encourages the NN to learn a compact, but meaningful, representation of the data. In the instance of motorcycle insurance policies, the latent encoding may contain a hidden representation of the vehicle characteristics and its owner's risk profile.

Our decoder mirrors the architecture of the encoder but operates in reverse, generating outputs that match the input data. The latent space is mapped back to the initial space with a SELU activation function and a Lecun normal initializer. Because of the heterogeneity in the distributions of our input data (see Section 3.3), the decoding layer outputs

are sliced before being passed through tanh and softmax activation functions with Xavier initializers in the final layer of our decoder.

The RMSprop optimizer with default learning rate value $\alpha = 0.001$ (which determines the speed of the optimizer's descend on the error curve) gave better results on our training data than the often-recommended Adam optimizer.

The main hyperparameters of the (mini-batch) gradient descent optimization algorithm are the batch size and the number of epochs. The larger the datasets, the more time-consuming and computationally expensive each training step becomes. In Keras, batches of the randomly suffled dataset are instead use for each gradient update. Smaller batch sizes (e.g., 16, 32, 64) introduce more noise and randomness to the (more frequent) parameter updates, which can have regularizing effects and improve the model's generalization - and therefore the quality of synthetic policies. However, training with smaller batches requires more iterations and may take longer overall. Conversely, larger batch sizes (e.g., 128, 256, 512, 1024) can accelerate training but may result in less noisy weight updates and potentially hinder the model's generalization. After experimenting with different batch sizes and monitoring the model's performance (i.e., reconstruction accuracy, latent space properties and synthetic data quality), we find that a batch size of 256 and 75 epochs ensure that each batch provides enough representative examples from the minority classes while remaining within acceptable computation time (20 minutes).

4.3 Reconstruction quality

Thanks to the KL divergence regularizer term, our VAE was able to learn multivariate independent latent variables. As shown in Fig.13 (in Appendix), the linear correlations between the latent encodings are close to 0. The latent variables are also centered around 0 with standard deviations close to 1 (see Table 9 and Fig.15 in Appendix).

Thanks to the reconstruction loss, our VAE model was also able to reconstruct the original data fairly accurately. To evaluate the quality of the reconstructed data, a reverse scaling is applied to the decoder's output $\hat{\mathbf{x}}$; a reverse one-hot-encoding is applied to the reconstructed categorical classes, and a reverse min-max transformation followed by a reverse quantile transformation is applied to the reconstructed continuous variables. As shown in Fig.16 (in Appendix), the frequency of each observed category in the reconstructed data matches with that of the original data. Fig.17 (in Appendix) also shows that the densities of the original and reconstructed continuous data are almost perfectly overlapping.

The following Fig.7 and Fig.8 stress the beneficial effect of the quantile transformation on the distributions of both the reconstructed and synthetic continuous data. Without applying our normal quantile transformation to the continuous variables (assumed to follow non-Gaussian multimodal distributions), the distributions of the reconstructed and synthetic variables deviate from the original. Fig.7 shows that the reconstructed distributions (in dark blue) clearly fail to capture the multiple modes of the variable *Duration* and do not perfectly capture the two modes of the *VehicleAge*'s distribution.

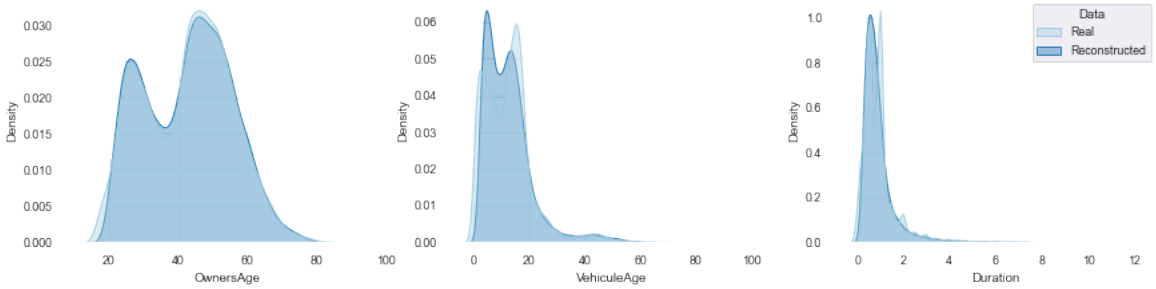

Figure 7: Reconstructed (in dark blue) *Duration*, *OwnersAge* and *VehicleAge* without quantile transformation compared to their real distributions (in light blue).

The impact of multimodal input distributions is even more pronounced when looking at the synthetic data distributions in Fig.8 obtained by feeding random noises to the decoder's network (see Section 4.4). When training our VAE on our pre-processed data without a normal quantile transformation scaler, the synthetic distributions (in dark blue) completely fail to reproduce the modes of the original distributions (in light blue).

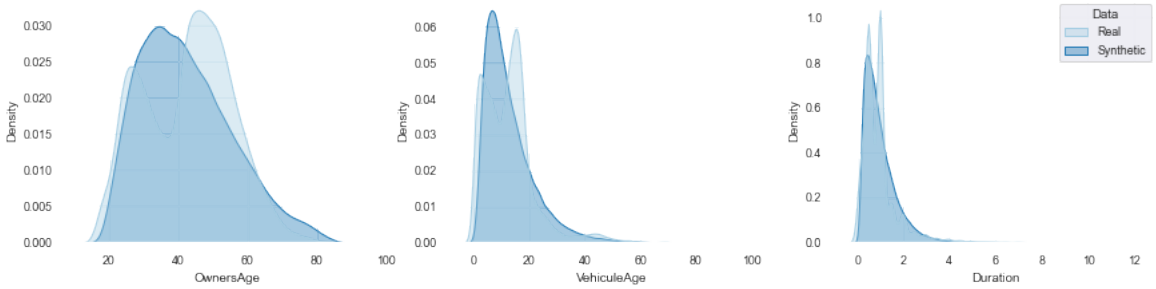

Figure 8: Synthetic (in dark blue) *Duration*, *OwnersAge* and *VehicleAge* without quantile transformation compared to their real distributions (in light blue).

The normal quantile transformation we proposed to scale our numeric data greatly improves the distributions of both the reconstructed (see Fig.17 in Appendix) and synthetic continuous data (see Fig.11 in Section 4.4).

4.4 Synthetic data generation

The trained decoder network can further be used as an unsupervised generative model to create new insurance policies that mimic the real policies patterns. This can become a valuable tool when real-world data is hard to collect or cannot be used due to privacy or regulatory concerns. Whether the data is real or artificial is irrelevant, what matters are the structure, statistical characteristics and patterns inside the data. Ideally, synthetic data should preserve the data distributions and correlation structure.

In their seminal papers, neither Kingma and Welling (2013) nor D. J. Rezende, Mohamed and Wierstra (2014) seem to specify the best way to generate new data points.

In their research on β-VAE, Higgins et al. (2017) do mention that *the most informative latent units of β-VAE have the highest KL divergence from the unit Gaussian prior*, confirming at least that the posterior distribution does not perfectly match the Gaussian prior $p(\mathbf{z}) \sim \mathcal{N}(0, \boldsymbol{I})$ and the difference matters. As illustrated in the left plot of Fig.9, although the latent variables are roughly centered around 0 with standard deviations close to 1 (see Table 9 in Appendix), they do not necessarily fit neatly into a standard bell curve. Applying a normal quantile transformation $\Phi^{-1}(F(Z))$ to each latent variable Z ensures their densities are bell-shaped (see right plot of Fig.9).

With that knowledge in mind, we generate new policies by first sampling from the prior distribution $p(\mathbf{z}) \sim \mathcal{N}(0, I)$. By doing so, samples $\mathbf{z}$ from the whole data distribution is drawn. Because of the non-bell shaped distributions of the latent codes $\mathbf{z}$, sampling from $\mathcal{N}(0, I)$ might fail and under sample or omit some regions of the true distribution, oversample others, and even provide inputs not covered by the aggregate posterior. We thence propose to apply, to the normally distributed samples $\mathbf{z}$, the inverse quantile transformations of the latent variables.

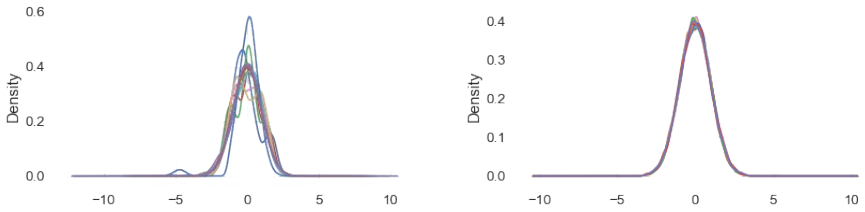

Figure 9: Densities of the $d = 15$ latent variables, before and after normal quantile transformation.

The new samples $\mathbf{z}$ are then feeded as input to the decoder. The quality of the so-generated synthetic policies are evaluated with respect to the statistical characteristics of the original portfolio. Fig.10 and Fig.11 shows that the distributions of the synthetic data matches those of the original data.

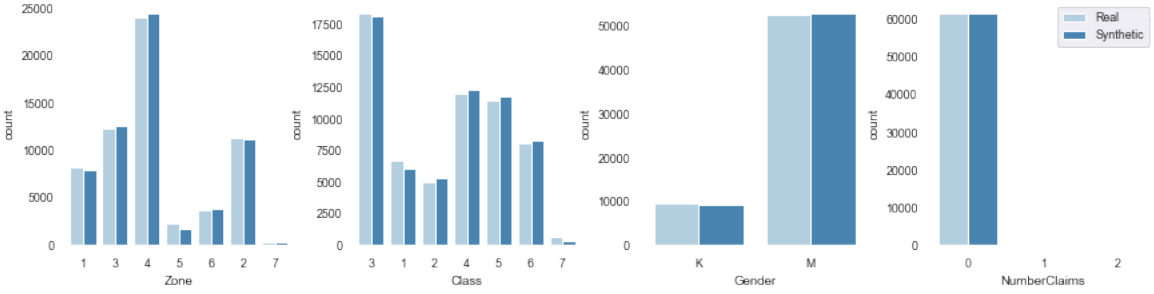

Figure 10: Comparison of real (light blue) and synthetic (dark blue) categorical data distributions (frequency of each observed category).

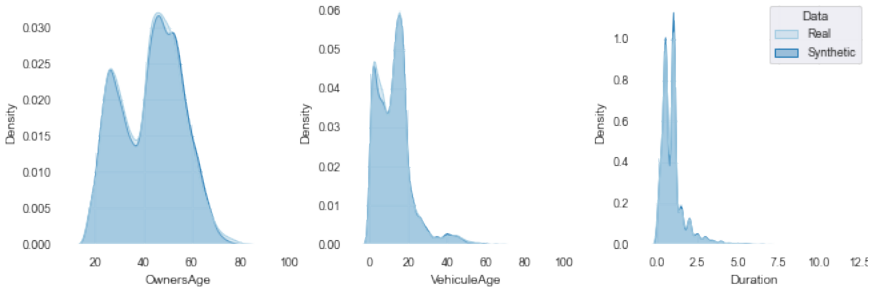

Figure 11: Comparison of real (light blue) and synthetic (dark blue) continuous data distributions.

We evaluate the ability of our VAE model to capture pairwise correlations between continuous variables by computing Pearson and Spearman correlations. Table 3 shows the original correlations between continuous variables are preserved in the synthetic dataset. The scores in Table 3 were obtained by computing $1 - 0.5\left|\text{corr}^{real} - \text{corr}^{synthetic}\right|$.

Pearson ρ	OwnerAge	VehicleAge	Spearman r_s	OwnerAge	VehicleAge
VehicleAge	real: 0.0755 synthetic: 0.077 score: 99.92%		VehicleAge	real: 0.0671 synthetic: 0.0777 score: 99.47%	
Duration	real: 0.1173 synthetic: 0.1334 score: 99.19%	real: -0.0278 synthetic: 0.0526 score: 95.98%	Duration	real: 0.1694 synthetic: 0.1689 score: 99.98%	real: 0.0787 synthetic: 0.0608 score: 99.11%

Table 3: Pearson and Spearman correlations between real and synthetic continuous data.

The synthetic and original continuous data distributions are further compared using the Kolmogorov-Smirnov (KS) statistic-based metric. The results are reported in Table 4. The output values are quite close to 1, indicating that the distributions (the real CDFs and the synthetic CDFs values) are almost identical.

	OwnersAge	VehicleAge	Duration
KS metric	0.98	0.99	0.96

Table 4: Kolmogorov-Smirnov statistic-based metric between real and synthetic continuous data columns.

The boundaries adherence and statistical similarities reported in Table 5 are also verified. All synthetic data are within the min and max bounds of the real data, and we observe highly similar values for the mean, standard deviations, and different quantiles.

	t	OwnersAge		VehicleAge		Duration	
		real	synthetic	real	synthetic	real	synthetic
Statistical similarity	mean	42.5438	42.8048	12.5584	12.6940	11.0079	0.9938
	std	12.9549	12.9446	9.6846	9.6194	1.0621	0.9759
	min	16.0	16.0	0.0	0.0	0.002740	0.002740
	25%	31.0	31.0	5.0	6.0	0.4904	0.4959
	50%	44.0	44.0	12.0	12.0	0.8877	0.9178
	75%	52.0	53.0	16.0	17.0	1.0	1.0
	max	92.0	83.0	99.0	87.0	11.9945	11.7653
Statistical score = $1 - \frac{\lvert {}^{t}real - {}_{t}fake \rvert}{\max^{real} - \min^{real}}$	mean	1.0		1.0		1.0	
	std	1.0		1.0		0.99	
	median	1.0		1.0		1.0	

Table 5: Statistic t similarity between real and synthetic continuous data and statistical score for the mean, standard deviation, and median.

The quality of the synthetic categorical data is compared using the Chi-Squared test based metric. The results reported in Table 6 are once again an indication of the high-quality of our synthetic data. Note, however, that due to the low number of policies (25 out of 62 289) with two reported claims, we often fail to generate qualitative artificial policies with two claims (hence the 67% for the *NumberClaims* modalities coverage).

	Zone	Class	Gender	NumberClaims
Chi-Squared test	0.99	0.98	1.0	1.0
Modalities coverage	100%	100%	100%	67%

Table 6: Chi-squared test, and classes coverage between real and synthetic categorical variables.

We further compare the contingency similarity of the categorical variables in Table 7. These scores were obtained by means of a contingency table which displays frequencies for combinations of two categorical columns.

	Zone	Class	Gender
Class	0.96%		
Gender	0.98%	0.97%	
NumberClaims	0.98%	0.98%	0.99%

Table 7: Contingency score between real and synthetic categorical variables.

To evaluate whether our synthetic insurance portfolio could be used as a substitute for the original portfolio in actuarial analyses, we compare the Poisson deviance obtained with two different GLMs. The Poisson deviance is defined as:

$$D_{Poisson} = 2\sum_{i=1}^{n} N_i \left(\frac{\nu_i}{N_i}\hat{\lambda}_i - \left(\log \frac{\hat{\lambda}_i \nu_i}{N_i} + 1 \right) 1_{\{N_i \geq 1\}} \right)$$

The first GLM is trained on the original insurance portfolio and is used to predict the claim frequencies of the original policies. The Poisson deviance obtained using the 'observed' claim frequencies N_i^{real}/ν_i^{real} (where N_i is the number of claims in variable *NumberClaims* and ν_i the policy's exposure in variable *Duration*), and the predicted frequencies by the first GLM is equal to 5 691.01.

The second GLM is trained on the *synthetic* portfolio and is used to predict the claim frequencies of the *original* policies. We were able to fit a GLM to the synthetic data because both the continuous variable *Duration* and the multinomial variable *NumberClaims* were feeded as input to our VAE. The Poisson deviance obtained using the 'observed' claim frequencies of the synthetic policies N_i^{fake}/ν_i^{fake} (obtained by extracting *NumberClaims* and *Duration* from the *generative* decoder's outputs) and the predicted frequencies of the *original* policies (obtained using the second GLM trained on the synthetic portfolio) is equal to 5 961.84. Table 8 reports the Poisson deviances and the difference (in %) from the Poisson deviance obtained with the first GLM. The smaller this variation (%), the more our synthetic policies can be considered as a strong substitute to the real policies.

We also further assess the quality of the reconstructed portfolio by fitting a third GLM on the *reconstructed* portfolio and by predicting the claim frequencies of the *original* policies. The Poisson deviance was obtained using the 'observed' claim frequencies of the reconstructed policies $\hat{N}_i/\hat{\nu}_i$ (obtained by extracting *NumberClaims* and *Duration* from the decoder's outputs) and the predicted frequencies of the *original* policies (obtained using the third GLM trained on the reconstructed portfolio) is equal to 5 713.09, which is relatively close to the Poisson deviance obtained with the first GLM. This strengthens our previous quality analysis of the reconstructed portfolio.

	GLM model	Poisson deviance	%
Real data	1	5691.01	
Synthetic data	2	5961.84	4.76%
Reconstructed data	3	5713.09	0.39%

Table 8: Poisson deviances.

We finally assess to wich extent policyholders could be identified in the synthetic data. To do so, we ensure the generated synthetic policies were not copies of the original ones. The fake and original portfolio both contain 62 289 policies. In the fake portfolio, 2.55% of the fake policies are duplicate and 97.45% of the policies are unique. Among the unique fake policies, 1.82% are copies of the real policies and 98.18% of the synthetic policies are not copies of the real policies. On a side note, any repeated values between the real and synthetic data occur by random chance. The real, sensitive values thus remain anonymous as it is not possible to distinguish them from the fake ones - without access to the real portfolio.

5 Conclusions

This article explored the application of VAEs for high-dimensional and multitype insurance data. The objective of the study was to leverage the power of VAEs to extract meaningful and low-dimensional representations of an insurance portfolio to generate artificial data. Thanks to our quantile transformations and NNs architecture, we were able to generate realistic and diverse samples of synthetic insurance policies characterised by multinomial and multi-modal non-Gaussian variables. The ability of our VAE to uncover the underlying structure in the input data offers a promising avenue for insurance companies to enhance their decision-making processes by facilitating tasks such as risk assessment and claim prediction, and overcoming privacy concerns and potential biases associated with the use of authentic policyholder information. It is worth noting that while VAEs assure a bright outlook for insurance analytics, there are still challenges to overcome. In particular, the research highlighted the significance of appropriate model architecture and hyperparameters tuning to achieve optimal results. Careful consideration should be given to the selection of activation functions, layer sizes, regularization techniques, and optimization algorithms to ensure the effective training and performance of VAEs on insurance data.

For future research, the latent codes obtained from our VAE model could also serve as a valuable input for a subsequent unsupervised regression step (e.g., clustering), provided that the target variable (e.g., *NumberClaims*) is withdrawn from the encoder's input. By utilizing the blurred latent representation of the original rating factors, a clustering algorithm could become an important tool to overcome antidiscrimination laws or regulations that prohibit explicit usage of sensitive attributes, such as gender or ethnicity, while still revealing underlying patterns and groupings within the data. Finally, our generative model could also be adapted to create 'fair' synthetic portfolios that are less influenced by historical biases and discrimination and have a better representativeness of the world, leading to more equitable pricing and coverage decisions.

Acknowledgements

This work was supported by the Excellence of Science (EoS) under Grant ID 40007517.

References

[1] Hervé Abdi and Lynne J Williams. "Principal component analysis". In: *Wiley interdisciplinary reviews: computational statistics* 2.4 (2010), pp. 433–459.

[2] Jinwon An and Sungzoon Cho. "Variational autoencoder based anomaly detection using reconstruction probability". In: *Special lecture on IE* 2.1 (2015), pp. 1–18.

[3] Christoph Baur et al. "Deep autoencoding models for unsupervised anomaly segmentation in brain MR images". In: *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part I 4*. Springer. 2019, pp. 161–169.

[4] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[5] Jacob Deasy, Nikola Simidjievski and Pietro Liò. "Constraining variational inference with geometric jensen-shannon divergence". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 10647–10658.

[6] Łukasz Delong and Anna Kozak. "The use of autoencoders for training neural networks with mixed categorical and numerical features". In: *ASTIN Bulletin: The Journal of the IAA* (2021), pp. 1–20.

[7] Michel Denuit, Donatien Hainaut and Julien Trufin. *Effective Statistical Learning Methods for Actuaries III: Neural Networks and Extensions*. en. Springer Actuarial. Cham: Springer International Publishing, 2019. ISBN: 978-3-030-25826-9 978-3-030-25827-6. DOI: 10.1007/978-3-030-25827-6. URL: http://link.springer.com/10.1007/978-3-030-25827-6 (visited on 16/06/2023).

[8] Andrea Ferrario, Alexander Noll and Mario V Wuthrich. "Insights from inside neural networks". In: *Available at SSRN 3226852* (2020).

[9] Xavier Glorot, Antoine Bordes and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 315–323.

[10] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep learning*. MIT press, 2016.

[11] Ian J Goodfellow et al. "Generative Adversarial Nets". In: *stat* 1050 (2014), p. 10.

[12] Cheng Guo and Felix Berkhahn. "Entity embeddings of categorical variables". In: *arXiv preprint arXiv:1604.06737* (2016).

[13] Donatien Hainaut. "A neural-network analyzer for mortality forecast". In: *ASTIN Bulletin: The Journal of the IAA* 48.2 (2018), pp. 481–508.

[14] Donatien Hainaut. "A self-organizing predictive map for non-life insurance". In: *European Actuarial Journal* 9.1 (2019), pp. 173–207.

[15] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: *International conference on learning representations*. 2017.

[16] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[17] Mark A. Kramer. "Nonlinear principal component analysis using autoassociative neural networks". en. In: *AIChE Journal* 37.2 (1991), pp. 233–243. ISSN: 1547-5905. DOI: 10.1002/aic.690370209. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/aic.690370209 (visited on 08/05/2023).

[18] Matt J Kusner, Brooks Paige and José Miguel Hernández-Lobato. "Grammar variational autoencoder". In: *International conference on machine learning*. PMLR. 2017, pp. 1945–1954.

[19] Yann LeCun et al. "Efficient backprop". In: *Neural networks: Tricks of the trade*. Springer, 2002, pp. 9–50.

[20] Xiaopeng Li and James She. "Collaborative variational autoencoder for recommender systems". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2017, pp. 305–314.

[21] Chao Ma et al. "VAEM: a deep generative model for heterogeneous mixed type data". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11237–11247.

[22] Lars Mescheder, Sebastian Nowozin and Andreas Geiger. "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks". In: *International conference on machine learning*. PMLR. 2017, pp. 2391–2400.

[23] Alfredo Nazabal et al. "Handling incomplete heterogeneous data using vaes". In: *Pattern Recognition* 107 (2020), p. 107501.

[24] Alexander Noll, Robert Salzmann and Mario V Wuthrich. "Case study: French motor third-party liability claims". In: *Available at SSRN 3164764* (2020).

[25] Razvan Pascanu, Tomas Mikolov and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *International conference on machine learning*. Pmlr. 2013, pp. 1310–1318.

[26] Yunchen Pu et al. "Variational autoencoder for deep learning of images, labels and captions". In: *Advances in neural information processing systems* 29 (2016).

[27] Danilo Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *International conference on machine learning*. PMLR. 2015, pp. 1530–1538.

[28] Danilo Jimenez Rezende, Shakir Mohamed and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models". In: *International conference on machine learning*. PMLR. 2014, pp. 1278–1286.

[29] Suwon Suh and Seungjin Choi. "Gaussian copula variational autoencoders for mixed data". In: *arXiv preprint arXiv:1604.04960* (2016).

[30] Naftali Tishby, Fernando C Pereira and William Bialek. "The information bottleneck method". In: *arXiv preprint physics/0004057* (2000).

[31] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE." In: *Journal of machine learning research* 9.11 (2008).

[32] Lei Xu, Maria Skoularidou et al. "Modeling Tabular data using Conditional GAN". In: (Oct. 2019). arXiv:1907.00503 [cs, stat]. URL: http://arxiv.org/abs/1907.00503.

[33] Lei Xu and Kalyan Veeramachaneni. "Synthesizing Tabular Data using Generative Adversarial Networks". In: (Nov. 2018). arXiv:1811.11264 [cs, stat]. DOI: 10.48550/arXiv.1811.11264. URL: http://arxiv.org/abs/1811.11264.

[34] Weidi Xu et al. "Variational autoencoder for semi-supervised text classification". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[35] Shengjia Zhao, Jiaming Song and Stefano Ermon. "Towards deeper understanding of variational autoencoding models". In: *arXiv preprint arXiv:1702.08658* (2017).

Appendix

Proof. [Proposition 1] As $\int q_\phi(\mathbf{z}|\mathbf{x})d\mathbf{z} = 1$, the marginal log-likelihood is rewritten as follows:

$$\ln p_\theta(\mathbf{x}) = \int q_\phi(\mathbf{z}|\mathbf{x})\ \ln p_\theta(\mathbf{x})d\mathbf{z}\,.$$

Given that $p_\theta(\mathbf{x}) = \frac{p_\theta(\mathbf{x},\mathbf{z})}{p_\theta(\mathbf{z}|\mathbf{x})}$ (Bayes theorem), we can develop the integral in the following manner:

$$\begin{aligned}
\ln p_\theta(\mathbf{x}) &= \int q_\phi(\mathbf{z}|\mathbf{x})\ \left(\ln p_\theta(\mathbf{x},\mathbf{z}) - \ln p_\theta(\mathbf{z}|\mathbf{x})\right) d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x})\ \left(\ln p_\theta(\mathbf{x},\mathbf{z}) - \ln q_\phi(\mathbf{z}|\mathbf{x}) + \ln q_\phi(\mathbf{z}|\mathbf{x}) - \ln p_\theta(\mathbf{z}|\mathbf{x})\right) d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \ln\left(\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right) d\mathbf{z} + \int q_\phi(\mathbf{z}|\mathbf{x}) \ln\left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right) d\mathbf{z} \\
&= D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\,||\,p_\theta(\mathbf{z}|\mathbf{x})\right) + \mathcal{L}(\mathbf{x};\theta,\phi)\,.
\end{aligned}$$

□

Proof. [Proposition 2] By definition, the KL divergence is

$$D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})\,||\,p_\theta(\mathbf{z})\right) = \mathbb{E}_{q_\phi}\left[\ln q_\phi(\mathbf{z}|\mathbf{x})\right] - \mathbb{E}_{q_\phi}\left[\ln p_\theta(\mathbf{z})\right]\,,$$

where the densities are equal to

$$\begin{cases}
p_\theta(\mathbf{z}) & = \frac{\exp\left(-\frac{1}{2}\mathbf{z}^\top\mathbf{z}\right)}{(2\pi)^{d/2}}\,, \\
q_\phi(\mathbf{z}|\mathbf{x}) & = \frac{\exp\left(-\frac{1}{2}\sum_{k=1}^{d}\frac{\left(z_k-\mu_{z,k}(\mathbf{x})\right)^2}{\sigma_{z,k}(\mathbf{x})^2}\right)}{(2\pi)^{d/2}\sqrt{\prod_{k=1}^{d}\sigma^2_{z,k}(\mathbf{x})}}\,.
\end{cases}$$

A direct calculation allows us to infer that for $\mathbf{z}\in\mathbb{R}^d$

$$\begin{aligned}
\mathbb{E}_{q_\phi}\left(\ln q_\phi(\mathbf{z}|\mathbf{x})\right) &= \mathbb{E}_{q_\phi}\left[-\frac{d}{2}\ln(2\pi) - \frac{1}{2}\sum_{k=1}^{d}\ln\sigma^2_{z,k}(\mathbf{x}) - \frac{1}{2}\sum_{k=1}^{d}\frac{(z_k-\mu_{z,k}(\mathbf{x}))^2}{\sigma_{z,k}(\mathbf{x})^2}\right] \\
&= -\frac{d}{2}\ln(2\pi) - \frac{1}{2}\sum_{k=1}^{d}\ln\sigma^2_{z,k}(\mathbf{x}) - \frac{1}{2}\mathbb{E}_{q_\phi}\left[\sum_{k=1}^{d}\frac{(z_k-\mu_{z,k}(\mathbf{x}))^2}{\sigma_{z,k}(\mathbf{x})^2}\right] \\
&= -\frac{d}{2}\ln(2\pi) - \frac{1}{2}\sum_{k=1}^{d}\ln\sigma^2_{z,k}(\mathbf{x}) - \frac{1}{2}\sum_{k=1}^{d}\mathbb{E}_{q_\phi}\left[\frac{(z_k-\mu_{z,k}(\mathbf{x}))^2}{\sigma_{z,k}(\mathbf{x})^2}\right] \\
&= -\frac{d}{2}\ln(2\pi) - \frac{1}{2}\sum_{k=1}^{d}\left(1+\ln\sigma^2_{z,k}(\mathbf{x})\right)\,,
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}_{q_\phi}\left(\ln p_\theta(\mathbf{z})\right) &= \mathbb{E}_{q_\phi}\left[-\frac{d}{2}\ln\left(2\pi\right) - \frac{1}{2}\mathbf{z}^\top\mathbf{z}\right] \\
&= -\frac{d}{2}\ln\left(2\pi\right) - \frac{1}{2}\mathbb{E}_{q_\phi}\left[\mathbf{z}^\top\mathbf{z}\right] \\
&= -\frac{d}{2}\ln\left(2\pi\right) - \frac{1}{2}\sum_{k=1}^{d}\mathbb{E}_{q_\phi}\left[z_k^2\right] \\
&= -\frac{d}{2}\ln\left(2\pi\right) - \frac{1}{2}\sum_{k=1}^{d}\left(\sigma_{z,k}\left(\mathbf{x}\right)^2 + \mu_{z,k}\left(\mathbf{x}\right)^2\right) .
\end{aligned}$$

□

Algorithm 2 Quantile normal transformer used to scale continuous variables (following, e.g., multimodal non-Gaussian distributions).

Input: observed values $\{x_1, x_2, \ldots, x_n\}$ of a continuous variable X whose empirical CDF is denoted $F\left(X\right)$ (where n is, e.g., the number of policies).

Rank estimation: the CDF $F\left(X\right)$ is estimated by means of the rank index.

$n_q \leftarrow$ Set up the desired number of quantiles n_q used in the estimation of the CDF.

$q \leftarrow$ Create the reference set of quantiles $q = \left\{q_1, q_2, \ldots, q_{n_q}\right\}$. These quantile of reference represent the desired percentiles of the target distribution G and are used to discretize the CDF of X. They can be evenly spaced, such as $q_i = \frac{i-1}{n_q - 1} \times 100$ for $i = 1, 2, \ldots, n_q$.

$\boldsymbol{x} \leftarrow$ Sort the input vector $\boldsymbol{x}$ in ascending order: $\boldsymbol{x} = \left\{x_{(1)}, x_{(2)}, \ldots, x_{(n)}\right\}$ where $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$.

for each percentile value q_i:

$k_i \leftarrow$ Calculate the empirical rank $k_i = \frac{q_i}{100}\left(n-1\right)$ of the q_i-th desired percentile.

$p_i \leftarrow$ Determine the quantile rank, that is the position of the q_i-th percentile within the sorted values $\boldsymbol{x}$ through interpolation $p_i = \mathbb{I}_{\{k_i \in \mathbb{Z}\}} \times \left(x_{k_i} + \left(x_{k_i+1} - x_{k_i}\right)\left(k_i - \lfloor k_i \rfloor\right)\right) + \mathbb{I}_{\{k_i \notin \mathbb{Z}\}} x_{\lfloor k_i \rfloor}$ where $\mathbb{Z}$ denotes the set of integer.

$\frac{p_i}{n-1} \leftarrow$ Scale the quantile rank to the range $[0, 1]$ to obtain the estimated empirical distribution $p = F\left(X\right)$.

end for

$\Phi^{-1}\left(p\right) \leftarrow$ Map to a standard normal distribution by applying the inverse of the desired distribution $\mathcal{N}\left(0, 1\right)$, denoted Φ^{-1}.

$\frac{\Phi^{-1}(p) - \mu_{\Phi^{-1}(p)}}{\sigma_{\Phi^{-1}(p)}} = \left(x_i\right)_{i=1,\ldots,n}^{transformed} = X^{transformed} \leftarrow$ Scale the transformed values to have a zero mean and unit variance.

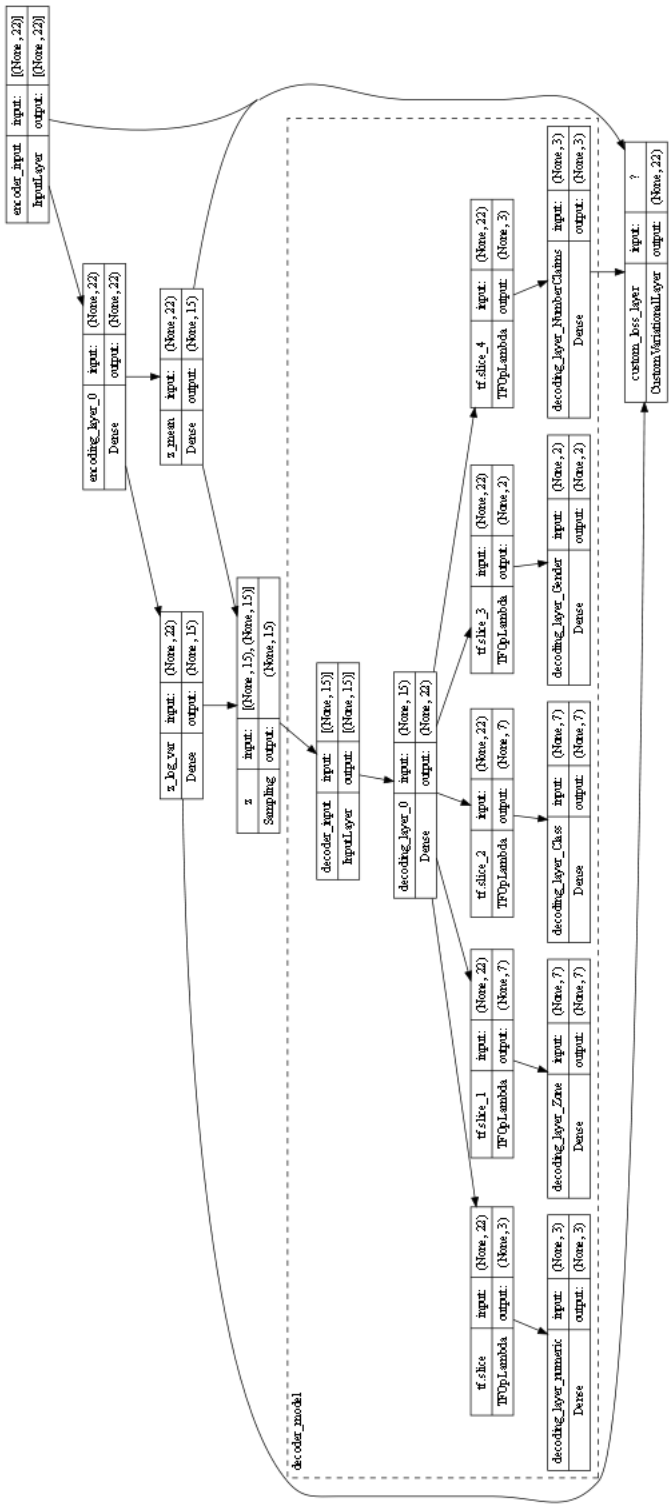

Figure 12: Model architecture.

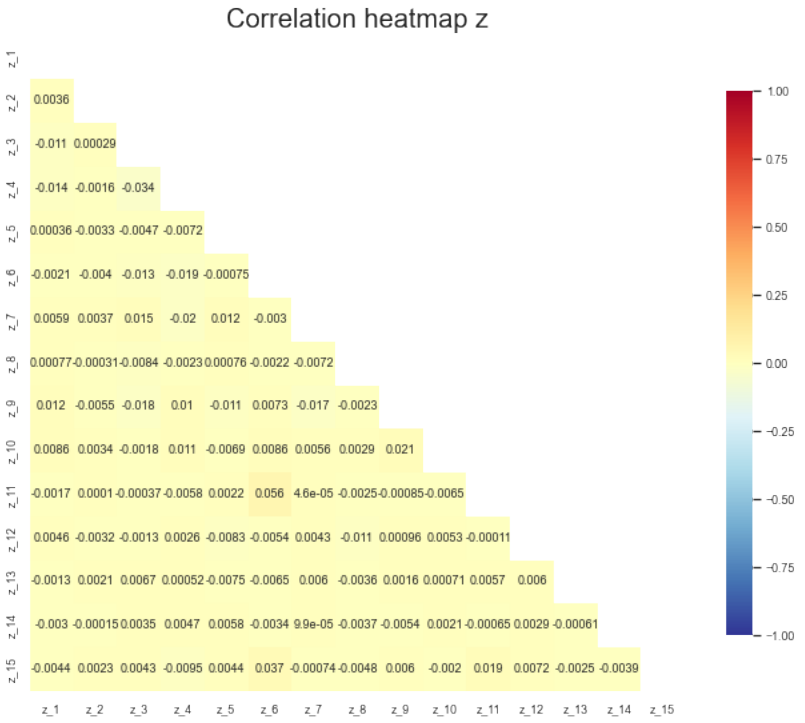

Figure 13: Pearson's correlation between the latent codes $(Z_j)_{j=1,\ldots,d}$.

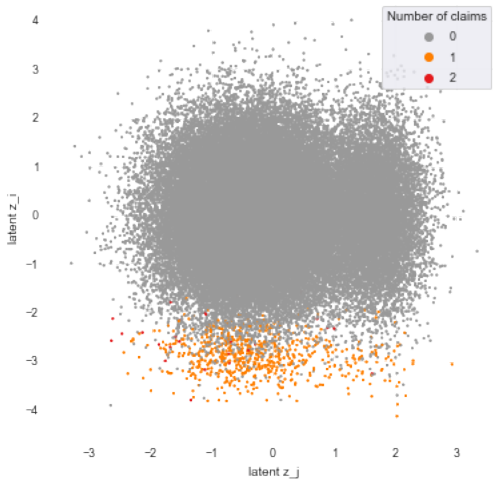

Figure 14: Scatter plot between the observed values of two latent variables Z_i and Z_j.

	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_7	Z_8	Z_9	Z_{10}	Z_{11}	Z_{12}	Z_{13}	Z_{14}	Z_{15}
Mean	-0.04	0.00	0.01	0.02	-0.01	-0.00	-0.02	0.01	0.03	-0.01	0.00	-0.00	0.00	-0.00	0.00
Standard deviation	1.00	1.00	0.99	1.00	1.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 9: Latent means and standard deviations.

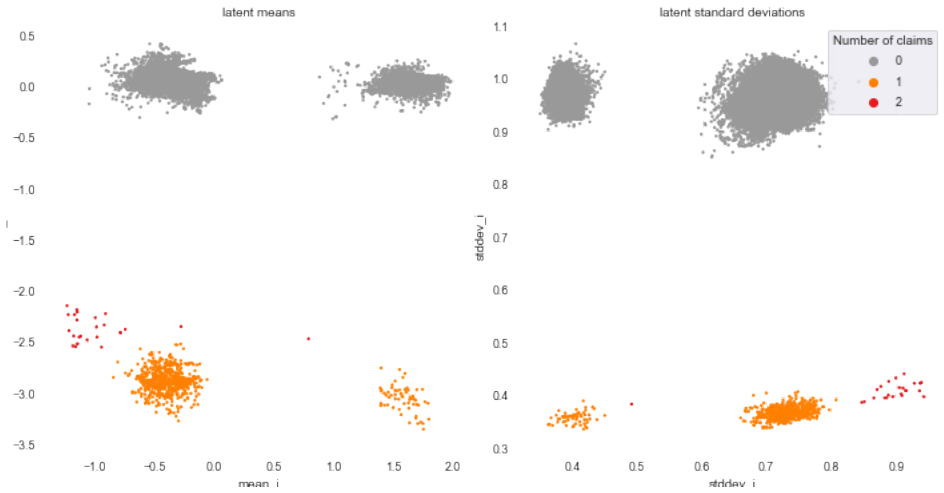

Figure 15: Scatter plot of the latent means and standard deviations between the observed values of two latent variables Z_i and Z_j.

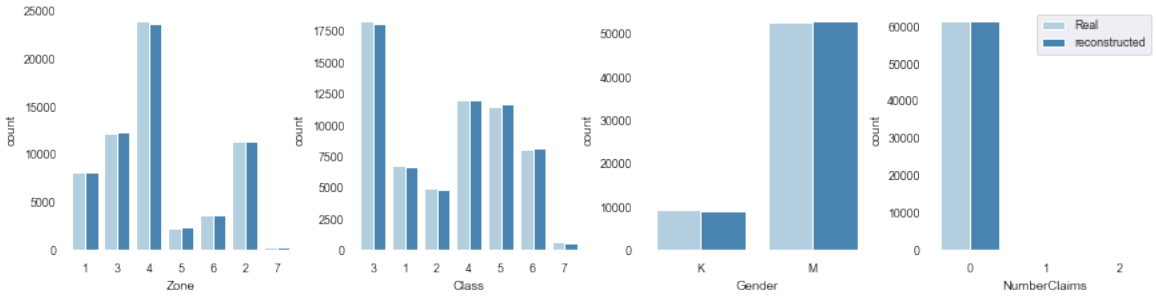

Figure 16: Comparison of real (light blue) and reconstructed (dark blue) categorical data distributions (frequency of each observed category).

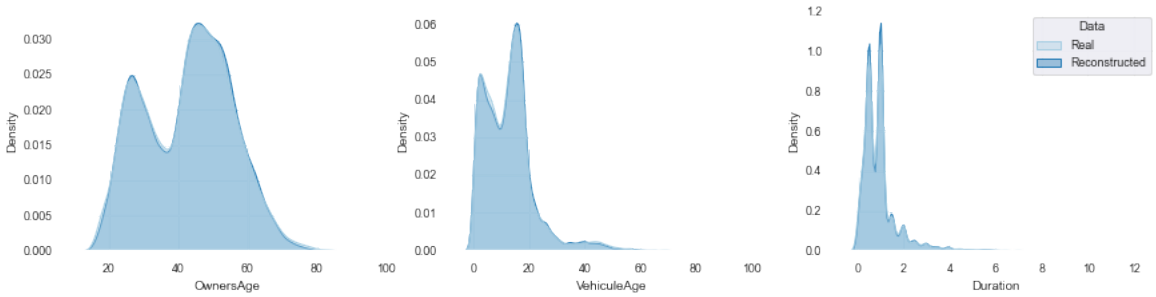

Figure 17: Comparison of real (light blue) and reconstructed (dark blue) continuous data distributions.